

Development and Evaluation of Protocols for the Operation of Wireless Ad-Hoc Networks with Quality-of-Service Requirements

PhD Thesis

*Thesis approved by the Department of Computer Science at
RPTU Kaiserslautern-Landau for the award of the Doctoral Degree*

Doctor of Engineering (Dr.-Ing.)

to

Christopher Kohlstruck

Date of Defense: December 15th, 2023
Dean: Prof. Dr. Didier Stricker
Reviewer: Prof. Dr. Reinhard Gotzhein
Reviewer: Prof. Dr. Ferhat Khendek

„Auf einmal merkt man, dass man Schritt für Schritt die ganze Straße gemacht hat. Man hat gar nicht gemerkt wie, und man ist nicht außer Puste.“

Beppo Straßenkehrer

Für Malika, Lilly und Tobi

Abstract

This thesis focuses on the operation of reliability-constrained routes in wireless ad-hoc networks. A complete communication protocol that is capable of guaranteeing a statistical minimum reliability level would have to support several functionalities: first, routes that are capable of supporting the specified Quality of Service requirement have to be discovered. During operation of discovered routes, the current Quality of Service level has to be monitored continuously. Whenever significant deviations are detected and the required level of Quality of Service is endangered, route maintenance has to ensure continuous operation. All four functionalities, route discovery, route operation, route maintenance and collection and distribution of network status information, will be addressed in this thesis.

In the first part of the thesis, we propose a new approach for Quality-of-Service routing in wireless ad-hoc networks called r_{\min} -routing, with the provision of statistical minimum route reliability as main route selection criterion. To achieve specified minimum route reliabilities, we improve the reliability of individual links by well-directed retransmissions, to be applied during the operation of routes. To select among a set of candidate routes, we define and apply route quality criteria concerning network load.

High-quality information about the network status is essential for the discovery and operation of routes and clusters in wireless ad-hoc networks. This requires permanent observation and assessment of nodes, links, and link metrics, and the exchange of gathered status data. In the second part of the thesis, we present cTEx, a configurable topology explorer for wireless ad-hoc networks that efficiently detects and exchanges high-quality network status information during operation.

In the third part, we propose a decentralized algorithm for the discovery and operation of reliability-constrained routes in wireless ad-hoc networks called dR_{\min} -routing. The algorithm uses locally available network status information about network topology and link properties that is collected proactively in order to discover a preliminary route candidate. This is followed by a distributed, reactive search along this preselected route to remove imprecisions of the locally recorded network status before making a final route selection. During route operation, dR_{\min} -routing monitors routes and performs different kinds of route repair actions to maintain route reliability in order to overcome varying link reliabilities.

Contents

1. Introduction	1
1.1. Routing in Ad-Hoc Networks	2
1.2. Quality-of-Service Routing in Ad-Hoc Networks	2
1.3. Reliability-Constrained Routing in Ad-Hoc Networks	3
2. r_{\min}-Routing – Discovery of Reliability-constrained Routes in Wireless Ad-Hoc Networks	5
2.1. Terminology	6
2.2. Network Model	8
2.3. Route Discovery	9
2.3.1. Motivation	9
2.3.2. Link Quality Metric	11
2.3.3. SMTX – A New Routing Metric	12
2.3.4. Route Selection Heuristic	19
2.3.5. Route Operation	24
2.3.6. Routing Metrics for Wireless Ad-Hoc Networks	25
2.4. Simulation Experiments	27
2.4.1. Experiment Setup	28
2.4.2. Assessment of r_{\min} -Routing	30
2.4.3. Comparison with Other Metrics	34
2.5. Conclusion	39
3. cTE_x – Decentralized Topology Detection using a Link State Algorithm	41
3.1. cTE _x Algorithm Overview	43
3.1.1. Link Status Detection	43
3.1.2. Distribution of Topology Information	45
3.2. Reference Link Quality Metric for cTE _x	47
3.3. Formal Specification of cTE _x Algorithm	48
3.3.1. Network Model	49
3.3.2. Functions, Global Parameters and State Variables of cTE _x	50
3.3.3. Reception and Processing of Probe Frames	52
3.3.4. Transmission of Probe Frames	56
3.3.5. Exchange of Network Status Information	58
3.3.6. Processing of Network Status Information	60

3.4.	EAR – Efficient and Accurate Link Monitor	61
3.4.1.	Link Quality Metric	61
3.4.2.	Measurement Schemes	62
3.4.3.	Communication Overhead	64
3.4.4.	Accuracy of EAR	64
3.5.	cTE _x and EAR	65
3.6.	Simulation Experiments	66
3.6.1.	Simulator Setup	66
3.6.2.	Accuracy of Link Quality Measurements	68
3.6.3.	Topology Distribution Latency	70
3.6.4.	Overhead	72
3.6.5.	Testbed Experiments	75
3.7.	Related Work	77
3.8.	Conclusion	78
4.	dR_{min}-Routing – Decentralized Discovery, Operation and Maintenance of Bandwidth-constrained Routes	81
4.1.	The dR _{min} Routing Protocol	82
4.1.1.	Network Status Information	83
4.1.2.	Preselection of a Route Candidate	84
4.1.3.	Cooperative Discovery of an Initial Route	85
4.1.4.	Route Repair during Operation	86
4.1.5.	Pseudocode of the dR _{min} -Routing Algorithm	87
4.2.	Assessment of dR _{min} -Routing	95
4.2.1.	Experiment Setup	95
4.2.2.	Analysis of Route Reliabilities	96
4.2.3.	Latency and Overhead of Route Search	96
4.2.4.	Route Discovery Success Ratio	97
4.3.	Comparison of dR _{min} -Routing with OLSR	98
4.3.1.	Comparison of Route Reliability	99
4.3.2.	Comparison of Network Throughput	101
4.4.	Preparations for Real-World Experiments	101
4.4.1.	The Gilbert-Elliott Model	103
4.4.2.	Time Token Bucket Algorithm	104
4.4.3.	Preliminary Experiments and Future Work	105
4.5.	Related Work	105
4.6.	Conclusion	106
5.	A Demonstrator and Runtime Controller for a Small Demonstration Network	109
5.1.	Testbed Setup	110
5.2.	Runtime Control of a Protocol Stack	111

5.3. Visualization of Network State	113
5.4. Showcase Experiments using the Demonstrator	114
5.5. Tools and Frameworks for Experiment Evaluation	120
5.6. Conclusion	122
6. Conclusions	125
6.1. Summary	125
6.2. Limitations and Future Work	127
A. Appendix A	129
B. Appendix B	135
Author's Contributions	141
Bibliography	143

List of Figures

2.1. Example topology	10
2.2. Example Topology for Greedy Choice Property	18
2.3. Example Topology for f_{sp}	21
2.4. 11-node Example Topology	24
2.5. Reliability of SMTX routes	31
2.6. Reliability of SP routes	35
2.7. Reliability of routes found with ETX and ETOP.	36
2.8. Reliability of routes found with ML metric.	37
2.9. Match Ratio per Reliability Target Level	37
3.1. Timing of Probe Creation	58
3.2. Mode Transition Diagram of EAR	62
3.3. Performance of EAR's Passive, Cooperative and Active modes	64
3.4. Topology T1.	65
3.5. Architecture of the WiPS framework	67
3.6. Topologies T1 and T2 from Original EAR Paper	68
3.7. Delivery Ratio over Link A to B for cTEx and EAR	69
3.8. Delivery Ratio over Link B to C for cTEx and EAR	69
3.9. 36-node Lattice Topology	71
3.10. Topology Information Distribution Latency	72
3.11. Comparison of Calculated and Measured Overhead	74
3.12. 11-node Real-World Testbed Topology	75
3.13. Real-World Link Quality Estimation	76
3.14. Network Topology Snapshot View	76
4.1. Example Topology	85
4.2. Route Repair in dR_{min} -Routing	96
4.3. Route Discovery Latency	97
4.4. Route-not-Found Ratio	98
4.5. Route Reliability with dR_{min} -Routing and OLSR	99
4.6. Reliability Comparison of 1448 Routes	100
4.7. Comparison of Average Reliabilities	100
4.8. Comparison of Network Throughput	101
4.9. Gilbert-Elliott Model	104
5.1. Demonstrator Testbed	110

List of Figures

5.2. Runtime Control Mechanism Architecture	112
5.3. Topology plot.	114
5.4. Demonstrator Web Interface Screenshot	115
5.5. Scatter, Graph and Tree Plot Visualizations	116
5.6. Information Flow Diagram	120
5.7. Grafana Dashboard Screenshot	123
A.1. Topologies for r_{\min} -Routing simulations	130
A.2. Topologies for r_{\min} -Routing simulations	131
A.3. Topologies for r_{\min} -Routing simulations	132
A.4. Topologies for r_{\min} -Routing simulations	133

List of Tables

2.1. Terminology	7
2.2. Properties of Boosted Edges and Paths	10
2.3. Properties of Paths in Fig. 2.3, for $r_{min} = 0.8$	21
2.4. Routing Metrics Values for Paths from Fig. 2.3	26
2.5. r_{min} -Routing Route Selection Parameters	32
3.1. List of Functions used in Pseudocode Descriptions of cTE _x	50
3.2. List of Global Configuration Parameters of cTE _x	51
3.3. List of State Variables of each cTE _x Node	52
3.4. List of State Variables of Link State Automata	55
3.5. Parameters of Overhead Experiments	73
3.6. Parameters of Real-World Experiments	75
4.1. Effective Reliabilities in Fig. 4.1	85
4.2. Definitions used in Listings 4.1 to 4.5	89
4.3. Helper Functions used in Listings 4.1 to 4.5	90
5.1. Available Demonstrator Chart Types	115
5.2. Example Playbook for Showcasing dR _{min} -Routing	118
B.1. Playbook for Showcasing dR _{min} -Routing Route Repair	136
B.2. Playbook for Showcasing cTE _x Topology Distribution	137
B.3. Playbook for Showcasing cTE _x Passive Probing	138

Listings

2.1. Pseudocode for Calculation of $SMTX_{p,r}^{l_{max}}$	15
2.2. Helpers $r(p)$, $cost(p)$ and $q(e)$	15
3.1. cTE _x Link State Automaton, part 1	53
3.2. cTE _x Link State Automaton, part 2	54
3.3. Generation of Explicit Probe Frames	57
3.4. Embedment of Network Status Information	59
3.5. Processing of Network Status Information	60
4.1. Processing of REQUEST Messages	91
4.2. Processing of RESPONSE Messages	92
4.3. Processing of ABORT Messages	92
4.4. Processing of TRANSPORT Messages	93
4.5. Events <code>conn_req</code> and <code>data_req</code>	93
4.6. Sending TRANSPORT Messages and Route Repair	94
5.1. State Variable Report Example	113
5.2. InfluxDB Line Protocol Example	121
5.3. Flux Query Example	122

List of Acronyms

ACK	Acknowledgement
AODV	Ad-hoc On-Demand Distance Vector
BAP	Broadcast-based Active Probing
CDF	cumulative distribution function
CTS	Clear-to-Send
DCF	Distributed coordination function
DSR	Dynamic Source Routing
EAR	Efficient and Accurate link-quality monitor
ECC	Error-correcting Codes
ETOP	Expected Number of Transmissions On a Path
ETT	Expected Transmissions Time
ETX	Expected Number of Transmissions
EWMA	Exponentially Weighted Moving Average
FDR	frame delivery ratio
HTTP	Hypertext Transfer Protocol
HWMP	Hybrid Wireless Mesh Protocol
IEEE	Institute of Electrical and Electronics Engineers
JSON	JavaScript Object Notation
LABQ	Link Availability-based QoS-aware routing
LSA	Link state automaton
MAC	Medium access control
MPR	Multi-Point Relay

List of Acronyms

OB-EWMA Outlier Bounded Exponential Weighted Moving Average

OLSR Optimized Link State Routing

PCF Point Coordination Function

PDR packet delivery ratio

PHY Physical layer

QMRB-AODV QoS Mobile Routing Backbone over AODV

QOLSR QoS-enhancement for the OLSR protocol

QoS Quality of Service

RTS Ready-to-Send

SMTX Smallest Maximum Number of Transmissions

TBP Ticket Based Probing

TDMA Time Division Multiple Access

TTBA Time Token Bucket Algorithm

UDP User Datagram Protocol

WLAN Wireless Local Area Network

WMEWMA Window Mean with EWMA

WSN Wireless Sensor Network

1. Introduction

Communication in a wireless network is inherently error-prone. This is different from most wired networks, where one may assume that a single communication attempt is usually successful. For wireless communication in an ad-hoc network (i.e. a spontaneously created wireless network without prior setup of special infrastructure), the way of avoidance and handling of failed communication attempts is a defining characteristic of almost all modern protocols at the physical, data link and network layer.

The reason behind the fragility of wireless communication is simple: the communication medium is shared, and successful reception requires a certain minimum signal strength. It is shared not only between all participants of a single network, but all members of all networks in vicinity. Even devices not designed for wireless communication at all, such as microwave ovens, emit electromagnetic waves that significantly impact wireless communication.

The number of measures devised to facilitate sharing of the communication medium is high. It ranges from obvious measures such as spatial or temporal separation, to sophisticated modulation and encoding schemes for the information to be transmitted to make wireless communication more reliable.

With some protocols, the network can be operated in a deterministic fashion, e.g. by assigning exclusive time slots for communication to each node in the network. This determinism can be leveraged to provide an application with real-time guarantees about the Quality of Service (QoS), which is a requirement by many industry applications. Those guarantees however can only be met as long as all emitters of (sufficiently powerful) electromagnetic waves in proximity of the network adhere to the same protocol, which implies the assumption of the so-called single network property. As the communication medium however is shared, as in many real world scenarios, this property usually does not hold. This applies especially to wireless ad-hoc networks, as assuring absence of any interfering networks, and more generally interfering emitters of electromagnetic waves is not straight-forward and is comparable to the prior set up of network infrastructure, which ad-hoc networks by definition do not require. This justifies a closer look at wireless communication protocols that incorporate functionalities to cope with dynamic levels of interference from arbitrary emitters.

One of the major protocol families that incorporates functionalities for assuring a specific QoS level for multi-hop communication in wireless ad-hoc networks is routing protocols. Routing is a core functionality of multi-hop net-

1. Introduction

works in general, and of wireless ad-hoc networks in particular. The purpose of routing is the discovery and operation of routes to provide end-to-end communication between sets of nodes. A route is a tree of links, where a link is a direct connection between a pair of nodes. Depending on the application context, routes may be required to satisfy certain QoS requirements, for instance, regarding throughput, delay, reliability, and resource consumption. In this thesis, we will focus on reliability-constrained routing, where routes have to satisfy minimum statistical reliability targets.

1.1. Routing in Ad-Hoc Networks

Routing is an essential functionality for directing communication across large networks. However, with the vast array of applications for wireless ad-hoc networks, finding a single routing protocol suitable for every application is challenging. As a result, several routing protocols have emerged over years of extensive research, each specifically designed to address the unique requirements of different wireless communication domains.

The demands and challenges posed on routing protocols are vast and diverse. On the one hand, many applications require a minimum quality of service with respect to latency, reliability, jitter or bandwidth. A network might also consist of several different types of nodes, varying in power, battery size or other capabilities (a so-called heterogeneous network). An especially hard challenge to a routing protocol poses node mobility, which is relevant with communication protocols for a swarm of flying drones for instance.

As addressing all these demands by a single protocol is close to impossible, especially as some of the demands might be almost mutually exclusive (minimal power consumption and minimal latency for instance), a common approach is to define the intended application domain of a routing protocol beforehand. Thus, only the demands common to the specified application domain need to be met. A consequence of this is the evidently vast number of specialized routing protocols available today. Unsurprising to the experienced reader, the already large family of specialized routing protocols for wireless ad-hoc networks will gain another member in the course of this thesis.

1.2. Quality-of-Service Routing in Ad-Hoc Networks

QoS routing plays a crucial role in networks that support distributed applications with real-time requirements, including distributed control and production systems. Its primary objective is to find routes that meet specific constraints on performance, reliability, and resource consumption. By taking into account these constraints, QoS routing reduces the number of potential route candidates

compared to pure best effort routing, thereby leading to a selection of routes with more predictable behavior.

QoS route discovery relies on network status information, which may either be collected proactively or reactively at runtime. This information could be centralized in a routing manager or distributed across a network's nodes. Ultimately, the quality of QoS route selection depends on the amount, type, accuracy, and timeliness of the available network status data.

However, achieving QoS requirements in wireless networks presents a unique set of challenges due to the dynamic and fragile nature of wireless communication links. A wireless link can frequently break and restore multiple times within seconds, making it necessary to continuously monitor the link state to determine its suitability for a route is supposed to meet specific QoS requirements.

1.3. Reliability-Constrained Routing in Ad-Hoc Networks

The reliability of links in wireless ad-hoc networks varies due to several factors such as distance, terrain, signal strength, and interference. Therefore, the choice of proper links to form a route is of crucial importance for reliability-constrained routing. Unlike wired networks, selecting the shortest routes based on the number of hops is not recommended in wireless networks due to the preference given to weaker and therefore less reliable links. Instead, it is better to prioritize routes with sufficiently strong links as those with weak links necessitate more retransmissions and are prone to failing. Therefore, wireless networks require careful consideration in selecting the right links in order to form reliable routes.

To increase the perceived strength or reliability of a link, retransmission of lost frames is a common approach. In WiFi networks, the same frame may be transmitted up to 7 times. Knowing this, we can conclude that the perceived reliability of a communication link strongly depends on the current level of abstraction. For instance, a distributed application that runs on top of a wireless ad-hoc network may experience a perfect link to a destination node, whereas on link level 30% of transmitted frames are not received correctly.

In this thesis, we will explore the impact of individual retransmission budgets adjusted for the raw reliability of each link in a route on a route's end-to-end reliability. We show that through optimal distribution of a fixed budget of transmissions, careful and continuous monitoring of link state and sophisticated route repair mechanisms operation of routes that satisfy minimum reliabilities targets set by the application is possible.

The structure of this thesis is as follows. In Chapter 2, a route discovery algorithm called r_{\min} -routing for routes that satisfy a minimum reliability level is

1. Introduction

introduced. We prove the optimality of our transmission budget allocation and define a routing metric for efficient search of feasible routes. As r_{\min} -routing assumes availability of complete and up-to-date topology information, a configurable topology detection and distribution protocol called cTEx was developed and is presented in Chapter 3. The chapter includes a complete specification of the protocol, a thorough evaluation in simulation and testbed experiments including a thorough comparison to a related protocol from the literature. Chapter 4 introduces dR_{\min} -routing, a complete, decentralized routing protocol that supports discovery, operation and maintenance of r_{\min} routes. In the course of the chapter, dR_{\min} -routing is specified semi-formally in form of pseudocode, implemented, evaluated in simulation experiments and compared with the often cited OLSR protocol. A summary of preconditions for a real world evaluation of dR_{\min} -routing is provided towards the end of the chapter. As illustration of the capabilities of complex communication protocols to a layman audience can be challenging, a demonstrator testbed and framework was designed and is presented in Chapter 5. In this chapter the extensible design and architecture of the framework for runtime control of a protocol stack is outlined. We also introduce so-called playbooks, which describe a tested sequence of actions on the demonstrator for showcasing specific communication protocol capabilities. Chapter 6 contains a summary of the findings of this thesis and an outlook with remarks on future work.

2. r_{\min} -Routing – Discovery of Reliability-constrained Routes in Wireless Ad-Hoc Networks

As outlined in the introduction, the concept of reliability-constrained routing is an aspect of Quality of Service in wireless ad-hoc networks: Routes are selected such that a specified quality requirement, such as reliability, is satisfied. In this chapter, we show an approach for the discovery of routes in wireless ad-hoc networks that satisfy a specified reliability target during route operation.

Reliability is an important requirement with many applications. Imagine a factory where production is monitored wirelessly using data provided by sensors. Some loss of sensor data might be tolerable to assuring safe operation. In case the level of tolerable loss is exceeded, production has to be stopped. Often the level of tolerable loss can be specified precisely enough such that a Quality of Service requirement can be constructed, which can then be ensured by the communication protocol. Another application are scenarios, where error-correcting codes (ECC) are used to reconstruct lost messages, using redundancy information embedded in received messages. The degree of loss where reconstruction is possible is known exactly, as it depends on the amount of redundancy information included in each message. The corresponding reliability requirement is then enforced by the communication protocol. Whenever lost messages can be reconstructed instead of having to be re-transmitted, a request for a re-transmission, which would have to be sent in the opposite direction of the data flow, is saved. This also reduces delay caused by re-transmission requests. Furthermore, such a request may be lost. This approach is especially useful when communication is asymmetric, i.e. the sender almost exclusively sends, whereas the sink is supposed to receive only (e.g. in case of multicast or one-way communication links).

A complete communication protocol that is capable of guaranteeing a statistical minimum reliability level would have to support several functionalities: first, routes that are capable of supporting the specified Quality of Service requirement have to be discovered. This requires collection and exchange of accurate network status information, such as link qualities. During operation of discovered routes, the current Quality of Service level has to be monitored continuously. Whenever significant deviations are detected and the required level of

2. r_{\min} -Routing

Quality of Service is endangered, route maintenance has to ensure continuous operation. Route maintenance includes the decision whether a route can be repaired or needs to be re-established. All four functionalities, route discovery, route operation, route maintenance and collection and distribution of network status information, will be addressed in this thesis.

In this chapter a solution for the first functionality, the discovery of routes, is proposed. As the objective is to find routes satisfying a *minimum* level of *reliability*, the proposed approach is called r_{\min} -routing [KMG19]. We begin by establishing some common terminology as well as a network model, which is also the base of most formalisms introduced in later chapters of this thesis. The model is designed to capture the influence of re-transmissions on the perceived reliability of a link. This is an important aspect of r_{\min} -routing: to increase the perceived reliability of specific links, a budget of transmission attempts is distributed over the links of a route. Thus, weaker links can be compensated for such that a route becomes capable of satisfying a specified reliability target.

Next, a novel routing metric is introduced to quantify the aptness of a route candidate for the fulfillment of the specified reliability requirement. This is followed by a detailed description of the discovery phase of r_{\min} -routing, where viable route candidates are searched in a global view of the network state. As the number of all possible paths between a source and destination quickly exceeds the computationally feasible, several filtering steps are applied to reduce the set of route candidates. In a final step of filtering a single route is selected that is expected to satisfy the minimum reliability target.

Towards the end of this chapter, we will present simulation experiments showing that the route selection heuristic in combination with the newly developed metric leads to routes that satisfy the specified reliability target. In addition, we show that existing routing metrics that are not tailored to the requirements of reliability-constrained routing, fail to continuously meet reliability targets.

2.1. Terminology

As several terms specific to the networking community, such as *packet* and *frame*, are sometimes used interchangeably though subtle differences matter in some contexts, we will establish a common terminology in this section. An overview of the established terms for each layer of abstraction is provided in Table 2.1.

In the network model specified in Section 2.2, the network is modeled as a directed graph of *vertices*, connected through *edges*. A sequence of vertices connected directly through directed edges is called a *path*. The outcome of a single *communication attempt* along an edge in the network model is captured as *probability of success*.

In the context of communication protocols, wireless communication interfaces

Table 2.1: Established terminology in tabular form, per layer of abstraction.

Network Model	MAC Layer	Network Layer
Vertex	Node	Network Address
Edge	Link	–
Path	Route	Connection / Established Route
Probability of Success	Reliability	–
Communication Attempt	Transmission	Transport
–	Frame	Packet

installed on physical network *nodes* exchange data over wireless communication *links*. Though several wireless communication interfaces per node are possible, we assume one communication interface per node, unless stated otherwise. A wireless communication link has two properties: a *channel* and a *data rate*. Each physical node in the network is represented by a vertex in the network model, and each link corresponds to a directed edge in the graph. In analogy to paths from the network model, a sequence of interconnected links is called a *route*. A communication attempt via an edge between vertices in the network model is called a *transmission* over a link between physical nodes, consequently. The sequence of bits that is sent in a single transmission is called a *frame*. The *probability of success* of communication attempts over edges between vertices in the network model is a value we can in practice only approximate through observations, so we refer to the respective concept as *reliability* of a link between network nodes to reflect that fact in the context of communication protocols. Different from the probability of success for communication attempts, which is a clearly defined mathematical concept, the reliability of a link leaves room for interpretation. In this thesis, we assume the reliability to be the value assigned to each link by a reliability-focused link quality metric of some sort.

One layer of abstraction above, at the network level, *packets* are exchanged between *network addresses* via *connections*, or *established routes*. In analogy with communication interfaces and network nodes, we assume a single network address per communication interface, unless specified otherwise. A packet is a sequence of bits formatted for communication over a sequence of wireless links (a route), which is eventually transmitted in one or more frames. A sequence of packets that travels in the same direction along an established route between a pair of addresses is a *stream*.

The established terms will be used consistently throughout this thesis and shall also serve the reader as indicators of the current level of abstraction.

2.2. Network Model

In standard literature [vS10, Zwe16], a communication network is often modeled as a graph $G = (V, E)$, with vertices $v \in V$ and edges $e \in E$, where $E \subseteq V \times V$, where vertices and edges model physical nodes and links, respectively. This definition is suitable for modeling simpler networks, where the cost of traversing an edge is equal for all edges. For the definition of complex quality requirements however, the model turns out to be insufficient. A possible initial remedy to this deficit is the introduction of a function $w : E \rightarrow \mathbb{R}^+$, which assigns a weight to each edge. With w , the model supports the definition of more sophisticated link quality metrics: w can be defined to capture latency, load or reception strength of links between network nodes. With an appropriate routing metric, the model supports selection of routes satisfying specified quality requirements. It is however, lacking flexibility regarding the set of edges. Currently, the set of edges is just a collection of pairs of vertices (v_1, v_2) , with $v_1, v_2 \in V$: no additional properties of the set of edges, such as asymmetry, can be specified.

Route discovery in r_{\min} -routing is based on a network model $G = (V, r)$, with vertices $v \in V$ and reliability function

$$r : V \times V \rightarrow \mathbb{R}_{0,1} \quad (2.1)$$

Based on r , the set of asymmetrical edges is derived as

$$E =_{df} \{(v, v') \in V \times V \mid r(v, v') > 0\} \quad (2.2)$$

With this definition, links modeled as edges in G are inherently asymmetrical and the set E is equivalent to the set of node pairs where transmission attempts can be successful. For edges $e = (v, v') \in E$, we will abbreviate $r(v, v')$ to r_e . We will further call a sequence of edges p , with $p = \langle (v_0, v_1), (v_1, v_2), \dots, (v_{n-1}, v_n) \rangle \in E^+$ a *path*. Usually, we write path p as $\langle v_0, v_1, v_2, \dots, v_n \rangle$ for legibility. Note that the length $|p|$ of a path p is always the number of hops, i.e. the number of edges in the path. We also restrict r to single-hop communication only: For $v, v' \in V$, we assume $r(v, v') = 0$, if v' is not in communication range of v . To model multi-hop communication along a path p , we introduce $r_p : E^* \rightarrow \mathbb{R}_{0,1}$. A key assumption is that the sequential traversal of edges e in a path p is a Bernoulli trial, such that

$$r_p = \prod_{e \in p} r_e \quad (2.3)$$

holds.

To this point, the exact semantics of the quality function r was left open. As our aim is to discover routes with a specified minimum end-to-end reliability, we define r_e as the probability that a single communication attempt between the pair of vertices connected through e succeeds. A consequence from Eq. (2.3) is

that the success probability of communication attempts is assumed to be independent and identically distributed. In reality, this assumption is too strong and necessitates careful measures to retain validity of the model.

As stated above, r_e captures the probability of success for a single communication attempt. Especially in wireless networks, links are inherently unreliable and re-transmissions are common. Based on the introduced network model and the assumption that repeated communication attempts are Bernoulli trials, the probability of success after up to $n \in \mathbb{N}^+$ attempts is equal to the inverse of the probability of all n attempts failing, which gives

$$r_{e,n} \stackrel{\text{def}}{=} 1 - (1 - r_{e,1})^n \quad \text{where } r_{e,1} = r_e \quad (2.4)$$

for the probability of success of n communication attempts over edge e . We call edges with $n > 1$ *boosted*.

For a given path $p \in P$ and up to n_e communication attempts on edge $e \in p$, we obtain the maximum number of attempts n_p over all edges in p , and end-to-end reliability of p under up to n_p communication attempts r_{p,n_p}

$$n_p = \sum_{e \in p} n_e \quad (2.5)$$

$$r_{p,n_p} = \prod_{e \in p} r_{e,n_e} \quad (2.6)$$

2.3. Route Discovery

The process of route discovery in r_{\min} -routing relies on accurate and up-to-date topology information, including precise estimates of the reliability of each link in the network. For this section, we assume this information is locally available on each node. A sophisticated protocol for collection and distribution of information on network topology and network status is introduced in Chapter 3.

2.3.1. Motivation

The core objective of r_{\min} -routing is to discover routes that satisfy a specified reliability target. This reliability target r_{\min} imposes a minimum end-to-end reliability requirement over a specified time frame d on the selected route, i.e. at any point in time, the ratio of *packets received* to *packets sent* over the last d seconds is greater than or equal to r_{\min} .

Fig. 2.1 depicts a small topology, where each edge e is annotated with the respective success probability r_e of a single communication attempt. For a path between vertex v_1 and v_3 , this topology contains two options: the path $p_1 = \langle v_1, v_3 \rangle$, with aggregated path reliability $r_{p_1} = 0.7$ and the path $p_2 = \langle v_1, v_2, v_3 \rangle$,

2. r_{min} -Routing

with $r_{p_2} = 0.8075$. If we specify the minimum reliability r_{min} for the path between v_1 and v_3 as 0.7, all cycle-free paths in the given topology satisfy this requirement. For $0.7 < r_{min} \leq 0.8075$, only the longer path is possible.

To support more demanding end-to-end reliability targets, r_{min} -routing applies *link boosting* (in analogy to boosting of edges as introduced with the network model in Section 2.2), to increase the perceived reliability of a link. If we specify $r_{min} = 0.9$, boosting of edges is necessary to find a feasible path. For the small example topology depicted in Fig. 2.1, we will increase the number of communication attempts globally to 2 per edge.

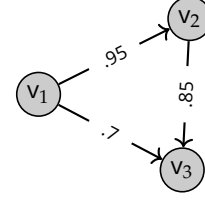


Figure 2.1: Example topology

Table 2.2: Path and edge properties after boosting

Path p	Edge e	$r_{e,1}$	n_e	$r_{e,2}$	n_p	r_{p,n_p}	
p_1	$\langle v_1, v_3 \rangle$	(v_1, v_3)	.70	2	.910	2	.910
p_2	$\langle v_1, v_2, v_3 \rangle$	(v_1, v_2)	.95	2	.998	4	.985
		(v_2, v_3)	.85	2	.978	4	.985

By assigning one additional communication attempt to the edges (v_1, v_2) and (v_2, v_3) , path p_2 becomes feasible and meets the specified r_{min} requirement. One additional attempt over edge (v_1, v_3) is also sufficient to render the second candidate path p_2 feasible. The changed path and edge properties after application of the boost mechanism are listed in Table 2.2.

So through boosting, previously unfeasible paths can be made feasible for a given reliability target. In this example, we increased the limit for communication attempts *globally*. Hence, initial transmission attempts on every link are retried once, when a failure is detected. However, this approach of increasing a global transmission limit does not scale for larger networks and more demanding reliability targets.

With a global limit of communication attempts of 2 from the example above we observe that the boosted reliability $r_{p_2,4}$ of path $p_2 = \langle v_1, v_2, v_3 \rangle$ is 0.985 after the assignment of extra attempts, which is well above the reliability target of 0.9. In fact, p_2 would still meet the specified r_{min} requirement, with $n_{(v_1, v_2)} = 1$. However, for r_{min} requirements lower than 1, some failed communication attempts (and consequently lost frames and packets) can be tolerated. Any extra attempts boost the path in question to an unnecessarily high reliability level. Hence, we call paths such as p_2 , where a lower limit of communication attempts still satisfies the r_{min} requirement *over-provisioned*. As wireless networks operate

on a shared medium, over-provisioning should be avoided, as additional transmissions are not necessary to satisfy the reliability requirement but reduce bandwidth available to other nodes. A transmission limit configured globally would lead to excessive over-provisioning. This calls for a more elaborate strategy, which increases the transmission limit only on specific links, where feasibility is still assured while over-provisioning is kept at a minimal level.

At this point, we can summarize our observations as two requirements on the path selection strategy and for the assignment of communication budgets:

- *Feasibility*: The reliability target r_{min} can be satisfied, i.e. there exists, for each edge $e \in p$, a value n_e such that $r_{p,n_p} \geq r_{min}$.
- *Optimality*: Bandwidth consumption is minimized (no over-provisioning), i.e. n_p is minimal.

We call a path p *feasible*, if it can match the reliability target r_{min} ; otherwise, it is unfeasible. We further call p *optimal*, if only the minimum number of communication attempts n_e necessary to achieve r_{min} over p is assigned to each edge $e \in p$. The terms *feasible* and *optimal* are also used with regards to routes, where they convey the same semantics.

The requirements for routes discovered with r_{min} -routing are now quite clear. The requirements regarding the different components of r_{min} -routing with respect to the fulfillment of the specified route properties are stated and addressed in the following sections:

- r_{min} -routing requires accurate estimates of link reliabilities to select routes, so a link quality metric supporting this requirement is necessary (Section 2.3.2).
- Routes should be both feasible and optimal regarding the specified reliability target. A routing metric that accommodates both requirements is needed (Section 2.3.3).
- As an exhaustive search for feasible and optimal paths is computationally unfeasible, a strategy for careful preselection of viable paths is necessary and introduced (Section 2.3.4).
- For an expressive performance evaluation of the devised route discovery strategy, we will consider several additional routing metrics for wireless ad-hoc networks from the literature for comparison (Section 2.3.6).

2.3.2. Link Quality Metric

r_{min} -routing requires accurate estimates of link reliability to select routes. How these estimates are obtained exactly, is not part of the specification of r_{min} -

2. r_{min} -Routing

routing. r_{min} -routing however imposes several requirements upon the link quality metric.

- *Normalized* The value range of the metric is $\mathbb{R}_{0,1}$ and perfect links have the value 1.0. This is the simplest of the requirements to fulfill and is needed for network model compatibility.
- *Aggregatability* The product of the reliability of links of a route should be indicative of the end-to-end reliability of the route¹. This requirement is also needed for compatibility with the network model introduced in Section 2.2.
- *Actuality* The metric value of a link is predictive of the outcome of the next transmission attempt. As routes are searched for future operation, ideally, the metric value should aim to predict future link behavior.
- *Conservativeness* The actual probability of success for the next communication attempt is preferably higher than the value of the metric. This requirement is detrimental to the goal of optimal routes with minimal bandwidth consumption. If the link metric was perfectly accurate, and the assumption of transmission attempts behaving like Bernoulli trials made in the network model was realistic, this requirement could be dropped. As achieving both is unlikely, slightly higher bandwidth consumption is preferred over higher fail rate of established routes.

For the remainder of this chapter, we use the packet delivery ratio (PDR) of the recent past as link quality metric. PDR satisfies the requirements *normalized*, *aggregatability* and *actuality*. A more sophisticated solution is introduced in one of the following chapters in this thesis as part of a complete protocol implementing r_{min} -routing.

2.3.3. SMTX – A New Routing Metric

In this section, we introduce the novel routing metric SMTX (smallest maximum number of transmissions), and derive an algorithm to compute SMTX from a path p and reliability target r_{min} . SMTX captures, for a given path, the smallest number of communication attempts that is needed to statistically guarantee a specified end-to-end reliability target. When applied to each of several candidate paths, selecting the path with lowest SMTX value satisfies both requirements stated in Section 2.3.1: Feasibility and optimality. Feasibility is achieved

¹This requirement could be dropped, if there was a protocol that accurately and proactively estimated the end-to-end reliability of routes directly (i.e. without deriving end-to-end reliability from estimates of link reliabilities). We are not aware of such a protocol.

because SMTX defines the necessary communication budget (and its optimal distribution) to achieve the reliability target, while optimality is achieved by selecting the path with lowest budget requirement.

The formal definition of SMTX for a path p and a reliability target r_{min} is concise and straightforward:

$$SMTX_{p,r_{min}} = \min\{n_p \in \mathbb{N} \mid r_{p,n_p} \geq r_{min}\} \quad (2.7)$$

As introduced in Section 2.2, r_{p,n_p} is the product $\prod_{e \in p} r_{e,n_e}$ of the reliabilities of each edge e in p , including a limit of communication attempts of n_e . $SMTX_{p,r_{min}}$ is defined as the minimal n_p (which is the sum over all n_e of edges in the path) sufficient to achieve r_{min} .

In its current form, $SMTX_{p,r_{min}}$ has no upper bound. As this can introduce difficulties with regards to designing an algorithm to calculate the metric value, an additional, bounded definition is provided:

$$SMTX_{p,r_{min}}^{n_{max}} = \min\{n_{max} \cdot |p|, n_p \in \mathbb{N} \mid r_{p,n_p} \geq r_{min}\} \quad (2.8)$$

This definition introduces an upper limit n_{max} , which is used as the metric value whenever $n_p > n_{max} \cdot |p|$. This also means that a result of $SMTX_{p,r_{min}}^{n_{max}} = n_{max} \cdot |p|$ implies that p is not feasible. The previous notation from Eq. (2.7) is equivalent to $SMTX_{p,r_{min}}^{\infty}$, such that we will refer to Eq. (2.8) only, from here on.

For the purpose of deriving an algorithm that calculates $SMTX_{p,r_{min}}^{n_{max}}$, we can think of n_p as a budget, which needs to be distributed optimally over all edges in p . Initially, the budget can be set to the length $|p|$ of p , as at least one communication attempt is required even for edges with perfect probability of success. The algorithm would then need to increase the budget until the reliability target r_{min} is met for the complete path p , or the maximum value $n_{max} \cdot |p|$ for the communication budget is reached. Whenever the budget is incremented, one of the edges is assigned one additional communication attempt. An optimal number of attempts is achieved, if we always allocate another attempt to the edge where this additional attempt results in the strongest proportional increase of success probability. The proportional increase $q(k, r_e)$ in success probability r_e , when allocated another attempt, for an edge e that currently has k attempts allocated can be expressed as:

$$q(k, r_e) = \frac{r_{e,k+1}}{r_{e,k}} \quad (2.9)$$

We show now, that $q(k, r_e)$ is decreasing in k , for all $r_e \in (0, 1)$. The restriction of r_e to the interval $(0, 1)$ is sensible, as a link with $r_e = 1$, i.e. a link of perfect

2. r_{\min} -Routing

quality won't have to be boosted and a non-existing link (i.e. $r_e = 0$) cannot be boosted. Monotonicity is an important property as this implies that $q(k, r_e) < q(k+1, r_e)$, such that a link budget of size b may be selected only after all lower budget levels are already tried.

Lemma 2.3.1. $q(k, r_e)$ is a decreasing function of k for $r_e \in (0, 1)$.

