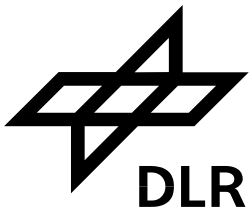


Position-Based Routing in Dynamic Mesh Networks for Applications with Micro-UAVs

Bachelorarbeit

für die Prüfung zum
Bachelor of Engineering

im Studiengang Informationstechnik
an der Dualen Hochschule Baden-Württemberg Mannheim

von
Lukas Magel

Bearbeitungszeitraum	04.07.2016 bis 26.09.2016
Kurs, Matrikelnummer	MA-TINF13ITIN, 9273080
Ausbildungsfirma	Deutsches Zentrum für Luft- und Raumfahrt e. V.
Abteilung, Standort	KN-COS, Oberpfaffenhofen
Betreuer der Ausbildungsfirma	Thomas Wiedemann
Gutachter der Dualen Hochschule	Prof. Dr. Rainer Colgen

Eidesstattliche Erklärung

Gemäß §5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011.

Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Oberpfaffenhofen, den 22. September 2016

Abstract

Many of the ad hoc routing protocols developed in the recent years are intended for the use in large-scale, static networks but provide poor performance in small, dynamic environments with moving nodes. This thesis proposes a new routing algorithm which employs a special position-aided metric to fill this gap.

The presented metric rates the wireless links between nodes in the network by predicting the path loss which the radio signals experience during propagation. The path loss is estimated using an empirical log-distance propagation model based on the Euclidean distance between two nodes. The model is maintained live and refitted with new measurements on-the-fly. The resulting link ratings are periodically distributed in the network and used by each node to build and maintain a graph of the network. The graph is used by the individual nodes to determine the most stable paths in the network and to forward messages accordingly.

The algorithm is successfully evaluated through multiple outdoor experiments which represent likely scenarios in a dynamic network. Compared to state-of-the-art protocols, it delivers considerably better results in dynamic environments which include moving nodes.

Zusammenfassung

Viele der Ad Hoc Routing Protokolle, welche in den letzten Jahren entstanden sind, wurden speziell für den Einsatz in statischen Umgebungen mit vielen einzelnen Knoten entwickelt und eignen sich nur eingeschränkt für den Einsatz in Netzwerken mit hoher Dynamik und sich bewegenden Knoten. Der in dieser Bachelorarbeit vorgeschlagene Algorithmus verwendet eine speziell für den Einsatz in dynamischen Netzwerken konzipierte, positionsgestützte Metrik, um diese Lücke zu füllen.

Die Metrik bewertet die Verbindungen zu anderen Knoten im WLAN Netzwerk basierend auf dem Pfadverlust, welchem die elektromagnetischen Wellen während der Übertragung ausgesetzt sind. Der Pfadverlust wird mit Hilfe eines empirischen Log-Distance Modells geschätzt, welches kontinuierlich mit live aufgenommenen Messungen aktualisiert wird. Die Bewertungen der Verbindungen zwischen den Knoten werden periodisch im Netzwerk verteilt und von jedem Knoten verwendet, um einen vollständigen Graphen des Netzwerks zu konstruieren und aktuell zu halten. Der Graph wird von den Knoten genutzt, um die stabilsten Routen im Netzwerk zu bestimmen und Pakete anhand dieser Routen durch das Netzwerk zu leiten.

Der vorgeschlagene Algorithmus wurde erfolgreich in mehreren Outdoor Experimenten getestet, welche typische Szenarien in einem mobilen Netzwerk nachstellen sollten. Im Vergleich mit aktuellen State-of-the-Art Routing Protokollen erzielte der Algorithmus deutlich bessere Resultate in einem Netzwerk mit sich bewegenden Knoten.

Contents

List of Figures

List of Tables

List of Abbreviations

MAV	**M**icro **A**erial **V**ehicles
DLR	**D**eutsches Zentrum für **L**uft- und **R**aumfahrt (German Aerospace Center)
API	**A**pplication **P**rogramming **I**nterface
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IP	**I**nternet **P**rotocol
ISO	**I**nternational **O**rganization for **S**tandardization
OSI	**O**pen **S**ystem **I**nterconnection
TCP	**T**ransmission **C**ontrol **P**rotocol
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
AP	**A**ccess **P**oint
OLSR	**O**ptimized **L**ink **S**tate **R**outing
B.A.T.M.A.N.	**B**etter **A**pproach **T**o **M**obile **A**dhoc **N**etworking
DSR	**D**ynamic **S**ource **R**outing
AODV	**A**d-Hoc **O**n-Demand **D**istance **V**ector Routing
GPS	**G**lobal **P**ositioning **S**ystem
UDG	**U**nit **D**isk **G**raph
GPSR	**G**reedy **P**erimeter **S**tateless **R**outing
GOAFR	**G**reedy **O**ther **A**daptive **F**ace **R**outing
IMU	**I**nertial **M**easurement **U**nit
RAM	**R**andom-**A**ccess **M**emory
USB	**U**niversal **S**erial **B**us
ROS	**R**obot **O**perating **S**ystem