Proof. We show that the derivative of $q(k, r_e)$ is less than 0.

$$\begin{aligned} \frac{\partial q(k, r_e)}{\partial k} &= \frac{\partial}{\partial k} \left(\frac{1 - (1 - r_{e,1})^{k+1}}{1 - (1 - r_{e,1})^k} \right) \\ &= \frac{r_{e,1}(1 - r_{e,1})^k \log(1 - r_{e,1})}{(1 - (1 - r_{e,1})^k)^2} \end{aligned}$$

As $r_{e,1} \in (0, 1)$, and $\log(x) < 0$ for $x \in (0, 1)$, we know $\frac{\partial}{\partial k}(q(k, r_e)) < 0$. \square

Based on $r_{e,n}$, the probability of success $r_{e,n+1}$ over edge e when allocated another communication attempt can be expressed as:

$$r_{e,n+1} = 1 - (1 - r_{e,n}) \cdot (1 - r_{e,1}) \quad (2.10)$$

As the increase in probability of success of an edge under allocation of one additional communication attempt $r_{e,n+1} - r_{e,n} = r_{e,1} \cdot (1 - r_{e,n})$ is always greater than 0, we know that $r_{e,n}$ is strictly increasing under allocation of additional communication attempts for edges e , where $0 < r_e < 1$. An algorithm that iteratively allocates each transmission to the link e , where $q(k, r_e)$ is maximal, therefore results in an optimal allocation sequence, as this approach maximizes r_{p,n_p} . The last term of below equation is maximal for maximal $q(k, r_e)$, so r_{p,n_p} is, too.

$$\begin{aligned} r_{p,n_p} &\stackrel{(2.6)}{=} \prod_{e \in p} r_{e,n_e} \\ &= \prod_{e \in p} \left(r_{e,1} \cdot \frac{r_{e,2}}{r_{e,1}} \cdot \frac{r_{e,3}}{r_{e,2}} \cdots \frac{r_{e,n_e}}{r_{e,n_e-1}} \right) \\ &\stackrel{(2.9)}{=} \prod_{e \in p} \left(r_{e,1} \cdot \prod_{k=1}^{n_e-1} q(k, r_e) \right) \\ &= \underbrace{\left(\prod_{e \in p} r_{e,1} \right)}_{r_p} \cdot \underbrace{\prod_{e \in p} \prod_{k=1}^{n_e-1} q(k, r_e)}_{\text{maximize}} \end{aligned}$$

```

1 // p is a sequence of edges, while an edge is a struct
2 // with attributes r and tx, where r is the un-boosted
3 // reliability and tx = 1 initially
4
5 // Initialize transmission counters
6 for e in p { e.tx = 1 }
7
8 while r(p) < r_min and
9     cost(p) < n_max * p.length {
10    let e_max = p[0]
11    for e in p {
12        if q(e) > q(e_max) {
13            e_max = e
14        }
15    }
16
17    e_max.tx += 1
18 }

```

Listing 2.1: Pseudocode for calculation of $SMTX_{p,r_{min}}^{n_{max}}$

```

19 func r(p) {
20     // Calculate reliability of p, incl. boosts
21     let rel = 1.0
22     for e in p {
23         // pow(a, b) := ab
24         rel *= 1 - pow(1 - e.r, e.tx)
25     }
26     return rel
27 }
28
29 func cost(p) {
30     // Calculate sum over all e.tx in p
31     let sum = 0
32     for e in p { sum += e.tx }
33     return sum
34 }
35
36 func q(e) {
37     // Calculate proportional increase if e is allocated
38     // one additional transmission
39     let r_next = 1 - pow(1 - e.r, e.tx + 1)
40     let r_now = 1 - pow(1 - e.r, e.tx)
41     return r_next / r_now
42 }

```

Listing 2.2: Helpers r(p), cost(p) and q(e)

2. r_{min} -Routing

An algorithm in the form of pseudocode that finds an optimal allocation sequence for a given path is provided in Listing 2.1. To determine $SMTX_{p,r_{min}}^{n_{max}}$, we start with $e.tx = 1$ for all edges. The **while**-loop (line 8) terminates either when the reliability target (captured as r_{min}) is met, or when the sum over all transmissions assigned to edges exceeds the budget determined by n_{max} . In case neither r_{min} is met, nor the budget $n_{max} \cdot |p|$ is exhausted, a search for the edge with maximum proportional increase in $q(k, r_e)$ is executed in lines (11-15). Finally, the optimal edge is assigned the additional transmission and the conditions in line 8 are checked again. In case p is unfeasible, meaning the available budget $n_{max} \cdot |p|$ is not sufficient to boost p such that r_{min} can be met, the second part of the condition in line 8 terminates the procedure. After the procedure terminates, the optimal distribution of communication attempts is recorded in form of the tx attribute values of the edges. To tell whether the procedure terminated because p was unfeasible or not, one additional check whether $r(p) > r_{min}$ evaluates to true is necessary.

Properties of SMTX

In this section we will show that SMTX is monotonic and non-additive. We also show that the respective optimization problem of finding a path with minimum SMTX costs cannot be solved using a greedy algorithm, as SMTX does not satisfy the greedy choice property.

First, we show that $SMTX_{p,r_{min}}^{n_{max}}$ is a monotonic function of paths. I.e. for two given paths p_1, p_2 with p_1 the shorter path in case both paths are of different length, if p_1 only contains links of equal or better (i.e. higher) reliability under element-wise comparison than p_2 , the SMTX value of p_1 will be better (i.e. lower) than that of path p_2 . In other words, under the stated circumstances, SMTX is monotonic in the sense of that it preserves the ordering of p_1 and p_2 .

Lemma 2.3.2. $SMTX_{p,r_{min}}^{n_{max}}$ is a monotonic function of paths, i.e. for two paths p_1, p_2

$$p_1 \leq p_2 \implies SMTX_{p_1,r_{min}}^{n_{max}} \leq SMTX_{p_2,r_{min}}^{n_{max}}$$

holds. Let $|p_1| \leq |p_2|$ without loss of generality. We define \leq on the set of paths $P \subseteq E^*$ as a reflexive partial order relation

$$p_1 \leq p_2 \iff \forall i. 0 \leq i < |p_1| \implies r_{p_1[i]} \geq r_{p_2[i]}$$

Proof. Assume the contrary that $SMTX_{p,r_{min}}^{n_{max}}$ is a non-monotonic function of paths.

Let p_1, p_2 paths and if $SMTX_{p,r_{min}}^{n_{max}}$ is indeed non-monotonic, then

$$\begin{aligned}
& \exists p_1, p_2 \cdot p_1 \leq p_2 \wedge SMTX_{p_1, r_{min}}^{n_{max}} > SMTX_{p_2, r_{min}}^{n_{max}} \\
& \xrightarrow{2.8} n_{p_1} > n_{p_2} \\
& \xrightarrow{2.5} \sum_{e \in p_1} n_e > \sum_{e \in p_2} n_e \\
& \xrightarrow{2.4} \exists i. 0 \leq i < |p_1| \wedge r_{p_1[i]} < r_{p_2[i]} \not\leq p_1 \leq p_2
\end{aligned}$$

This contradicts our assumption and thus, SMTX is monotonic. \square

Above definition of \leq on the set of P implies a partial order only and no total order, as \leq is not defined for all pairs of paths.

Next we will show that SMTX is a non-additive function of paths. We define addition with respect to paths intuitively as the concatenation of the respective edge sequences. Two paths can only be concatenated, if the last vertex of the first path and the first vertex of the second path are the same vertex.

Lemma 2.3.3. $SMTX_{p,r_{min}}^{n_{max}}$ is a non-additive function of paths.

Proof. Assume the contrary that $SMTX_{p,r_{min}}^{n_{max}}$ is an additive function. Let $p_1 = \langle v_1, v_2 \rangle$ with $r_{(v_1, v_2)} = 0.9$, $p_2 = \langle v_2, v_3 \rangle$ with $r_{(v_2, v_3)} = 0.9$ and $p_3 = p_1 + p_2 = \langle v_1, v_2, v_3 \rangle$. If SMTX is indeed additive, then

$$SMTX_{p_1, 0.9}^{\infty} + SMTX_{p_2, 0.9}^{\infty} = SMTX_{p_3, 0.9}^{\infty}$$

should hold. However, $SMTX_{p_1, 0.9}^{\infty} = 1$ and $SMTX_{p_2, 0.9}^{\infty} = 1$, so $SMTX_{p_1, 0.9}^{\infty} + SMTX_{p_2, 0.9}^{\infty} = 2$ but $SMTX_{p_3, 0.9}^{\infty} = 4$. This contradicts our assumption and thus, SMTX is non-additive. \square

Furthermore, the corresponding optimization problem of finding a path of minimal SMTX cost does not satisfy the greedy choice property, which is together with the optimal substructure property a requirement for the applicability of greedy algorithms.

Lemma 2.3.4. The minimum SMTX cost problem does not satisfy the greedy choice property, i.e. there are cases, where a globally optimal choice cannot be obtained by greedily selecting the locally optimal choice.

Proof. We assume the contrary that SMTX satisfies the greedy choice property and construct a counter example. In Fig. 2.2, the path of minimum SMTX cost

2. r_{min} -Routing

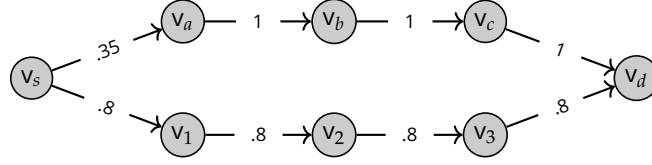


Figure 2.2: Greedy choice property counter example

from vertices v_s to v_d , for $r_{min} = 0.8$ is $p_1 = \langle v_s, v_a, v_b, v_c, v_d \rangle$, as $SMTX_{p_1, 0.8}^\infty = 7$, while $SMTX_{p_2, 0.8}^\infty = 8$ with $p_2 = \langle v_s, v_1, v_2, v_3, v_d \rangle$.

However, when making the greedy choice of the local optimum, starting at v_s , edge (v_s, v_1) is picked, which is not part of the global optimum. This is a contradiction to the assumption that by selecting the locally optimal choice we obtain a globally optimal solution. \square

Theorem 2.3.5. *The minimum SMTX cost problem cannot be optimally solved by a greedy algorithm.*

Proof. An optimization problem can be optimally solved by a greedy algorithm if two properties hold: the optimal substructure property and the greedy choice property. Thus, the proof is immediate by Lemma 2.3.4. \square

The fact that the minimum SMTX cost problem cannot be optimally solved using greedy algorithms is very important, as this also rules out common shortest-path algorithms such as Dijkstra's algorithm.

Time Complexity of SMTX

To determine the runtime complexity of calculating $SMTX_{p, r_{min}}^{n_{max}}$ we analyze the algorithm provided in Listing 2.1. The outer loop (line 8) is bounded by $|p|$, as exactly one additional communication attempt is assigned per iteration and the loop is terminated after $n_{max} \cdot |p|$ assignments in the worst case. To find the edge with maximum impact (lines 11-15), $|p|$ comparisons are required, while each comparison takes constant time. It is however possible to improve the runtime complexity of these comparisons by storing the results of $q(e)$ as a max-heap: After the initial run, only a single value (the value for the edge assigned the additional communication attempt) needs to be updated per iteration. With a max-heap, retrieval of the optimal edge to boost is possible in constant time. Using this optimization, the runtime complexity of lines 11-15 is reduced from $O(|p|)$ to $O(\log p)$. This results in an average-case time complexity of

$$O(|p| \cdot \log |p|) \quad (2.11)$$

for calculating SMTX for a single path p . In practice however, SMTX is used to search for the best among several paths. As the optimization problem for finding a path with minimal SMTX value does not have the greedy choice property (see Lemma 2.3.4), a greedy algorithm (such as Dijkstra’s shortest path) cannot be applied. An alternative would be exhaustive search, however the time complexity of an exhaustive search of all paths between a pair of vertices in the network, of which there are $(|V| - 2)!$ in a complete graph, is $O(|V|^2 \cdot (|V| - 2)!)$. The assumption of a complete graph surely is a worst-case scenario, albeit realistic for small-area networks, where all nodes are in communication range of each other.

From these considerations we can conclude that an exhaustive search is computationally infeasible in some and algorithmically inelegant in most cases, as the vast majority of evaluated paths will resemble significant detours. Hence, an alternative to exhaustive search is introduced in the following section.

2.3.4. Route Selection Heuristic

As stated in Section 2.3.3, an exhaustive search for feasible and optimal paths is computationally unfeasible, as the number of paths between two vertices in a graph grows exponentially with the number of edges. This results in a vast search space even for smaller networks. To keep algorithm runtime within manageable bounds, we apply the following strategy: First, a set of candidate paths is generated using an efficient k -shortest-paths algorithm supporting additive edge weights. This set of candidates is then filtered using several criteria, where candidates with unwanted properties are discarded, e.g. paths with an excessive length, or paths containing edges that have a low probability of successful communication attempts. It is important that each of the filter criteria is not overly strict, so only sub-optimal paths are filtered and that each criterion can be computed efficiently. After the filtering steps, SMTX is applied to the remainder of candidates, to assure feasibility and optimality (see Section 2.3.1) of the final path selection. For each step, a formal definition will be provided.

In the first step, Yen’s algorithm [Yen71] is used to determine the set of k -shortest paths, with $k \in \mathbb{N}$ for given source and destination vertices v_s and v_d of V . As Yen’s algorithm supports additive edge weights only, we cannot use SMTX, which we have shown to be non-additive. Instead, we use the inverse of each edge’s success probability $\frac{1}{r_e}$ as weight. This is a sensible choice, as we prefer routes to be reliable as well as short. As a consequence, the set of paths produced by Yen’s algorithm contains the k paths where

$$\sum_{e \in p} 1/r_e \tag{2.12}$$

2. r_{min} -Routing

is minimal. The time complexity of Yen's algorithm is $O(k \cdot |V| \cdot (|E| + |V| \cdot \log |V|))$ [BELR07].

Step 1. Given source and destination vertices v_s and v_d of V , and $k \in \mathbb{N}$, we first compute a set of route candidates $P_{C_1}(v_s, v_d) \subset P$ as follows:

$$P_{C_1}(v_s, v_d) = P_{min} \subset P, \text{ where } |P_{min}| = k \\ \wedge \forall p \in P_{min} \cdot \forall p' \in P \setminus P_{min} \cdot w(p) \leq w(p')$$

$$\text{with } w(p) =_{df} \sum_{e \in p} 1/r_e$$

These k -shortest paths with respect to w are calculated using Yen's algorithm [Yen71] and are therefore cycle-free.

The value for k needs to be chosen carefully: If it is too small, the set of candidates might not contain any feasible paths that satisfy r_{min} although such paths might exist in G . On the other hand, the larger k is chosen, the more paths with sub-par reliability are included in the set, significantly increasing algorithm runtime without contribution to the quality of the path selected in the final step.

To reduce the size of set P_{C_1} , we will filter and discard paths in P_{C_1} that do not satisfy certain criteria. The time complexity of all path properties checked during the filter steps should be low.

In Step 2, we discard all paths that exceed a length limit. We choose the limit relative to the length n_{sp} of the shortest path between v_s and v_d with respect to hop count. As the hop count metric gives preference to longer and therefore less reliable hops, we not only consider paths p of length n_{sp} , but also longer routes up to a maximum length of $\lceil f_{sp} \cdot n_{sp} \rceil$, with a suitable factor $f_{sp} \geq 1$.

Step 2. Given a set of paths P_{C_1} , remove paths that exceed a length limit:

2a. Determine the length n_{sp} of the shortest path (w.r.t. hop count) between v_s and v_d in P .

2b. Compute

$$P_{C_2}(v_s, v_d) = \{ p \in P_{C_1}(v_s, v_d) \mid |p| \leq \lceil f_{sp} \cdot n_{sp} \rceil \}$$

We suggest to choose $f_{sp} = 3$, for the following reasons: Fig. 2.3 shows three line topologies with two, three and four vertices, respectively. The Euclidean distance between v_s and v_d is the same in each topology. What varies is the number of vertices placed between v_s and v_d . Using a simulation model, we have determined the success probability of each edge based on its length, i.e. the Euclidean

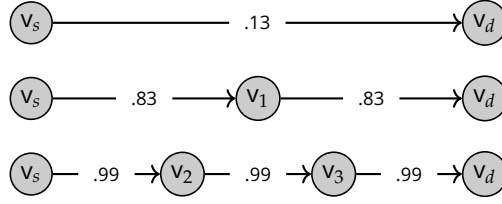


Figure 2.3: Three paths between two vertices. Edges with higher success probabilities result in more efficient paths, despite higher number of hops.

Table 2.3: Properties of paths in Fig. 2.3, boosted for $r_{min} = 0.8$

Path p	Edge e	$r_{e,1}$	n_e	r_{e,n_e}	n_p	r_{p,n_p}
$\langle v_s, v_d \rangle$	(v_s, v_d)	0.13	12	.812	12	.812
$\langle v_s, v_1, v_d \rangle$	(v_s, v_1)	0.83	1	.830	3	.806
	(v_1, v_d)	0.83	2	.971	3	.806
$\langle v_s, v_2, v_3, v_d \rangle$	(v_s, v_2)	0.99	1	.990	3	.970
	(v_2, v_3)	0.99	1	.990	3	.970
	(v_3, v_d)	0.99	1	.990	3	.970

distance between the edge's respective source and destination vertices. The path $p_1 = \langle v_s, v_d \rangle$ using the directed edge between v_s and v_d has very poor reliability of 0.13, so $r_{p_1} = 0.13$. The 2-hop path $p_2 = \langle v_s, v_1, v_d \rangle$ already provides an increased path reliability r_{p_2} of 0.69. With the 3-hop path $p_3 = \langle v_s, v_2, v_3, v_d \rangle$, we even get $r_{p_3} = 0.97$. The Euclidean distance covered is the same in all three scenarios, what varies is only the number of hops (i.e. edges) used to bridge this distance. Now, f_{sp} should be chosen such that p_3 remains in the candidate set, as p_3 is the most efficient path (see Table 2.3). By choosing $f_{sp} \geq 3$, the candidate set for paths between v_s and v_d would include p_3 , as $n_{sp} = 1$ and $|p_3| = 3 \leq \lceil f_{sp} \cdot n_{sp} \rceil$. Including paths longer than p_3 (e.g. with $f_{sp} = 4$) would not further improve the situation, as the reliability of each individual link of p_3 is close to 1 already. Based on these observations, we conclude that paths with a length of up to $3 \cdot n_{sp}$ should be kept as route candidates in this early stage of route selection.

Next, in Step 3, all paths that contain edges with low success probability are discarded, as the corresponding links for these edges are likely to become bottlenecks during route operation and usually require significant boosting. As we consider heavy boosting of links an emergency measure that also comes with a cost, we aim to keep the requirement for boosts in candidate paths at a low level.

2. r_{min} -Routing

Step 3. Given a set of candidate paths P_{C_2} , we keep only paths with sufficiently strong edges. In other words, we filter out all paths that contain at least one weak edge. Formally:

$$P_{C_3}(v_s, v_d) = \{ p \in P_{C_2}(v_s, v_d) \mid \forall e \in p. 1/r_e \leq n_{maxAvg} \}$$

Here, n_{maxAvg} denotes the maximal average number of communication attempts required for successful reception. For instance, by setting $n_{maxAvg} = 2.0$, paths with at least one edge e of success probability $r_e < 0.5$ are filtered out.

So after Step 3, all overly long paths and paths containing very weak edges are discarded from the candidate set. In an implementation of this route selection heuristic, the filtering steps 2 and 3 may be merged and executed efficiently using a single iteration over the set of candidate paths. In the next step, the remaining paths are filtered according to their efficiency at supporting the specified reliability requirement r_{min} .

Step 4. Given a set of candidate paths $P_{C_3}(v_s, v_d)$ and a reliability target r_{min} , we keep only paths that are sufficiently bandwidth-efficient, i.e. paths with a small value for $SMTX_{p,r_{min}}^\infty$. Formally:

$$P_{C_4}(v_s, v_d) = \{ p \in P_{C_3}(v_s, v_d) \mid SMTX_{p,r_{min}}^{n_{max}} \leq f_{smtx} \cdot n_{smtx,min} \wedge n_{max} \cdot |p| > f_{smtx} \cdot n_{smtx,min} \}$$

where

$$n_{smtx,min} = \min \{ SMTX_{p,r_{min}}^{n_{max}} \mid p \in P_{C_3}(v_s, v_d) \wedge SMTX_{p,r_{min}}^{n_{max}} < n_{max} \cdot |p| \}$$

$$n_{max} = f_b$$

The intention of Step 4 is the following: After this step, the set of candidates should contain only paths that are capable to support the required reliability target r_{min} , with sufficient efficiency. This purposefully includes paths less efficient than the optimal path, such that further filter criteria may be applied in subsequent steps. This enables us to support additional quality requirements, captured by additional metrics. Whether the efficiency of a path is sufficient is controlled by the parameter f_{smtx} . A path is excluded from the set of candidates, if the number of necessary communication attempts exceeds that of the most efficient path in the set by more than the factor f_{smtx} . To only keep the most efficient paths in the set of candidates, $f_{smtx} = 1.0$ can be set. We illustrate this filtering step by means of a small example: Consider the following

sequence $S = [7, 8, 8, 9, 10, 13, 17]$ of SMTX metric values for paths in the candidate set $P_{C_3}(v_s, v_d)$ connecting a pair of vertices v_s, v_d . For f_{smtx} we choose the value of 1.2. For the given S , the value of $n_{smtx, min}$ is 7, so $f_{smtx} \cdot n_{smtx, min} = 8.4$. Consequently, all paths where SMTX exceeds 8.4 are discarded and the filtered sequence S' is $[7, 8, 8]$.

Furthermore, we limit the average budget of communication attempts per edge to f_b , e.g. $f_b = 4$. This acts as a bound on the metric value not least to provide an upper limit on runtime complexity for calculation of SMTX. Care has to be taken that the product $f_b \cdot |p|$ is actually greater than $f_{smtx} \cdot n_{smtx, min}$, otherwise unfeasible or sub-optimal paths may be included in $P_{C_4}(v_s, v_d)$.

Route selection stops if $P_{C_4}(v_s, v_d)$ is empty or contains only one element, which then becomes the selected route. Otherwise, we continue with Step 5. As values returned by SMTX are integer, there may remain several, equally well-suited candidate paths (with respect to SMTX) after Step 4. Depending on the value of f_{smtx} there might be even more sufficiently efficient candidate paths. Hence, to make a final selection, an additional step applying a metric which provides more granularity is necessary:

Step 5. Depending on the routing objective, we now sort the paths in P_{C_4} and select the path ranked top. Any routing metric can be applied here, as all remaining paths satisfy the reliability requirement.

Selection Heuristic Example

To illustrate the process of filtering the set of candidate paths for route selection, we work with the example 11-node topology depicted in Fig. 2.4. The objective will be to find a route between nodes 9 and 10, with $r_{min} = 0.9$.

In Step 1, the k shortest paths with respect to weight function $\sum_{e \in p} 1/r_e$, are calculated. For this example, we assume $k = 300$. Due to the small size of the topology, there are only 239 possible paths between nodes 9 and 10, so after Step 1, the set of candidates contains all 239 possible paths.

During Step 2, all paths that exceed the length of the shortest path by a factor of 3 or more are discarded from the initial set. In the given example, the shortest path p_{sp} (with respect to hop count) is $\langle 9, 2, 6, 10 \rangle$, so $|p_{sp}| = 3$. Consequently, all paths with length > 9 are discarded. In this topology, from the initial 239, only 1 path is affected and therefore removed, leaving a remainder of 238 paths in the set.

In Step 3, all paths that contain at least one edge e , with $r_e \leq 0.5$ (which corresponds to $n_{maxAvg} = 2.0$) are filtered out. Hence, we avoid links where failure of a single transmission attempt is more likely than its success. This criterion affects 173 paths, such that after Step 3, 65 paths remain in the set of candidates.

2. r_{min} -Routing

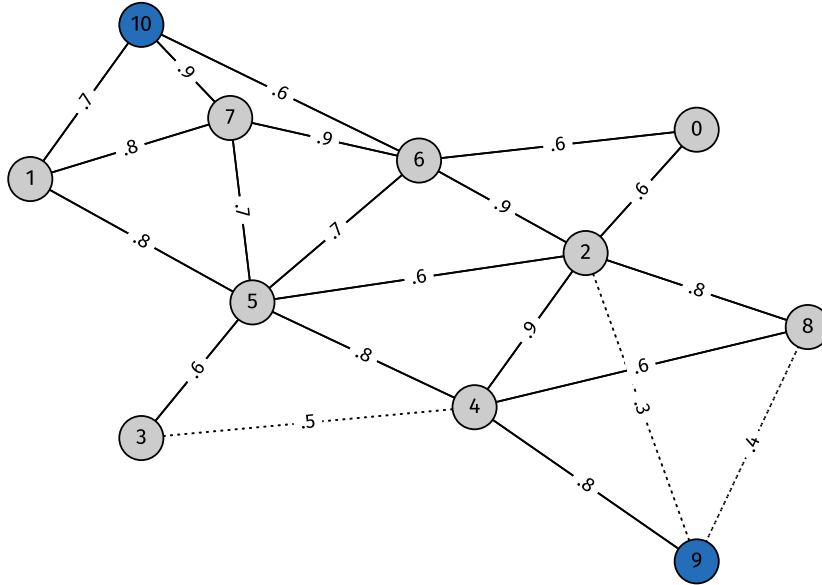


Figure 2.4: Example topology with 11 nodes for illustration of the route selection heuristic. Dotted lines indicate a weak link ($r \leq 0.5$).

We now apply SMTX to the 65 paths left and filter inefficient paths with respect to meeting the reliability target as Step 4. The most efficient path p_{ef} is $\langle 9, 4, 2, 6, 7, 10 \rangle$ with $SMTX_{p_{ef}, 0.9}^{\infty} = 10$. We choose $f_{smtx} = 1.3$, so all paths p , where $SMTX_{p, 0.9}^{\infty} > 13$ are removed. For the example topology depicted in Fig. 2.4 this leads to removal of another 59 paths, leaving 6 paths.

In Step 5, the final selection among any remaining path candidates is made. Which metric is used in this step is left open intentionally, such that further quality criteria can be applied, without endangering the specified route reliability target.

2.3.5. Route Operation

The route selected according to the heuristic described in Section 2.3.4 is now ready for operation. The selected route (i.e. the sequence of network addresses) as well as the corresponding vector of transmission limits is embedded in every packet sent along the established route. Upon reception, each intermediate node along the route extracts the correct transmission limit from the vector and forwards the packet via unicast, while enforcing that limit. With each transmission attempt, an acknowledgement (ACK) is requested. If the ACK is not received, which is detected using a timeout, the transmission is retried until all attempts are exhausted. In case the final attempt also fails, the frame is lost (usually along with the corresponding packet). Any further attempts of transferring the

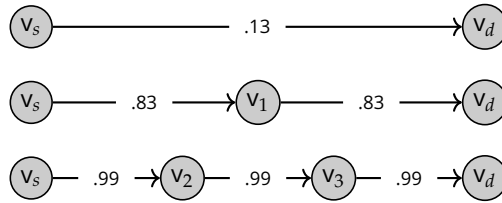


Figure 2.3: Small example topology (repeated from page 21)

complete packet to the destination node of the route need to be triggered by the application layer.

It is important to note that this simple approach of route operation described above is functionally incomplete regarding various aspects: Monitoring of the achieved end-to-end reliability and repair of breaking routes are not handled, while adequate handling of both is vital in environments with dynamic link qualities. Also, there is no mechanism for shutting down an established route after it is no longer needed. These aspects and more will be addressed in Chapter 4 of this thesis, where a functionally complete implementation of r_{\min} -routing is introduced. In this first description of the approach, these details are left unspecified on purpose, however.

2.3.6. Routing Metrics for Wireless Ad-Hoc Networks

Reliability-constrained Quality of Service routing is an established sub-topic in the wireless routing community. Though to our knowledge there is no solution that claims to support statistical reliability guarantees comparable to those provided by r_{\min} -routing, there are several link and routing metrics that are based on reliability. In this section, we will address ETX (Expected number of Transmissions), a routing metric introduced by de Couto et al. in [DCABM03] as well as the routing metric ETOP (Expected number of Transmissions On a Path), which is itself based on ETX. As another example for a link metric focused on reliability we will address ML (Minimum Loss), together with the classic SP (Shortest Path) metric. All metrics will be evaluated in simulation experiments and compared against SMTX as employed in r_{\min} -routing in a later section of this chapter.

Shortest Path The most prevalent routing metric is Shortest Path (SP) in hops, which minimizes the number of hops between source v_s and destination v_d :

$$SP_p = |p| \quad (2.13)$$

2. r_{min} -Routing

Table 2.4: Properties of paths in Fig. 2.3, with the optimal choice according to each metric underlined.

Path	ETX	ETOP ₄	SMTX _{0.9} [∞]	ML	SP
$\langle v_s, v_d \rangle$	7.69	7.35	17	0.13	<u>1</u>
$\langle v_s, v_1, v_d \rangle$	<u>2.40</u>	<u>2.41</u>	4	0.69	2
$\langle v_s, v_2, v_3, v_d \rangle$	3.03	3.03	<u>3</u>	<u>0.97</u>	3

While this metric has its merits in wired networks, it is not the first choice for wireless networks, as it gives preference to longer and therefore less reliable links. This is also illustrated in the example in Fig. 2.3 together with Table 2.4: with the SP metric, the least reliable path $\langle v_s, v_d \rangle$ is favored, as it covers the most distance per hop. To achieve a more usable reliability level of e.g. 0.9, the single link in the favored path would require an expected number of 17 transmission attempts per frame.

Expected Transmission Count Another routing metric is Expected Transmission Count (ETX). ETX has been published in [DCABM03] and is especially suitable for wireless mesh networks. In the original paper, ETX is defined as the multiplicative inverse of a link's packet delivery ratio (PDR) and therefore captures the expected number of transmissions required to successfully transmit over the link in question. We slightly adapt this definition to fit our network model and use the success probability of an edge r_e instead of the PDR. The formal definition is as follows:

$$ETX_p = \sum_{e \in p} 1/r_e \quad (2.14)$$

As does SMTX, ETX considers link reliabilities. However, different from SMTX, no reliability target is considered. To assess the quality of a route, ETX uses only the average number of transmissions for successful delivery. By choosing a route with the smallest ETX value, average bandwidth consumption is minimized. Another drawback is that ETX is only accurate if there is no upper bound n_{max} for the number of transmissions on a link [JEH⁺12]. For instance, with $n_{max} = 4$, a link e with $r_e = 0.2$ would require an average number of transmissions of $1/r_e = 5$, which would not be feasible. Therefore, ETX yields meaningful results only for high quality links.

Expected number of Transmissions On a Path In [JEH⁺12], ETX is adapted to correctly reflect the effect of an existing upper bound K for the number of transmissions, as well as the position of links in the route. This is based on the observation that a weak link close to the source of a route does not waste as much transmissions upon loss of frames when compared to an equally reliable link closer to the destination, as a failed end-to-end attempt is always retried from the source node. The result of this work is called Expected number of Transmissions On a Path, ETOP in short.

$$ETOP_{p,K} = \sum_{i=0}^{n-2} (E_i \frac{\rho_{i+1}}{\rho_n}) + K \frac{1 - \rho_n}{\rho_n} + E_{n-1} \quad (2.15)$$

Here, K is the maximum number of communication attempts for each edge, including the initial attempt. E_i is the expected number of communication attempts necessary to communicate successfully over edge (v_i, v_{i+1}) and ρ_i is the probability that each of the first i edges is traversed with at most K attempts per edge.

Although ETOP fixes some of the shortcomings of ETX, it also inherits some of its drawbacks. Link reliabilities are considered, however no reliability target can be specified.

Minimum Loss A straightforward routing metric that considers link reliabilities achieved by a single transmission is Minimum Loss (ML) [PTMS⁺06], which we define here for reasons of comparison:

$$ML_p = r_p = \prod_{e \in p} r_e \quad (2.16)$$

For route selection, this simple metric is certainly a candidate. Experiments by the original authors show that in combination with dynamic source routing (DSR) as routing protocol, ML has benefits over ETX. However, different from SMTX, no reliability target is considered.

2.4. Simulation Experiments

For evaluation and analysis of SMTX, extensive simulation experiments were carried out. The set of topologies used during these experiments is random generated with a uniform distribution of nodes. Links are modeled as Bernoulli processes², where the probability of success of a single communication attempt

²Refer to Section 2.2 for some implications this model has.

2. r_{min} -Routing

depends only on the distance between the communicating nodes plus a random propagation loss that is independent of the distance between the nodes. The additional random propagation loss applied to each link is necessary to create asymmetric links. During execution of experiments, the simulated reliability of each link is static. Furthermore, no internal or external interference is modeled. Transceivers are assumed to switch back and forth between transmission and reception modes instantaneously, such that these blind periods do not cause any additional frame losses. Frame size is not simulated and consequently has no influence on the success probability of communication attempts.

2.4.1. Experiment Setup

Topology Generation To achieve the necessary level of generality, 200 topologies of varying size and density were generated. Topology sizes range from 10 to 50 nodes, while the network density as the normalized ratio of existing links to possible links ranges from 0.3 to 0.6. For each of these topology classes, 10 variants were generated:

$$\underbrace{|\{10, 20, 30, 40, 50\}|}_{\text{Topology sizes}} \times \underbrace{|\{0.3, 0.4, 0.5, 0.6\}|}_{\text{Network densities}} \cdot \underbrace{10}_{\text{Variants}} = 200 \quad (2.17)$$

The process for generation of a topology consists of several steps: first, random node positions are drawn from a uniform distribution. Then for each pair of nodes, the respective link's reliability is determined by calculating the propagation loss between the node pair (see next paragraph for details). For most experiments, only links with reliability > 0.5 are kept. If after this step the network model graph is strongly connected, i.e. there is a directed path from every vertex to every other vertex, the graph is stored and used in experiments, otherwise it is discarded. To obtain topologies of desired density, the area for node placement is increased whenever enough generated topologies resulted in graphs of duplicate density. Figures showing all 200 generated topologies can be found in Figs. A.1 to A.4.

Propagation Loss The simulated path loss for each link is static and calculated offline using a so-called log-distance propagation model. This model is an extension to the Friis free space model and adds parametrization to account for different degrees of obstruction and shadowing by means of a path-loss distance exponent. The resulting path loss L_e for a link represented by edge e in the model is calculated as follows:

$$L_e = L_0 + 10 \cdot n \log_{10} \frac{d_e}{d_0} \quad (2.18)$$

In our model, the path loss L_0 at reference distance $d_0 = 1$ m is set to 46.677 dB and the path loss distance exponent n is set to 3.0, which resembles a level of shadowing and obstruction similar to what is typically found in a production factory hall. L_e is calculated for distance d_e , which is the Euclidean distance in meters between source and destination nodes of the respective link. This path loss L_e is then subtracted from the initial transmission power P_{tx} , which in our experiments is set to 20 dB.

In addition to distance-based loss, some random loss is applied to simulate asymmetric links. This additional random loss is drawn from a normal distribution $\mathcal{N}(\mu, \sigma)$ with mean $\mu = 0$ dB and standard deviation $\sigma = 16$ dB. If this random loss was also applied directly and offline, the reliability of each link would be either 0 or 1, depending on whether the nodes are close enough to compensate for path as well as random loss. As a network of binary links is not our intention, a link's reliability value is determined by calculating the probability that the random loss drawn from the normal distribution $\mathcal{N}(\mu, \sigma)$ is low enough for the transmission to be decoded by the destination node's transceiver. In other words, we calculate the probability that the remaining signal strength after application of both path loss and random loss is higher than the receiver's reception sensitivity P_{rx} , which is set to -106 dB. This is done by evaluating the cumulative distribution function³ (CDF) of the normal distribution $\Phi_{\mu, \sigma}$ at position $x = P_{tx} - L_e - P_{rx}$. Finally, the reliability of a link captured by the success probability r_e of the respective edge e in the network model is calculated as follows:

$$r_e = \Phi_{\mu, \sigma}(P_{tx} - L_e - P_{rx}) \quad (2.19)$$

Link Properties As mentioned above, the simulated reliabilities of links are static. Hence, the outcome of a transmission attempt is a Bernoulli process, just as defined in Section 2.2, where the network model was introduced. This implies also statistical independence of subsequent transmissions attempts. Hence, we can assume that the expected packet delivery ratio (PDR) over some link is equal to r_e of its corresponding edge and further that the expected end-to-end PDR of a whole route is equal to r_p (or r_{p, n_p} if there is link boosting) of the corresponding path p .

³The CDF of a random variable X evaluated at x is the probability that X will take a value less than or equal to x .

2.4.2. Assessment of r_{min} -Routing

The main objective of r_{min} -routing is to find routes that satisfy the specified reliability target. Hence, this is also where the focus lies in this first sequence of experiments. In the introduction of r_{min} -routing in Section 2.3 this goal was summarized as the requirement for routes to be *feasible*. To evaluate whether a selected route is actually feasible, i.e. the reliability target is met, we request routes from every node to every other node in the network and record whether the specified end-to-end reliability target is met in the simulations. Routes are requested with different r_{min} targets specified, to verify that the requested reliability is also met for ambitious reliability requirements. The desired outcome with respect to feasibility of routes would be that the requested minimum reliability is always met, i.e. there are no routes, where the packet delivery ratio (PDR) falls below the r_{min} target.

Another aspect to be assessed is, whether routes discovered with r_{min} -routing are also efficient, i.e. *optimality*⁴ is achieved by avoiding *over-provisioning*. We consider a route over-provisioned if the route's end-to-end reliability exceeds the specified reliability target significantly. In the introduction to Section 2.3, we explored some of the implications of over-provisioning and why it should be avoided. For achieving optimality, the desired outcome would be that the PDR of a route is as close as possible to the specified r_{min} target, though never below.

Results of this first sequence of experiments are depicted in Figs. 2.5a to 2.5d. The figures show aggregated results from simulation runs over all 200 topologies. Box-plots⁵ are used to visualize the range of end-to-end route reliability over the length of the respective routes. We observe that both *optimality* and *feasibility* are achieved. For very short routes, optimality is harder to achieve, as the size of the assigned transmission budget is always integer. So the constellation, where the reliability of a selected route is significantly higher than what was requested, while assignment of one less transmission would not satisfy the required reliability target anymore, occurs quite frequently. This constellation becomes less frequent with increasing route length, as the additional degrees of freedom for the distribution of the transmission budget allow for more fine-grained control of the resulting end-to-end reliability. Another takeaway from the results is the decrease in over-provisioning for routes of higher end-to-end reliability. This observation is also a consequence of the additional degrees of freedom while distributing the transmission budget. Albeit this time the additional degrees of freedom stem from the higher number of transmissions that need to be assigned to achieve the more ambitious reliability targets. With respect to feasibility the route selection process leads to flawless results: there is

⁴Refer to Section 2.3 for a definition of optimality and over-provisioning.

⁵The box extends from the lower to upper quartile, the colored horizontal line indicates the median. The whiskers cover the whole range of the data.