1 Introduction

In natural disaster situations like tsunamis or earthquakes, disaster relief teams need to be able to quickly assess the situation at hand and take appropriate actions. In order to do so, the teams require reliable information such as aerial imagery of the disaster. Micro Aerial Vehicles (MAVs) present an efficient and convenient measure to support teams in the gathering of air based information. They are low cost, very maneuverable and can easily be deployed without the need for a runway or additional ground personnel besides an operator. In amateur as well as professional photography, MAVs have already proven their advantages over conventional measures and found widespread application. Due to these characteristics, they are especially well suited for deployment in a swarm of multiple MAVs. In such a formation, they can quickly cover large areas and take precise measurements while at the same time being more robust and less prone to failure than a single aircraft.

The institute of Communications and Navigation within the German Aerospace Center (DLR) develops intelligent algorithms that enable swarms of MAVs to function in a decentralized manner and solve a common task without support of a ground-based, coordinating instance. This task includes but is not limited to the sensing of an area for disaster relief purposes. Other applications comprise the exploration of an a priori unknown environment to create an obstacle map, a height profile or a survey of the toxic pollution concentration of an area. The decentralized nature of the swarm makes it more robust as there is no single point of failure and individual MAVs, also called agents, can be exchanged easily.

1.1 Motivation

A swarm of multiple agents can only function properly and pursue a certain goal if it is equipped with a proper communication link. This link is used by the agents to exchange information and data regarding the progress of the task. Like the swarm itself, the communication link must also be designed in a decentralized manner. This means that individual agents in the

swarm communicate directly with each other using only the radio equipment carried by them. They are not allowed to utilize any type of common infrastructure such as a ground-based relay. This restriction can significantly impact the mobility and range of the swarm, as two agents must reside in their mutual radio range in order to communicate with each other. If the underlying algorithm assumes an n-to-n communication between all agents, the swarm range is even further limited to a circle with the same radius as the radio range of a single agent. The restriction can however be mitigated significantly if one agent does not only function as origin or destination of data but also forwards data not intended for itself. This concept is pursued by dynamic routing protocols. They discover links between nodes in a wireless network on-the-fly and enable communication between distant peers by relaying the data through other nodes. Research in the field of dynamic routing protocols has produced several different implementations which target specific applications. One possible use case is the creation of so called mesh networks to group together the access points of private households and form a city-wide network.

Applications in swarm exploration do however represent a challenging environment for such routing protocols. The individual agents move at significant speeds and force the protocol to quickly react to changes in the topology. Experiments using established dynamic routing protocols have shown that their applicability for swarm exploration is limited due to the mobile nature of the swarm. Instead, a new approach is required, which can react more quickly to changes in the topology and determine stable routes in a dynamic environment.

1.2 Overview

This thesis proposes a new routing algorithm, intended to provide a better performance in mobile networks which operate in a line-of-sight environment, such as swarms of MAVs. The proposed algorithm detects and rates links between different agents in the network based on the path loss that a communication signal experiences on its path from one agent to another. The path loss of an electromagnetic signal describes the reduction in power density that the signal experiences as it propagates through space. Multiple publications have shown that the path loss of a link correlates with its stability and hence provides a means to rate the quality of a connection. In order for the algorithm to exploit this correlation, it must be able to estimate the path loss. One possible approach would be to measure the path loss of transmitted signals live and use the result directly as rating for the link. Since the path loss measurement can

fluctuate heavily over the course of less than a second, this approach would not provide a stable rating. Instead, a model is used which approximates the given conditions over a longer period of time and therefore yields a more stable rating. Since each agent in the swarm already carries a GPS receiver and hence knows the distances to all other agents in the swarm, the proposed algorithm estimates the expected path loss based on the distance between two agents.

The correlation between distance and path loss is described through an empirical log-distance model, which approximates the expected path loss for a given distance. In order for this model to provide meaningful results, its parameters must be fitted with existing measurements of the path loss for given distances. Since the propagation conditions can change drastically for different environments and times, these measurements are taken live, on-the-fly, and the parameters of the model function are periodically refitted with these new measurements to reflect the changing conditions. The parameter fitting is performed using a least squares regression, which calculates the set of parameters that minimizes the error between model and measurements and therefore fits the current conditions best.

Every agent in the swarm maintains such a model function for each other agent in its radio range, and uses it to periodically rate the links which originate from these agents. Links which exhibit a high path loss are considered unstable and are not used for communication. The resulting set of stable links and their corresponding rating is called the neighborhood table of each agent and is distributed in the swarm. Every agent uses its own neighborhood table as well as the received tables to build a graph of the network. In it, the vertices represent the agents and the edges denote the links between the agents. The algorithm uses this graph to determine the most stable paths in the network to other nodes which are not in direct communication range. Whenever an agent intends to send data to a peer, it consults these paths and forwards the data to the next agent in the path leading to the destination. This forwarding of data is called routing and is performed for all packets no matter if they were sent by a local process on the agent itself or originated from a different peer. The network graph and the corresponding paths are updated periodically whenever new neighborhood tables are received from the other agents in the network.