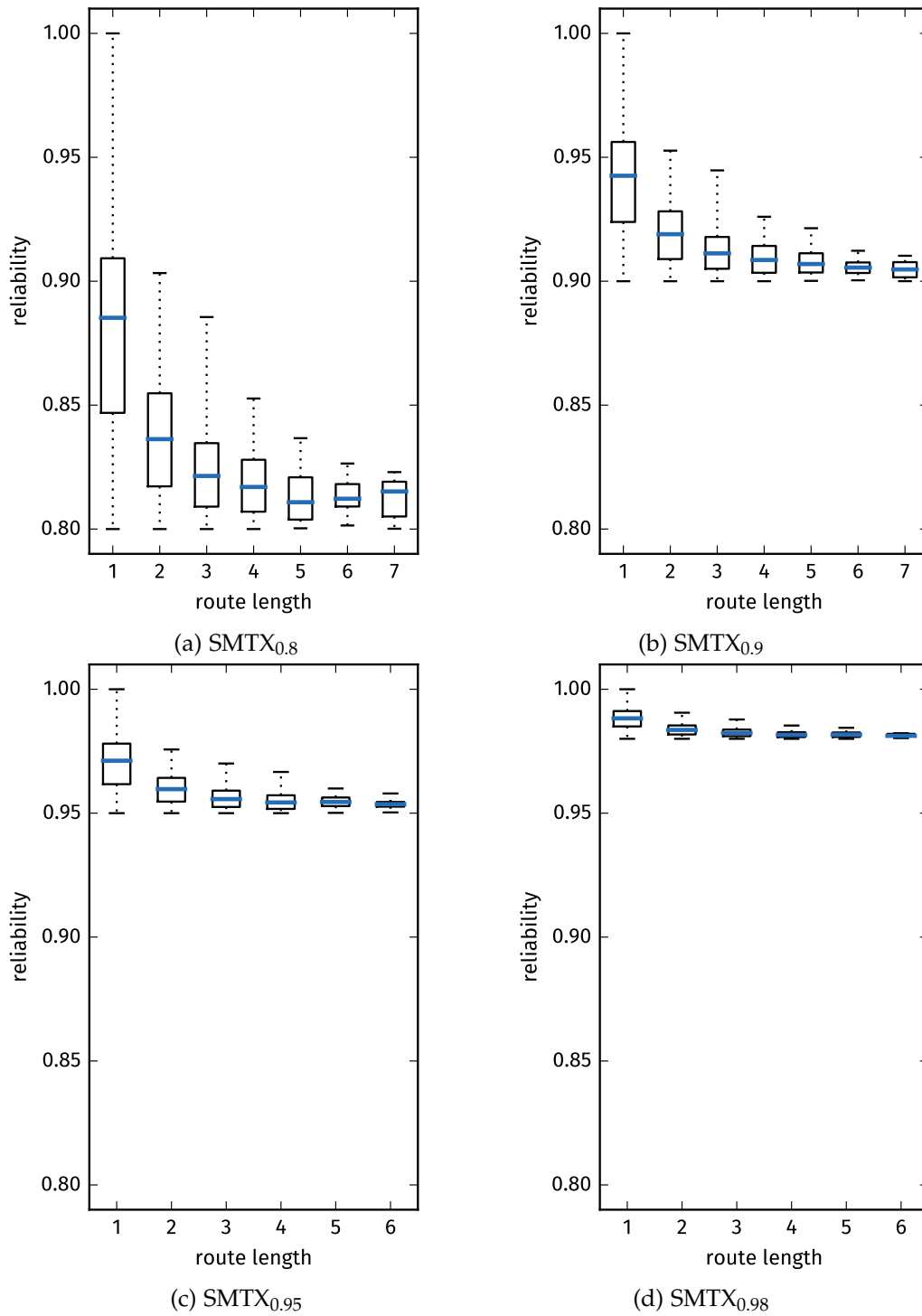


Figure 2.5: Reliability of routes found with SMTX metric, under different configurations of r_{\min} .

2. r_{\min} -Routing

Table 2.5: Parametrization of r_{\min} -routing selection heuristic.

Parameter	Value	Description
k	100	Number of initial path candidates, controls size of search space
f_{sp}	3	Shortest path factor, controls removal of long routes (relative to shortest route)
n_{maxAvg}	2.0	Maximum of expected communication attempts per edge, controls removal of routes with weak links
f_b	4	Budget of communication attempts per edge (for satisfaction of r_{\min} target), controls removal of inefficient routes
f_{smtx}	1.3	Tolerance regarding efficiency of feasible paths, relative to most efficient feasible path

not a single route (note also the lack of downward outliers), where the specified target reliability was not met. It is important to note that this achievement is due to the assumption of static link reliabilities and statistical independence of the success of transmission events. Later in this thesis, these assumptions will be dropped, which requires additional measures.

To conclude this first assessment of r_{\min} -routing, we will examine whether r_{\min} -routing is capable of discovering a sufficiently reliable route when such a route exists, i.e. whether a path that could satisfy the requirement is detected, or if the selection heuristic is too restrictive and discards too many viable candidates. Whether a potentially viable route is detected depends on several factors: First, there is the size of the search space. In Section 2.3.4, the initial candidate set is initialized using a k-shortest-paths algorithm. The parameter k configures the number of paths in the set and consequently the size of the search space. Is k chosen too small, the candidate set might not contain any viable paths. A very large k leads to higher computational overhead. Another aspect to consider is the quality level of links in the network topology that is searched. If links are sufficiently reliable, there are often several feasible routes in the network. As we need to find only one, chances improve that a feasible path is contained in the initial candidate set with increasing link reliabilities in the network. Consequently a feasible route is much harder to find in network topologies where reliability of most links is insufficient to support the end-to-end reliability target. It is also clear that there are networks, where especially high reliability requirements cannot be satisfied at all for some node pairs, while keeping transmission overhead within a pre-defined transmission budget. In this case, even a com-

plete search (with $k = \infty$) would not lead to the discovery of a feasible route, as no such route exists. On the other hand, no routing algorithm will manage to maintain routes between these node pairs with the requested reliability in that case. At last, there is the selection heuristic (as detailed in Section 2.3.4) itself. After a first step, where the set of candidates is initialized, it is filtered rigorously. Paths of excessive length and paths that contain weak edges are discarded. If this filtering is configured too strictly, all feasible path candidates might have been discarded before the final selection step.

In all our experiments the r_{\min} -routing selection heuristic parameters were configured as shown in Table 2.5.

- The first parameter is k , which controls the number of initial path candidates. The configured value of 100 is a compromise between size of search space and computation effort and works well, as experiments show.
- The shortest path factor f_{sp} controls the maximum length of paths to be kept in the candidate set. As detailed in Section 2.3.4, a choice of $f_{sp} = 3$ provides a good balance between route length and link reliability.
- With n_{maxAvg} a maximum for the expected number of transmissions per link can be configured. If the expected number of necessary transmissions for successful reception via a link exceeds n_{maxAvg} , the respective path candidate is discarded. By setting $n_{maxAvg} = 2.0$, all paths with at least one edge e with $r_e < 0.5$ are removed from the set of candidates. This is a sensible choice in general, as this excludes only edges where a single communication attempt is more likely to fail than to succeed. For some networks however, it might be necessary to increase n_{maxAvg} to allow routes with weaker links, e.g. when the overall link reliability is low and route search terminates without a result often. We note that in this series of simulation experiments, filtering with n_{maxAvg} has no effect on the set of candidates, as links with $r_e < 0.5$ are already discarded from generated topologies.
- The end-to-end transmission budget available for distribution in SMTX is controlled by f_b . When comparisons between different routing metrics regarding route reliability are performed, f_b should be set to the global per-link transmission limit (which is the enforced limit for non-SMTX metrics) as the number of allowed transmission attempts has strong influence on the end-to-end reliability of a route. Keeping f_b and the global per-link transmission limit synchronized is of key importance to fairness of comparison. In existing wireless technologies, the maximum number of transmission attempts is commonly also set to a global default value, independent of link reliabilities. For instance, in IEEE 802.11, the default values are between 4 and 7 depending on the size of the frame to be transmitted

2. r_{min} -Routing

and whether the size category qualifies for transmission with RTS/CTS or not [IEEE16]. Hence, we choose $f_b = 4$ for our simulations, too.

- At last, the SMTX tolerance factor f_{smtx} was set to 1.3, which gives a tolerance of 30 % for the efficiency (i.e. number of required attempts to satisfy r_{min}) of feasible path candidates. Candidate paths that would satisfy the r_{min} target, but require >30 % attempts when compared to the most efficient feasible path are therefore removed from the candidate set. It is important to note that the SMTX tolerance factor has no influence on whether a route is found at all. It controls only filtering of excess feasible candidates before a final route decision is made in the last step. For our experiments, we picked ETOP as metric for the final selection step.

With this configuration r_{min} -routing was capable of finding a feasible route for every pair of nodes and every reliability target in all simulation runs. This is remarkable, though it is important to note that the used topologies are purposefully generated in a way that ensures that feasible routes exist (see Section 2.4.1).

2.4.3. Comparison with Other Metrics

The experiments from the previous section show that with r_{min} -routing, routes that satisfy different reliability requirements are discovered consistently across a variety of network configurations. To illustrate novelty of this achievement, we compare the results obtained with r_{min} -routing with other routing metrics. In this section, the routes discovered with r_{min} -routing, which uses the novel routing metric SMTX are compared to routes selected with ETX, ETOP, ML and SP. Formal definitions of all metrics used for this comparison are provided in Section 2.3.6.

In Figs. 2.6a, 2.6b, 2.7a, 2.7b and 2.8 the range of reliabilities of all routes detected by the respective routing metric is visualized in the form of box-plots. Setup of this series of experiments is the same as with the previous sequence and is described in the preceding section. The global per-link transmission limit is set to 4, i.e. after 4 failed transmission attempts a frame is dropped and not re-transmitted further. A transmission limit of 4 per link also corresponds to the configured end-to-end transmission budget used in r_{min} -routing, where f_b was also set to 4. This ensures a fair comparison, as the available transmission budget influences the perceived end-to-end route reliability strongly.

The routes discovered with the SP metric (see Fig. 2.6a) are the shortest among all evaluated metrics, which is expected as SP minimizes the number of hops necessary for each packet to reach the destination node. The end-to-end reliabilities of routes selected with the SP metric are quite high, with most routes achieving a reliability of at least 0.85. In an extra run (depicted in Fig. 2.6b), where only very unreliable links ($r_e < 0.2$) are discarded from the network, the

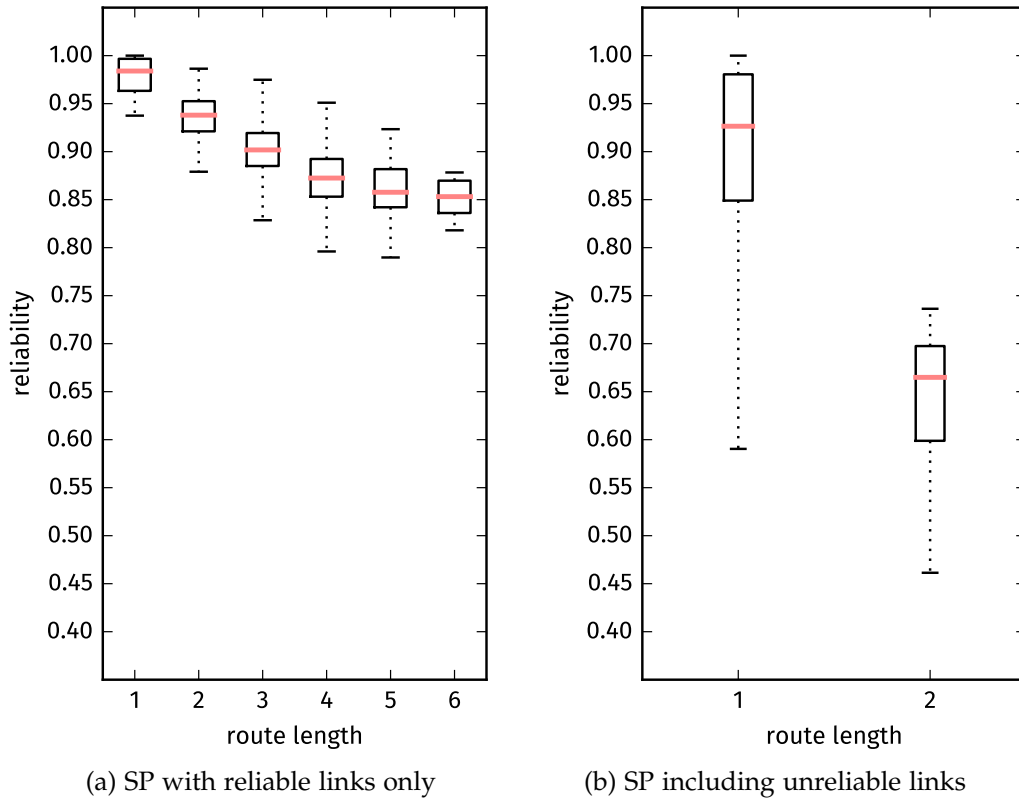


Figure 2.6: Reliability of routes found with SP, 2.6a with reliable links only ($r \geq 0.5$) and 2.6b including unreliable links ($r \geq 0.2$).

2. r_{min} -Routing

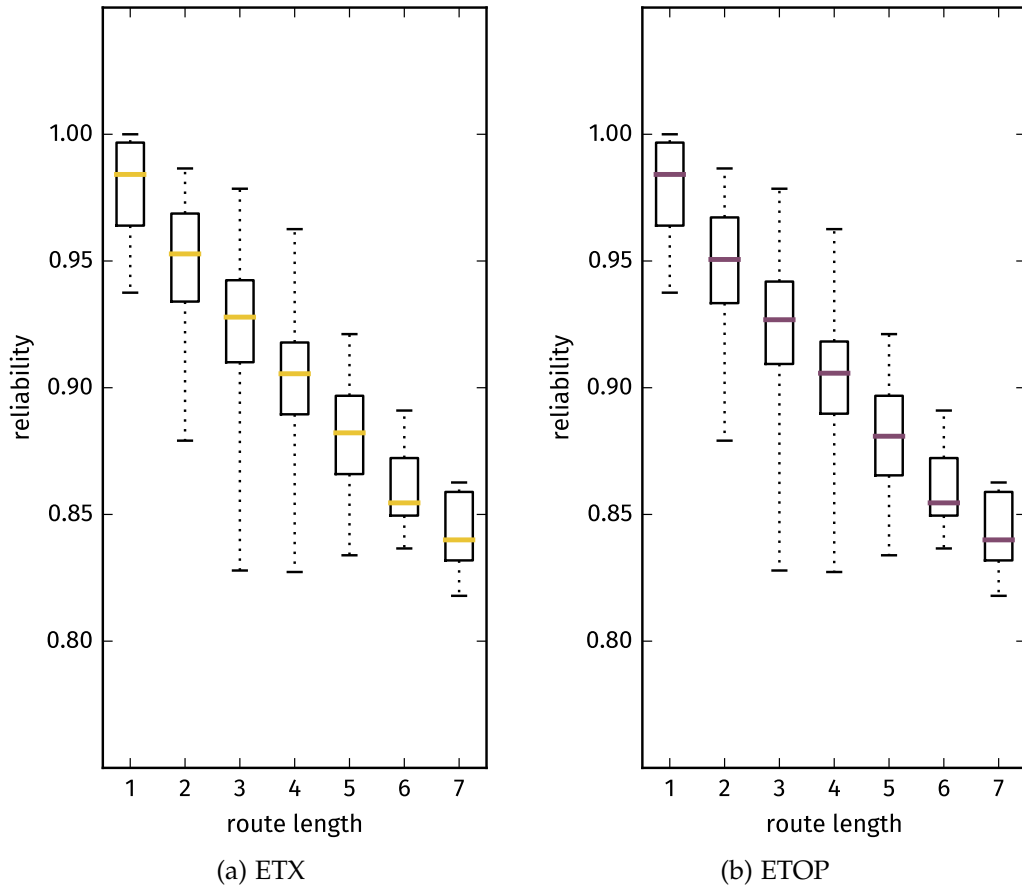


Figure 2.7: Reliability of routes found with ETX and ETOP.

reliability of routes is significantly lower. In addition, length of routes is shorter than in the previous run, as the additional links present in this topology directly connect nodes previously not connected. This striking preference of short and unreliable routes is due to the fact that SP does not consider link quality in any form during route selection. As long and therefore less reliable links cover greater distances per hop in the network, these links are preferred with SP. This result confirms results from many other works, where metrics minimizing hop count prove unfit for networks that contain unreliable links.

For the closely related ETX and ETOP metrics, (Figs. 2.7a and 2.7b) results are very similar and are therefore discussed jointly: short routes exhibit high reliabilities, which decline as the lengths of selected routes increase. This is expected, as the longer the bridged distance, the more likely it is that a weaker link is included in the route. Both ETX and ETOP do not apply any measures to weaken the impact of less reliable links on the end-to-end reliability of the whole route, hence the visible decline in reliability for longer routes. The overall

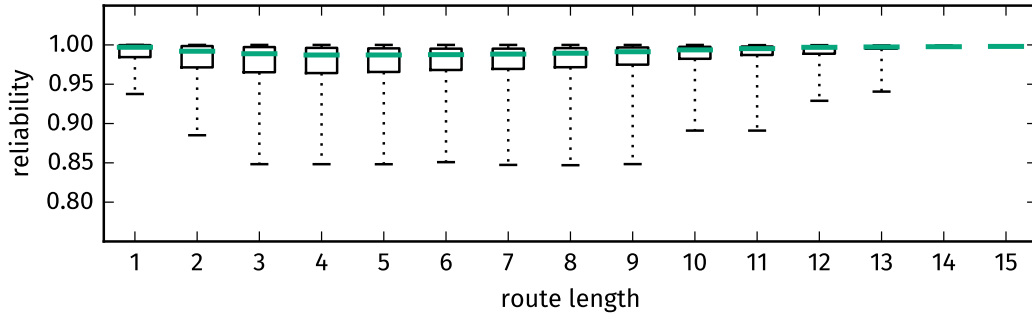


Figure 2.8: Reliability of routes found with ML metric.

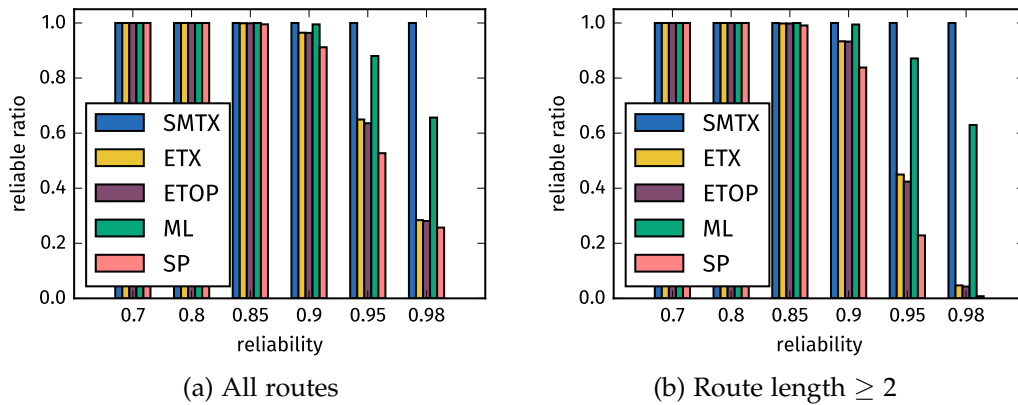


Figure 2.9: Ratio of routes that match the respective reliability target on the x-axis. In Fig. 2.9b, single hop routes are ignored.

length of selected routes is very similar to that observed with SMTX.

With ML, results are different. Median route reliability is very high across all route lengths, though there are some downward outliers. What is unique for the ML metric is a significantly increased length of selected routes. Due to the definition of ML, close to perfectly reliable links cause only a diminishing decrease on the metric's value and are therefore included in routes very often. There is also no punishing effect for excessive route lengths, as long as each link's reliability is close to 1. In practice, network-wide application of ML leads to excessive use of the most reliable links in the center of the network, increasing the risk of congestion.

In Figs. 2.9a and 2.9b, the ratio of routes matching different reliability targets is depicted for all metrics. Note that in Fig. 2.9a, this ratio is calculated over all routes selected by the respective metric, which includes a high count of trivial single hop routes. We observe that the ratio of routes matching the respective reliability target is close to 100% up to a reliability target of 0.85 for all metrics. With SMTX, all routes satisfy their respective target, even with ambitious

2. r_{min} -Routing

reliability requirements of up to 0.98. Routes found with the ML metric proved quite reliable in previous experiments, already. This observation is confirmed to a degree in this experiment, too. The reliability target of 0.9 is still met by all ML-routes. The two highest targets prove to be too ambitious for routes detected with ML: 0.95 is met by 90 %, 0.98 by a mere 65 %. ETX and ETOP again, show very similar results in this experiment. A reliability target of up to 0.85 is met by almost all routes found with ETX or ETOP. The higher targets of 0.95 and 0.98 are only met by about 60 % and 30 % of routes, respectively. Routes found with the SP metric are similarly reliable, with a ratio of reliability not too far off that achieved by ETX and ETOP. On first look, this seems surprising, as link reliabilities are not considered at all in the SP metric. We suspect that this level of performance of the SP metric stems in part from a high number of very short routes being included in the experiment. As the densities of the random generated topologies range from 0.3 up to 0.6, we can estimate that about 45 % of routes detected using the SP metric are single-hop routes. As detailed in Section 2.4.1, a pair of nodes is only considered connected if the direct link between them has a reliability of at least 0.5. Each transmission is tried up to 4 times, which results in a perceived minimum reliability of 0.94 (according to Eq. (2.6)) for all single-hop routes in the network. Even for two-hop routes, the perceived minimum end-to-end reliability is still slightly above 0.87.

To assess the strength of the influence of very short routes on the ratio of reliable routes in Fig. 2.9a, we evaluated this experiment another time, with all single-hop routes filtered from the results. Results of this second evaluation are depicted in Fig. 2.9b. Up to a reliability target of 0.85, results are very similar to the previous run, while the ratio of reliable routes declines significantly with all metrics apart from SMTX for reliability targets > 0.95 . The ratio of reliable routes selected by SP for a reliability target of 0.95 drops by about 30 points from 52 % down to 22 %, solely due to exclusion of single-hop routes from the data set. Furthermore, without single-hop routes, only very few routes ($< 3\%$) selected by ETX, ETOP or SP satisfy a reliability target of 0.98. ML-selected routes are not affected negatively by exclusion of single-hop routes from the results. This is consistent with the observation that with ML, longer routes are preferred as the metric does not sanction route length as long as the links are of high quality. To conclude this chain of thought, the influence of single-hop routes on the aggregated ratio of reliable routes is significant for metrics ETX, ETOP and SP. Of the assessed metrics, only SMTX selects routes in a way that maintains a very high ratio of reliable routes across all specified reliability targets and route lengths.

2.5. Conclusion

In this chapter, we have introduced r_{\min} -routing, a route selection algorithm capable of discovering routes that satisfy a specified minimum reliability target, called r_{\min} . It was shown that through so-called boosting of some less reliable links, the perceived reliability of individual links can be improved to a degree that makes ambitious end-to-end route reliabilities possible. As an increase of the global per-link transmission limit is undesirable for various reasons, r_{\min} -routing incorporates an algorithm for optimal distribution of a transmission budget among links of a route. This way, end-to-end route reliabilities can be improved without increasing the global transmission limit.

To evaluate whether r_{\min} -routing lives up to what is promised, extensive simulation experiments were carried out. The key takeaway from these experiments is that even ambitious statistical reliability targets can be satisfied, while overprovisioning is avoided. In addition, r_{\min} -routing is capable of finding feasible routes with remarkable consistency. Other metrics evaluated, such as ETX, ETOP and ML result in significantly less reliable routes, they fail when ambitious end-to-end reliability requirements are to be met, or overprovision.

However, r_{\min} -routing is not designed to be a functionally complete routing protocol. There are no mechanisms for route monitoring and maintenance, which are crucial as soon as link reliabilities are dynamic. Key requirements for application of r_{\min} -routing, such as the availability of accurate network state information, as well as a link quality metric that provides aggregatable, normalized and conservative link quality estimates of sufficient actuality are left unspecified and still have to be addressed.

Because of several simplifying assumptions made for the simulation experiments, further evaluation of the proposed route discovery and selection algorithm is necessary once a more complete implementation of r_{\min} -routing is available. In the following chapter, a first step in that direction is taken: we will introduce a protocol for the acquisition and distribution of sufficiently accurate and up-to-date network status information, along with a link quality metric that satisfies the requirements stated in this chapter for use with r_{\min} -routing.

3. cTEx – Decentralized Topology Detection using a Link State Algorithm

For communication functionalities of wireless ad-hoc networks such as route discovery or cluster formation, network status information is to be collected and updated. Depending on the specifics of routes and clusters, this information may range from aggregated data to detailed information about network topology, consisting of nodes, links, and paths. Furthermore, link and path properties, e.g., regarding reliability, delay, throughput, and load, may be required.

In the previous chapter, we have introduced r_{\min} -routing, a route selection mechanism for routes that satisfy a configurable minimum reliability level. During specification of this mechanism, some requirements for applicability of the presented approach were stated. In particular, availability of accurate and up-to-date information on the reliability of individual links is among these requirements. A solution that provides the necessary network status information required for application of r_{\min} -routing is presented in this chapter.

Functionality to discover and update network status information is typically a part of routing and clustering protocols. This has the advantage that the amount of data to be collected can be tailored to the needs of the specific protocol. However, less attention is often paid to the quality of the collected network status data, as the focus is on the main functionality, i.e. routing or clustering. For instance, classical routing protocols for wireless ad-hoc networks such as Dynamic Source Routing (DSR) [JM96] or Ad-hoc On-Demand Distance Vector (AODV) routing [PR99] use links that have been detected by a single successful unidirectional transmission – the broadcast of a route request frame – for route formation. Furthermore, they try to establish short routes (in terms of the number of hops), which gives preference to weaker (and therefore less reliable) links, as these links can cover larger distances than stronger (and therefore more reliable) links. Therefore, more retransmissions are required during route operation, and routes tend to break. This problem is to some degree remedied by Quality-of-Service (QoS) routing protocols, which collect information about link properties, too. Examples are QoS Mobile Routing Backbone over AODV (QMRB-AODV) [IPQ06] and Link Availability-based QoS-aware (LABQ) [YMSF07] routing. Yet, the functionality for network status discovery of these protocols is very specific,

3. *cTEx – Decentralized Topology Detection using a Link State Algorithm*

and it is not straightforward to reuse it in different protocols.

High-quality information about the network status is crucial for the discovery and operation of routes and clusters in wireless ad-hoc networks. Generally, it depends on the application context – e.g., best effort routing, QoS routing, clustering – what amount of network status is required. The importance of different attributes of the network status information also depends on the requirements of the respective application. High accuracy of status information for instance is a common requirement and requires continuous observation and assessment of nodes, links, and link metrics, and distribution of gathered status data. However, especially in networks where energy resources are constrained, stability of the distributed information is often valued higher than adaptivity (i.e. up-to-dateness) and high accuracy is to be weighed against the increased amount of effort it incurs. In addition, some applications (such as r_{\min} -routing) profit from conservative estimates, such that the probability the environment actually supports the communicated quality levels is higher. As a consequence, a proposed solution should be carefully designed for a high degree in configurability in order to support many different requirement profiles. Therefore, we argue that network status discovery and exchange is a core functionality of wireless ad-hoc networks that should receive special attention.

In this chapter, we present *cTEx*, a configurable **T**opology **E**xplorer for wireless ad-hoc networks. *cTEx* consists of functionality for the efficient detection and exchange of high-quality network status information. To this end, *cTEx* is composed of link metrics, link probing algorithm, and status exchange protocol, which are parameterized to match the given application context. The structure of the chapter is as follows: First, a general overview of the *cTEx* algorithm is provided. This is followed by a presentation of the OB-EWMA link quality metric, which is the default link quality metric used by *cTEx*. Then, the complete *cTEx* algorithm is specified in detail supported by formal notation. *cTEx* shares some principles with a protocol from the literature called Efficient and Accurate link-quality monitor (EAR), hence an outline of the EAR protocol is provided before we move on to a conceptual comparison of *cTEx* and EAR. This section is followed by an evaluation of the *cTEx* protocol in simulation as well as real-world experiments, along with a performance comparison between *cTEx* and EAR. Towards the end of the chapter, we provide an overview of related work and some concluding thoughts.

3.1. **cTEx** Algorithm Overview

cTEx – the configurable Topology Explorer – is a modular protocol for the efficient detection and exchange of network status information. In this section, we will provide an overview of its two main components: link status detection and distribution of topology information.

3.1.1. Link Status Detection

In its link detection component, *cTEx* uses hybrid probing, a combination of the concepts active probing and passive probing, to estimate link quality. Both active and passive probing are classes of techniques for deriving the quality of a link [LL17]. As suggested by the name, active probing techniques involve the creation and transmission of probe messages of some kind, whereas passive probing techniques derive link quality by mere observation, e.g. of physical layer properties such as received signal strength. For the applicability of passive probing techniques, some network traffic is usually required, though mechanisms controlling the generation of this traffic are not part of the passive probing technique.

The advantage of active probing is that the knowledge about the topology does not degrade during times where the network is used sparsely for application data. At specific intervals, extra probe messages can be transmitted and used for a continuous evaluation of link quality. This comes with the drawback that in situations of high network load, the probe messages decrease the capacity of the network for application data.

Passive probing techniques usually have no influence on a network's capacity. The metrics used for link quality estimation like the delivery ratio of frames or physical parameters are acquired by observation only. An active participation in the network is no part of passive probing techniques. Consequently, link quality and topology information degrades during periods where parts of the network are unused, as passive probing techniques rely on metadata created from regular transmissions.

Hybrid probing techniques aim to combine advantages of both techniques, while trying to minimize their drawbacks. In *cTEx*, active probing is applied only when there are extended periods without transmissions. In situations of high network load, no additional traffic for probing of links is created, as there are enough transmissions generating metadata. In combination, these mechanisms lead to a constant stream of metadata to base the link quality estimations on.

Different from many other link quality estimation techniques, *cTEx* does not evaluate outgoing, but incoming links. The most important advantage of this approach is that by observing ingress traffic, a single transmission of a node can

3. cTEx – Decentralized Topology Detection using a Link State Algorithm

be used to improve the link quality estimation for all of its neighbors. By operating each node’s transceiver in monitor mode, this advantage also holds in case of unicast transmissions. With approaches where outgoing links are evaluated, a single transmission either contributes to the link quality estimation towards a single destination node only, or requires some kind of separate acknowledgement from all destination nodes.

cTEx bases its estimations of link quality (and by extension its view of the topology) solely on observations of the *frame delivery ratio*¹ (FDR). At this time, physical parameters such as the received signal strength are not used for link quality estimation. As mentioned above, the interpretation of hybrid probing employed in cTEx is that the protocol remains passive, as long as regular transmissions occur and only resorts to an active mode, when there are no transmissions generated by upper layers for a configured amount of time.

The goal of this mechanism is to ensure at all times that there is enough traffic such that accurate estimations of link quality can be made continuously. The impact on network load is kept low, as each node transmits dedicated probe frames only in the absence of application traffic. However, this mechanism is not perfect in that regard: for instance, a node that is mainly receiving will continuously transmit dedicated probe frames. This would be tolerable, if the node is expected to also transmit application data in the future, but this is not true in all scenarios. A sink node for example is generally neither a source node nor an intermediate hop of a routed network packet. With this mechanism the sink node will continuously transmit dedicated probe frames to enable evaluation of its otherwise unused outgoing link, which, at least in this case, is an unnecessary strain on network capacity. To alleviate this issue one might disable active probing for nodes known to not use their outgoing links, such as dedicated sink nodes. This is, however, also detrimental to the efficacy of mechanisms in cTEx for distribution of topology information, which rely on a node’s egress traffic.

To process the stream of transmission metadata collected by each node for each of its neighbors, cTEx divides time into so-called FDR-slots of a configurable length. At the end of each FDR-slot, the received transmission metadata is processed and an FDR value for the current slot is calculated. To detect potential loss of frames, cTEx requires every transmission to carry a sequence number. This allows detection of failed transmissions by searching for gaps in the stream of sequence numbers and lets us calculate the ratio of successful frame transmissions. The result of this first step of processing is a vector of FDR values, with one value for each FDR-slot.

In the second and final step the vector of FDR values is processed according

¹On the network layer, this metric is well-known and called *packet delivery ratio* (PDR). In this context however, PDR would not be the correct term, as cTEx solely monitors MAC layer frame delivery, ignoring any potential aggregations of individual frames into packets. See Section 2.1 for details on terminology used in this thesis.

to the specification of a link quality metric. To maintain a high degree of modularity, cTEx is designed to work with any metric that operates on this stream of FDR values provided by cTEx. This metric controls key aspects of the nature of link quality values, e.g. reactivity, conservativeness or stability. For example, to achieve maximum reactivity, a metric might simply choose the most recent FDR value in the vector as link quality. To obtain a more smoothed sequence of values not as much affected by outliers, one might prefer to factor in values from several FDR-slots and calculate an average. In case the link quality is also required to be a conservative estimate, the vector of FDR values may need to be filtered for upward outliers first. We find that the the optimal choice of link quality metric strongly depends on the environment the link quality values are to be used. As cTEx aims to be a configurable, universal protocol for topology exploration with wide applicability we consider a configurable link quality metric a mandatory feature. Nonetheless, cTEx comes with a link quality metric called Outlier Bounded Exponential Weighted Moving Average (OB-EWMA) [MG21], which will be outlined in Section 3.2.

3.1.2. Distribution of Topology Information

In addition to providing mechanisms for generation of high quality network status information cTEx also incorporates a sophisticated scheme for fast and efficient network-wide distribution of topology information. Topology information is embedded in form of so-called *topology chunks* into network frames. One topology chunk describes the current state of a single link and contains identifiers for source and destination node, the link's reliability value, and a timestamp. This information is condensed into a data structure that occupies only 5 Bytes per link. Depending on the combination of distribution strategies, topology chunks for every link are embedded in every frame, only periodically at specific intervals, or only for links that experienced a significant change in quality since last embedment.

Distribution Strategies

For every packet that passes through the cTEx layer (from top to bottom) and every probe message created in active mode by the cTEx layer itself, an algorithm decides based on the configured distribution strategies, whether topology chunks are embedded into the resulting network frames and if so, for which links.

If the strategy `PERIODICALLY_FULL` is configured, topology information for every link known to a node is embedded at specific intervals. That includes links for which the respective node is *authoritative* (any direct incoming links), as well as link information relayed from other nodes, for which this node is *non-*

3. cTEx – Decentralized Topology Detection using a Link State Algorithm

authoritative. Depending on the size of the network, the size of data to be added may exceed the maximum size for a frame. In this case, subsequent frames are used to embed remaining topology chunks. With respect to a node’s memory resources however, topology size usually remains manageable. The protocol is designed for networks of up to 100 nodes, which results in a maximum number of $100 \cdot (100 - 1) = 9900$ entries in the list of (asymmetric) links in case of a fully connected network. As each entry can be stored in 5 Bytes, a single² topology table should never exceed a size of 49.5 Kilobytes. However, to transmit this topology of maximum size with 802.11 WiFi frames, at least 23 frames are required to embed all chunks.

One drawback of the PERIODICALLY_FULL strategy is that drastic changes in link quality just after the interval triggered may take a while to be reported to other nodes and propagate through the network. A more reactive strategy could help remedy that. The second distribution strategy available in cTEx called ONLY_UPDATED provides the reactivity PERIODICALLY_FULL is lacking. Based on two thresholds³, one for link improvement and one for link deterioration, a link’s state is included in the next frame. For this strategy, the last reported quality value of a link is stored. Before a frame is finalized, the current quality of each link is compared to the quality value last reported. If their difference exceeds any of the configured thresholds, the link is queued for embedment in the frame. This way, significant changes of link quality are allowed to propagate quickly through the network, while links of more or less constant quality cause almost no additional distribution overhead.

The two distribution strategies are non-exclusive and can be combined. Should the need arise, more strategies can be added easily.

Topology View Updates

cTEx maintains an internal data structure that reflects the current view of the topology. This view is kept up-to-date with link quality estimates performed by the node itself (for links the node is authoritative for) and via received topology chunks embedded into network frames received from neighboring nodes. For every frame that is received, the dedicated cTEx monitoring component checks whether topology information is embedded. Any embedded chunks are extracted and decoded. Topology chunks for links that were previously unknown to this node trigger the creation of that link in the node’s topology view. Existing links are updated when comparison of the timestamp embedded in the

²cTEx supports maintenance of several topology tables, as it is often necessary to manage different topology views per communication channel, frame size category and transmission rate.

³We use two thresholds here to optionally allow faster reaction towards link deterioration without increasing overhead incurred by communicating improvements of link quality, as some protocols (e.g. r_{\min} -routing) need to react when a link drops below a minimum link quality.

topology chunk with the timestamp of the corresponding link in the node's local topology view indicates the received topology chunk contains more recent information.

3.2. Reference Link Quality Metric for cTEx

cTEx is designed to work well with r_{\min} -routing (see Chapter 2), a route selection algorithm that aims to find routes that satisfy a minimum reliability target. The approach relies on high quality network status information and on the availability of link quality estimates that satisfy the requirements aggregatability, actuality and conservativeness. cTEx provides the framework to obtain and distribute these link quality estimates, whereas the applied link quality metric must be constructed in a way that ensures the resulting estimates are normalized, aggregatable, sufficiently recent as well as conservative.

The OB-EWMA link quality metric, introduced in [MG21] manages to produce link quality estimates of sufficient quality to be used in the context of r_{\min} -routing. In this section, an outline of the mechanisms and ideas employed in OB-EWMA will be provided, along with a summary of the evaluation of OB-EWMA in the context of r_{\min} -routing as carried out in [MG21].

OB-EWMA works on sequences of FDR values, where each FDR value is calculated as the ratio of successfully received frames over a time period of length d_{fs} , called an FDR-slot. For a link e , we name the resulting sequence of FDR values fdr^e . A single OB-EWMA link quality estimate for some link e is then calculated over a sequence of sliding windows that move over the vector fdr^e of FDR values. The number of moving windows is n_{mw} and the size of each window (i.e. the number of spanned FDR-slots) is s_{mw} . To filter upward and downward outliers and in order to obtain a conservative estimate, only the p -percentile of the FDR-slots in each window is used. As usual, an exponential weighting of $n_{mw} - 1$ previous windows is achieved by introducing a factor α . The complete formula that defines the OB-EWMA metric is given below:

$$\text{OB}_p^e(n_{mw}, s_{mw}) = \text{empirical } p\text{-percentile from the set} \\ \{fdr^e[k] \mid n_{mw} - s_{mw} + 1 \leq k \leq n_{mw}\}$$

$$\text{OB-EWMA}_{\alpha,p}^e(n_{mw}, s_{mw}) = \begin{cases} \text{undefined}, & \text{if } n_{mw} < s_{mw} \\ \text{OB}_p^e(n_{mw}, s_{mw}), & \text{if } n_{mw} = s_{mw} \\ \alpha \cdot \text{OB-EWMA}_{\alpha,p}^e(n_{mw} - 1, s_{mw}) + \\ (1 - \alpha) \cdot \text{OB}_p^e(n_{mw}, s_{mw}), & \text{if } n_{mw} > s_{mw} \end{cases} \quad (3.1)$$

3. *cTEx* – Decentralized Topology Detection using a Link State Algorithm

The stated properties of the metric were verified in extensive real-world experiments by its original authors with the following setup: The testbed consisting of 11 nodes is placed in a university building. To collect data on the quality of links in the network, 802.11g WiFi frames were transmitted at regular intervals between each pair of nodes and the outcome of each transmission was recorded. These data are then used to calculate OB-EWMA values in an offline analysis.

The results show that OB-EWMA is indeed capable to provide accurate conservative estimates of link quality, with sufficient adaptability. For the stronger links in the network (average FDR > 0.8), results show the quality value resulting from OB-EWMA is an overestimate of the actual FDR value for 12% of FDR-slots, so OB-EWMA values can be considered conservative estimates.

In a separate experiment, the aptness of OB-EWMA link quality estimates for operation of r_{\min} -routes (see Chapter 2) was tested. The values provided by OB-EWMA were used to set the transmission budget for each link in a single 3-hop route. The reliability target was set to $r_{\min} = 0.95$. During route operation, the reliability of the established route (i.e. ratio of packets⁴ received to packets sent) during each FDR-slot was calculated. The results suggest stable route operation is possible, though the reliability target is missed in some slots. Evaluation of OB-EWMA in the context of r_{\min} -routing is continued in Chapter 4 of this thesis.

3.3. Formal Specification of *cTEx* Algorithm

In Section 3.1 a general overview of the mechanisms employed in the *cTEx* protocol was provided. In this section, a detailed specification supported by formal notation of the core functionalities is introduced along with some background on the design and implementation decisions made. The protocol specification consists of four core functionalities:

- Link detection
 - Reception and processing of probe frames (Section 3.3.3)
 - Transmission of probe frames (Section 3.3.4)
- Maintenance of a local view of the global network status
 - Exchange of network status information (Section 3.3.5)
 - Processing of network status information (Section 3.3.6)

⁴In this scenario, it is in fact the packet delivery ratio, as the r_{\min} target refers to a ratio of delivered network layer packets, not frames.

3.3.1. Network Model

The network model at the foundation of all formal notations used in this and the following sections is largely the same as introduced in Section 2.2 of Chapter 2. Special care has to be taken when referring to the concepts of link reliability and quality. The reliability r_e of an edge e in the network model is the success probability of only the next communication attempt. The quality q^e of a link (identified by its corresponding edge e in the model) is the output value of a link quality metric at some point in time.

Compared to the model introduced in Chapter 2, we extend the definition of the communication topology slightly to include network status for different points in time, channels, transmission rates and frame sizes. Consequently, we model communication topology as a tuple $G = (V, q, c, r_{tx}, s_f)$, consisting of a set of vertices V , a statistical quality function $q : \text{Time} \times V \times V \rightarrow \mathbb{R}_{0,1}$ for single-hop communication, a channel c , a transmission rate r_{tx} , and a frame size s_f . Based on q , the set of (asymmetrical) links at time t is derived as

$$E(t) =_{df} \{(t, v, v') \in \text{Time} \times V \times V \mid q(t, v, v') > 0\} \quad (3.2)$$

and thus also captures the dynamic of network status over time. This model comprises several aspects that are of particular importance in wireless ad-hoc networks:

- *Asymmetrical links e .* In wireless ad-hoc networks, links are often asymmetrical, resulting, e.g., from interference and reflections. In the model, this is captured by a set of directed edges $E(t)$.
- *Statistical quality function q .* In many scenarios, it is not sufficient to model links as either existing or non-existing. Therefore, we adopt a statistical approach and associate a quality value $q(t, v, v')$ with each pair of vertices at time t . If this value is greater than zero, the pair of vertices constitutes an edge $e = (t, v, v')$. By having a time value as parameter of q , we can model varying link quality. The definition of q is to be chosen to fit the scenario. For instance, for reliability-constrained routing, quality can be defined as packet delivery ratio or as expected number of transmissions.
- *Channel c .* An important factor that influences link quality is the quality and utilization of the communication channel, i.e. the portion of the electromagnetic spectrum that is used for frame transmissions. For technologies supporting channel hopping, it is not sufficient to model the communication topology independently of the used channel. Rather, one topology per channel is to be considered.

3. cTEx – Decentralized Topology Detection using a Link State Algorithm

Table 3.1: Functions used in pseudocode descriptions of cTEx

Name	Type	Description
$node()$	Address $\rightarrow V$	Function mapping a node address to the corresponding element of V . Inverse of $addr()$.
$addr()$	$V \rightarrow$ Address	Function mapping an element of V to the corresponding node address. Inverse of $node()$.
$metric()$	Real[] \rightarrow Real	Link quality metric
$size()$	Void \rightarrow Real	Obtain size of a frame
$embed()$	Tuple \rightarrow Void	Reduce and embed topology information into a frame

- *Transmission rate r_{tx}* . In IEEE 802.11 networks, different transmission rates can be used. For instance, broadcasting may use a low data rate, e.g., 2 Mbps, that is more tolerant to bit errors than higher rates used for unicasting. Therefore, ideally, one topology per transmission rate is to be considered.
- *Frame size s_f* . Link quality depends on frame size, where the frame delivery ratio decreases with increasing frame size. Therefore, ideally, one topology per frame size should be considered. This, however, is not feasible in practice, as this would produce an extreme amount of management traffic. Therefore, a small number of typical frame sizes may be considered.

To keep topology detection manageable, the number of channels, transmission rates, and frame sizes should be kept small. For instance, a single channel could be used, and one or two typical frame sizes and transmission rates could be chosen.

3.3.2. Functions, Global Parameters and State Variables of cTEx

Tables 3.1 to 3.4 list functions, global parameters, and state variables of nodes and link state automata used in the specification of the cTEx algorithms. Following, we provide some information about their meaning and usage.

Table 3.1 contains five functions. The function $node()$ is defined to obtain the sending node $v \in V$ from its node address $f.snd$ contained in every received frame f . Its inverse operation is handled by $addr()$. The function $metric()$ determines how link quality is to be derived. It is defined to operate on a sequence of FDR values. For instance, $metric()$ can be instantiated by most metrics introduced in the previous Chapter 2, such as ETX or ML. In the context of this thesis

Table 3.2: Global configuration parameters of cTEx

Name	Type	Description
d_{fs}	Duration	FDR-slot duration
$r_{pf,min}$	Real	Minimum rate of probe frames
d_{pf}	Duration	Maximum time between two subsequent probe frames, $d_{pf} = (r_{pf,min})^{-1}$
d_{term}	Duration	Termination delay in case of link inactivity
$q_{diff\uparrow}$	Real	Minimum difference of reported and current (lower) link quality to trigger a report, a negative value
$q_{diff\downarrow}$	Real	Minimum difference of reported and current (higher) link quality to trigger a report, a positive value
d_{rep}	Duration	Maximum time between two reports of a link

the reference link quality metric OB-EWMA [MG21] as outlined in Section 3.2 is used unless stated otherwise. The $size()$ function calculates the size in bytes of any given frame, while $embed()$ is a helper function encapsulating the process of reduction and embedment of a link state tuple into a frame.

Table 3.2 lists global configuration parameters of cTEx. Local time is structured into FDR-slots of duration d_{fs} . An FDR value is calculated at the end of each FDR-slot, based on the received probe frames in this slot. Probe frames are transmitted with the minimum rate $r_{pf,min}$. If for a duration of length d_{pf} no frame is sent, a dedicated explicit probe frame is created to maintain $r_{pf,min}$. The parameter d_{term} determines after what duration of link inactivity a link state automaton terminates. The parameter q_{diff} determines when the difference between reported and current link quality of a link e should trigger a status report, i.e. the link's changed status should be communicated to the rest of the network. Similarly, d_{rep} specifies the maximum time allowed between two subsequent reports of the same link. These last two parameters control the two distribution strategies available, event-triggered and time-triggered (previously introduced as ONLY_UPADTED and PERIODICALLY_FULL, respectively).

Table 3.3 summarizes the state variables of a node v . Each node v maintains an initially empty set lsa^v of link state automata lsa^e for incoming active links e . An automaton lsa^e is created and added to lsa^v when a new link e is detected, and terminated and removed from lsa^v when an existing link e is deemed inactive. The network status information of node v is represented by an initially empty

Table 3.3: State variables of each cTEx node

Name	Type	Description
lsa^v	Set of LSA	Set of link state automata lsa^e of node $v \in V$, $e = (v', v) \in V \times V$. Initially, $lsa^v = \emptyset$
T^v	Set of Tuple	Network status of node $v \in V$. Initially, $T^v = \emptyset$
c_{local}^v	Clock	Local clock
$timer_{pf}^v$	Timer	Explicit probing timer
n_{txCtr}^v	Int	Counter for probe frames sent by v

set T^v of tuples T^e , further detailed in Section 3.3.5. Nodes have a local clock c_{local}^v used to set timers and to timestamp update events. The timer $timer_{pf}^v$ is set to trigger creation of explicit probing frames, in case conventional outgoing traffic is insufficient to maintain $r_{pf,min}$. With n_{txCtr}^v all frames sent by node v are counted.

Table 3.4 lists the state variables of link state automata lsa^e . The current link quality q^e is updated at the end of each FDR-slot, using the function $metric()$. The link quality q^e of link e is associated with a sequence number $n_{q,seq}^e$, which is updated every time the link quality is re-computed. This way, nodes receiving network status information can determine which reported quality value is the most recent one. The counter n_{fs}^e keeps track of the current FDR-slot number, starting with 0. State variables n_{seqLow}^e and $n_{seqHigh}^e$ keep track of the lowest and highest sequence number of probe frames received in the current FDR-slot so far. The counter n_{rxCtr}^e retains the number of probe frames received in this FDR-slot. Time t_{start}^e is the starting point of the current FDR-slot. In the array fdr^e , the history of FDR values collected in the corresponding FDR-slots is collected, to be used to calculate q^e . Timers $timer_{fs}^e$ and $timer_{term}^e$ are set to trigger actions at the end of an FDR-slot and in case of link inactivity, respectively.

3.3.3. Reception and Processing of Probe Frames

Nodes listen for probe frames in order to detect and assess links. Received probe frames are processed to derive status information about incoming links. For each incoming link e , channel c , transmission rate r_{tx} , and frame size category s_{cat} , nodes create and maintain a link state automaton. In this section, we will omit the distinction of links by c , r_{tx} , and s_{cat} , to simplify the exposition. With this simplification we can write the link state automaton that handles link e as lsa^e , instead of $lsa^{e,c,r_{tx},s_{cat}}$.

Listings 3.1 and 3.2 show transitions of link state automata. Here, transitions

```

1 when receive  $f$  then {
2   // Derive respective link from sender and receiver addresses
3   let  $e = (\text{node}(f.\text{snd}), v)$  in {
4     if  $\text{lsa}^e \in \text{lsa}^v$  then {
5       // Link is already known, state variables are
6       // initialized
7       // Increment frame counter of current slot
8        $n_{\text{rxCtr}}^e = n_{\text{rxCtr}}^e + 1$ 
9       // Update max sequence number of current slot
10       $n_{\text{seqHigh}}^e = f.\text{txCtr}$ 
11      // Reset termination timer
12      set  $\text{timer}_{\text{term}}^e = c_{\text{local}}^v + d_{\text{term}}$ 
13    } else {
14      // Link is new, create link state automaton
15      // and initialize state variables
16      create  $\text{lsa}^e$ 
17      // Add lsa to set
18       $\text{lsa}^v = \text{lsa}^v \cup \{\text{lsa}^e\}$ 
19
20      // Initialize state variables
21       $q^e = 0$ 
22       $n_{\text{fs}}^e = 0$ 
23       $n_{\text{rxCtr}}^e = 1$ 
24       $n_{\text{seqLow}}^e = n_{\text{seqHigh}}^e = f.\text{txCtr}$ 
25       $n_{q,\text{seq}}^e = 0$ 
26       $t_{\text{start}}^e = c_{\text{local}}^v$ 
27
28      // Initialize link state tuple in topology view  $T^v$ 
29       $T^v = T^v \cup \{(e, q^e, n_{q,\text{seq}}^e, \text{true}, 0, c_{\text{local}}^v, c_{\text{local}}^v)\}$ 
30
31      // Initialize FDR-slot timer
32      set  $\text{timer}_{\text{fs}}^e = t_{\text{start}}^e + d_{\text{fs}}$ 
33      // Initialize termination timer
34      set  $\text{timer}_{\text{term}}^e = c_{\text{local}}^v + d_{\text{term}}$ 
35    }
36  }
37 }

```

Listing 3.1: Pseudocode for the link state automata of cTEx, part 1

3. cTEx – Decentralized Topology Detection using a Link State Algorithm

```

38 when expiry  $timer_{fs}^e$  then {
39     // Current FDR-slot is over, so calculate its frame
40     // delivery ratio and re-initialize state for next slot
41     // Determine number of frames sent over this link
42      $n_{sent} = \max(\lfloor r_{pf,min} \cdot d_{fs} \rfloor, n_{seqHigh}^e - n_{seqLow}^e)$ 
43     // Calculate and store FDR value for last slot
44      $fdr^e[n_{fs}^e] = \min(1, \frac{n_{rxCtr}^e}{n_{sent}})$ 
45     // Re-calculate link quality with metric over updated  $fdr^e$ 
46      $q^e = \text{metric}(fdr^e)$ 
47
48     // Re-initialize state variables for next slot
49      $n_{rxCtr}^e = 0$ 
50      $n_{fs}^e = n_{fs}^e + 1$ 
51      $n_{seqLow}^e = n_{seqHigh}^e$ 
52      $t_{start}^e = t_{start}^e + d_{fs}$ 
53      $n_{q,seq}^e = n_{q,seq}^e + 1$ 
54
55     // Create updated link tuple, with  $q_{rep}^e$  and  $t_{rep}^e$  unmodified
56     // and  $t_{upd}^e$  set to local time
57      $T_{upd}^e = \{(e, q^e, n_{q,seq}^e, T^e.b_{rep}^e \text{ or } q_{diff\uparrow} < (T^e.q_{rep}^e - q^e) < q_{diff\downarrow},$ 
58          $T^e.q_{rep}^e, T^e.t_{rep}^e, c_{local}^v)\}$ 
59     // Update  $T^v$  with  $T_{upd}^e$ 
60      $T^v = T^v \setminus \{T^e\} \cup T_{upd}^e$ 
61
62     // Reset FDR-slot timer
63     set  $timer_{fs}^e = t_{start}^e + d_{fs}$ 
64 }
65
66 when expiry  $timer_{term}^e$  then {
67     // No frames were received over link  $e$  for duration  $d_{term}$ ,
68     // so this link is considered broken
69     terminate  $lsa^e$ 
70      $lsa^v = lsa^v \setminus \{lsa^e\}$ 
71 }

```

Listing 3.2: Pseudocode for the link state automata of cTEx, part 2

Table 3.4: State variables of each link state automaton lsa^e

Name	Type	Description
q^e	Real	Current link quality
$n_{q,seq}^e$	Int	Sequence number of current link quality
n_{fs}^e	Int	FDR-slot number
n_{seqLow}^e	Int	Lowest sequence number of probe frames received in this FDR-slot
$n_{seqHigh}^e$	Int	Highest sequence number of probe frames received in this FDR-slot
n_{rxCtr}^e	Int	Counter for probe frames received in this FDR slot
t_{start}^e	Time	Local start time of this FDR-slot
fdr^e	Real[]	Array of FDR values observed in FDR-slots
$timer_{fs}^e$	Timer	FDR-slot timer
$timer_{term}^e$	Timer	Termination timer

are triggered by the reception of probe frames and the expiration of timers: When node $v \in V$ receives a probe frame f with sender node address snd and probe sequence number $txCtr$, it checks whether for the incoming link $e = (node(f.snd), v)$, a link state automaton already exists (line 4). In this case, the counter n_{rxCtr}^e is incremented (line 8), and $n_{seqHigh}^e$ is updated (line 10). In addition, the termination timer is set (line 12). If no link state automaton for link e exists, lsa^e is created and initialized (lines 16-26), followed by initializations of the respective link state tuple T^e in T^v (line 29) and FDR-slot and termination timers.

When $timer_{fs}^e$ expires (line 38 in Listing 3.2) at the end of the current FDR-slot n_{fs}^e , the frame delivery ratio is determined (lines 42-44), and the function $metric()$ is applied to pdr^e , yielding the current reliability estimate q^e (line 46). Subsequently, the next FDR-slot is initialized (lines 49-53), and the updated link status is incorporated into the local topology view T^v (lines 57-60), where access of an element $elem$ of a tuple T is denoted as $T.elem$. The network status information of a node is represented as a set T^v of tuples $T^e = (e, q^e, n_{q,seq}^e, b_{rep}^e, t_{upd}^e, t_{rep}^e)$, where $e = (v_1, v_2) \in V \times V$ and q^e and $n_{q,seq}^e$ as described in Table 3.4. b_{rep}^e indicates whether a link is to be published according to the event-triggered distribution strategy (see below), t_{upd}^e is the local time of the last link status update, and t_{rep}^e is the local time this link status was reported last. Finally, $timer_{term}^e$ is

reset to indicate the link is still active.

When $timer_{term}^e$ expires (line 66), this indicates a longer period of frame losses for link e , as every received frame resets $timer_{term}^e$ in line 34 of Listing 3.1. In this case, the link state automaton is terminated and removed from lsa^v .

In cTEx, nodes operate in monitor mode (also called promiscuous mode) in order to overhear all incoming (implicit and explicit) probe frames. This has the advantage that unicast frames are actually perceived as broadcast frames, which avoids the drawback of a fixed and low transmission rate.

3.3.4. Transmission of Probe Frames

cTEx-nodes ensure a regular stream of (implicit or explicit) *probe frames* to enable the detection of links by direct neighbors. A probe frame is any unicast frame that is created to either transport data from a higher layer packet (an *implicit* probe frame) or to serve as a dedicated probe message (an *explicit* probe frame). To reduce traffic, unicast frames of regular or management traffic are interpreted as implicit probe frames. Only when there is not enough unicast regular or management traffic, dedicated explicit probe frames are sent. Thus, a minimum rate of probing is supported, with short reaction times to create explicit probe frames. The maximum rate depends on the frequency of implicit probe frames, with no enforced upper bound. To detect probe frames, nodes have to operate in monitor mode (also called promiscuous mode), where all frames regardless the destination address are received by the wireless communication interface and handed over to the protocol stack. Implicit probe frames (i.e. conventional unicast traffic) are sent to the MAC address of an existing node. Explicit probe frames are sent to an unused MAC address. Each probe frame f carries sender node address snd , transmission rate r_{tx} , frame size s_f , and probe sequence number $txCtr$.

Creation of explicit probe frames by request of the cTEx layer follows specific rules that ensure a specified minimum rate $r_{pf,min}$ of outgoing frames. In Listing 3.3 the mechanism is described in pseudocode. The transitions in Listing 3.3 are triggered upon transmission of frames and by expiry of timers. Refer to Tables 3.1 to 3.3 for descriptions of the functions and state variables used.

During initialization of a network node, $timer_{pf}^v$ is configured to expire whenever there is a period of length d_{pf} , where no probe frame was transmitted. Every transmitted unicast frame will reset this timer (see line 7). When $timer_{pf}^v$ expires (line 10), a new explicit probing frame is created (line 12) and initialized (lines 14-17). When this explicit probe frame is finally transmitted, it also resets this timer (again per line 7). An illustration of the resulting stream of frames is depicted in Fig. 3.1: A red vertical bar indicates transmission of an implicit probe frame, usually triggered by creation of a new application layer packet. Whenever the period between subsequent implicit frames exceeds d_{pf} ,

```

1 when send  $f$  then {
2     // Increment frame counter
3      $n_{txCtr}^v = n_{txCtr}^v + 1$ 
4     // Set frame counter field of frame  $f$ 
5      $f.txCtr = n_{txCtr}^v$ 
6     // Reset timer that triggers next explicit probe frame
7     set  $timer_{pf}^v = c_{local}^v + d_{pf}$ 
8 }
9
10 when expiry  $timer_{pf}^v$  then {
11     // Create new explicit probe frame of length  $s_f$ 
12     create  $f$ 
13     // Set tx rate, frame size, sender and destination addresses
14      $f.r_{tx} = r_{tx}^v$ 
15      $f.s_f = size(f)$ 
16      $f.snd = addr(v)$ 
17      $f.dst = UNUSED\_ADDRESS$ 
18
19     // Trigger send event for new frame
20     emit send  $f$ 
21 }

```

Listing 3.3: Pseudocode describing the generation of explicit probe frames

3. cTE_x – Decentralized Topology Detection using a Link State Algorithm

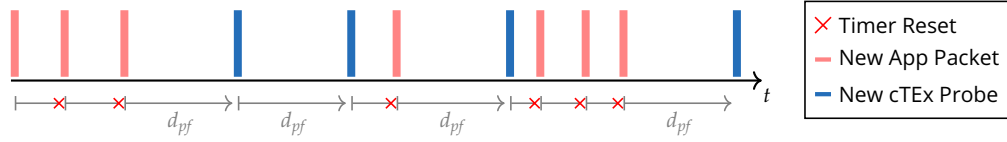


Figure 3.1: Timing of probe creation (cTE_x active mode) under sporadic egress application layer traffic.

an explicit probe frame (blue vertical bars) is transmitted to maintain $r_{pf,min}$.

Only unicast frames qualify as probe frames, as link quality measurements based on broadcast frames have been described to yield inaccurate results, due to different PHY settings and a fixed and low transmission rate [KS06]. Another benefit is that unicast usually enables several transmission rates. Furthermore, probe frames are sent without RTS/CTS, and with automatic retransmission disabled. With these settings, we keep control over retransmissions, which are triggered from a higher layer.

For each topology to be determined, i.e. per channel, transmission rate, and frame size category, and for each sending node, probe frames f carry consecutive probe sequence numbers $txCtr$. This way, all receiving nodes can detect frame losses and determine the actual frame delivery ratio (see Section 3.3.3). In particular, no feedback of receiving nodes to the sender, e.g. ACKs or accumulated ACKs, is required for this purpose. This improves the accuracy of link assessment, as it is not biased by the possible loss of ACKs.

3.3.5. Exchange of Network Status Information

Local network status information may be extracted from the link state automata of a node and reported to other nodes. Non-local information may be received, transformed, and reported to other nodes. All (implicit and explicit) probe frames may carry network status information. When reported via probe frames, tuples $T^e \in T^v$ are reduced to $T_{rep}^e = (e, q^e, n_{q,seq}^e)$, so b_{rep}^e , t_{upd}^e and t_{rep}^e are not communicated between nodes.

cTE_x applies two strategies to exchange network status information contained in T^v with nodes in 1-hop neighborhood of v . A description of both strategies in the form of pseudocode is given in Listing 3.4.

The event-triggered strategy is applied when the quality value q^e of a link changes significantly compared to the previous value, i.e. when b_{rep}^e is true for some link e (see line 57 of Listing 3.2 for incoming local links and line 9 of Listing 3.5 for external links). When the next probe frame is to be transmitted, the set of tuples $T^e \in T^v$ where b_{rep}^e is set is extracted from T^v (line 3), reduced to tuples T_{rep}^e , and included in the probe frames (lines 7-18). Furthermore, for all reported tuples T^e , b_{rep}^e is set to *false* (line 13), and t_{rep}^e is set to the local time c_{local}^v

```

1 when send  $f$  then {
2     // Set of significantly changed links (event-triggered)
3      $T_{sig} = \{T_e \in T_v \mid b_{rep}^e = \mathbf{true}\}$ 
4     // Set of links due for periodic distribution (time-triggered)
5      $T_{due} = \{T_e \in T_v \mid t_{rep}^e < (c_{local}^v - d_{rep})\}$ 
6
7     for  $T_{tx}^e$  in  $T_{sig} \cup T_{due}$  {
8         // We embed until frame is at maximum capacity
9         if ( $\text{size}(f) + \text{TOPO\_CHUNK\_SIZE}$ ) <  $\text{MAX\_FRAME\_SIZE}$  {
10            // reduce and embed  $T_{tx}^e$  into  $f$ 
11             $\text{embed}(T_{tx}^e, f)$ 
12            // Update report flag and timestamp
13             $T_{tx}^e.b_{rep}^e = \mathbf{false}$ 
14             $T_{tx}^e.t_{rep}^e = c_{local}^v$ 
15        } else {
16            break
17        }
18    }
19 }

```

Listing 3.4: Pseudocode describing the embedment of network status information into outgoing frames

(line 14).

The time-triggered strategy ensures all links known to a node are reported at least once in a time interval of length d_{rep} , i.e. all tuples $T^e \in T^v$ not reported in the last d_{rep} seconds are selected for reporting (line 5). This information is included in the next probe frame(s) to be transmitted. As in the event-triggered case, for all reported tuples T^e , b_{rep}^e is set to *false*, and t_{rep}^e is set to the local time c_{local}^v (lines 7 and 18). Time-triggered reporting ensures that new network nodes can synchronize their network status information also for stable links, which are not reported by the event-triggered strategy. Furthermore, it enables nodes to detect inactive (i.e. broken) links (see Section 3.3.6).

As probe frames have a maximum frame size, not all link status information scheduled for reporting may fit into a single frame. Here, reporting happens in the order of update times, given by t_{upd}^e . In this case, several probe frames are required for the exchange. In our implementation, a tuple T_{tx}^e has a size of 5 bytes only.

```

1 when receive  $f$  then {
2     // Extract topology information from received frame
3      $T_{rx} = f.tuples$ 
4
5     // Set of new links, unknown to this node up to now
6      $T_{new} = \{(T_{rx}^e.e, T_{rx}^e.q^e, T_{rx}^e.n_{q,seq}^e, true, 0, c_{local}^v, c_{local}^v)$ 
7          $| T_{rx}^e \in T_{rx} \wedge \nexists T_{local}^e \in T^v . T_{local}^e.e = T_{rx}^e.e\}$ 
8     // Set of known links with an update in the received frame
9      $T_{upd} = \{(T_{rx}^e.e, T_{rx}^e.q^e, T_{rx}^e.n_{q,seq}^e, q_{diff\uparrow} < (T^e.q_{rep}^e - q^e) < q_{diff\downarrow},$ 
10         $T_{local}^e.q_{rep}^e, T_{local}^e.t_{rep}^e, c_{local}^v)$ 
11         $| T_{rx}^e \in T_{rx} \wedge T_{local}^e \in T^v \wedge T_{local}^e.n_{q,seq}^e < T_{rx}^e.n_{q,seq}^e\}$ 
12
13     // Remove links we have an update for
14      $T_v = T_v \setminus \{T_{local}^e \in T^v | \exists T_{rx}^e \in T_{upd} . T_{rx}^e.e = T_{local}^e.e\}$ 
15     // Add new and updated links
16      $T_v = T_v \cup T_{upd} \cup T_{new}$ 
17 }
```

Listing 3.5: Pseudocode describing the processing of received network status information for incoming frames

3.3.6. Processing of Network Status Information

Network status information extracted from the link state automata (i.e. gathered locally) and received from other nodes is transformed into a joint local view of the global network status. The amount of detail of this local view depends on the amount of collected network status information and the degree of abstraction.

Updates for locally evaluated links are continuously integrated into the local view T^v , as per lines 57-60 in Listing 3.2. When node v receives a probe frame with link status information, it also updates its network status T^v . This mechanism is described in Listing 3.5. For each $T_{rx}^e = (e, q^e, n_{q,seq}^e)$ contained in the probe frame, v checks whether there is a corresponding tuple $T_{local}^e \in T^v$. If not, a new tuple is inserted, with the values of T_{rx}^e (line 7). Otherwise, the existing tuple is updated, if $T_{local}^e.n_{q,seq}^e < T_{rx}^e.n_{q,seq}^e$ (line 9), i.e. if the link status is more recent. In this case, b_{rep}^e is set to *true* if the received update is significant (as determined by thresholds $q_{diff\uparrow}$ and $q_{diff\downarrow}$ line 9).

Periodically, outdated link status information is removed from T^v , i.e. $T^v = T^v \setminus \{T_{local}^e \in T^v | c_{local}^v - T_{local}^e.t_{upd}^e > d_{term}\}$. This way, broken links are detected by each node, without having to be reported.

3.4. EAR – Efficient and Accurate Link Monitor

Though conceived independently, cTEx shares some attributes with Efficient and Accurate link-quality monitor (EAR), published in [KS06]. There are however differences in key aspects of the protocols, resulting in cTEx being the more accurate, more modular and also less complex approach. In this section, EAR will be presented, before we focus on the differences and commonalities between EAR and cTEx in the next section.

EAR is devised by its authors as a measurement framework for link quality in wireless mesh networks. The core objectives of its design are low overhead, high accuracy and awareness of asymmetric wireless links. In order to achieve these goals the authors emphasize four key characteristics of their solution.

- *Hybrid approach*: Depending on the availability of existing network traffic, EAR applies either a *passive*, *cooperative* or *active* measurement scheme.
- *Unicast-based uni-directional measurement*: As broadcasts use a fixed and low data rate, which is usually more robust against bit errors, accuracy (and by extension applicability) of measurements can be improved by using unicast measurements, as unicast is typically used for communication of application data. In addition, each direction of a link is measured separately.
- *Distributed and periodic measurement*: As link quality often varies over time, link quality is re-evaluated periodically.
- *Cross-layer interaction*: EAR consists of two main components, one that collects information on the success and failure of transmissions (called outer EAR), and one that processes these transmission statistics and performs link quality estimation at the network layer (called inner EAR).

During operation, EAR first measures link quality during its *measurement period* using the appropriate measurement scheme (either active, passive or cooperative) for the respective link. The duration of this measurement period as suggested by the authors is 9 s. After the measurement period, nodes enter the *update period*, where link qualities are re-calculated and the measurement scheme for each link for the next measurement period is determined.

3.4.1. Link Quality Metric

In EAR, link quality estimates are based on the frame delivery ratio. A link's quality value is re-calculated at fixed intervals, namely at the end of each measurement period. The link quality metric used in EAR is a simple EWMA metric, defined as follows:

3. cTEEx – Decentralized Topology Detection using a Link State Algorithm

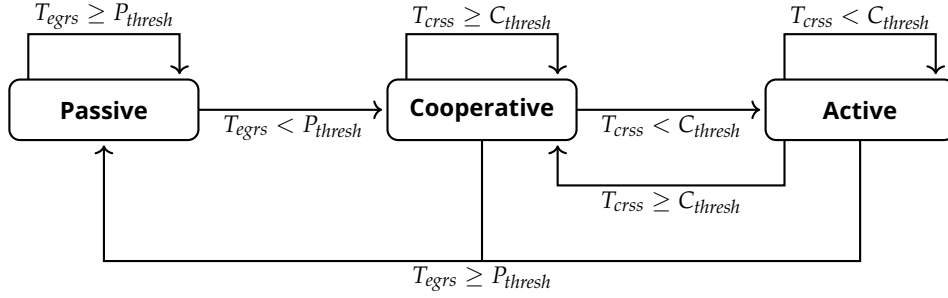


Figure 3.2: Transition mechanism between measurement schemes as applied in EAR. The direct transitions from cooperative and active modes back to passive take precedence. The necessary negation of $T_{egrs} \geq P_{thresh}$ for all other transitions is omitted for legibility. Source: Adapted from [KS06].

$$d_i = (1 - \alpha) \cdot d_{i-1} + \alpha \cdot \frac{n_s}{n_t} \quad (3.3)$$

Here, d_i is the smoothed delivery ratio for measurement period i and n_s and n_t are the number of successful transmissions and number of transmission and re-transmissions, respectively.

In addition to the link quality, EAR also collects statistics on the data rates used over each link. From this information, estimates of link capacity are derived, which can then be used in link quality metrics such as ETX [DCABM03] and ETT [DPZ04].

3.4.2. Measurement Schemes

EAR applies hybrid probing, i.e. depending on some conditions, links are evaluated using existing traffic only, or by actively sending dedicated probe messages. To keep overhead low, EAR uses the passive measurement scheme when possible and switches to cooperative mode in case there is no egress, but sufficient cross-traffic to the destination node. Only in absence of both, egress and cross-traffic, EAR operates in active mode.

Passive Mode

In passive mode, EAR uses only existing traffic between source and destination node of the link to be measured. The outer EAR component, which monitors the network device driver, collects statistics about successfully transmitted frames and manages counters for total number of transmission attempts and re-transmissions. After each measurement period, these values are communicated to

the inner EAR component, where Eq. (3.3) is used to derive current link cost, i.e. a link quality value. As is shown in Fig. 3.2, passive mode is used whenever the amount of egress traffic, captured by T_{egrs} , exceeds the threshold set for passive mode P_{thresh} .

Cooperative Mode

EAR's cooperative mode is applied when the amount of direct unicast traffic between two nodes is not sufficient (i.e. $T_{egrs} < P_{thresh}$, see Fig. 3.2). In this case, the source node v_{src} instructs the destination node v_{dst} of the link $e = (v_{src}, v_{dst})$ to be measured, to selectively overhear and count all frames sent by v_{src} to a third node v' for which traffic from v_{src} exceeds C_{thresh} . v' is selected by v_{src} such that the commonly used data rates for transmissions from v_{src} to v_{dst} and v' are similar. Node v_{src} uses a CooperateREQ message at the end of the update period to request cooperation from v_{dst} and to communicate v' . After the subsequent measurement period, v_{dst} sends results (i.e. the number n_s of successfully overheard frames between v_{src} and v') back to v_{src} using a CooperateREP message. v_{src} can now calculate the delivery ratio of frames for link e using the total number of frames n_t sent from v_{src} to v' and the number of frames n_s successfully overheard at v_{dst} .

Handling of retransmissions of frames is delicate in cooperative mode, as retransmissions (i.e. duplicate frames) are not received by the cooperative node (even in monitor mode). Hence, any frame for which the retry bit is set is ignored by EAR and does not increment counters n_t and n_s .

Furthermore, we note that in cooperative mode, the delivery ratio of non-acknowledged frames is measured. This is different from both active and passive modes, where the delivery ratio of only acknowledged frames is measured. Depending on the degree of asymmetry of links, this might cause significant inaccuracies during estimation of link quality.

Active Mode

In situations where there is neither sufficient egress- nor cross-traffic for a pair of nodes (i.e. $T_{crss} < C_{thresh}$), EAR transitions to active mode and starts sending dedicated probe messages. When existing egress and cross-traffic is insufficient between v_{src} and several of its neighbors, dedicated probe messages are sent between v_{src} and one other node v' only. The remaining nodes switch to cooperative mode by request from v_{src} and start overhearing the dedicated probe messages sent from v_{src} to v' . However, this optimization is only applied where data rates between all nodes are sufficiently similar.

In cases where links are idle (i.e. only little egress- and cross-traffic) and data rates are heterogeneous, only a subset of rates and links is probed per

3. *cTE*x – Decentralized Topology Detection using a Link State Algorithm

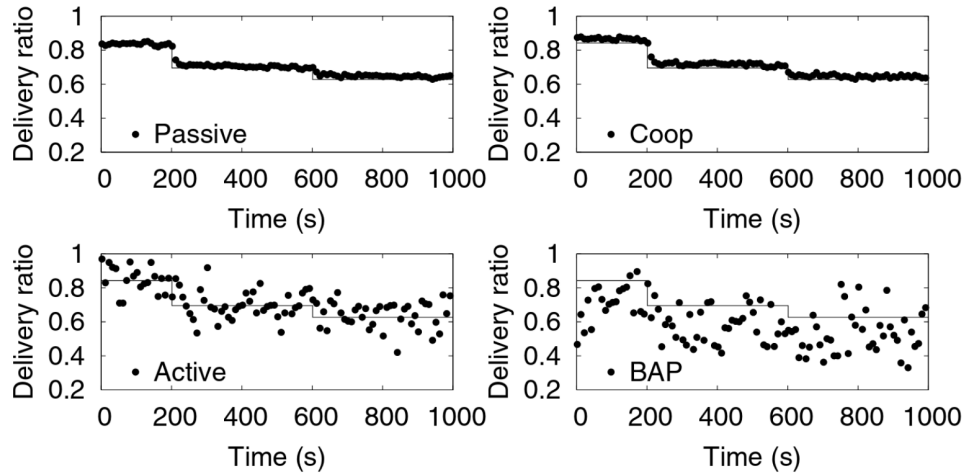


Figure 3.3: Performance of EAR in active, passive and cooperative mode versus Broadcast-based Active Probing (BAP) on dynamic link (A, B) of topology T1. Source: Adapted from [KS06].

measurement cycle. The lengths of intervals a single link is actively probed at depend on its activity and quality-variance, such that probing overhead incurred by stable or unused links is kept low.

3.4.3. Communication Overhead

The communication overhead incurred by EAR depends on the applied measurement scheme. With passive mode, the overhead is zero. Measurements are based on existing unicast traffic only, and no further messages are produced. Overhead strongly increases when EAR operates in active mode for some links, as explicit probe messages are created. Under the assumption of homogeneous data rates in the network however, each node may only apply active probing on a single link, as cooperative mode can be used to measure links to the remainder of its neighborhood. Existing unicast traffic between nodes that does not exceed thresholds P_{thresh} and C_{thresh} is not used by EAR.

3.4.4. Accuracy of EAR

The authors of EAR evaluate accuracy, scalability and awareness of link asymmetry of their protocol using simulation and real-world experiments. Results are compared to Broadcast-based Active Probing (BAP), an approach widely used e.g. for the adoption of link quality metrics Expected Number of Transmissions (ETX) and Expected Transmissions Time (ETT). In BAP, simple broadcast frames are sent at a fixed and low data rate of 2 Mbps. The delivery ratio of

these broadcast frames is then used to derive link quality.

Results show (see Fig. 3.3 for a performance plot copied from the original paper) that accuracy of EAR is very high in passive and cooperative modes. In active mode however, accuracy is lacking. When compared to the performance of BAP in this experiment, EAR's active mode is still an improvement. However, in scenarios where consistently accurate link quality estimates are a requirement, for instance for the application of r_{\min} -routing, the accuracy provided by EAR's active mode is insufficient.

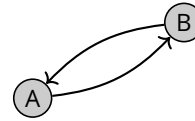


Figure 3.4: Topology T1.

3.5. cTEx and EAR

cTEx and EAR both are sophisticated protocols for topology detection. In addition, cTEx contains a component for the efficient distribution of the collected networks status information. EAR does not cover this functionality. In this section, commonalities as well as key differences between the two protocols are presented.

Both protocols use a metric similar to what is commonly known as PDR of unicast transmissions for link quality estimation. To maintain consistency with the terminology established in previous chapters of this thesis (see Section 2.1), we will refer to this metric as frame delivery ratio (FDR). We note that the authors of EAR refer to the same concept as PDR. As broadcast messages are limited to a low and fixed data rate, link quality estimates based on FDR of broadcast messages are usually inaccurate for unicast transmissions. The potential of opportunistic usage of existing network traffic to collect FDR statistics is leveraged by both protocols. This approach is key to maintain a low communication overhead for link quality estimation and also increases accuracy, as estimations are based on real network traffic. Links are modelled as asymmetric in both protocols. Furthermore, both protocols perform a continuous monitoring of detected links. This is in contrast with many other link quality metrics and topology detection mechanisms, where link quality is often evaluated only once in a dedicated measurement phase (e.g. in ETX [DCABM03]) or even reduced to a binary attribute based on the successful delivery of a single broadcast frame (e.g. in AODV [PR99]).

The key difference between both protocols lies in the respective measurement scheme applied. EAR operates in three different modes, depending on the amount of existing network traffic. In its *active* mode, unicast probing messages are transmitted to support FDR-based link quality estimation. In case of sufficient egress network traffic, EAR switches to *passive* mode, where only

3. cTEx – Decentralized Topology Detection using a Link State Algorithm

existing transmissions are used for estimation of link quality. These transmissions may be triggered by actions in other network management layers or by the application. As EAR uses the sender as the measuring node for the quality of outgoing links (in contrast with cTEx where the receiver estimates incoming links), an additional *cooperative* mode is necessary to optimize for the case where a measuring node has no egress traffic via the link that is to be measured, but has traffic to other nodes. In this situation, the destination node of the link to be measured is sent a CooperateREQ message and starts overhearing the traffic sent by the measuring node and collects statistics, which are then reported back to the measuring node where the link quality estimation is done. Application of cooperative mode significantly reduces overhead, as without it, any links with insufficient direct unicast traffic would have to be measured in active mode. In cooperative mode however, the un-acknowledged delivery ratio is measured. As the measurement mode applied by EAR may change after every measurement cycle, delivery ratios of un-acknowledged frames (from cooperative mode) are mixed deliberately with delivery ratios of un-acknowledged frames (from active and passive modes), which may introduce significant inaccuracies.

In cTEx, the receiving node is responsible for collecting transmission statistics and performing link quality estimation. This is achieved by operating each node’s transceiver in monitor mode at all times. With EAR, monitor mode is only necessary when a node is asked to cooperate. Due to cTEx’s frame generation mechanism, a scenario where network traffic is insufficient for link quality estimation does not occur and a cooperative mode is not necessary. cTEx always measures delivery ratio of un-acknowledged frames.

3.6. Simulation Experiments

In this section, cTEx is evaluated in simulation as well as real-world experiments. The analyzed performance metrics are accuracy of link quality measurements, latency of topology distribution and overhead incurred by the protocol. In the experiments on accuracy, cTEx is compared to EAR. In a last set of experiments, cTEx is run on real hardware to illustrate that the protocol works as intended in real world scenarios in addition to simulated environments.

3.6.1. Simulator Setup

Simulations are done in ns-3, a sophisticated, “discrete-event network simulator for internet systems”, as per its authors. The input topologies for the simulations are provided to the simulator as a list of tuples, where each tuple consists of three values: source node, destination node and raw reliability ($r_{raw,e}$ for a link e). The distinction between r_{raw} , as a raw statistical property of a simulated link, and

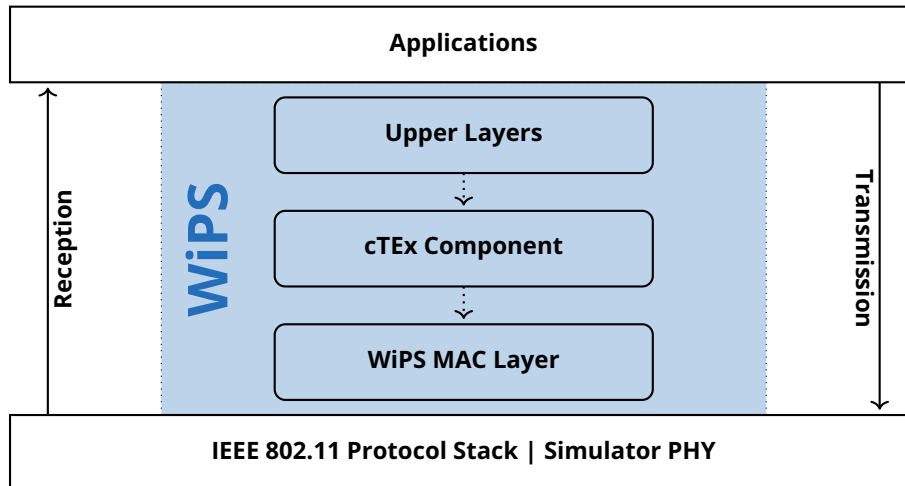


Figure 3.5: Architecture of the WiPS framework

the link's reliability, as the output of the link quality metric is important. In the specification of an input topology the simulator runs on top of, it is indeed just a statistical property of the link itself and directly corresponds to the probability of a successful communication attempt r_e of the network model (see Section 2.2 of Chapter 2). During operation of that link in a network and through the evaluation of the statistics collected during operation, the link is assigned a dynamic link quality value, which in case of cTEEx is determined using OB-EWMA in these experiments.

The described input topology data structure is then used to initialize a custom ns-3 propagation model. A propagation model in ns-3 is responsible for the calculation of path loss (also called path attenuation, for the attenuation a wave receives while it propagates. In this context, path refers to a single link). The higher the path loss, the lower is the received signal strength. To understand how the configured raw reliability of a link, given as a decimal number between 0 and 1.0, manifests itself in the simulator it is useful to follow the way of a single transmission frame through the layers of the simulator.

After a packet is created in the application layer, it passes through the layers of WiPS, see Fig. 3.5 for an overview of the WiPS architecture. The lowest WiPS layer is the WiPS MAC layer. There, the frame is constructed and passed on to the PHY layer of the simulator. For each simulated node, there is a virtual network device. Once the frame reaches this device of node v , the configured propagation model(s) become important. For every node $v_i \in V \setminus \{v\}$ in the network, the configured raw link reliability $r_{raw,(v,v_i)}$ is obtained. Then, for each node v_i , a random number between 0 and 1.0 is drawn from a uniform distribution. If this number is larger than the configured raw reliability $r_{raw,(v,v_i)}$, the frame's path loss is set to infinity, which prohibits reception of this frame by

3. *cTEx* – Decentralized Topology Detection using a Link State Algorithm

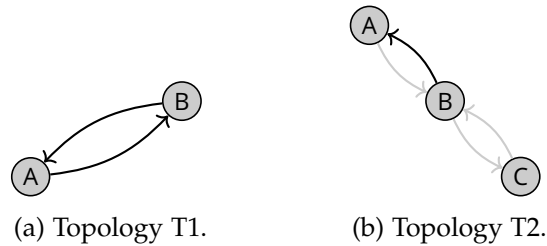


Figure 3.6: Topologies T1 and T2, as described in [KS06]. Black lines represent links with regular traffic (consisting of implicit probe frames), grey lines represent links without regular traffic.

node v_i . If the number drawn is smaller or equal to the configured raw reliability, no path loss is added.

It is important to note, that even if the dice roll described above succeeds and no path loss is added, the frame may still fail to be received due to collisions. At this point it becomes clear, why distinguishing between the raw reliability and the link quality as determined through the link quality metric is important: the raw reliability is the configuration for the result of the dice roll, in aggregation not all too different from the distance between two nodes, whereas the metric's link quality value is a dynamic attribute assigned to a link during operation of the network, which is influenced by the configured raw reliability as well as other factors like history or collisions.

This approach abstracts from the geographical positions of nodes. It might occur, that there is no possible mapping of all nodes onto a 2-D plane, where the distance between any two nodes is proportional to the raw reliability of their link. However, in the real world, path loss is not a function of only the distance between two nodes, either. Interference and obstacles of different materials influence the loss added to a frame during transit. In our case, this abstraction from geographical positions should not be detrimental to the applicability of results to real-world scenarios and simplifies the setup of experiments.

3.6.2. Accuracy of Link Quality Measurements

In several simulation experiments, we investigate the accuracy of link quality as measured with *cTEx* and compare it to the accuracy achieved by Efficient and Accurate link-quality monitor (EAR) [KS06]. For this, we have recreated experiments reported in [KS06], for topologies T1 and T2 shown in Fig. 3.6.

In topology T1, regular unicast traffic consisting of implicit probe frames in both directions is available to determine link quality, which causes EAR to operate in passive mode (see Section 3.4 for an introduction of EAR). In topology T2, regular unicast traffic is only sent from node B to node A, causing EAR to

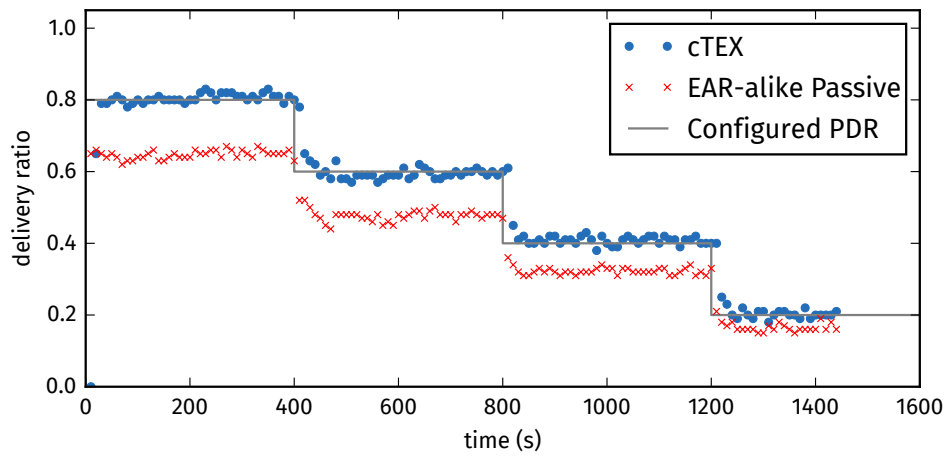


Figure 3.7: Delivery ratio for link B to A over time as measured by cTEx and a passive probing implementation similar to EAR's for topology T1.

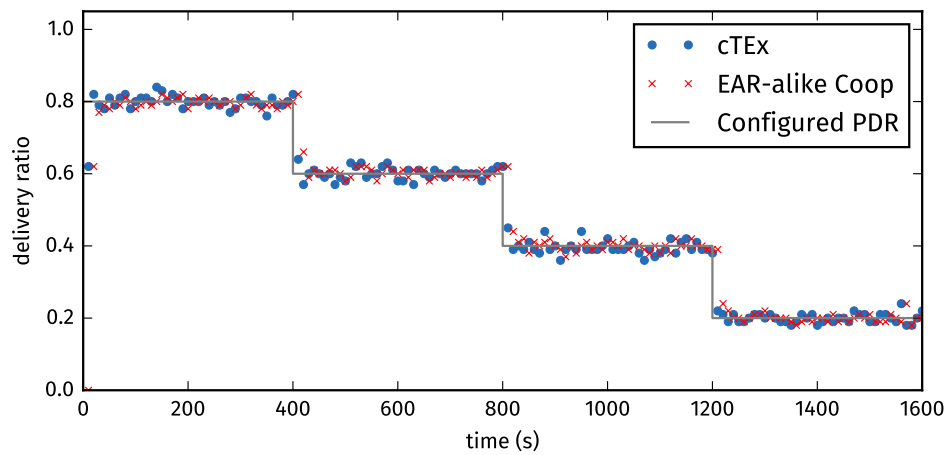


Figure 3.8: Delivery ratio for link B to C over time as measured by cTEx and a cooperative probing implementation similar to EAR's for topology T2.

adopt a cooperative mode with node C. Regular traffic is created by UDP-like data streams with a frame size of 35 bytes and a frame rate of 91 fps, yielding 25.4 kBit/s. During the experiment, the configured raw reliability of all links starts with 0.8 and drops by 0.2 every 400 s.

Fig. 3.7 shows the results of the experiment for topology T1 for the link from B to A. Here, the FDR measured with cTEx is always very close to the configured FDR, i.e. the measured link quality is highly accurate. However, the FDR measured with an EAR-alike implementation is significantly lower. This results from an important difference between cTEx and EAR: cTEx always measures incoming links, while EAR measures outgoing links. To detect frame loss, cTEx uses consecutive probe sequence numbers (see Sections 3.3.3 and 3.3.4), which makes feedback from receiving nodes obsolete. However, EAR requires such feedback, i.e. ACKs. It follows that EAR measures the ratio of frames that are both successfully received *and* ACKed, yielding a reduced FDR. We point out that this reduced accuracy of EAR cannot be compensated by increasing the rate of probe frames.

Fig. 3.8 shows the results of the experiment for topology T2 for the link from B to C. Here, the FDR measured with cTEx is again very close to the configured FDR, and so is the FDR measured with EAR. The reason is that in this experiment, EAR operates in cooperative mode, which uses an accumulated ACK at the end of a measurement cycle, i.e. a `CooperateREP` message from node C, but otherwise operates similar to cTEx for this particular link. Nevertheless, the accuracy for the link from B to A measured by EAR is similar to the one shown in Fig. 3.7. However, with EAR, nodes may change measurement modes after every measurement period. Hence, delivery ratios acquired from cooperative mode may be mixed with ratios acquired in active and passive mode. From Fig. 3.7 we observe that this it makes a significant difference, whether only acknowledged or also un-acknowledged frames are counted. Mixing both, introduces inaccuracies in the resulting link quality proportional to the asymmetry of the respective link. The authors of EAR neither propose a solution to this deficiency, nor acknowledge the problem in the original paper [KS06].

3.6.3. Topology Distribution Latency

Several simulation experiments on different topologies were carried out to evaluate the topology distribution latency. As topology distribution is not covered by EAR, we cannot compare cTEx and EAR for this series series of experiments. The results are illustrated on a lattice topology with 36 nodes as seen in Fig. 3.9, as the behavior is easiest to follow in a regular topology with limited density.

In this experiment, the raw reliability of all links is configured as 0 at the start of simulation. Over a period of 10 minutes, the configured raw reliability increases linearly from 0 to 1.0 (see the solid black lines in Figs. 3.10a and 3.10b).

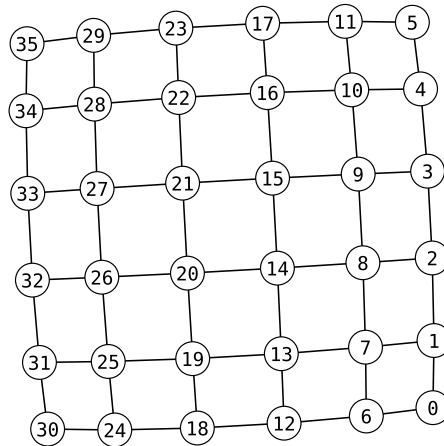


Figure 3.9: The graph used during the experiments, 36 nodes, 120 edges (due to directedness).

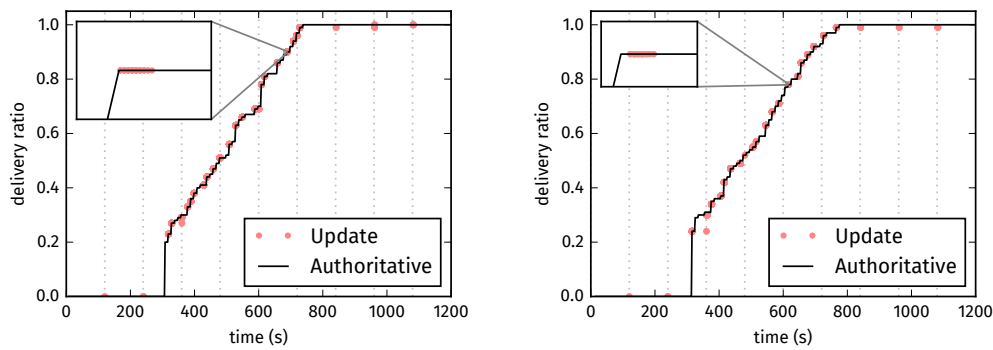
This continuous change is tracked by the link detection component of cTEx and the current quality values are distributed in the network. As each link’s quality constantly changes in a controlled way, the point in time a specific quality value is updated in a distant node (the red dots in Figs. 3.10a and 3.10b) can be used to estimate the topology distribution latency. The further away the dots are from the black line – in horizontal direction – the higher the latency.

It is important to note that topology distribution utilizes the same links we are evaluating, and is therefore also affected by the very low frame delivery ratio in the early stages of the experiment. Only a few frames containing topology information are received correctly, as the configured raw reliability of all links is still low. The frame delivery ratio improves as the whole network becomes more reliable and consequently, link status updates travel faster through the network.

For this experiment, cTEx was configured to embed a link’s status into a frame at least every 120 s (marked by the dotted vertical lines in Figs. 3.10a and 3.10b). Additional reports are triggered whenever a link’s quality changes by at least 2.5% in either direction from the last reported value. The configured link metric was OB-EWMA [MG21] as outlined in Section 3.2.

We observe that the latency for topology distribution is generally low. Apart from a few exceptions in the early phase of the simulation (over still unreliable links), link information propagates through the network of 36 nodes in a few seconds. In a network with reliable links, a piece of information on a link’s quality propagates one hop further through the network at least with the configured frequency for probe frames. With additional traffic even faster: If there is an average of only 10 fps per node, the link information propagates at 10 hops

3. cTEx – Decentralized Topology Detection using a Link State Algorithm



(a) Distribution latency of link quality of the link between nodes 0 and 1, as measured at node 1. (b) Distribution latency of link quality of the link between nodes 21 and 15, as measured at node 15.

Figure 3.10: Distribution latency of topology information in a 36 node graph. The dots in Figs. 3.10a and 3.10b mark the points in time other nodes update their topology w.r.t. the link quality on the y-axis. The dashed vertical lines mark a full topology dump event. For better readability, two excerpts of the diagrams are enlarged.

per second in every direction.

Link status information also propagates fastest along sequences of good links. As these links are also the most likely ones to be used during routing, the network reaches a usable state quickly.

3.6.4. Overhead

The maximum overhead for link detection with cTEx is linear in the number of nodes in the network. Each node v creates a maximum number of explicit probe frames based on the configured minimum probe rate $r_{pf,min}$. This maximum number of explicitly created probe frames is reduced by the number of implicit probe frames sent by node v , i.e. by the egress traffic generated by protocol layers above cTEx, yielding the actual overhead.

This is substantially different from EAR [KS06], which considers the traffic on each outgoing link separately. For EAR to operate in passive mode for an outgoing link (v, v') , node v has to produce enough egress traffic to node v' , i.e. $T_{egrs} \geq P_{thresh}$ (refer to Section 3.4 for details). If there is not enough egress traffic on any outgoing link, then node v chooses some link (v, v') for which it operates in active mode, and operates in cooperative or active mode for all other outgoing links. This implies that even if the sum of all egress traffic on outgoing links is above P_{thresh} , node v may have to operate in active mode on one or more links, which yields higher overhead compared to cTEx. Even worse, different

Table 3.5: Parameters of Overhead Experiments

Parameter	Value	Description
s_{pf}	16 bit	Net size of a probe frame, i.e. additional field <i>txCtr</i> (2 bytes)
d_{pf}	251 ms	Maximum time between subsequent probe frames, so $r_{pf,min} = 3.98 s^{-1}$
$s_{linkDesc}$	40 bit	Size of an encoded link status tuple T_{rep}^e
d_{rep}	120 s	Maximum time between two reports of a link
n_{nodes}	36	Number of nodes
n_{links}	120	Number of links

from cTEX, the overhead caused by the active mode on one link is not reduced by combined egress traffic of all outgoing links. Finally, if EAR does not record sufficient cross traffic on a link operated in cooperative mode, it switches to active mode for this link. This implies that the maximum overhead of EAR with respect to the number of probe frames is linear in the number of links in the network, and not in the number of nodes, as in case of cTEX.

It should be mentioned that different from EAR, cTEX appends some status data to each frame to exchange the network status among nodes, a functionality that is not addressed by EAR. This slightly increases the size of frames, which may reduce the delivery ratio. As the size of the status data is usually small, we consider this a sensible tradeoff.

Another important aspect when analyzing the overhead of a link estimation protocol is the amount of data caused by the protocol's management traffic. An upper bound for the total number of bits exchanged during link estimation and distribution with the time-triggered strategy (see Section 3.3.5) in cTEX is given by the following equations:

$$o_{det} = s_{pf} \cdot d_{pf}^{-1} \quad (3.4)$$

$$o_{dist} = s_{linkDesc} \cdot n_{links} \cdot d_{rep}^{-1} \quad (3.5)$$

$$o_{total} = (o_{det} + o_{dist}) \cdot n_{nodes} \quad (3.6)$$

A description of each parameter along with the value set is provided in Table 3.5. o_{det} and o_{dist} is the respective overhead in bps for detection and distribution per node. o_{total} gives the total overhead in bps for detection and distribution for the whole network.

In Fig. 3.11, Eq. (3.6) is experimentally validated, with parameters set as shown in Table 3.5. During the experiment, each node sends an explicit probe

3. cTE_x – Decentralized Topology Detection using a Link State Algorithm

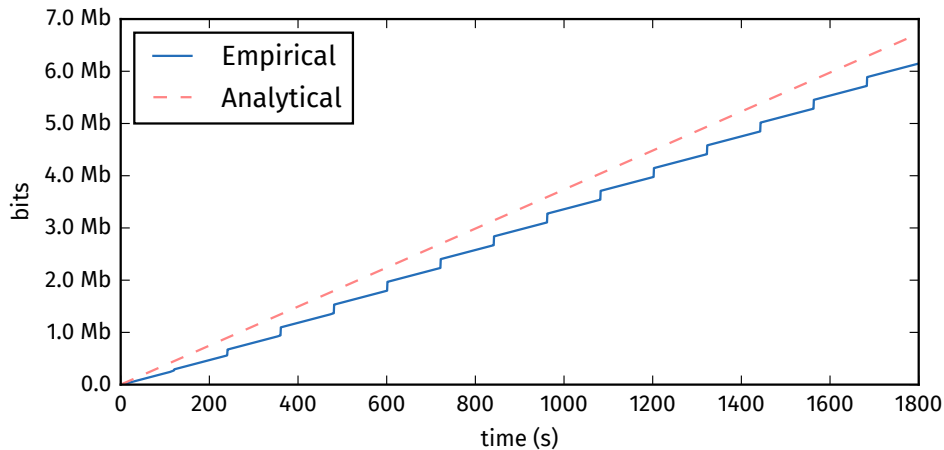


Figure 3.11: Comparison of calculated to measured overhead in bits. Overhead was measured on a 36 node graph with 120 links (due to directedness).

frame every 251 ms. To calculate the overhead o_{det} , we have assumed that regular unicast 802.11 WiFi frames were available for this purpose. These frames already carry sender address snd , transmission rate r_{tx} , and frame size s_f . Thus, the only extra field to be added is the probe sequence number $txCtr$, which requires 2 extra bytes as overhead s_{pf} . In addition, the topology is distributed by each node every 120 s, yielding o_{dist} .

The analytical line in Fig. 3.11 represents the upper bound and is determined by inserting the parameter values in Table 3.5 into Eqs. (3.4) to (3.6). The empirical line results from a simulation experiment using the same parameter values, with periodic topology distribution only (i.e. event-triggered link status reporting is turned off). The upper bound is not hit by the empirical results, because link status distribution does not always cover all links, especially in the early phase. Overhead of event-triggered link status distribution is bounded by the number of links, the interval the set T^v is updated by a node v , and the maximum frame rate.

As shown in Fig. 3.11, a network of 36 nodes and 120 links, operated over 30 minutes, causes a total accumulated overhead of slightly more than 6 Mbit, which is an average overhead of about 28.5bps per link to estimate and distribute link quality in this experiment. We note that this kind of assessment cannot be made for EAR, as EAR does not address network status exchange.

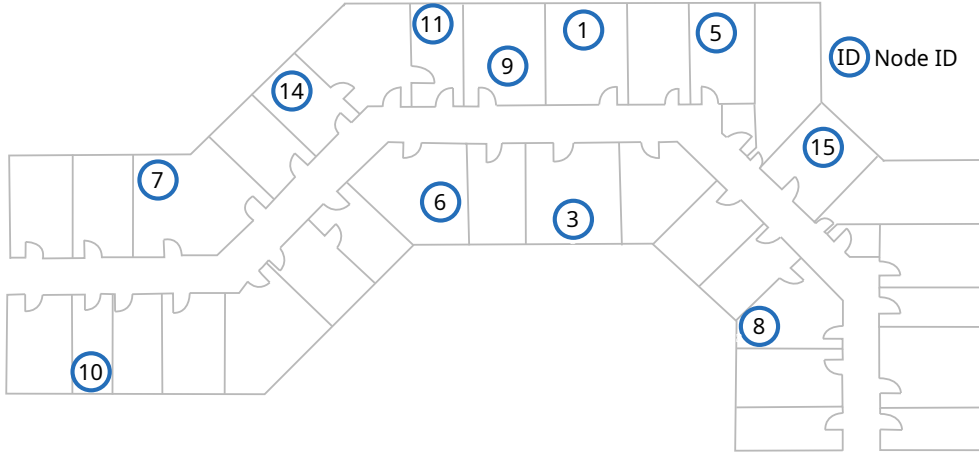


Figure 3.12: Node placement of our 11-node real-world testbed.

3.6.5. Testbed Experiments

To verify that cTEx also works in real-world scenarios, several experiments were carried out on our 11-node testbed. The nodes are placed in an office building and are permanently exposed to interference from several WiFi access points and client devices such as laptops and smartphones. The WiFi chipset used in all the nodes is Atheros AR9280, the nodes run Linux 5.3 with an unmodified ath9k wireless chipset driver. The placement of the nodes in the building is displayed in Fig. 3.12.

The cTEx framework was executed on the nodes for an hour at a time, while the experiment data was logged to a database and evaluated offline. The configuration parameters are listed in Table 3.6. The results from these experiments meet the expectations raised by the results from our simulations.

In Fig. 3.13, the resulting quality values over time for two exemplary pairs of

Table 3.6: Parameters of Real-World Experiments

Parameter	Value	Description
$metric()$	OB-EWMA	Link quality metric
d_{fs}	12 s	FDR slot duration
d_{pf}	251 ms	Maximum time between subsequent probe frames, so $r_{pf,min} = 3.98 s^{-1}$
q_{diff}	0.025	Minimum difference of reported and current link quality to trigger a report
d_{rep}	120 s	Maximum time between two reports of a link

3. cTEx – Decentralized Topology Detection using a Link State Algorithm

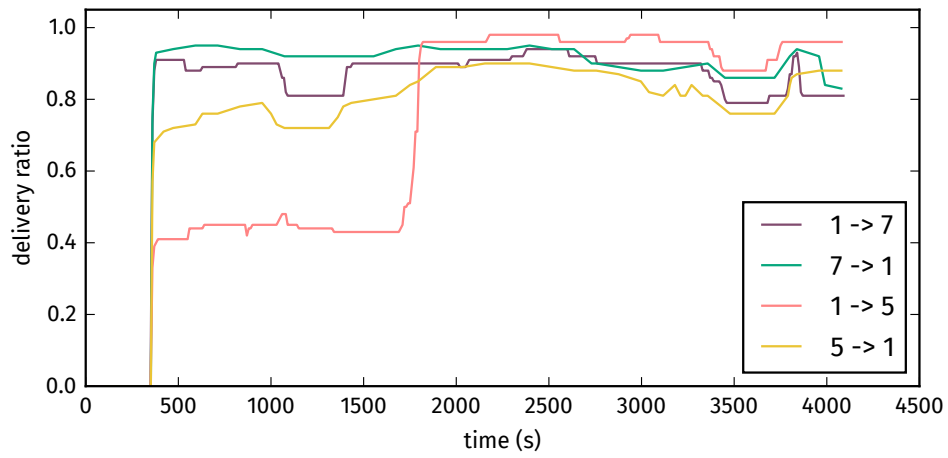


Figure 3.13: Results from one of the testbed experiments for a selection of links.

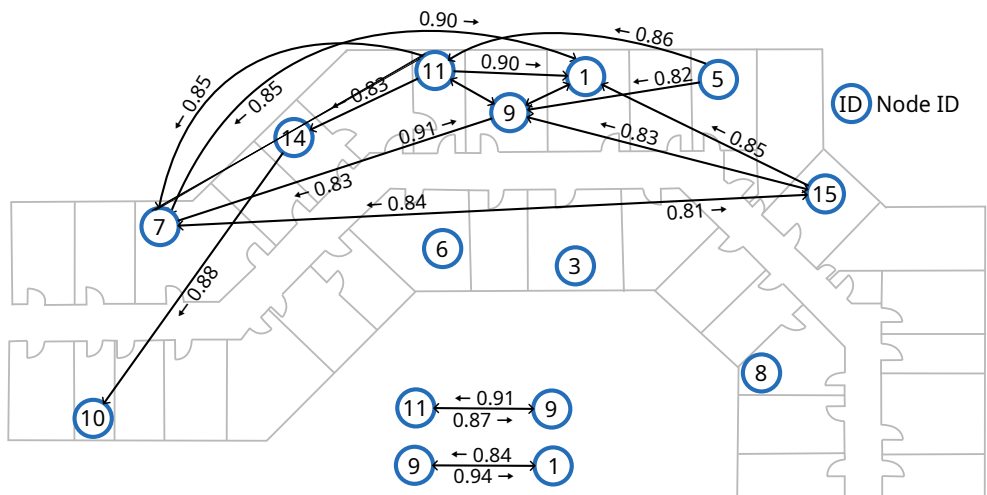


Figure 3.14: Snapshot view of the network topology as detected by cTEx. Links with quality < 0.8 are not displayed.

links are depicted. The results show that real-world links are inherently asymmetric, which is reflected by cTEx. As an example, compare the two links between nodes 1 and 5. The general increase of link quality after about 1800s into this experiment can be attributed to lunchtime, where the building is less crowded. These characteristics were observed consistently in several runs of these experiments.

Fig. 3.14 shows a snapshot view of the network topology detected by cTEx. The experiment was run on a subset of 8 testbed nodes, with cTEx configured according to Table 3.6. For better readability, only links with a quality value $q^e > 0.8$, as determined by cTEx at this point in time, are depicted. Together with links not shown, the nodes form a network, i.e. between any pair of nodes, there is a path of communication links.

We note that different from simulation experiments, link qualities could not be controlled in testbed experiments. This is due to the office environment where the testbed experiments were conducted, with other nodes using the same frequency spectrum. Also, to assess the accuracy of link quality measurements in the testbed experiments, feedback about the successful operation of reliability-constrained routes would be needed, which was not available. However, given the high accuracy of link quality measurements in our simulation experiments, we are confident that cTEx will perform well in real-world scenarios, providing high-quality network status information for routing protocols and clustering algorithms. The real-world performance of cTEx in the context of r_{\min} -routing will be evaluated in the following chapter, where a complete routing protocol is introduced.

3.7. Related Work

The core of cTEx is its link probing algorithm. Network airtime is precious, so the overhead messages caused by a link probing algorithm should be minimal, especially in high load situations, while still yielding high-quality status data. Hence, several protocols use existing network traffic to probe links. This strategy is often called passive mode, in contrast to an active mode, where additional frames are created for link probing. In [KS06], an Efficient and Accurate link-quality monitor (EAR) is proposed as a sophisticated link measurement framework for multi-hop wireless mesh networks. EAR adaptively employs three measurement schemes and operates in monitor mode to opportunistically use existing unicast network traffic to estimate link quality by calculating the packet delivery ratio (PDR). Additional probing packets⁵ are only created when necessary. However, two of EAR's three measurement schemes calculate the PDR

⁵This is the term used in the original work. With respect to the terminology established in this thesis, the correct term would be *frame*.

of acknowledged packets, whereas its cooperative scheme does not. In combination with the adaptive switching between schemes, this potentially leads to inconsistent measurements. Links for which there is neither sufficient egress nor cross-traffic are probed separately in EAR’s active mode, which comes with a considerable message overhead.

The concept of overhearing existing unicast traffic for link probing is also employed in more recent work. In [LCZH17], the concept is applied in Wireless Sensor Networks (WSNs) by counting wake-up frames. The authors evaluated their protocol as an integration into the Four Bit Wireless Link Estimator [FGJL07], a popular link estimation framework with wide applicability from Zigbee WSNs to 802.11 WiFi networks.

The choice of link metric heavily depends on the intended application of the network, to provide enough flexibility for a wide range of applications. *cTEx* does not assume a specific link metric. Any metric based on packet (or frame) probing data should be compatible. This includes many software-based link quality estimators [BKM⁺12] such as ETX [DCABM03] and most of its variants and implementations [TMK15, DPZ04, JEH⁺12, ZSA10] as well as Four-Bit [FGJL07] or WMEWMA [WTC03].

In related works, the mechanisms for link probing, determination of link quality according to a link metric, and distribution of status information are usually designed as part of a routing protocol, e.g. as part of 802.11s with Hybrid Wireless Mesh Protocol (HWMP) [IEEE11]. HWMP is a routing protocol with proactive and reactive components, where airtime multiplied by the expected number of transmissions (ETX) is used as link metric, and routing information is distributed by flooding. It is left, however, unspecified how the error rate used to define ETX is determined.

3.8. Conclusion

In this chapter *cTEx* was introduced, a protocol for the efficient detection of links and distribution network status information. *cTEx* uses existing network traffic to reduce overhead incurred by the protocol and to improve accuracy of the derived link status information. With respect to the particular link metric employed as part of *cTEx*, many link metrics based on packet (or frame) delivery ratio data can be applied. The reference link quality metric of *cTEx*, Outlier Bounded Exponential Weighted Moving Average (OB-EWMA) provides accurate, normalized and conservative link quality estimates, especially well-suited for r_{\min} -routing algorithms.

As *cTEx* shares similarities with a protocol from literature called Efficient and Accurate link-quality monitor (EAR), various aspects of *cTEx* and EAR were compared throughout this chapter. In conclusion, *cTEx* can be viewed as supe-

rior solution to the problem of acquiring link status information, as it is more efficient with respect to overhead, simpler in its implementation with respect to the link estimation mechanism and more accurate with respect to the derived link quality values.

In addition, cTEx features mechanisms to not only detect links, but also distribute this acquired information among all nodes of the network. By appending small chunks describing a link's status, either periodically or on-demand, into network frames, network status information is distributed in the network in a quick and bandwidth-efficient manner. Both, distribution speed and bandwidth efficiency are supported by the requirement that cTEx nodes operate their transceiver in monitor mode, in order to receive all frames sent by nodes in communication range, not only frames with the respective node's destination address.

Though cTEx is intended as a configurable and universal topology exploration protocol, its design was heavily influenced by the objective of providing sufficiently accurate network status information for r_{\min} -routing algorithms. Hence, in the following chapter, we will introduce dR_{\min} -routing, a complete, decentralized routing protocol for wireless ad-hoc networks that supports discovery, operation and maintenance of r_{\min} -routes (i.e. routes of a minimum end-to-end reliability with statistical guarantee), while relying on high quality network status information provided by cTEx.

4. dR_{\min} -Routing – Decentralized Discovery, Operation and Maintenance of Bandwidth-constrained Routes

In the introductory section of this thesis four core functionalities for a communication protocol capable of guaranteeing a statistical minimum reliability level for end-to-end communication were identified. The functionalities stated are:

1. Route discovery
2. Route operation
3. Route maintenance
4. Collection and distribution of network status information

The r_{\min} -routing algorithm, introduced in the Chapter 2, covers the core of the route discovery process. This includes selecting optimal route candidates and assigning transmission budgets to each link along the route. In Chapter 3 of this thesis, we introduced the cTEX protocol for network topology detection and distribution. With cTEX, the fourth functionality is addressed. This leaves functionalities for route operation and maintenance as well as the decentralization of route discovery open.

In this chapter dR_{\min} -routing [KG22] is introduced. dR_{\min} -routing implements the r_{\min} -routing route discovery algorithm in a decentralized manner and adds mechanisms for route operation and maintenance. It performs a distributed, reactive search along a route candidate preselected with r_{\min} -routing to remove imprecisions of the locally available network status before making a final route selection. While routes are operated, dR_{\min} -routing monitors them and performs different kinds of dynamic route repair actions to maintain route reliability in order to cope with varying link reliabilities. By utilizing the cTEX-protocol, dR_{\min} -routing becomes a functionally complete routing protocol capable of handling route discovery, operation, and maintenance while meeting the minimum route reliability level set by the application.

4. dR_{min} -Routing

The wireless communication technology we target with dR_{min} -routing is WiFi, for its wide availability as an off-the-shelf product at affordable prices. In addition, in its latest iterations, WiFi supports very high link rates of up to 9.6 Gbps. On the other hand, only a best effort Quality of Service (QoS) level is supported by the IEEE 802.11 group of wireless communication standards that form the foundation of WiFi. Strictly speaking, this is valid for WiFi in DCF mode only. In PCF mode, QoS guarantees are supported. This facilitates operation of reliability-constrained routes. In wireless mesh networks however, PCF mode is unusable, as the mode restricts a network to single-hop communication.

This chapter is structured as follows: Following this introduction, we present dR_{min} -routing in Section 4.1. Similar to other protocols covered in this thesis, this section on dR_{min} -routing also includes a complete, semi-formal specification of the protocol. In Section 4.2 we assess dR_{min} -routing and compare it to OLSR, an often cited routing protocol from the literature, by extensive simulation experiments in Section 4.3. Experiments in a real-world testbed are out of the scope of this thesis, however, a summary of prior considerations for an evaluation of dR_{min} -routing on hardware is provided in Section 4.4. Related work is presented in Section 4.5. In the last part of this chapter, Section 4.6, a summary and discussion of the results are provided.

4.1. The dR_{min} Routing Protocol

dR_{min} -routing is a decentralized protocol for reliability-constrained routing in wireless ad-hoc networks. The objective of dR_{min} -routing is to discover, maintain, and operate routes with specified minimum route reliabilities. In accordance with [KMG19] and Chapter 2 of this thesis, we call this minimum route reliability r_{min} . It is defined as the end-to-end packet delivery ratio of an established route, which is measured over a moving time interval of specified length d_w . The reliability target is considered met, while there exists no time interval of duration d_w , where the ratio of received to transmitted packets over a route is less than r_{min} . To achieve this objective, the following measures are taken:

- Network status information about network topology and statistical link reliabilities is collected, updated, and exchanged (Section 4.1.1).
- Based on locally available network status information, the source node of a route preselects a route candidate providing a specified minimal route reliability (Section 4.1.2).
- Starting with the preselected route candidate, an initial route is determined by a distributed search, which removes imprecisions of the locally available network status (Section 4.1.3).

- To maintain route reliabilities, routes are monitored during operation, and several kinds of route repair actions are taken if the reliability target is (about to be) missed (Section 4.1.4).

4.1.1. Network Status Information

For the discovery of routes p with a specified minimal end-to-end reliability target r_{\min} , dR_{\min} -routing needs high quality information about network topology and link reliabilities. In a dynamic environment, this requires permanent monitoring and adaptation of the set of nodes, links, and link metric values, as well as the exchange of collected status data. For dR_{\min} -routing, we model a network as a graph $G = (V, q)$, where V is a set of vertices and $q : V \times V \rightarrow \mathbb{R}_{0,1}$ is a statistical single-hop reliability metric. This definition deviates from the network model used in previous chapters, as q is not defined as the success probability of a communication attempt, but as the estimated reliability value that is the output of a link quality metric. Hence, we will occasionally refer to $q(v, v')$ simply as the reliability¹ of edge (v, v') . This implicitly defines the set of edges $E =_{df} \{(v, v') \in V \times V \mid q(v, v') > 0\}$.

To detect and exchange network status information, we have adopted cTEx, a configurable topology explorer for wireless ad-hoc networks introduced in the previous chapter and published in [KG21]. For link detection, nodes transmit unicast probe frames with sequence numbers. Since nodes operate in monitor mode, all direct neighbors may receive and process these probe frames. By checking sequence numbers, nodes can detect frame loss without providing feedback to senders, which serves as a statistical basis for a sophisticated link metric q . To reduce bandwidth consumption, regular unicast frames are also interpreted as probe frames. Furthermore, nodes create and update their local view of the global network status by exchanging and processing network status data.

As link metric q , we have adopted Outlier Bounded Exponential Weighted Moving Average (OB-EWMA) [MG21]. OB-EWMA is based on frame delivery ratio (FDR), which makes it generally suitable for reliability-constrained routing. Furthermore, it reflects dynamic changes of FDR to cope with varying link reliability, is sufficiently stable to keep route changes manageable, and provides conservative estimates to avoid frequent route failures. For the link modelled as edge e and parameters $\alpha, p \in \mathbb{R}_{0,1}$, $q^e = \text{OB-EWMA}_{\alpha,p}^e(n_{mw}, s_{mw})$ is the link metric value of sliding window n_{mw} of size s_{mw} , taking the history of link metric values into account, where α is the weight that is given to the history, and p

¹Strictly speaking, this is a deviation from the terminology established in Section 2.1. As both, the term reliability and the value of $q(v, v')$, which is the output of a link quality metric, correspond to the MAC layer abstraction level we consider this deviation justified.

4. dR_{min} -Routing

defines the set of downward outliers. For more details on OB-EWMA, refer to Section 3.2 in Chapter 3 of this thesis.

In the description of the dR_{min} -routing algorithm in form of pseudocode in Section 4.1.5, the topology view (which is provided by cTEx) of a node modelled as vertex v is captured as the set T^v , which contains a tuple $T^e = (e, q^e)$ for each link known to v .

4.1.2. Preselection of a Route Candidate

When a route with a specified reliability target r_{min} is requested, the source node preselects a route candidate based on locally available network status information. In dR_{min} -routing, we have adopted r_{min} -routing (introduced in Chapter 2 and [KMG19]), a centralized algorithm for the discovery of reliability-constrained routes, for this purpose.

Given the network model $G = (V, q)$ and the derived set of edges E (see Section 4.1.1), the set $P \subset E^*$ denotes the set of all cycle-free paths in G , i.e. the set of all unicast routes. Assuming statistical independence of transmission events², the reliability q^p of a path $p \in P$ is given as the product of the reliabilities q^e of edges e along p .

With increasing length of a path p in hops, the product q^p decreases, if reliabilities q^e are smaller than 1. This reduces the chances of discovering routes that satisfy r_{min} . Here, r_{min} -routing improves perceived link reliabilities by well-directed retransmissions, thereby increasing the number of route candidates. Assuming statistical independence of transmission events, the success probability of up to n_{max}^e (re)transmissions of a frame on the link modelled as edge e is increased to $q_{n_{max}^e}^e = 1 - (1 - q^e)^{n_{max}^e}$, thereby increasing the reliability $q_{n_{max}^e}^p$ of a path p to the product of reliabilities $q_{n_{max}^e}^e$ of edges e along p . For a pair of vertices $s, d \in V$, r_{min} -routing determines a candidate path p from s to d such that (i) $q_{n_{max}^p}^p \geq r_{min}$ and (ii) the maximum number of communication attempts n_{max}^p on p , i.e. the sum of n_{max}^e of the edges along p , is minimal, provided there exists a path that corresponds to a sequence of sufficiently strong links and is sufficiently bandwidth-efficient.

The calculations described above for preselection of a route candidate are captured as $Rmin(T_v, s, d, r_{min})$ in the description of the dR_{min} -routing algorithm provided in Section 4.1.5.

To illustrate effective reliability of links and corresponding routes, Table 4.1 lists the respective reliability values for different levels for n_{max}^p , where $p = \langle v_1, v_2, v_3, v_4 \rangle$ as depicted in Fig. 4.1. In each row of Table 4.1, n_{max}^e and $q_{n_{max}^e}^e$ are underlined for the edge that is assigned the additional transmission. The rela-

²In practice, this assumption is too strong; therefore, dR_{min} -routing applies a conservative link metric q and further measures to cope with a certain degree of deviation from this property.

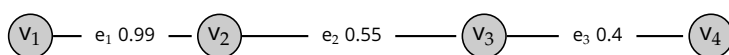


Figure 4.1: Example topology

Table 4.1: Effective reliability of edges in Fig. 4.1

n_{max}^p	$n_{max}^{e_1}$	$q_{n_{max}^{e_1}}^{e_1}$	$n_{max}^{e_2}$	$q_{n_{max}^{e_2}}^{e_2}$	$n_{max}^{e_3}$	$q_{n_{max}^{e_3}}^{e_3}$	q^p
4	1	.99	1	.55	<u>2</u>	<u>.64</u>	.34
5	1	.99	<u>2</u>	<u>.80</u>	2	.64	.51
6	1	.99	2	.80	<u>3</u>	<u>.78</u>	.62
7	1	.99	<u>3</u>	<u>.91</u>	3	.78	.71
8	1	.99	3	.91	<u>4</u>	<u>.87</u>	.78
9	1	.99	3	.91	<u>5</u>	<u>.92</u>	.83
10	1	.99	<u>4</u>	<u>.96</u>	5	.92	.88
11	1	.99	4	.96	<u>6</u>	<u>.95</u>	.91

bility target value r_{min} is set to 0.9, which requires a total transmission budget of 11 transmissions to reach in this example. As stated above, the route candidate selection is based solely on topology information local to the respective source node (vertex v_1 in case of this example), which may be outdated. The process of refinement of this route candidate is covered in the following section.

4.1.3. Cooperative Discovery of an Initial Route

To preselect a route candidate satisfying r_{min} , the source node relies on its local view T_v of the global network status. This local view may not be completely accurate, as network status data exchanged via cTEx may have aged towards the end of update periods or due to loss of probe frames containing such data. Therefore, dR_{min} -routing performs a cooperative search along the preselected route candidate p in order to improve the accuracy of the initial route. A description of this process in the form of pseudocode is provided in Section 4.1.5, Listings 4.1 to 4.5.

To start the cooperative search, the source node creates a REQUEST packet carrying the addresses of source node s and destination node d , route candidate $p = (e_1, \dots, e_n)$, reliability target r_{min} , the duration of the target time interval d_w , a ticket counter ctr_t , and the initial number of tickets ctr_i . Tickets represent permissions for concurrent route searches. In our experiments, we have set ctr_i to 4. For each edge $e_i \in p$, q^{e_i} and $n_{max}^{e_i}$ are included in the REQUEST packet (in

4. dR_{min} -Routing

form of vectors \vec{q}_p and \vec{n}_p). Then, the REQUEST packet is sent as a unicast frame to the next node, with the acknowledgment mechanism enabled. This ensures that lost REQUEST packets are retransmitted, which improves the reliability of the route search and differs from reactive routing protocols that use broadcast transmissions for this search.

When an intermediate node v_i receives a REQUEST packet, it recalculates q^{e_i} and $n_{max}^{e_i}$ of the next link, based on its local network status, and replaces these values in the REQUEST packet to be sent. Furthermore, it performs a route search with r_{min} -routing, with itself as source node, based again on its local network status T_{v_i} . For this search of an optimal suffix route $p_{v_i,d}$ from v_i to d , the reliability target r_{min} needs to be adapted, such that the combination of the prefix route p_{s,v_i} from s to node v_i with the suffix route $p_{v_i,d}$ still satisfies the initial r_{min} requirement. As the product of reliabilities of p_{s,v_i} and $p_{v_i,d}$ needs to be greater or equal to r_{min} , we calculate $r'_{min} = q_{n_{max}^{p_{s,v_i}}}^{p_{s,v_i}} / r_{min}$. If this alternative route p' (the concatenation of p_{s,v_i} and $p_{v_i,d}$) is different from the route recorded in the REQUEST packet, it is considered better, as it is based on newer topology information and the following action is taken:

- If the next hop of the alternative route is different from the next hop of the route of the REQUEST packet and $ctr_t > 1$, another REQUEST packet is created containing the route so far concatenated with the alternative route, and the tickets are split among the REQUEST packets.
- The remainder of the route recorded in the REQUEST packet is replaced by the alternative route otherwise.

Then, all REQUEST packets are sent to their next hop. Processing of received REQUEST messages in intermediate nodes is also described in pseudocode in lines 19-50 in Listing 4.1 of Section 4.1.5.

Upon reception of a REQUEST packet, the destination node compares ctr_i and ctr_t . If tickets are missing, i.e. $ctr_i > ctr_t$, it waits for the reception of further REQUEST packets. When all REQUEST packets have arrived or when a maximum waiting time has expired, the destination node selects the best route as the initial route. Then, it creates a RESPONSE packet containing the initial route and sends it as a unicast frame on the backward route to notify the source node and intermediate nodes.

4.1.4. Route Repair during Operation

Once a route over path $p = (e_1, \dots, e_n)$ has been established, route operation starts, i.e. TRANSPORT messages carrying a payload are transferred from source to destination. dR_{min} -routing monitors active routes during operation in order

to obtain feedback whether their actual reliability is sufficient to maintain the end-to-end reliability target r_{min} .

As a first measure, dR_{min} -routing observes each link of a route as edge e_i of the corresponding path p to check whether the actual link reliability still matches the required link reliability. In case there is a significant drop, i.e. the actual link reliability falls below the reliability requirement for this link established during route discovery, $n_{max}^{e_i}$ is increased. Each route is assigned a budget for this purpose, which is initially set to twice the number of edges in p , and is carried by messages along the route. This mechanism enables an immediate route repair, without the need for a new route search. If the initial budget has been consumed, the route is aborted by sending an ABORT message to the source as well as the destination node of the route, and a new route is searched (see Sections 4.1.2 and 4.1.3).

As a second measure, dR_{min} -routing monitors the packet delivery ratio (PDR) along the route to check whether the reliability target r_{min} is still satisfied for moving time windows of duration d_w , where d_w is specified when the route is requested. By monitoring sequence numbers, the PDR of the current window can be calculated at every hop along the route, including the destination node. When a node detects a PDR value below r_{min} , the route is aborted and reestablished. A description of route repair during operation is given in Section 4.1.5, Listing 4.6, lines 117-158.

4.1.5. Pseudocode of the dR_{min} -Routing Algorithm

In Listings 4.1 to 4.5, a slightly simplified version of the algorithm for dR_{min} -routing is depicted. All mathematical terms used are listed in Table 4.2, whereas a description of several helper functions used in the algorithm can be found in Table 4.3. The receive events (keyword `when`) for each of the four message types REQUEST, RESPONSE, ABORT and TRANSPORT, together with application layer requests for connection and data transmission provide the structure of the algorithm.

In Section 4.1.3 the message types responsible for cooperative discovery of an initial route, REQUEST and RESPONSE, were introduced. During route repair (Section 4.1.4), ABORT messages are used to communicate a route failure to source, destination and intermediate nodes. TRANSPORT messages are the only messages that carry a payload and are used for regular communication along established routes. In addition to the payload, each TRANSPORT message carries a budget ctr_r for route repairs, as stated in Section 4.1.4.

As mentioned above, the depicted pseudocode describes a slightly simplified version of the dR_{min} -routing algorithm. Processing of a fifth message type (called DONE) for cleanly shutting down established routes is omitted for the sake of brevity. Another aspect omitted from this version is handling of the special case of non-existence of a route with sufficient quality. In this case, the request-

4. dR_{min} -Routing

ing application layer is informed that no route was found. It is then up to the requesting layer to retry the search, possibly with an increased number of tickets. The cleanup procedure for stale half-established routes is also omitted, as well as queuing of TRANSPORT messages during route discovery and repair. The selection of functionalities to be omitted from this specification in pseudocode was done based on novelty of concepts and complexity. The omitted parts mentioned above are nonetheless vital to the implementation, albeit conceptually straight-forward and of low complexity.

Table 4.2: Definitions used in Listings 4.1 to 4.5

Definition	Description
$p_{s,d} = (e_1, \dots, e_n)$	Path between s and d , where $e_1 = (s, v_1)$ and $e_n = (v_{n-1}, d)$
$\vec{n}_p = (n_{max}^{e_1}, \dots, n_{max}^{e_n})$	Vector of transmission limits for TRANSPORT frames
$\vec{q}_p = (q^{e_1}, \dots, q^{e_n})$	Vector of link qualities values for path p
$rid = \text{unique identifier}$	Route identifier, unique for each initial route request
$R_{s,d} = (rid, p_{s,d}, \vec{n}_{p_{s,d}}, \vec{q}_{p_{s,d}}, r_{min}, d_w)$	Route tuple
n_{mgmt}	Transmission limit for management frames (REQUEST, RESPONSE, ABORT)
$n_{rx}(R_{s,d}, t)$	Number of packets sent over $R_{s,d}$ during last d_w seconds before t
$n_{tx}(R_{s,d}, t)$	Number of packets received over $R_{s,d}$ during last d_w seconds before t
$pdr(R_{s,d}, t) = \frac{n_{rx}(R_{s,d}, t)}{n_{tx}(R_{s,d}, t)}$	Definition of packet delivery ratio. Route reliability targets are met, as long as $\forall t. pdr(R_{s,d}, t) \geq r_{min}$
$q^e = \text{OB-EWMA}_{\alpha, p}^e(n_{mw}, s_{mw})$	Current link quality as determined and maintained by cTEX, see Section 4.1.1 and [KG21].
$T^e = (e, q^e)$	Status of link $e = (v, v') \in V \times V$.
$T^v = \text{set of } T^e$	Network status of $v \in V$.
$Rmin(T^v, s, d, r_{min})$	Function, returning a route $R_{s,d}$, satisfying r_{min} (see Section 4.1.2 and [KMG19] for details). Based on information local to v .
$REQ_{s,d} = (R_{s,d}, ctr_t, ctr_i)$	REQUEST packet tuple
$RSP_{s,d} = (R_{s,d})$	RESPONSE packet tuple
$ABRT_{s,d} = (R_{s,d})$	ABORT packet tuple
$TRNS_{s,d} = (R_{s,d}, ctr_r, data)$	TRANSPORT packet tuple
R_{est}	Set of established routes $R_{s,d}$
R_{wait}	Set of half-established routes $R_{s,d}$
R_{mon}	Set of tuples $(R_{s,d}, T_v)$ of monitored route and network state

4. dR_{min} -Routing

Table 4.3: Helper Functions used in Listings 4.1 to 4.5

Definition	Description
send()	Send packet to the next hop as ACKed unicast, with specified transmission limit.
reliability()	Calculate boosted reliability $q_{n_{max}}^p$ of provided route.
next_hop()	Extract next hop of provided node in provided route.
reverse()	Reverse $p_{s,d}$, $\vec{n}_{p_{s,d}}$ and $\vec{q}_{p_{s,d}}$ of provided route.
prefix()	Extract a prefix of provided route up to provided node.
best()	Select the route with minimal n_{max}^p still satisfying r_{min}
get_est()	Get route with provided rid from set R_{est}
search_done()	Check if all tickets were received, i.e. compare sum over ctr_t for rid of provided route request, with ctr_i and check whether the maximum waiting time has elapsed. In each case, route search is complete and true is returned.
check()	Check status of the provided monitored route against current T_v . Returns REPAIR if the actual link reliability falls below the required level, FAULT if the r_{min} target is not met in last time window of duration d_w and OK otherwise.
boost()	Calculate number of transmissions necessary to reach the same level of link quality as when the provided route was established.
conn_req()	Application layer request for a connection satisfying provided r_{min} requirement. Triggers route request.
conn_ind()	Indicate to application layer that a connection is established.
data_req()	Application layer request to transport data along established route with provided rid .
data_ind()	Indicate to application layer that new data was received on an established connection.

```

1 when receive  $REQ_{s,d}$  then {
2   if  $d = v$  {
3     // -> This node is the destination node
4     // Add request to the waiting set
5      $R_{wait} \cup \{REQ_{s,d}.R_{s,d}\}$ 
6     if search_done( $R_{wait}, REQ_{s,d}$ ) {
7       // -> All route requests sent by the source were
8       // -> received, select a route
9       // Create set of route candidates and select best
10      let  $C = \{r \in R_{wait} \mid r.rid = REQ_{s,d}.R_{s,d}.rid\}$ 
11       $RSP_{s,d} = (\text{best}(C))$ 
12       $RSP_{s,d}.R_{s,d} = \text{reverse}(RSP_{s,d}.R_{s,d})$ 
13      // Send RSP message with selected route,
14      // notify application and clean up waiting set
15      send( $RSP_{s,d}, n_{mgmt}$ )
16      conn_ind( $rid, s, d, r_{min}$ )
17       $R_{wait} \setminus \{r \in R_{wait} \mid r.rid = RSP_{s,d}.R_{s,d}.rid\}$ 
18    }
19  } else {
20    // -> This node is not the destination, but an
21    // -> intermediate hop
22    // Construct alternative route from s to d, with
23    // fixed prefix and optimized suffix
24    let  $p' = \text{prefix}(REQ_{s,d}.R_{s,d}, v)$ 
25     $r'_{min} = \text{reliability}(p') / r_{min}$ 
26     $R'_{s,d} = p' + Rmin(T^v, v, d, r'_{min})$ 
27
28    if next_hop( $R'_{s,d}, v$ )  $\neq$  next_hop( $REQ_{s,d}.R_{s,d}, v$ )
29      // -> Next hop of alternative route is different from
30      // -> route suggested in REQ message
31      if  $REQ_{s,d}.ctr_t > 0$  {
32        // -> Budget for parallel search is not yet exhausted
33        // Create another REQ message containing the
34        // alternative route and send it
35         $REQ_{s,d}.ctr_t = REQ_{s,d}.ctr_t - 1$ 
36         $REQ'_{s,d} = (R'_{s,d}, 1, ctr_t)$ 
37        send( $REQ'_{s,d}, n_{mgmt}$ )
38      } else {
39        // -> Budget for parallel search is exhausted
40        // Create REQ message containing the alternative
41        // route, send it and exit without forwarding
42         $REQ'_{s,d} = (R'_{s,d}, 0, ctr_t)$ 
43        send( $REQ'_{s,d}, n_{mgmt}$ )
44        return
45      }
46    }
47    // Forward the received REQ message along
48    // route suggested inside
49    send( $REQ_{s,d}, n_{mgmt}$ )
50  }
51 }

```

Listing 4.1: Pseudocode describing the processing of REQUEST messages

4. dR_{min} -Routing

```
52 when receive  $RSP_{s,d}$  then {
53   if  $s = v$  {
54     // -> This is the source node, add route to set
55     // -> of established routes
56      $R_{est} \cup \{ \text{reverse}(RSP_{s,d}.R_{s,d}) \}$ 
57   } else {
58     // -> This is an intermediate node, forward RSP message
59     send( $RSP_{s,d}$ ,  $n_{mgmt}$ )
60   }
61
62   // In either case, start monitoring outgoing links for
63   // the respective route
64    $R_{mon} \cup \{ (\text{reverse}(RSP_{s,d}.R_{s,d}), T^v) \}$ 
65 }
```

Listing 4.2: Pseudocode describing the processing of RESPONSE messages

```
66 when receive  $ABRT_{s,d}$  then {
67   // On route abort, remove the respective route from set
68   // of monitored routes
69    $R_{mon} \setminus \{ (R_{s,d}, T^v) \mid R_{s,d}.rid = ABRT_{s,d}.R_{s,d}.rid \}$ 
70
71   if  $s = v$  {
72     // -> This is the source node
73     // Remove route from set of established routes
74      $R_{est} \setminus \{ ABRT_{s,d}.R_{s,d} \}$ 
75   }
76
77   if  $(v \neq s)$  and  $(v \neq d)$  {
78     // -> This node is an intermediate node
79     // Forward ABRT message
80     send( $ABRT_{s,d}$ ,  $n_{mgmt}$ )
81   }
82 }
```

Listing 4.3: Pseudocode describing the processing of ABORT messages

```

83 when receive  $TRNS_{s,d}$  then {
84   if  $d = v$  {
85     // -> This is the destination node
86     // Extract and pass received data to application layer
87     data_ind( $TRNS_{s,d}.R_{s,d}.rid, TRNS_{s,d}.data$ )
88   } else {
89     // -> This is an intermediate node
90     // Forward the message along the established route
91     send_transport( $TRNS_{s,d}$ )
92   }
93 }

```

Listing 4.4: Pseudocode describing the processing of TRANSPORT messages

```

94 // This event is triggered by the application layer
95 // to request a new connection to a node
96 when conn_req ( $s, d, r_{min}$ ) then {
97   // Find a route candidate
98    $R_{s,d} = Rmin(T^v, s, d, r_{min})$ 
99   // Construct REQ message for candidate route
100   $REQ_{s,d} = (R_{s,d}, ctr_i, ctr_i)$ 
101  // Send request message
102  send( $REQ_{s,d}, n_{mgmt}$ )
103 }
104
105 // This event is triggered by the application layer to
106 // send data via an established connection
107 when data_req ( $rid, data$ ) then {
108   // Obtain established route
109    $R_{s,d} = get\_est(R_{est}, rid)$ 
110   // Initialize repair budget
111    $ctr_r = 2 \cdot R_{s,d}.p_{s,d}.hops$ 
112
113   // Create and send TRNS message to transport data
114    $TRNS_{s,d} = (R_{s,d}, ctr_r, data)$ 
115   send_transport( $TRNS_{s,d}$ )
116 }

```

Listing 4.5: Pseudocode describing the processing of application layer events
conn_req and data_req

4. dR_{min} -Routing

```

117 def send_transport( $TRNS_{s,d}$ ) {
118     // Before each transmission of a TRNS message,
119     // health of the corresponding route is checked
120     switch check( $TRNS_{s,d}.R_{s,d}, T^v, R_{wait}$ ) {
121         case OK:
122             // -> Route is ok
123             // Transmit message with retransmission
124             // limit configured after route discovery
125             send( $TRNS_{s,d}, n_{e,max}$ )
126         case REPAIR:
127             // -> Route needs repair, i.e. the respective outgoing
128             // -> link's quality deteriorated since route discovery
129
130             // Calculate number of necessary boosts to maintain
131             // reliability link had after route discovery
132             let  $n'_{e,max} = \text{boost}(TRNS_{s,d}.R_{s,d}, T^v, R_{wait})$ 
133             if ( $TRNS_{s,d}.ctr_r - n'_{e,max} > 0$ ) {
134                 // -> Repair budget allows to apply boost
135                 // Update repair budget in TRNS message and send
136                 // with additional boost
137                  $TRNS_{s,d}.ctr_r = TRNS_{s,d}.ctr_r - n'_{e,max}$ 
138                 send( $TRNS_{s,d}, n'_{e,max}$ )
139             } else {
140                 // -> Repair budget is exhausted
141                 // Send message without additional boost
142                 send( $TRNS_{s,d}, n_{e,max}$ )
143             }
144         case FAULT:
145             // -> Route has failed, i.e. specified  $r_{min}$  target
146             // -> was not met in the last  $d_w$  seconds
147
148             // Create and send ABRT message in forward direction
149              $ABRT_{s,d} = (TRNS_{s,d}.R_{s,d})$ 
150             send( $ABRT_{s,d}, n_{mgmt}$ )
151             // Create and send ABRT message in backward direction
152              $ABRT'_{s,d} = (\text{reverse}(TRNS_{s,d}.R_{s,d}))$ 
153             send( $ABRT'_{s,d}, n_{mgmt}$ )
154
155             // Remove route from set of monitored routes
156              $R_{mon} \setminus \{(R_{s,d}, T^v) \mid R_{s,d}.rid = ABRT_{s,d}.R_{s,d}.rid\}$ 
157     }
158 }

```

Listing 4.6: Pseudocode describing the process for sending TRANSPORT messages including route repair

4.2. Assessment of dR_{\min} -Routing

dR_{\min} -routing claims to discover, maintain and operate routes with specified minimum route reliabilities for time windows of a specified duration. We have devised extensive experiments to support this claim and show that dR_{\min} -routing is a viable alternative to existing routing protocols. In this section, we will provide results showing that

- routes discovered and operated with dR_{\min} -routing match their specified reliability targets over the specified time windows,
- the latency introduced by the cooperative route discovery mechanism used in dR_{\min} -routing (see Section 4.1.3) is low, and usually smaller than 10 ms,
- and that the employed route discovery mechanism finds suitable routes even when the network is under load.

4.2.1. Experiment Setup

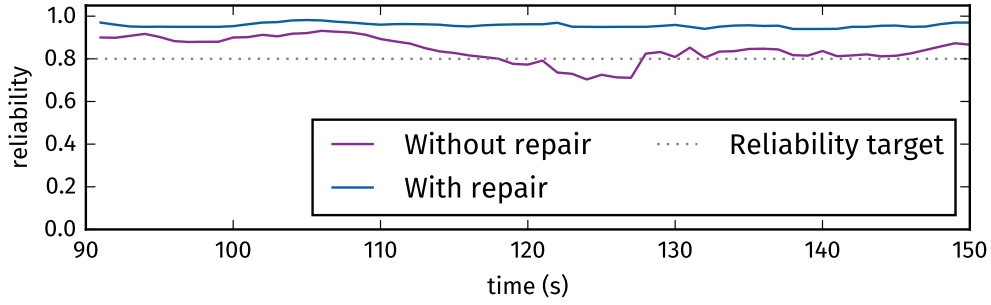
The experiments were carried out in the discrete event network simulator $ns-3^3$, using *YansWifiPhy* for simulation of the PHY layer and *AdhocWifiMac* for the MAC layer. The WiFi standard was set to 802.11g, with a constant transmission rate of 54 MBit/s for unicast as well as broadcast frames. To model propagation loss, the class *LogDistancePropagationLossModel* from $ns-3$ was used, configured with an exponent of 2.0 and a reference loss of 46 dB at 1 m, in addition to a random propagation loss drawn from a normal distribution with a mean of 0 dBm and a variance of 8 dBm.

The experiments on dR_{\min} -routing were carried out with our own implementation of a network and MAC layer protocol on top of *AdHocWifiMac*, to allow for the necessary control over retransmissions. For the OLSR experiments carried out in Section 4.3, the protocol implementation that comes with $ns-3$ was used, with UDP as the transport protocol and IP as the underlying network layer protocol.

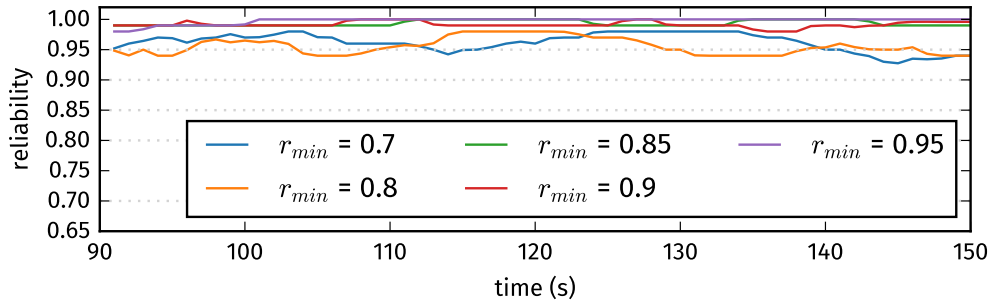
The topologies used in the experiments are all random generated, with a uniform distribution of the static nodes' positions. The nodes are scattered over a square area of 500 m width for the smaller, and 1000 m width for the larger topologies. To obtain interesting node pairs to evaluate the routing performance of the protocols, we calculated the node pairs with the greatest Euclidean distance. This approach avoids trivial routes between close nodes, which pose no real challenge for most routing protocols.

³Source code is available at <https://gitlab.com/nsnam/ns-3-dev>

4. dR_{min} -Routing



(a) Comparison of dR_{min} -routing with and without route repair.



(b) Five routes operated simultaneously between the same node pair, with different r_{min} targets.

Figure 4.2: Experiments illustrating the necessity of route repair, and successful matching of different reliability targets.

4.2.2. Analysis of Route Reliabilities

To achieve the specified reliability targets, dR_{min} -routing applies two measures: use of a stable, adaptive, and conservative link metric for route discovery, and continuous monitoring with route repair mechanisms, which are applied when the end-to-end reliability target r_{min} is (about to be) missed. Fig. 4.2a shows by the example of a single, selected route that with route repair, the reliability target was matched throughout route operation. The displayed reliability values are the ratio of successfully received packets at the destination node, calculated for a moving window of $d_w = 10$ s. In Fig. 4.2b, the reliabilities of several routes operated simultaneously between the same pair of nodes, with different target reliabilities r_{min} is depicted. The configured reliability targets are always matched. The topology consists of 30 nodes, all routes transport a stream of datagrams of 50 KiB/s.

4.2.3. Latency and Overhead of Route Search

The cooperative route discovery process introduces a latency for each route search, depending on the length of route candidates in hops. In a series of

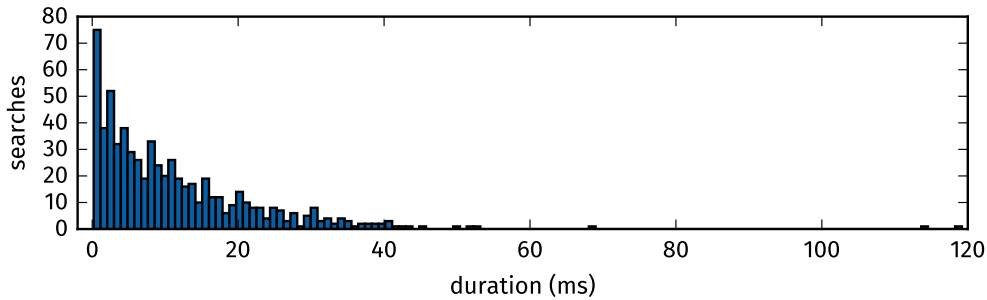


Figure 4.3: Distribution of latency caused by route discovery for successful searches in a 30-node topology with dR_{\min} -routing.

experiments on a 30-node topology, the latency between transmission of the REQUEST message and reception of the respective RESPONSE message at the source node was recorded. The results in Fig. 4.3 show that the latency introduced by route discovery is almost always (>98 %) below 40 ms and in 57 % of cases below 10 ms. In the 62 experiments of this series, the number of simultaneous routes ranges from 1 to 35, and the data rate of the operated routes varies between 2.5 KiB/s and 75 KiB/s. In Fig. 4.3, only successful route searches are depicted. A route search is successful if the destination node was reached and responded with a RESPONSE message that was received by the source node before the request timed out.

The management overhead of dR_{\min} -routing in terms of bandwidth consumption is determined by the overhead for topology detection and exchange with cTEX, and the overhead for cooperative route discovery. In [KG21], the overhead of cTEX amounts to 26.5 bps per link in a network of 36 nodes and 120 links. The overhead for cooperative route discovery is bounded by the maximum number of exchanged REQUEST and RESPONSE frames, which depends on the number of links of route candidates, the length of the selected route, and the maximum number of retransmissions per link. Here, the number of route candidates is limited by the number of tickets. In summary, the management overhead of dR_{\min} -routing is low.

4.2.4. Route Discovery Success Ratio

dR_{\min} -routing claims to discover and maintain routes that satisfy a given end-to-end reliability target. Depending on the quality of the protocol, the topology and existing traffic, such a route might not be found for every pair of nodes. Hence, the ratio of source and destination node pairs for which a suitable route is found is an important performance criterion for any routing protocol. dR_{\min} -routing manages to find routes in most environments, given the network contains a sufficient amount of reliable links and has enough free bandwidth to accommodate the additional traffic. To illustrate this, a sequence of experiments

4. dR_{\min} -Routing

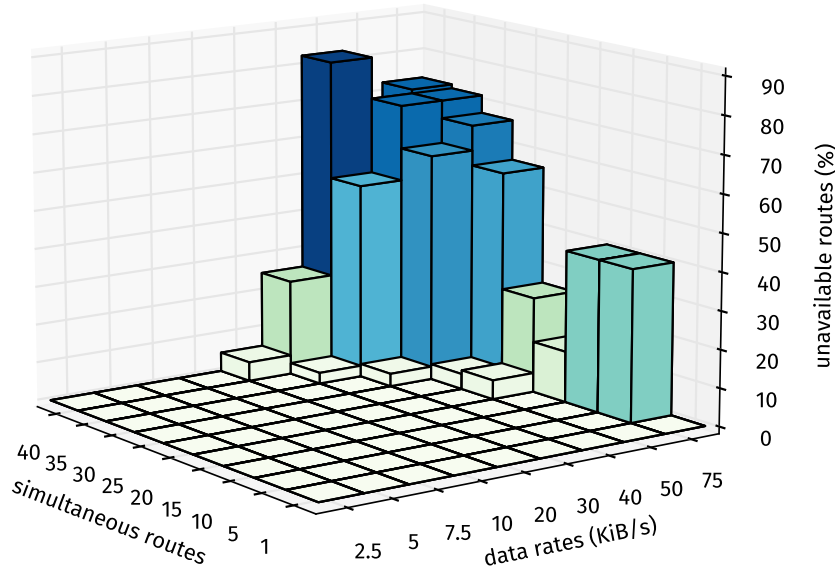


Figure 4.4: Ratio of node pairs, where no route satisfying the specified reliability requirement of $r_{\min} = 0.8$ could be found.

with different numbers of simultaneously operated routes under different data rates were carried out on several 30-node topologies with a field width of 500 m. The source and destination node pairs were picked randomly, while nodes were only picked a second time after every other node was picked once. The aggregated results are displayed in Fig. 4.4. We observe that dR_{\min} -routing is capable of discovering suitable routes in the majority of cases analyzed. In cases where data rates >40 KiB/s in combination with a high number of routes operated in parallel (>30) are requested, dR_{\min} -routing struggles to find routes. On the other hand, the network consists of 30 nodes placed in a $500\text{ m} \times 500\text{ m}$ area, which suggests that the problem stems in part from a lack of available bandwidth.

4.3. Comparison of dR_{\min} -Routing with OLSR

We now compare the performance of dR_{\min} -routing and Optimized Link State Routing (OLSR) [CJ03]. OLSR optimizes the control overhead of the classical link state algorithm for wireless local area networks. For this, the concept of multipoint relays (MPRs) is introduced and applied. MPRs provide an efficient way for broadcasting control traffic by reducing the number of transmissions. This works especially well in dense networks. Furthermore, OLSR determines routes with the smallest number of hops. A mature implementation of OLSR is available for the network simulator ns-3. The setup for all experiments in this section has been stated in Section 4.2.1.

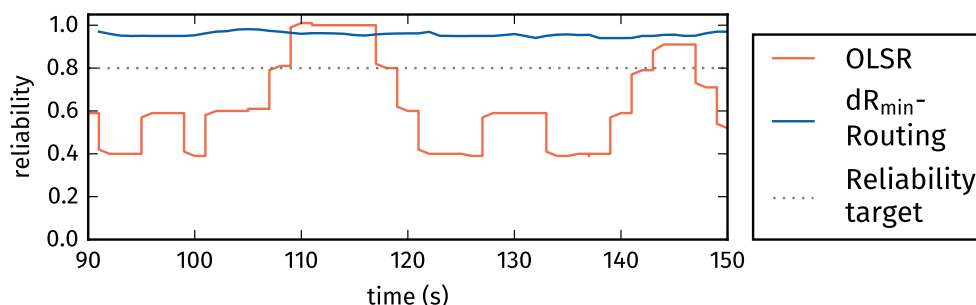


Figure 4.5: Comparison of route reliability over time for the same node pair in a 30-node topology, with $r_{\min} = 0.8$.

4.3.1. Comparison of Route Reliability

In Section 4.2, it was shown that dR_{\min} -routing can find and operate routes that conform to specified reliability targets. To show that finding and operating these routes requires a specialized routing protocol, the reliability of routes detected by OLSR was analyzed. In Fig. 4.5, the reliability of routes between a single node pair is displayed. For the route discovered by OLSR, the reliability oscillates and is generally much lower than for the route discovered with dR_{\min} -routing. It is important to note that the upper limit of MAC layer transmissions for a frame is the same for both setups.

To evaluate whether the specified reliability targets are met for arbitrary routes and under a variety of network conditions, another set of experiments was carried out. On a 30-node topology, between 1 and 40 routes were operated simultaneously with data rates between 2.5 KiB/s and 75 KiB/s. In total, 1448 routes were evaluated, the achieved reliability of each route with dR_{\min} -routing was recorded and is displayed in Fig. 4.6a. For comparison, the same experiments were run with OLSR, which does not employ any mechanisms to discover particularly reliable routes. The resulting reliability of each route is shown in Fig. 4.6b. In very rare cases, dR_{\min} -routing may not detect deteriorating route reliability in time to repair the route before it falls below the specified target. Routes operated with OLSR between the same pairs of nodes are mostly unreliable. This is due to the simple topology detection mechanism of OLSR, and because OLSR favors short routes in terms of hops.

To further evaluate the reliability of routes detected and maintained with dR_{\min} -routing for different topology sizes and to compare them with OLSR, we have introduced 5 topology size classes ranging from 10 to 50 nodes. For each size class, we have generated 10 topologies. For each topology, 5 node pairs with maximum Euclidean distance were calculated. Between each of these node pairs, a packet stream of 20 packets per second, with a payload of 128 Bytes each is generated, resulting in a payload traffic of 2.5 KiByte/s per route. The reliability of a route was measured as the packet delivery ratio (PDR) over

4. dR_{\min} -Routing

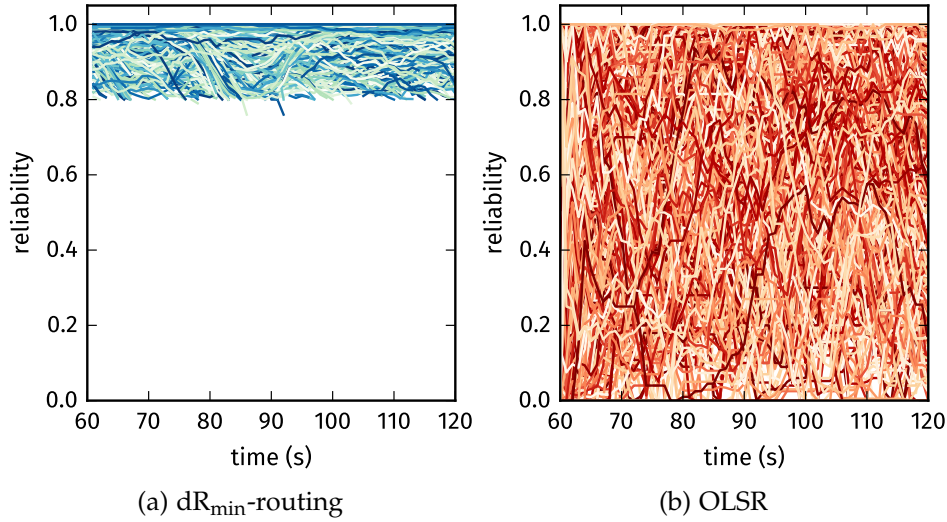


Figure 4.6: Comparison of route reliabilities of 1448 routes operated in a 30-node topology of size $500\text{ m} \times 500\text{ m}$. The dR_{\min} -routing reliability target for all routes was set to 0.8.

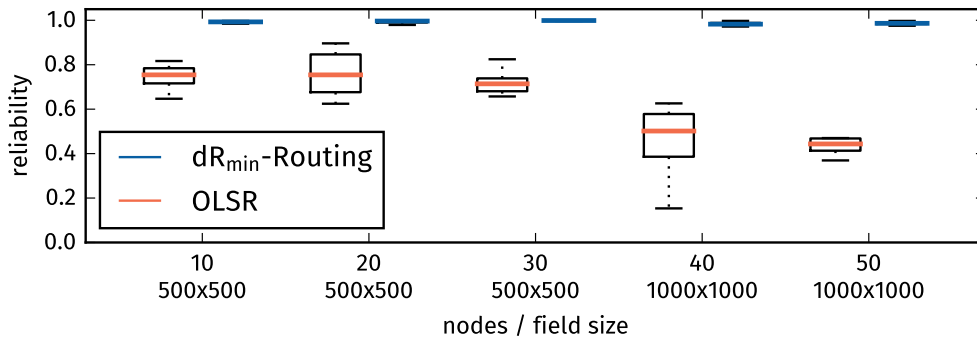


Figure 4.7: Comparison of average reliabilities of dR_{\min} -routing and OLSR

a moving window of length 10s. The obtained PDR sequence for each route is then aggregated using the *arithmetic average*. The resulting average reliabilities of all routes in a single topology are then also aggregated into a single value using the *arithmetic average*, such that 10 aggregated reliability values remain over a total of 10 topologies with the same number of nodes each. The results are depicted in Fig. 4.7 and show that routes operated with dR_{\min} -routing have a very high reliability over all five topology size classes analyzed. This is different for OLSR, which also degrades with larger topology sizes.

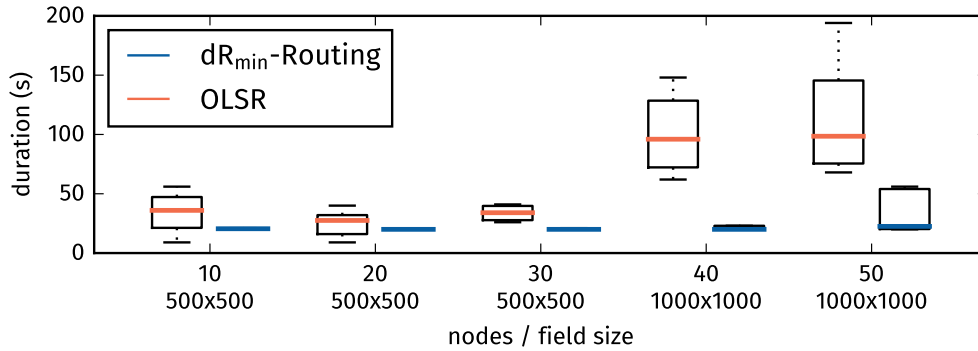


Figure 4.8: Required time to transmit 1 MiB through the network in different topologies (10 topologies per size class).

4.3.2. Comparison of Network Throughput

To illustrate that routes discovered and operated with dR_{min}-routing can support applications where network throughput is of concern, we devised another series of experiments. Network throughput was analyzed by transmitting 1 MiB between the two most distant nodes in the network for each topology. The required time for this transfer is measured for each topology, resulting in 10 values per topology size class. The results are depicted in Fig. 4.8.

For the dR_{min} experiment, a packet with 512 Byte payload was created every 10 ms at the source node, as the mechanisms employed by dR_{min} for route reliability monitoring and bandwidth reservation benefit from periodic traffic. As it takes about 20 s to generate 1 MiB of data at 51,200 B/s, a lower limit is visible for the transfer durations in the dR_{min}-routing results in Fig. 4.8. The OLSR experiment was configured to use the same payload size per packet, but without any delay between the generation of consecutive packets. For the smaller topologies, OLSR has higher throughput in some cases, the median experiment however already indicates a network throughput that is below the minimum throughput achieved over all runs with dR_{min}-routing. As the number of nodes or the network area increases, the difference in network throughput between OLSR and dR_{min}-routing becomes wider. In the experiments with 50 nodes, the median transfer duration for 1 MiB with dR_{min}-routing is less than a quarter of the time necessary with OLSR.

4.4. Preparations for Real-World Experiments

In the previous section we summarized the results from simulation experiments in the network simulator ns-3. The simulator was used to evaluate dR_{min}-routing performance with respect to reliability, throughput and route discovery latency. As the results and especially the comparison of dR_{min}-routing with

4. dR_{\min} -Routing

the OLSR protocol give reasons for the dR_{\min} protocol to be considered for real application scenarios, a thorough evaluation in real world experiments is the next step. A full evaluation of dR_{\min} -routing in a real world testbed is beyond the scope of this thesis. We did however look into the necessary preparations and caveats for a move of the dR_{\min} -protocol from a simulated to a real-world environment. Several aspects need addressing for this transition:

- Success of transmission events over simulated links (without collisions) is statistically independent and identically distributed, which we cannot assume for real world transmissions.
- The single network property is valid for the simulated environment, it may not be for a real world testbed.

The first aspect, regarding statistical independence of success of transmission events, is a common assumption made in the context of reliability-focused protocols. However, success of a transmission is usually not statistically independent. In scenarios of periodic traffic patterns, or in sparse networks, the sequence of transmission failure and success is often bursty [SKAL08], i.e. a successful transmission is more likely to be followed by several more successful transmissions. Also, a transmission failure is more likely to be followed by another failure as well. The assumption of independence of transmission success, however, is at the base of the link boosting mechanism of dR_{\min} -routing. As dR_{\min} -routing relies heavily on link boosting to achieve target reliability levels, this issue needs to be addressed with great care, in order for dR_{\min} to work reliably. In [Mat22], the question of what statistical properties a link is required to exhibit for the boost mechanism (i.e. targeted retransmissions) to work is posed and answered in great detail. The authors evaluate efficacy of link boosting for links with different statistical properties and with synthetic as well as real world data. Three different statistical properties of wireless links are presented and evaluated:

- the μ -value for statistical independence, which is a component of the discrete-time 2-state Markov model published in [Gil60, Ell63] and known as the Gilbert-Elliott model,
- p and z -values, as components from the Wald-Wolfowitz Runs-test [Bra68], which indicate whether the sequence of transmission success and failure is random,
- and at last the Chi-square goodness-of-fit test of the observed event outcomes against a Bernoulli distribution.

The statistical properties are then compared with respect to their capacity to predict the efficacy of link boosting, i.e. whether a link that is supposedly boosted to a specific reliability level actually meets the specified reliability target.

The results suggest that using the Gilbert-Elliott model for monitoring statistical independence in form of the μ -value is sufficient to form a decision on whether a specific link's statistical properties fully support link boosting directly, only after application of additional measures such as an increased retransmission budget, or not at all. By incorporating the μ -value as an additional metric into the route selection process, links that do not support boosting with sufficient bandwidth efficiency can be identified and avoided. As the authors of [Mat22] obtained their results from an offline analysis of real testbed transmission event data, the conclusions may only be valid for scenarios, where failed transmission events do not trigger an immediate retransmission. Success or failure of the retransmission is modelled by repurposing the outcome of the subsequent transmission made by the node, possibly occurring several hundred milliseconds later, depending on the transmission interval. This is an important takeaway, as this ensures a continuously uniform distribution of transmission events over time and does not lead to a cluster of (re-)transmissions as soon as some frames are dropped.

Another conclusion drawn by the authors of [Mat22] is that in the analyzed scenarios, most links exposed the necessary statistical properties to allow link boosting. This suggests that promising simulation results for a routing protocol such as dR_{\min} -routing, which relies on boostability of links, may actually apply (with limits) to real world scenarios.

4.4.1. The Gilbert-Elliott Model

As stated above, the μ -value as part of the Gilbert-Elliott model can be used to measure the boostability of a wireless link. Hence, a short summary is provided in the following paragraphs.

The Gilbert-Elliott model is defined as a discrete-time 2-state Markov chain and was introduced by Gilbert in [Gil60] and extended later by Elliott in [Ell63]. We map the two two states of the Markov chain Fig. 4.9, named *Good* and *Bad*, to successful transmission attempt and failed transmission attempt, respectively. The probability the state switches, for instance the probability that a successful transmission is followed by a failed one, is captured by p_b , whereas p_g corresponds to the probability that a failed transmission is immediately followed by a successful transmission attempt.

Channel memory $\mu \in [0, 1]$ can then be calculated as

$$\mu = 1 - p_b - p_g \quad (4.1)$$

and expresses the degree of statistical independence of a link. If for a given sequence of transmission events of a link, channel memory value μ is close to 1, the respective link is bursty and success and failure of transmission events on

4. dR_{min} -Routing

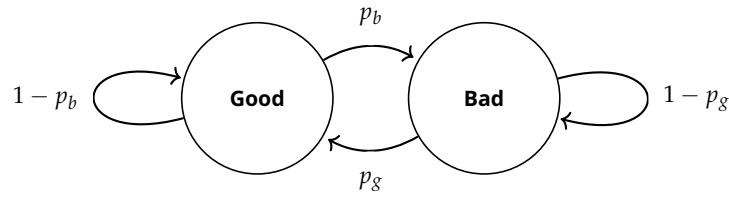


Figure 4.9: Gilbert-Elliott model

that link strongly depend on the outcome of the previous attempts. A μ close to zero indicates a link with statistically independent transmission outcomes, whereas a μ close to -1 suggests that the link is oscillatory.

A scenario not covered in [Mat22] is one where routes are actually operated in a real testbed network. Only a periodic traffic pattern, where transmissions are carried out by each node at semi-regular intervals was evaluated. This traffic pattern however is different from the patterns induced by nodes forwarding packets back and forth along an established route. The frame intervals are variable and set by the application, and several routes may be in operation simultaneously, leading to heterogeneous load patterns at the nodes in the network. As a result, the distribution of transmission events over time may deviate from an uniform distribution, which may be detrimental to the boostability of links required by dR_{min} -routing.

A possible countermeasure is provided in another chapter of [Mat22], where Time Token Bucket Algorithm (TTBA) is introduced. TTBA is a traffic shaping algorithm that can be applied to improve randomness of transmission events over time, or in other words, helps to reduce the burstiness of links and subsequently improves their statistical independence property.

4.4.2. Time Token Bucket Algorithm

The TTBA uses a token bucket to distribute transmissions over time. This works as follows: For each transmission, the expected medium occupancy time is calculated. Then, the algorithm checks whether the corresponding number of tokens is available in the token bucket and if so, consumes the tokens. In case the remaining token budget is insufficient to pay for the medium occupancy time, the transmission is delayed and no tokens are consumed. The refill rate of the token bucket and token size control the total medium occupancy time permitted to each node. The maximum size of the bucket controls the maximum burst length, as transmissions are performed only while the size and number of tokens in the bucket is sufficient.

To decouple the timing of transmission events from the clocking of both application and token bucket refill timer, the token bucket refill interval can be

configured with a lower and upper bound. The algorithm then picks a random interval after each refill event. Simulation experiments by the authors suggest this mechanism reduces the chance of frame loss due to collisions and results in a higher frame delivery ratio than static refill. Further simulation experiments show that this dynamic refill mechanism also improves the statistical link properties (as captured for instance by the μ -value from the Gilbert-Elliott model) necessary for successful link boosting.

Under the proper configuration of protocol parameters, TTBA effectively distributes transmissions over time and thus reduces the burstiness of wireless links. As a reduction in burstiness implies an increase in a link's statistical independence property, TTBA may serve as a supplementary protocol below dR_{\min} -routing to assist in ensuring links are boostable.

4.4.3. Preliminary Experiments and Future Work

The implementation of the dR_{\min} -routing protocol used in the simulation experiments of this chapter (see Section 4.2) can be used for real-world experiments with minimal additional effort. Preliminary experiments suggest that the promising results from the simulation environment do not immediately carry over to real-world scenarios. Thus, future work should focus on integration of a monitoring mechanism for the statistical properties of links required for successful link boosting (based on the Gilbert-Elliott model) into dR_{\min} -routing. In addition to this, we recommend adding traffic shaping based on the Time Token Bucket Algorithm to further improve the statistical properties of links for link boosting.

4.5. Related Work

The pattern of initiating a route discovery reactively with a request message, in combination with fixing and communicating the route decision with a corresponding response message sent by the destination node is also used in the classical Ad-hoc On-demand Distance Vector (AODV) [PR99] algorithm and Dynamic Source Routing (DSR) [Joh94] protocol. Both protocols also require nodes to monitor established routes and indicate route failures with route error messages. dR_{\min} -routing employs a similar, though much more sophisticated monitoring and repair mechanism. Different from AODV, dR_{\min} -routing uses directed unicast messaging instead of flooding the network with request messages for route discovery. dR_{\min} -routing also applies a much more flexible approach for topology exploration and link quality estimation, which adds a proactive component to the reactive approach of DSR and AODV.

In more recent works, quality of service in wireless networks is of increasing

4. dR_{\min} -Routing

interest. QOLSR [MF07] extends the well-known OLSR protocol with metrics for delay and bandwidth and proposes additional MPR selection heuristics. In [YMSF07] a link availability-based QoS-aware (LABQ) routing protocol is introduced, where the availability of links between mobile nodes is monitored to inform the route decision and produce routes with high availability and energy efficiency. The link quality metric is ETX [DCABM03], which is a well-known metric for ad-hoc wireless networks with some similarities to OB-EWMA, which is used in dR_{\min} -routing, as both are based on the measured packet (or frame) delivery ratio of a link. The focus with LABQ however, lies on the discovery of routes with high availability. ETX is not used to the extent, where a specific end-to-end route reliability target would be supported.

Optimized Link State Routing v2 (OLSRv2) [Cla14] is a revision of the original routing protocol for mobile ad hoc networks, OLSR. OLSRv2 supports arbitrary link metrics for route discovery, with the requirement that the metric is additive. This provides support of link metrics which can support some QoS requirements. Maintenance and repair of routes in accordance with the selected link quality metrics and QoS specifications, however, would require extensions to the protocol.

The cooperative discovery of an initial route of dR_{\min} -routing has some similarities with Ticket Based Probing (TBP) [CN99]. TBP addresses delay- and bandwidth-constrained routing. When a route is requested, a number of green and yellow tickets are issued for a concurrent route search. Green tickets are permissions to search for cost-effective routes satisfying the QoS-requirement, yellow tickets increase the probability of success by searching for an optimal path, without considering cost. To preselect next hops, TBP assumes availability of distance vectors for delay, bandwidth, and cost. However, TBP is not functionally complete, and does not address reliability-constrained routing.

4.6. Conclusion

In this chapter we introduced dR_{\min} -routing, a functionally complete routing protocol that aims to provide applications with routes that satisfy the reliability requirement requested by the application. The dR_{\min} -routing protocol uses the r_{\min} -routing algorithm to extract reliable routes from topology information and includes mechanisms for monitoring and repair of established routes. We use the cTEx protocol in combination with the OB-EWMA link quality metric for topology detection and distribution to obtain an accurate view of the topology of the network, including normalized and conservative estimates of each link's quality.

In order to facilitate reproduction of our results, a detailed and semi-formal specification of dR_{\min} -routing is provided in the form of pseudocode. As all

additional components of dR_{\min} -routing, namely the r_{\min} -routing algorithm, the cTE_x protocol for topology detection and the OB-EWMA link quality metric are fully specified at a similar level of detail, an implementation of the dR_{\min} -routing protocol functionally equivalent to the one used in this thesis may be created by anyone in the research community.

dR_{\min} -routing was assessed in simulation experiments, which also include a comparison with the OLSR protocol.

5. A Demonstrator and Runtime Controller for a Small Demonstration Network

In previous chapters, two protocols for wireless ad-hoc networks were introduced: the cTEx protocol for topology detection and dR_{\min} -routing, a functionally complete routing protocol for discovery, operation and maintenance of routes that satisfy a specified reliability target. These protocols were specified at detail, implemented and evaluated in various simulation and real-world experiments. The results of this analysis were then published in research articles and are discussed over several sections of this thesis. For the introduction of the complex concepts behind communication protocols for wireless ad-hoc networks to a community of fellow researchers, this approach is well-suited.

However, in order to showcase core mechanisms and capabilities of these communication protocols to an audience without a deep background in the corresponding field of research, a less scientific and more practice-oriented approach would be helpful. Hence, a demonstrator framework and testbed was devised. The goals of this endeavor were set as follows:

- *Visualization of network state:* As the use case of the demonstrator is showcasing specific capabilities of complex communication protocols to a layman audience, dynamic visualizations of network state are necessary to keep impact of the respective communication protocols on the network state traceable.
- *Modular and extensible:* Architecture of the demonstrator framework should be devised such that the demonstrator can be used to showcase several different protocols, while integration of new protocols should be possible with low implementation effort.
- *Simple and robust setup:* The demonstrator hardware and software should be simple to set up, dismantle and configure, and work reliably e.g. when set up at an exhibition.
- *Few hardware components:* In order for the demonstrator setup to be still portable, only a few network nodes, an ethernet switch and cables as well

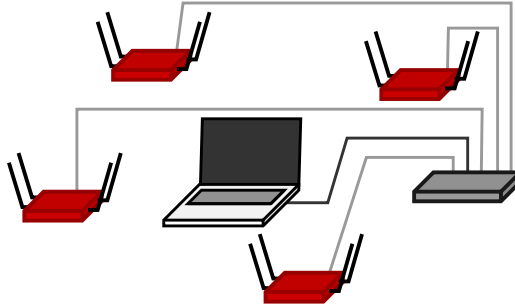


Figure 5.1: Demonstrator testbed setup.

as a laptop should be sufficient for showcasing. No infrastructure apart from an energy source should be required.

The demonstrator testbed setup is explained in Section 5.1. In the process of creating the demonstrator framework, a modular mechanism for runtime control of a protocol stack was implemented (see Section 5.2). For visualization of network state and control of the demonstrator through the operator, a web-based interface was implemented (details in Section 5.3). At last, a series of showcase experiments for each protocol was devised (Section 5.4).

In the last section of this chapter (Section 5.5), a short introduction to the tools and frameworks used during evaluation of the numerous experiments carried out in the context of this thesis is provided.

5.1. Testbed Setup

The demonstrator testbed consists of four network nodes, a laptop and an ethernet switch. A diagram of the testbed setup is shown in Fig. 5.1. The network nodes are PC Engines APU2¹, which are equipped with two PCIe WiFi Cards² and three Ethernet ports. Two antennas are attached to each WiFi Card. The four network nodes run Linux in version 5.10. The laptop is used to control the testbed and can be of any type and operating system, as the only requirements are availability of a web browser and an ethernet port. To prevent interference through the exchange of control and status information between the network nodes and the demonstrator interface this information is exchanged over a separate wired network. To provide said wired network, an ethernet switch and five cables are necessary. Network nodes, laptop and switch are connected by cables as shown in Fig. 5.1.

¹AMD Embedded G series GX-412TC, 1 GHz quad Jaguar core, 2 GiB RAM

²Compex WLE200NX, compatible with 802.11n, ath9k Linux kernel driver

5.2. Runtime Control of a Protocol Stack

To illustrate the capabilities of the reliability-focused protocol stack for topology detection and routing introduced in this thesis, we designed a series of showcase experiments. These showcase experiments are implemented utilizing a demonstrator application framework for small wireless ad-hoc networks, that allows for runtime control and configuration of protocol and application parameters. Runtime control of protocol parameters is a key prerequisite, as it is necessary to transfer a network into a sequence of states where a communication protocol's impact on network behavior and performance becomes apparent.

The core of the demonstrator framework is the component for runtime control of protocols. This component is protocol-agnostic, such that implementation of runtime control capabilities for additional communication protocols requires only minimal modification (or none at all). To achieve this level of generality for the runtime control component, we developed a modular architecture where each protocol layer can advertise a set of actions that may be triggered by the operator of the demonstrator. For instance, a protocol that is an optional part of a layered protocol stack may allow for its deactivation (and activation) at runtime. During startup, such a protocol would *register* an action identifier string and a corresponding callback function implementation with the stack controller. The stack controller then communicates all available actions to the demonstrator application. The demonstrator application stores protocol and action identifiers and automatically adds a control element to the web-based user interface, through which the operator may issue the control command for the respective protocol layer. The command is then sent to the respective network node, where the stack-controller component schedules a *command* event for the respective protocol layer. When the event is processed, the new protocol parameter is set. The described mechanism also supports a single, optional string parameter per action. A visualization of this architecture is depicted in Fig. 5.2.

This architecture requires usually no changes at the demonstrator side to add runtime control capabilities for a protocol. At protocol side, the framework provides functions for registration of controllable parameters of a protocol. Reporting of controllable parameters is done by the stack-controller component during startup of the node.

In addition to any controllable parameters, protocols may emit state variable updates. These may include for instance the total number of transmissions a node has performed or current end-to-end reliability of an established route. The reported values of state variables are visualized as tables, plots or charts in the demonstrator user interface. The reporting mechanism for state variable updates is very similar to the mechanism for registering controllable parameters (see also Fig. 5.2): Publishing a state variable on the protocol side is as simple as calling the framework-provided *publish* function with three parameters, the first

5. A Demonstrator and Runtime Controller for a Small Demonstration Network

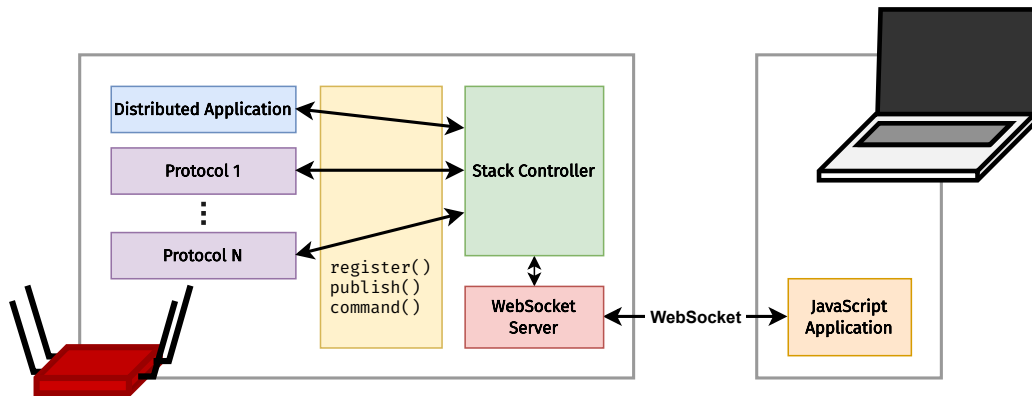


Figure 5.2: Architecture diagram of the runtime control mechanism.

is an identifier string for the respective state variable, the second is an identifier for the desired visualization type (e.g. table, graph or scatter plot, details in Section 5.3) and the last is a description of the current variable state, i.e. the value to be reported.

As the underlying protocol for the exchange of messages between the network nodes and the demonstrator application, we chose the WebSocket protocol. As the demonstrator is implemented as a web application, the chosen protocol should be well supported in common web browser's JavaScript engines. Another requirement is bandwidth efficiency of communication. Report of state variables in short intervals improves traceability of changes in protocol behavior via the user interface, however also increases load on the network connection between demonstrator and testbed. To keep the bandwidth requirement and processing times of the reporting mechanism at manageable levels, a lightweight protocol with low overhead is preferable. Another requirement is support for full-duplex communication, i.e. the server (in this case the program executed on the testbed node) may push data via an established WebSocket without an explicit prior request by the client (the demonstrator application running on the laptop). This is a requirement not met by half-duplex protocols such as HTTP, where the client needs to regularly poll the server. The WebSocket protocol satisfies all three requirements, i.e. JavaScript compatibility, bandwidth efficiency and full-duplex communication. We also considered plain HTTP for communication, however WebSockets are superior with respect to bandwidth efficiency, as no headers are used. As mentioned above, HTTP also supports half-duplex communication only.

5.3. Visualization of Network State

During protocol operation, the protocol implementation may report values of state variables to the demonstrator. As these reports are initiated at the nodes the protocol implementations run on (and not by the demonstrator), a full-duplex connection (e.g. through the WebSocket protocol) to the demonstrator is preferable to keep communication efficient, as with half-duplex connections (as used in HTTP) regular polling of each node by the demonstrator is necessary.

The web interface at the demonstrator supports a pre-defined set of visualizations. For each visualization type, there exists a format requirement for reporting values. A protocol implementation may report values of any state variable, as long as the reporting is done according to a specific format, depending on the desired visualization type for the reported value. For instance, each node may regularly report the reliability of all links it is aware of to the demonstrator node. When aggregated, link state information of all nodes in the network can be used to construct a graph, where each edge is annotated with the respective link reliability value. So for each link reliability report, we require some identifier for the kind of value that is reported, a timestamp, a chart type and the value itself. With respect to our example, where the reliability of an incoming link is reported, a suitable identifier would be `topology`, the timestamp would be a UNIX timestamp in milliseconds, the chart type is specified as an index into the list of available chart types, which is 2 for a topology plot, and at last the reported value, which is a tuple consisting of identifiers for source and destination nodes, as well as the quality of the link that connects both. The information is collected in an object and serialized as JSON before communication to the demonstrator. A JSON string that corresponds to the example discussed above is given in Listing 5.1.

```

1 {
2   "identifier": "topology",
3   "chart-type": 2,
4   "timestamp": 1671901200000,
5   "value": {
6     "source-node": 3,
7     "destination-node": 5,
8     "reliability": 0.85
9   }
10 }
```

Listing 5.1: Example of report of a single value in JSON format.

This JSON string is then sent to the demonstrator over the WebSocket connection. At the demonstrator the reporting node's network address is extracted from the message header and the JSON string is parsed. The reported value

together with the timestamp is then passed on to the respective plot function, where the value is visualized in the chart corresponding to the reported identifier, and according to the specified chart type along with any other values reported under the same identifier. Reported values are collected in separate data structures for each reporting node at the demonstrator. This way, plots can be grouped by reporting node.

In Fig. 5.3, an exemplary topology plot is depicted. The thickness of an edge reflects the quality of the respective link, as does the distance between nodes. The position of each node is calculated automatically using a gravity based spring layout mechanism, where the value assigned to each edge is used to model a force that pull source and destination node together. Between nodes without a direct link, a force that pushes both nodes apart is simulated. The resulting plots may not serve to provide insights on the real network's physical geometry, proximity of nodes in the plot however indicates reliable links, which often correlate with physical proximity in the real network.

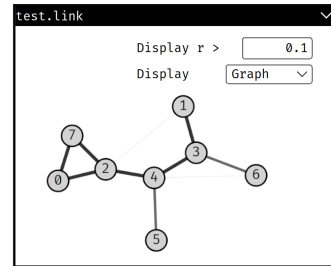


Figure 5.3: Topology plot.

All recently reported values are stored in data structures at the demonstrator and incorporated into the respective visualizations. Depending on the frequency state variables are reported, the stream of value reports may be of high volume. Visualizations are updated in a lazy manner, i.e. only when visible on the operator's viewport. A screenshot of the web-based demonstrator interface is depicted in Fig. 5.4.

As mentioned above, there is a fixed (yet easily expandable) set of visualization types available in the web interface of the demonstrator. A complete list of the supported plot types is provided in Table 5.1. Separate larger depictions of some of the visualization types in the demonstrator web interface are shown in Fig. 5.5

5.4. Showcase Experiments using the Demonstrator

The purpose of the demonstrator framework is to showcase the capabilities of complex communication protocols for wireless ad-hoc networks to a layman audience. To facilitate tracing network state changes for viewers, network state is visualized continuously in the demonstrator web interface and protocol parameters may be controlled and configured at runtime. The operator may also control the application that runs on top of the protocol stack and for instance

5.4. Showcase Experiments using the Demonstrator

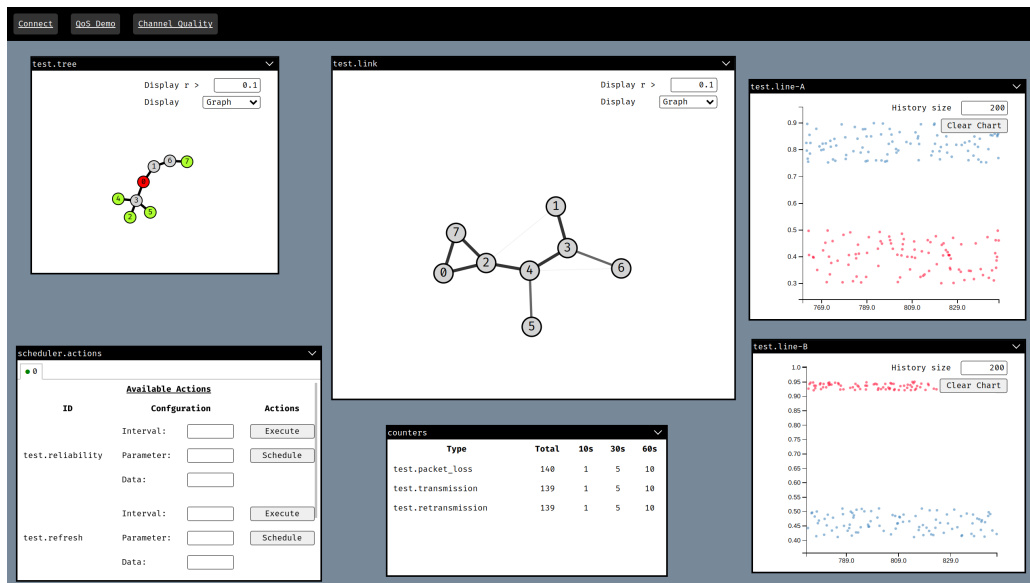
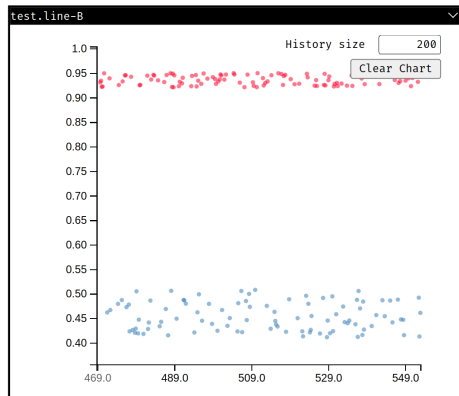


Figure 5.4: Screenshot of the demonstrator web interface.

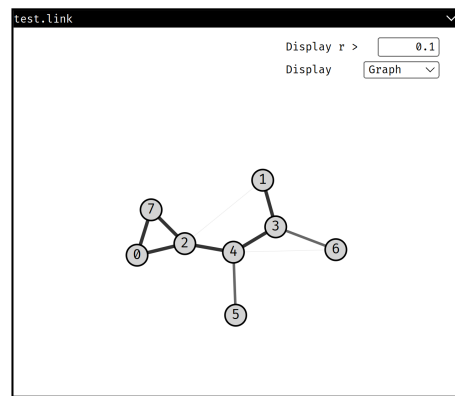
Table 5.1: Available Demonstrator Chart Types

Identifier	Chart type
1	Scatter plot (single sequence)
2	Graph
3	Table
4	WebView
5	Scatter plot (multiple sequences)
6	Tree

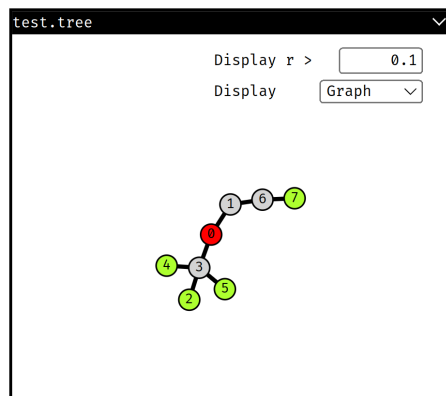
5. A Demonstrator and Runtime Controller for a Small Demonstration Network



(a) Screenshot of a scatter plot.



(b) Screenshot of a graph plot.



(c) Screenshot of a tree plot.

Figure 5.5: Screenshots of scatter, graph and tree plots. All are available visualization types in the demonstrator web interface.

schedule regular transfer of messages to a destination address.

As ad-hoc construction of expressive example experiments that adequately showcase a specific protocol capability poses a challenge, several tested experimentation playbooks were created which when followed illustrate the respective protocol capability. In this section, we will go over a single exemplary playbook. More playbooks are available in Tables B.1 to B.3.

In Table 5.2 an exemplary playbook is given in tabular form. We assume the following assumptions hold before the operator begins performing the steps declared in the playbook:


- Demonstrator is set up correctly (as depicted in Fig. 5.1), i.e. all nodes and the laptop are powered on and connected to the ethernet switch.
- All transceivers are tuned to the same channel and there are no strong emitters of signals at carrier frequencies also used by WiFi in the immediate vicinity of the testbed setup.
- Testbed nodes are placed very close to each other (within a circular area of diameter 2 m or less), so near perfect link reliabilities can be expected.

The protocol stack that runs on the demonstrator testbed for the exemplary playbook in Table 5.2 (and Tables B.1 to B.3 as well) consists of a custom MAC layer³ capable of controlling retransmissions, the cTEx protocol (see Chapter 3), the dR_{\min} -routing protocol (see Chapter 4) and an application layer. As mentioned in Section 5.2, each protocol layer may register actions that can be triggered directly from the demonstrator web interface. There exist actions for most of the configurable parameters of each protocol. In the playbooks included in this thesis however only a subset of the available actions is used:

- *MAC layer*: This layer registers actions to set and remove artificial reliability limits on a link. This includes setting the limit to 0, which causes the MAC layer to drop all frames received over the link. The limit is implemented in a straight-forward manner: For every frame received, a random number from $(0,1)$ is drawn from an uniform distribution. If the number drawn is higher than the configured reliability limit, the current frame is dropped. The default reliability limit is set to 1.0.
- *cTEx layer*: The cTEx protocol layer also provides several actions for remote control through the operator of the demonstrator: activation and deactivation of the protocol and configuration of a specific dissemination strategy (details in Section 3.3.5).

³The MAC layer works on top of an unmodified 802.11n WiFi MAC

Table 5.2: Example Playbook for Showcasing dR_{\min} -Routing

Step		Operator Action	Audience Instruction / Observations	Comments
1	0:00	Activate cTEx and dR_{\min} -routing protocol layers	Observe topology detection process	dR_{\min} layer is passive, no application packets by default
2	3:00	<i>Control MAC:</i> Set reliability limit of all links to 0.0, apart from links in path $\langle 0,1,2,3 \rangle$	Link quality estimations adapt to configuration. Updates propagate quickly through network	Refer audience to topology plot
3	4:00	<i>Control Application:</i> Schedule regular packets from 0 to 3 ($r_{\min} = 0.9$, $i = 0.5$ s, $d_w = 10$ s)	Observe successful route discovery and operation	Refer audience to <i>received packets</i> counter for destination node
4	4:30	<i>Control MAC:</i> Remove limit on link (1,3) in both directions	No change at established route	Refer audience to topology plot, where the re-enabled link should be visible again
5	5:30	<i>Control MAC:</i> Limit reliability on link (2,3) to 0.0	End-to-end route reliability and link quality estimation drop. Route is re-established via $\langle 1,2,3 \rangle$ before route fails.	Refer audience to table of established routes, topology and link quality plots

- *dR_{min} layer*: At the dR_{min}-routing layer actions exist for activation and de-activation, as well as for toggling the route repair mechanism (see Section 4.1.4).
- *Application layer*: The application layer registers an action to schedule the creation of messages (once, or at regular intervals). Creation of a packet triggers a route search with the parameters r_{min} and d_w . r_{min} is the requested target reliability of the route and d_w the moving window duration the reliability limit has to be met for.

The playbook from Table 5.2 is intended to showcase the route discovery mechanism of the dR_{min}-routing protocol (for a detailed explanation of the discovery mechanism see Sections 4.1.2 and 4.1.3). Route discovery is carried out by dR_{min}-routing for two different scenarios. An initial route discovery is performed whenever a route (of required reliability) to the destination node set in an application packet is not yet established. Additional route discovery may occur when an already established route is about to fail. In this playbook, both scenarios are covered: We first let dR_{min}-routing discover a route in a constrained topology and subsequently disable a link (via a MAC layer action) that is part of the established route to trigger an additional route discovery.

In step 1 the cTEx and dR_{min}-routing protocol layers are activated by the operator on all nodes in the demonstrator testbed. After this step the dR_{min}-routing layer is still passive, the application does not create packets by default. The cTEx layer however commences topology detection and distribution through regular transmission of dedicated probing frames, as there is no application layer traffic yet. Before the operator continues with step 2, initial topology detection by cTEx should be complete. In step 2, the operator instructs the MAC layer to disable all links that are not on the path $\langle 0, 1, 2, 3 \rangle$. After the respective set of actions is issued, the topology plot in the demonstrator web interface quickly reflects the changes in link reliabilities. As in its default configuration the cTEx protocol embeds topology chunks informing other nodes as soon as a significant change in link quality is detected, the updated topology propagates quickly through the network. The process of adaptation to the updated topology is complete once the topology plot shows a line topology. In step 3, the application layer at node 0 is configured to create regular messages with the destination address of node 3 every 0.5 s. The messages are configured with an r_{min} target of 0.9 over a window of 10 s. As the testbed topology is a line with nodes $\langle 0, 1, 2, 3 \rangle$, there is only a single route option. The expected delay between triggering the action at the application layer to the begin of route operation is very short, usually below 50 ms and will likely be perceived as instant by the audience. After step 3, there is a tabular visualization in the demonstrator's web interface with an entry for the total number of received packets at each node. The entry for received packets at

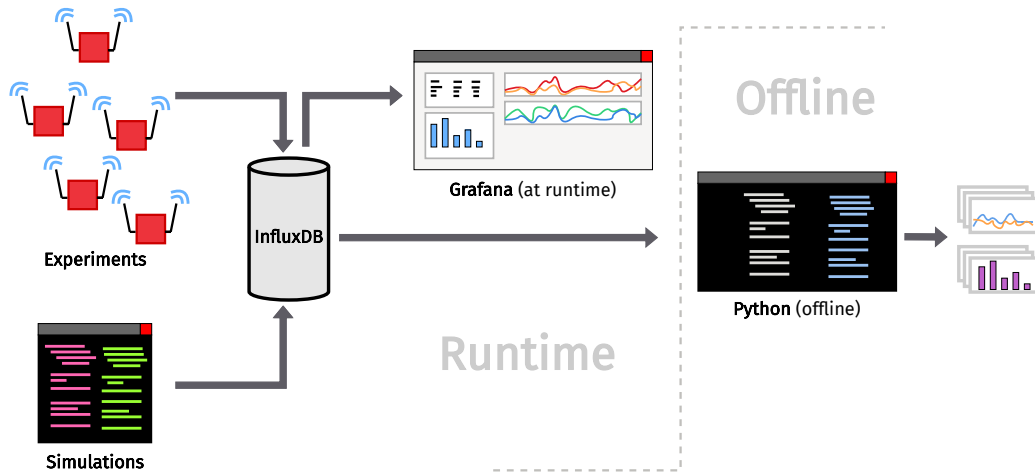


Figure 5.6: Information flow diagram of the experiment evaluation process.

node 3, which is the destination node of the route that was just established now increases twice per second. After the audience is convinced the route operation works as expected, the operator continues with step 4, where the link (1,3) is re-enabled via an action for the MAC layers of nodes 1 and 3. This does not affect the established route along $\langle 0, 1, 2, 3 \rangle$. The now enabled link between node 1 and 3 however reappears in the topology visualization. Before the operator proceeds with step 5, the estimated link reliability of link (1,3) should return to its original level (i.e. the level right before step 2). In the final step, reliability of link (2,3) is reduced to 0, again via an action to the MAC layers at nodes 2 and 3. As this link is part of the established route via $\langle 0, 1, 2, 3 \rangle$ the route repair mechanism of dR_{\min} -routing (at nodes 2 and 3) will soon initiate a shutdown of the established route. The next message that is created by the application layer on node 0 triggers the route discovery mechanism and a new route is established along the path $\langle 0, 1, 3 \rangle$. Again, this process is likely to be perceived as instantaneous by the audience. This concludes this exemplary playbook and the discussion of the demonstrator testbed.

5.5. Tools and Frameworks for Experiment Evaluation

As thorough evaluation of the communication protocols and algorithms presented in this thesis demands collection of detailed records of protocol and network state as well as dozens of performance metrics, several tools and frameworks for management of this data were applied in the process. Usage of these tools allows us to record experiment data in great detail first, and filter and analyze at a later stage. Fig. 5.6 illustrates the flow of data and also provides an overview of the topics covered in this section.

```
app_rx run_id="drmin-eval-17",route="82 -> 91",msg_id="JMoSo7" \
src=93,dst=91,size=512 16721712007163091000
```

Listing 5.2: InfluxDB line protocol example

At experiment runtime, protocol state, network state and performance metrics are recorded in the open source time series database *InfluxDB*⁴. An InfluxDB server accepts records (also called data *points*) over network via the so-called *line protocol*. A series of points that refer to the same sensor attribute form a so-called *measurement*. Each individual point consists of a measurement name, tags, and several fields of data in the form of keyed values. Points may also contain a timestamp. An example of a data point that corresponds to the reception of a message at the application layer is given in Listing 5.2. In this example, `app_rx` is the name of the measurement and `run_id`, `route` and `msg_id` are its tags. The field keys are `src`, `dst` and `size`. The timestamp in nanoseconds marks the end of the line. In relational database terms, a measurement (i.e. a collection of points with the same measurement name) is a table, and tags and fields represent columns. The difference between tags and fields is that tags are indexed, such that response times for queries that filter by tag are short.

Millions of these lines can be stored and queried efficiently with InfluxDB, without the need to have a database schema specified up front. The sophisticated mechanisms for automatic aggregation of lines for long-term archiving provided by InfluxDB are not used, as in this scenario all data are required at their original granularity for analysis.

Records stored in an InfluxDB database can be queried with the *Flux* query language. In Flux, measurements are selected, filtered and transformed through a chain of commands, where the output of the previous command is the input for the next. In Listing 5.3, an example script to obtain the number of received packets, aggregated over windows with a duration of 10s is provided. First, the bucket from which the records are to be fetched is specified, then the time range as a pair of Unix timestamps⁵. The subsequent `filter()` function uses the specified predicate to filter for records of a specific measurement name that are also tagged with the specified `run_id`. It also selects the column of interest, in this case `size`. The `window()` function assigns each point (i.e. row) that is output by the `filter()` function to a table that represents a 10s window, based on the timestamp of each row. The last function in the chain, `count()`, reduces each table to its respective size value.

Execution time for a single batch of experiments quickly exceeds several hours (for simulation as well as real-world experiments). Thus, it is often desirable

⁴Source code available at <https://github.com/influxdata/influxdb>

⁵A Unix timestamp is the number of seconds since January 1st, 1970 at 00:00 UTC.

```
1 from(bucket: "drmin/autogen")
2 |> range(start: 1672219930, stop: 1672220030)
3 |> filter(fn: (r) =>
4     r.run_id == "drmin-eval-17" and
5     r._measurement == "app_rx" and
6     r._field == "size")
7 |> window(every: 1s, duration: 10s)
8 |> count()
```

Listing 5.3: Flux query language example. This script calculates the number of packets received at the application layer over a moving 10s window.

to verify an experiment runs as expected already at runtime, so configuration mistakes are spotted early. For visualization of experiment data during runtime, the *Grafana*⁶ tool can be used. Grafana includes a web-based dashboard for visualization and analysis and has support for InfluxDB as database backend and also the Flux query language. This allows for reuse of most of the Flux scripts written to query data during offline analysis also for live monitoring, with only few modifications.

In Fig. 5.7, a screenshot of the Grafana dashboard is displayed. The dashboard supports filtering of data through definition of variables and time ranges. In our dashboard for instance, a variable definition that provides a list of available experiment `run_ids` was used to quickly select a specific experiment run for analysis. The `run_id` variable is automatically populated using the connected data source (in our case InfluxDB) and presented to the user as a dropdown selection menu.

For in-depth analysis of experiment data as well as tuning and debugging of the implemented communication protocols the Grafana dashboards lack flexibility. Hence, a combination of Flux (for querying InfluxDB) and Python (for data transformation and plotting) was used for this task. Python is an interpreted programming language with comprehensive support libraries for data analysis, data transformation, plotting and visualization.

5.6. Conclusion

The demonstrator framework presented in this chapter is intended to showcase complex communication protocols to an audience without a deep background in the respective field of research. However, a certain level of knowledge about communication protocols, wireless ad-hoc networks and the respective vocabulary of technical terms is still required. The requirements regarding the demon-

⁶Source code available at <https://github.com/grafana/grafana>

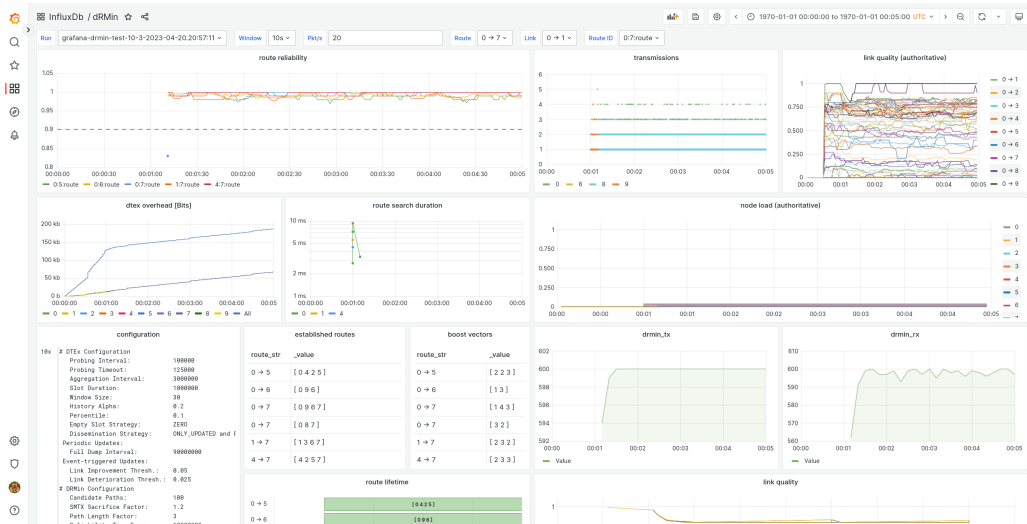


Figure 5.7: Screenshot of the Grafana dashboard.

strator testbed and framework stated in the introductory part of the chapter are addressed fully: The demonstrator web interface provides rich visualizations of network state, including real-time updates. The modular framework architecture facilitates extensions and addition of new protocols with low implementation effort. The testbed setup, consisting of only four network nodes, a laptop, 5 ethernet cables and a network switch is still portable, easy to set up and dismantle and requires no additional infrastructure apart from a source of electricity.

The exemplary playbook presented as part of this chapter illustrates how the demonstrator testbed can be used to showcase the route repair mechanism of dR_{\min} -routing. Additional examples of playbooks are provided in the appendix of this thesis.

Storage, processing, analysis and evaluation of vast amounts of experiment data has its challenges. Some of the tools and frameworks we used to overcome these challenges were introduced towards the end of this chapter. From the suggestions outlined there, the possibility for reuse of Flux query scripts from offline evaluation to visualize protocol behavior at runtime proved especially helpful.

6. Conclusions

6.1. Summary

Chapters 2 to 4 contain the principal contributions of this dissertation. In Chapter 2, the r_{\min} -routing algorithm is presented and evaluated. To discover r_{\min} -routes, a network model with statistical link reliabilities is utilized, where the link reliabilities are aggregated to model end-to-end route reliabilities. To meet the defined minimum route reliabilities, targeted retransmissions are employed to improve the reliability of individual links during route operation. To constrain bandwidth consumption, a transmission budget proportional to the number of hops in the route is distributed optimally with respect to the resulting end-to-end route reliability among the links of the route. A corresponding routing metric called SMTX for the efficient search for feasible routes is also defined in the chapter. As the minimum cost problem that corresponds to SMTX does not satisfy the greedy choice property, an efficient greedy route search algorithm does not exist. Hence, route discovery applies a heuristic that considers only a subset of paths. To further reduce the subset of candidate paths, a series of filtering steps targeted at removing sub-optimal paths is applied. At the end of the filtering process, all remaining paths satisfy the reliability target set by the application at a similar degree of bandwidth efficiency.

The r_{\min} -routing discovery algorithm is evaluated in simulation experiments on a variety of random-generated topologies of different sizes. The results are compared to those of a set of often cited route metrics from the literature, including shortest path and ETX. Results show that the r_{\min} -routing algorithm is capable of discovering routes that match ambitious reliability targets consistently. The level of consistency is not matched by any of the other routing metrics we evaluated.

A core foundation of r_{\min} -routing is the assumption that success of a particular communication attempt is statistically independent and identically distributed. However, this assumption does not hold for real networks in general and additional measures are necessary to retain validity of the underlying network model in reality.

r_{\min} -routing assumes the availability of complete, accurate and up-to-date topology information at every node of the network. In Chapter 3 we introduce cTEx, a protocol for topology detection and distribution that is supposed to meet this requirement set by r_{\min} -routing. cTEx uses both dedicated probing

6. Conclusions

frames and general network traffic for link quality estimation. The estimation of a link's quality takes place at the destination node. This involves operating the wireless transceiver in monitor mode and observing the sequence numbers in transmitted frames to calculate the frame delivery ratio. The resulting sequence of frame delivery ratios is then combined into a single value to represent the link quality metric. cTEx supposedly works with any link quality metric that bases its estimations on a sequence of frame delivery ratios. The chapter includes a reference link quality metric from literature, called OB-EWMA, which is also used for evaluation of cTEx. As cTEx shares some concepts with EAR evaluation includes a comparison of the two protocols. It is emphasized that the scopes of EAR and cTEx are not identical, as cTEx also includes a solution for the efficient distribution of obtained topology information. Experiment results show that cTEx in combination with OB-EWMA provides accurate, normalized and conservative link quality estimates. Key takeaways from the comparison with EAR are that cTEx uses less overhead, has a lower implementation complexity and provides a higher accuracy in link quality estimates. In addition to that, results indicate the topology information distribution is fast and bandwidth efficient.

r_{\min} -routing and the cTEx protocol both are applied by the protocol introduced in Chapter 4, dR_{\min} -routing. The chapter introduces a complete and decentralized implementation of a routing protocol that is capable of discovery, operation and maintenance of r_{\min} routes in a wireless ad-hoc network. For this, the route discovery mechanism from r_{\min} -routing is adopted and augmented through a subsequent cooperative refinement mechanism: A route request packet is sent along a candidate route based on topology information local to the respective source node, along the way, the candidate route is refined with topology information available at intermediate nodes that is more recent until the request packet reaches the destination node. Once a route is established it is continuously monitored. When a link's quality deteriorates significantly, the protocol may automatically slightly increase the transmission budget for the link temporally. If a route threatens to break, it is aborted preemptively. With the next packet a new route along a possibly different path is established. Local availability of topology information is contributed by cTEx. In simulation experiments, dR_{\min} -routing is evaluated and compared to OLSR, an often cited routing protocol for ad-hoc networks. The routes operated using dR_{\min} -routing match the specified reliability target and are more reliable than those discovered and operated by OLSR. A simulation experiment that compares the respective throughput of both protocols suggests dR_{\min} -routing is superior in this regard, too.

However, OLSR is a well-tested, mature and proven routing protocol for wireless ad-hoc networks and deployed in countless real-world applications. Hence, its fitness for purpose is beyond question. With respect to dR_{\min} -routing, only small, preliminary experiments were carried out on real hardware. The chap-

ter includes a discussion of necessary preconditions for a successful operation of dR_{\min} -routing on real hardware and elaborates on the consequences of assumptions made for the underlying network model. To retain validity of the network model, especially regarding the assumption that repeated communication attempts are Bernoulli trials, several measures are suggested. Using the Gilbert-Elliott model, the independence of success and failure of communication attempts on a link can be measured and quantified. A metric derived from the Gilbert-Elliott model may serve as an additional criterion during route discovery and may help to avoid links that do not exhibit the statistical properties necessary for effective link boosting. A second measure is traffic shaping. Distributing transmission events over time more evenly improves a link's statistical properties with respect to link boosting at the cost of increased end-to-end latencies.

In Chapter 5 a demonstrator framework and testbed setup with the purpose of showcasing the capabilities of the protocols presented in this thesis to a layman audience is introduced. The demonstrator supports addition of further protocols for showcasing with minimal implementation effort. The chapter also includes a playbook containing the description of steps to showcase the route repair functionality of dR_{\min} -routing. Towards the end of the chapter a survey of the tools and frameworks used in context of experiment evaluation and analysis for this thesis is given.

6.2. Limitations and Future Work

The lack of an evaluation of dR_{\min} -routing on real hardware is evident and should be addressed. For this, the additional measures to retain validity of the underlying network model as suggested in Section 4.4 have to be incorporated into the dR_{\min} -routing protocol. Implementation of a metric based on the μ value from the Gilbert-Elliott model is a rather straight-forward task, as the process of route selection already considers several different metrics. The second measure suggested likely requires significantly higher effort, as a compatible implementation of the suggested time token bucket algorithm has to be created and incorporated into the framework. This is to be followed by the often time-consuming process of tuning protocol parameters towards an optimal result.

The comparison between OLSR and dR_{\min} -routing should be extended with results from experiments on a real-world testbed. An implementation of OLSR is available, a replication of the simulation experiments carried out as part of this thesis on hardware will provide valuable insights on performance characteristics of both protocols and real-world applicability of our simulation results as well.

A. Appendix A

On the following pages, plots depicting the topology of the networks used for the evaluation of r_{\min} -routing in Section 2.4 are displayed. The topologies are all randomly generated with node position coordinates drawn from a uniform distribution.

A. Appendix A

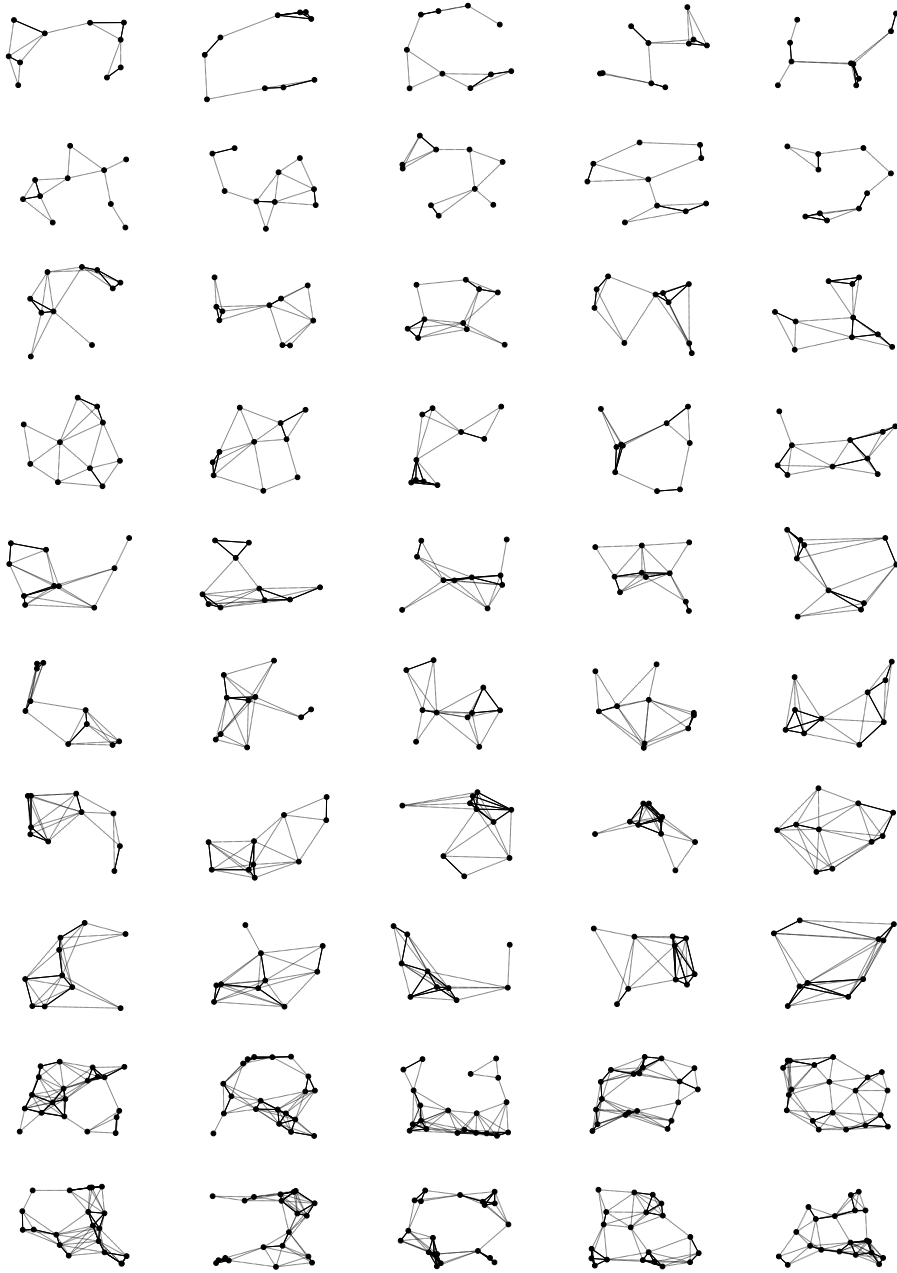


Figure A.1: Topologies for r_{\min} -Routing simulations

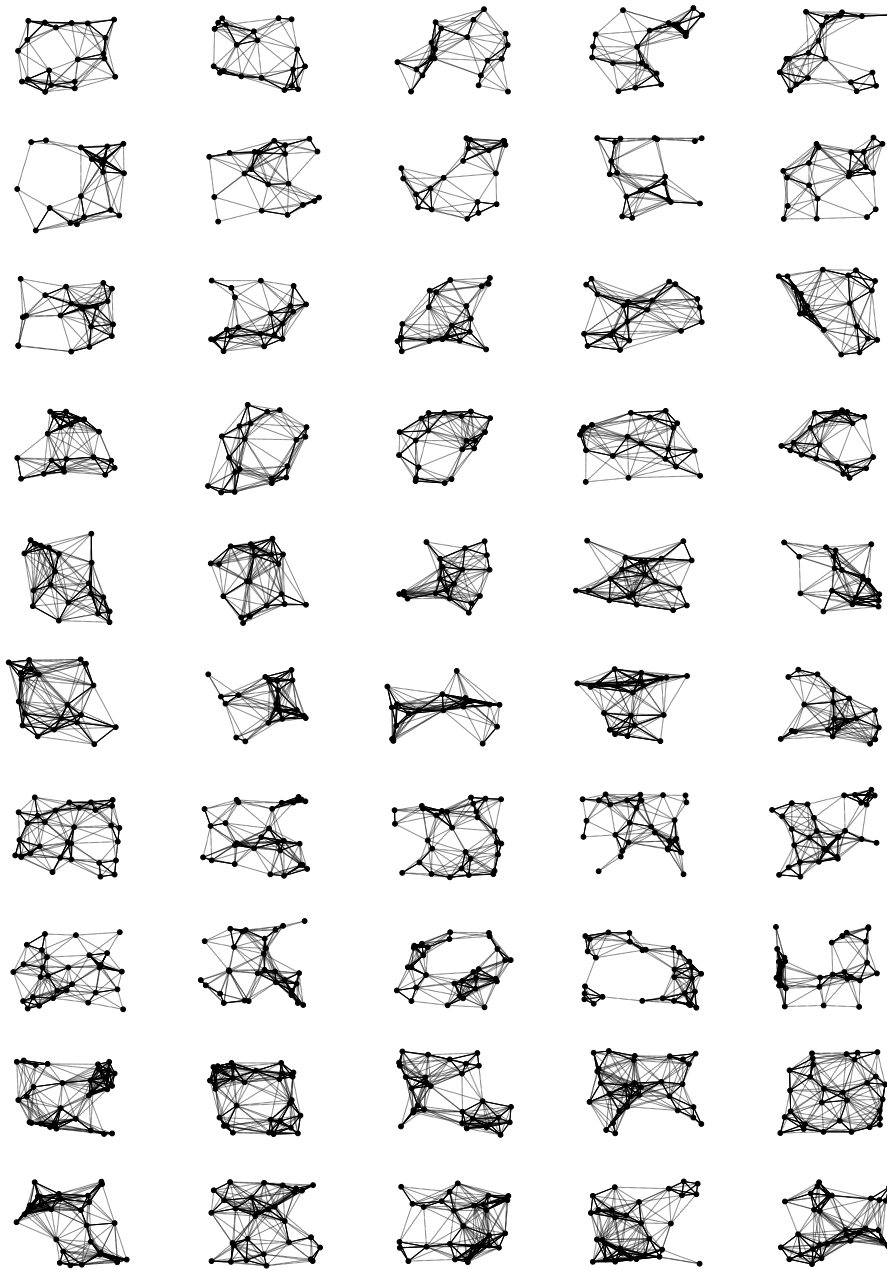


Figure A.2: Topologies for r_{\min} -Routing simulations

A. Appendix A

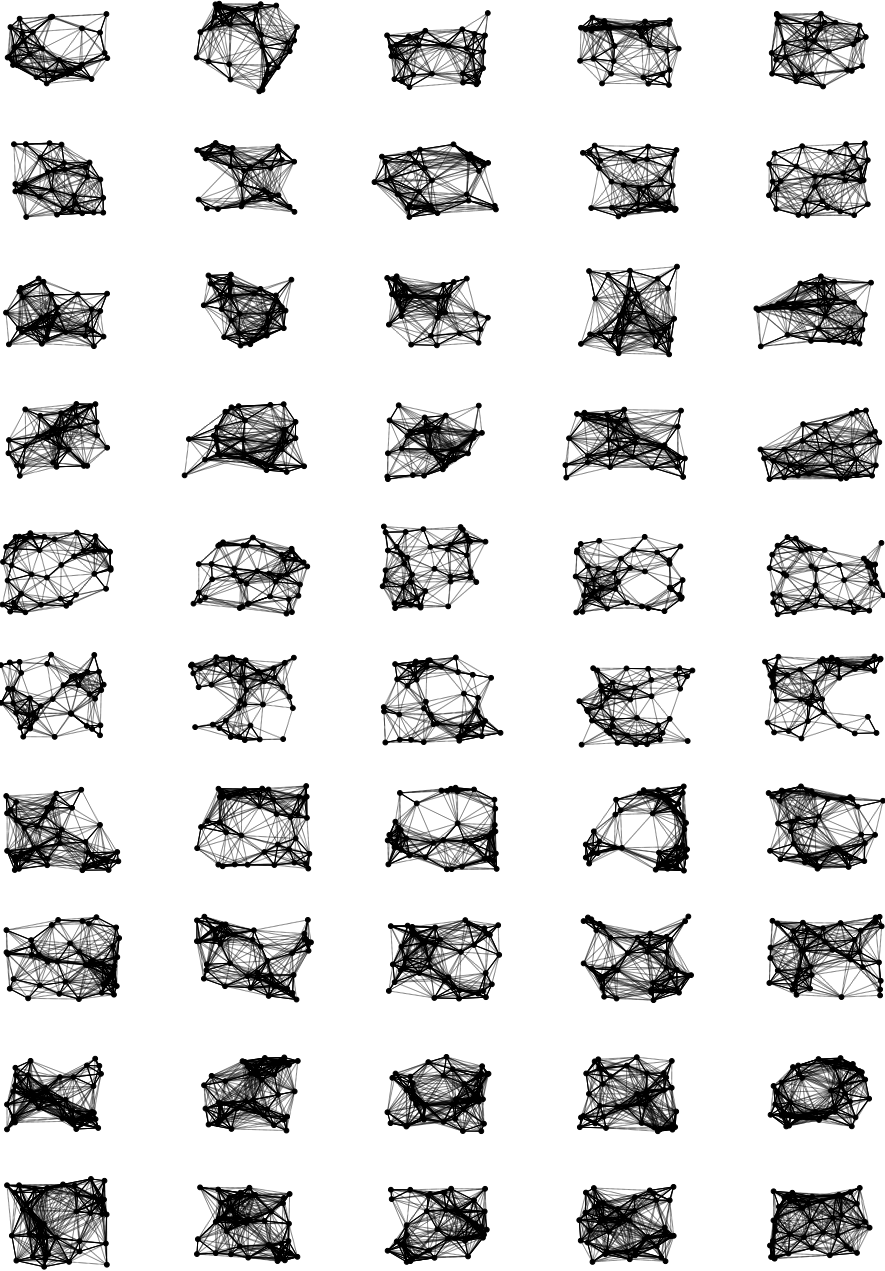


Figure A.3: Topologies for r_{\min} -Routing simulations

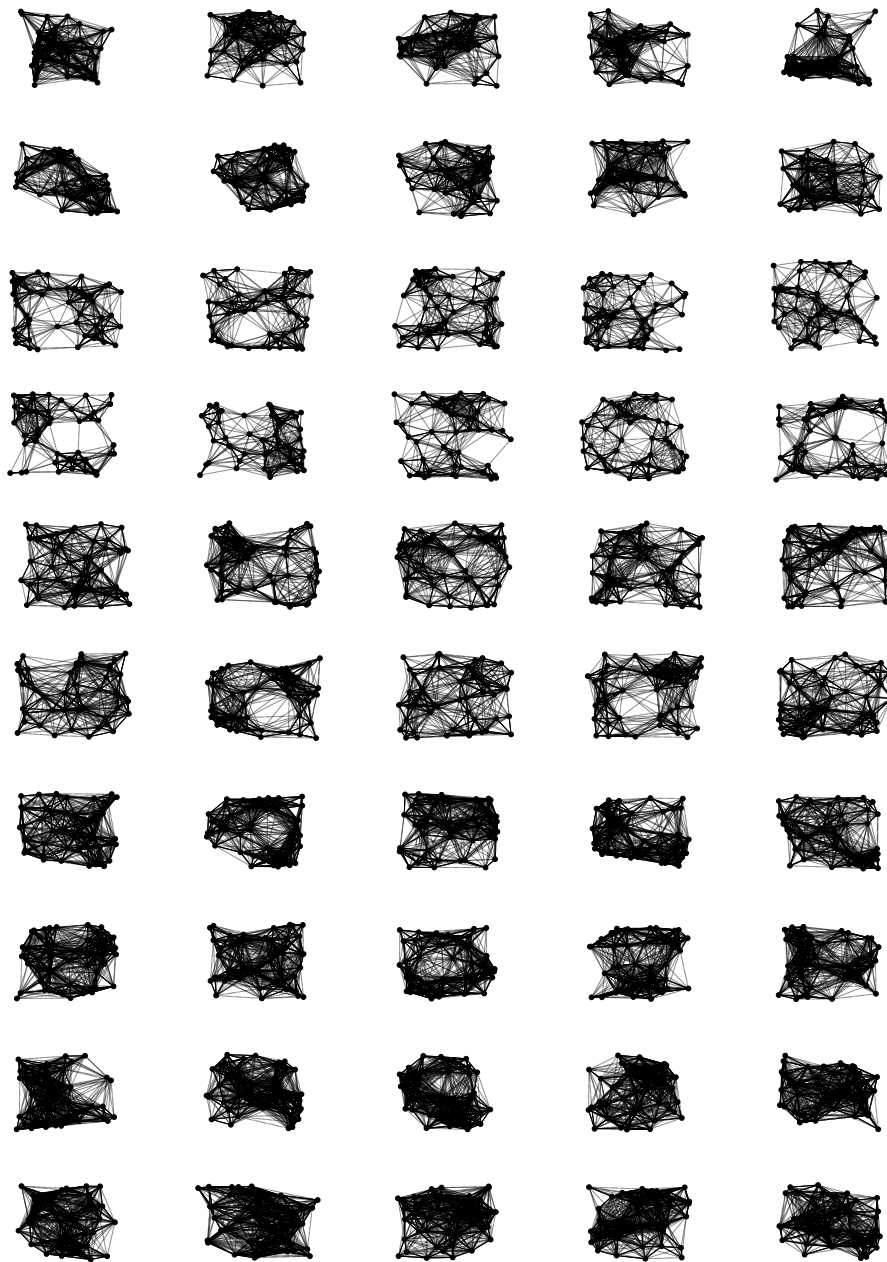


Figure A.4: Topologies for r_{\min} -Routing simulations

B. Appendix B

The tables displayed on the subsequent pages contain three playbooks intended to showcase the communication protocols developed in this thesis to an audience without a deep background in the respective field of research. A fourth playbook including a quick walkthrough is presented in Section 5.4.

Table B.1: Playbook for showcasing dR_{\min} -routing route repair mechanism


Step		Operator Action	Audience Instruction / Observations	Comments
1	0:00	Activate cTEx and dR_{\min} -routing protocol layers	Observe topology detection process	dR_{\min} layer is passive, route repair is enabled, no application packets by default
2	3:00	<i>Control MAC:</i> Limit reliability of all links at 0, except links in path $\langle 0, 1, 2, 3 \rangle$	Link quality estimations adapt to configuration. Updates propagate quickly through network	Refer audience to topology plot
3	4:00	<i>Control Application:</i> Schedule regular packets from 0 to 3 ($r_{\min} = 0.9$, $i = 0.5s$, $d_w = 10s$)	Observe successful route discovery and operation	Refer audience to <i>received packets</i> counter for destination node
4	5:00	<i>Control MAC:</i> Limit reliability of link (2,3) to 0.8	Link quality deteriorates, no change at established route	On-demand route repair allocates additional transmissions to keep the route working
5	5:30	<i>Control MAC:</i> Limit reliability on link (1,2) to 0.8	Link quality deteriorates, no change at established route	See above.
6	7:00	<i>Control MAC:</i> Restore realibility of links (1,2) and (2,3)	No change at established route, link quality increases	Refer audience to topology plot
7	8:00	<i>Control dR_{\min}:</i> Deactivate route repair mechanism	No change at established route	
8	8:10	<i>Control MAC:</i> Limit reliability of link (2,3) to 0.8	Link quality deteriorates, established route fails	As there is no route repair mechanism, the change in link quality causes end-to-end route reliability to drop below the configured target of $r_{\min} = 0.9$

Table B.2: Playbook for showcasing cTEx topology distribution mechanism



Step		Operator Action	Audience Instruction / Observations	Comments
1	0:00	Activate cTEx and dR_{\min} -routing protocol layers	Observe topology detection process	cTEx topology distribution is activated (event- and time-triggered), dR_{\min} layer is passive, no application packets by default
2	3:00	<i>Control MAC</i> : Limit reliability on link (2,3) to 0.5	Link quality estimation by node 3 adapts, other nodes are notified	Refer audience to per-node topology plots
3	4:00	<i>Control cTEx</i> : Configure topology distribution strategy to time-triggered only (every 1 minute)	No change in estimated link quality	
4	4:10	<i>Control MAC</i> : Restore reliability on link (2,3)	Link quality estimation by node 3 adapts, other nodes are <i>not</i> notified	Refer audience to per-node topology plots
5	5:10	<i>No action</i>	Nodes 0, 1, 2 receive updated link quality	Refer audience to per-node topology plots

Table B.3: Playbook for showcasing cTEx passive mode

Step		Operator Action	Audience Instruction / Observations	Comments
1	0:00	Activate cTEx and dR_{\min} -routing protocol layers	Observe topology detection process	cTEx starts topology exploration using active probing at rate 1/50 ms, dR_{\min} layer is passive, no application packets by default. Refer audience to probing frame counters.
2	1:30	<i>No action</i>	Topology exploration continues, probing frame counters increase	
3	3:00	<i>Control MAC:</i> Limit reliability of all links to 0, apart from links in path $\langle 0, 1, 2, 3 \rangle$	Link quality estimations adapt to configuration. Updates propagate quickly through network	Refer audience to topology plot
4	4:00	<i>Control Application:</i> Schedule regular packets from 0 to 3 ($r_{\min} = 0.9, i = 0.05\text{ s}, d_w = 10\text{ s}$)	Observe successful route discovery and operation	Refer audience to <i>received packets</i> counter for destination node
5	4:15	<i>Control Application:</i> Schedule regular packets from 3 to 0 ($r_{\min} = 0.9, i = 0.05\text{ s}, d_w = 10\text{ s}$)	Observe successful route discovery and operation	Refer audience to <i>received packets</i> counter for destination node
6	4:30	<i>No action</i>	Probe frame counters do not increase further as passive probing is used	
7	5:30	<i>Control MAC:</i> Restore reliability of all links	Previously non-existing links are discovered. Probe frame counters do not increase further as only passive probing is used	Refer audience to topology plot and probing frame counters.

Author's Contributions

- [KG21] C. Kohlstruck and R. Gotzhein. cTEx – A Configurable Topology Explorer for Wireless Ad-hoc Networks. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE, 2021.
- [KG22] C. Kohlstruck and R. Gotzhein. dRmin-Routing – A Decentralized Algorithm for Reliability-constrained Routing. In *2022 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1–9. IEEE, 2022.
- [KMG19] C. Kohlstruck, K. Mathews, and R. Gotzhein. rmin-Routing – Discovery and Operation of Routes in Wireless Ad-hoc Networks with Specified Statistical Minimum Reliabilities. In *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, pages 430–437, October 2019. ISSN: 0742-1303.

Bibliography

- [BELR07] Eric Bouillet, Georgios Ellinas, Jean-Francois Labourdette, and Ramu Ramamurthy. *Path Routing in Mesh Optical Networks*. John Wiley & Sons, 2007.
- [BKM⁺12] Nouha Baccour, Anis Koubâa, Luca Mottola, Marco Antonio Zúñiga, Habib Youssef, Carlo Alberto Boano, and Mário Alves. Radio link quality estimation in wireless sensor networks: A survey. *ACM Transactions on Sensor Networks*, 8(4), September 2012.
- [Bra68] James V Bradley. *Distribution-free statistical tests*. 1968.
- [CJ03] T. H. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626, October 2003.
- [Cla14] T. H. Clausen. The Optimized Link State Routing Protocol Version 2. RFC 7181, April 2014.
- [CN99] S. Chen and K. Nahrstedt. Distributed Quality-of-Service Routing in Ad Hoc Networks. *IEEE Journal on Selected areas in Communications*, 17(8):1488–1505, 1999.
- [DCABM03] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A High-throughput Path Metric for Multi-hop Wireless Routing. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, MobiCom '03*, pages 134–146, New York, NY, USA, 2003. ACM.
- [DPZ04] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in Multi-radio, Multi-hop Wireless Mesh Networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, MobiCom '04*, pages 114–128, New York, NY, USA, 2004. ACM.
- [Ell63] E. O. Elliott. Estimates of error rates for codes on burst-noise channels. *The Bell System Technical Journal*, 42(5):1977–1997, 1963.
- [FGJL07] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, and Philip Alexander Levis. Four-bit wireless link estimation. 2007.

Bibliography

- [Gil60] E. N. Gilbert. Capacity of a burst-noise channel. *The Bell System Technical Journal*, 39(5):1253–1265, 1960.
- [IEE11] IEEE. IEEE Standard for Information Technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements - Part 11. *IEEE Std 802.11s-2011 (Amendment to IEEE Std 802.11-2007)*, pages 1–372, September 2011. URL: <https://standards.ieee.org/ieee/802.11s/4243/>.
- [IEE16] IEEE. IEEE Standard for Information technology – Telecommunications and information exchange between systems Local and metropolitan area networks – Specific requirements - Part 11. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, December 2016. URL: <https://standards.ieee.org/ieee/802.11/5536/>.
- [IPQ06] Gabriel Ioan Ivascu, Samuel Pierre, and Alejandro Quintero. QoS Support based on a Mobile Routing Backbone for Ad-hoc Wireless Networks. In *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing, IWCMC '06*, pages 121–126, New York, NY, USA, July 2006. Association for Computing Machinery.
- [JEH⁺12] G. Jakllari, S. Eidenbenz, N. Hengartner, S. V. Krishnamurthy, and M. Faloutsos. Link Positions Matter: A Noncommutative Routing Metric for Wireless Mesh Networks. *IEEE Transactions on Mobile Computing*, 11(1):61–72, January 2012.
- [JM96] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Tomasz Imielinski and Henry F. Korth, editors, *Mobile Computing*, The Kluwer International Series in Engineering and Computer Science, pages 153–181. Springer US, Boston, MA, 1996.
- [Joh94] D. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In *1994 First Workshop on Mobile Computing Systems and Applications*, pages 158–163. IEEE, 1994.
- [KS06] Kyu-Han Kim and Kang G. Shin. On Accurate Measurement of Link Quality in Multi-hop Wireless Mesh Networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking, MobiCom '06*, pages 38–49, New York, NY, USA, September 2006. Association for Computing Machinery.

- [LCZH17] D. Liu, Z. Cao, Y. Zhang, and M. Hou. Achieving Accurate and Real-Time Link Estimation for Low Power Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 25(4):2096–2109, August 2017.
- [LL17] Christopher J. Lowrance and Adrian P. Lauf. Link Quality Estimation in Ad Hoc and Mesh Networks: A Survey and Future Directions. *Wireless Personal Communications*, 96(1):475–508, September 2017.
- [Mat22] Kiran Mathews. *Protocols and Algorithms for reliability-constrained Quality-of-Service Routing in IEEE 802.11 (WiFi) Networks in Contention Mode*. PhD thesis, Technische Universität Kaiserslautern, 2022.
- [MF07] A. Munaretto and M. Fonseca. Routing and Quality of Service Support for Mobile Ad Hoc Networks. *Computer Networks*, 51(11):3142–3156, 2007.
- [MG21] Kiran Mathews and Reinhard Gotzhein. OB-EWMA: A Link Metric for Reliability-constrained Routing in Wireless Networks. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7, March 2021. ISSN: 1558-2612.
- [PR99] C. E. Perkins and E. M. Royer. Ad-hoc On-demand Distance Vector Routing. In *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.
- [PTMS⁺06] Diego Passos, Douglas Vidal Teixeira, Débora C Muchaluat-Saade, Luiz C Schara Magalhães, and Célio V N Albuquerque. Mesh Network Performance Measurements. In *5th International Information and Telecommunication Technologies Symposium, 2006*, page 8, 2006.
- [SKAL08] Kannan Srinivasan, Maria A Kazandjieva, Saatvik Agarwal, and Philip Levis. The β -Factor: Measuring Wireless Link Burstiness. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 29–42, 2008.
- [TMK15] A. T. Tran, D. D. Mai, and M. K. Kim. Link Quality Estimation in Static Wireless Networks With High Traffic Load. *Journal of Communications and Networks*, 17(4):370–383, August 2015. Conference Name: Journal of Communications and Networks.
- [vS10] Maarten van Steen. *Graph Theory and Complex Networks*. Marten van Steen, 2010.

Bibliography

- [WTC03] Alec Woo, Terence Tong, and David Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys '03*, pages 14–27, New York, NY, USA, November 2003. Association for Computing Machinery.
- [Yen71] Jin Y. Yen. Finding the K-Shortest Loopless Paths in a Network. *Management Science*, 17(11):712–716, July 1971.
- [YMSF07] Ming Yu, Aniket Malvankar, Wei Su, and Simon Y. Foo. A link availability-based QoS-aware routing protocol for mobile ad hoc sensor networks. *Computer Communications*, 30(18):3823–3831, December 2007.
- [ZSA10] H. Zhang, L. Sang, and A. Arora. Comparison of Data-Driven Link Estimation Methods in Low-Power Wireless Networks. *IEEE Transactions on Mobile Computing*, 9(11):1634–1648, November 2010. Conference Name: IEEE Transactions on Mobile Computing.
- [Zwe16] Katharina A. Zweig. *Network Analysis Literacy*. Springer Vienna, 2016.

Christopher Kohlstruck

Curriculum Vitae

Education

- 2018-2023 **PhD in Computer Science** at TUK, Networked Systems Group
- PhD Thesis *Development and Evaluation of Protocols for the Operation of Wireless Ad-Hoc Networks with Quality-of-Service Requirements*
Supervisor: Prof. Dr.-Ing. Reinhard Gotzhein
- 2016-2018 **M.Sc. in Computer Science** at TUK, Grade 1.6
Specialization: Communication Systems, Minor: Business Administration
- Master Thesis *QoS Routing – Discovery and Assessment of Routes with Statistical Minimum Reliability*
Supervisor: Prof. Dr.-Ing. Reinhard Gotzhein and Dr.-Ing. Kiran Mathews
- 2011-2016 **B.Sc. in Computer Science** at TUK, Grade 2.3
Specialization: Distributed and Networked Systems, Minor: Business Administration
- Bachelor Thesis *Sensing in Cognitive Radio Networks*
Supervisor: Prof. Dr.-Ing. Reinhard Gotzhein and Dr.-Ing. Markus Engel

Experience

- 2018-2022 Research Associate at TUK, Network Systems Group
- 2016-2018 IT-Service at TUK, Faculty of Business Studies and Economics
- 2014-2017 Student Assistant and tutor at TUK, Network Systems Group

Peer-Reviewed Publications

- 2022 dRmin-Routing – A Decentralized Algorithm for Reliability-constrained Routing
C. Kohlstruck, R. Gotzhein
Proceedings of International Wireless Communications and Mobile Computing Conference (IWCMC 2022)
- 2021 cTEx – A Configurable Topology Explorer for Wireless Ad-hoc Networks
C. Kohlstruck, R. Gotzhein
Proceedings of International Conference on Computer Communications and Networks (ICCCN 2021)
- 2019 rmin-Routing – Discovery and Operation of Routes in Wireless Ad-hoc Networks with Specified Statistical Minimum Reliabilities
C. Kohlstruck, K. Mathews, R. Gotzhein
Proceedings of IEEE 44th Conference on Local Computer Networks (LCN 2019)
- 2018 The Selective Clustering Energy Detector for Cognitive Radio Networks – Conceptual Design and Experimental Assessment
C. Kohlstruck, M. Engel, R. Gotzhein
Proceedings of International Conference on Wireless Communications & Mobile Computing Conference (IWCMC 2018)