The performance of the routing algorithm is evaluated in several outdoor experiments that model different uses cases. This includes static as well as mobile scenarios. The results are evaluated individually and compared to existing results of other routing protocols where applicable.

The thesis is organized as follows: Chapter 2 provides an overview of the current state

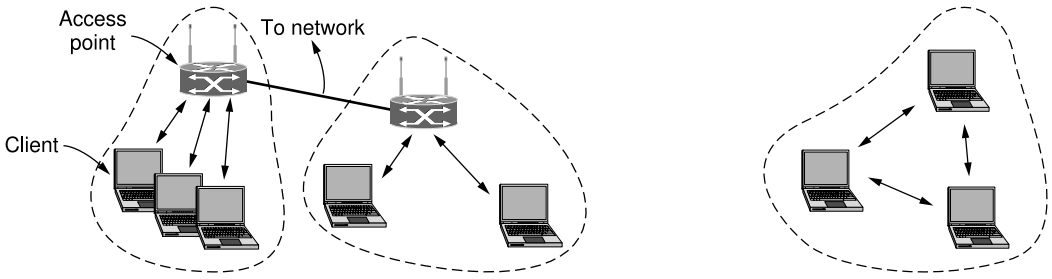
of the art concerning dynamic routing protocols and signal propagation models. The first section in chapter 2 outlines the different existing routing protocols as well as their strengths and weaknesses in relation to mobile networks. The second section presents different radio propagation models for path loss estimation and outlines the log-distance path loss model in greater detail. Chapter 3 provides a theoretical definition of the routing algorithm, which comprises the definition of the network graph used by the agents, the routing algorithm itself, and the least squares regression used to estimate the path loss model parameters. Chapter 4 describes the experiments which were conducted to evaluate the routing algorithm. The first section of chapter 4 outlines the experimental setup and the implementation of the routing algorithm used for the experiments. The following sections evaluate the proposed path loss metric as well as the routing algorithm itself. Chapter 5 puts the achieved results into perspective and provides an outlook.

2 State of the Art

The following chapter will first give an introduction to different dynamic routing protocol types, i.e. topology and position based protocols, as well as their suitability for swarm exploration. It then continues to explore the different approaches to model the propagation of radio signals and proposes the use of a simple log-distance model.

2.1 Routing in Ad Hoc Networks

In wireless local networking (WLAN), the 802.11 standard has become the predominant form of communication (see [1, pp. 299 sq.]). It features two different modes of operation: the infrastructure as well as the ad hoc mode. Figure 2.1 depicts both modes in contrast to each other. The infrastructure mode is used to associate multiple devices with an Access Point (AP), which provides access to a remote network like the Internet. All communication between the different devices as well as to and from the remote network is channeled through the AP, which functions as a network bridge.



(a) Infrastructure Mode **(b)** Ad Hoc Mode

Figure 2.1: WLAN 802.11 communication modes [1, p. 300]

Contrary to this, the ad hoc mode does not rely on a central gateway (see [1, pp. 299,389]). All devices are directly associated with one another and exchange messages peer-to-peer. This creates a decentralized network of independent nodes without requiring any pre-existing

infrastructure. In order for two nodes in the network to communicate with each other they must be in direct radio contact (see [1, p. 389]). This cannot always be achieved due to a limited radio range or obstacles, which block the propagation of the signal. Ad hoc routing protocols, also called mesh protocols, solve this shortcoming by turning every node in the network into a router. Such a node does not only act as origin or destination of messages but also forwards messages which are not intended for itself on their path from source to destination. As a result, a node in an ad hoc network can communicate with every other node as long as there exists a direct or indirect path between them. Contrary to wired networks, the topology of such a network can be highly dynamic, as node movement or even changing radio conditions have an impact on the links between the nodes in the network. Over the past years, several routing protocols have emerged which are tailored to handle these special conditions. The following sections give an overview of different types of routing protocols in ad hoc networks as well as their shortcomings.

The first group of ad hoc routing protocols is generally referred to as *topology routing protocols* (see [2, p. 621]) and will be described in further detail in the following section.

2.1.1 Topology Routing Protocols

Topology routing protocols are generally classified as either proactive or reactive protocols (see [3, p. 29]). Proactive protocols continuously detect routes to other nodes in the network, normally by the means of small *hello* frames that are periodically broadcasted to all neighbors. In contrast to this, reactive protocols only determine a route to a certain destination node when it is required. Both protocol types merely rely on the information they can gather through direct and indirect communication, or passive observation of the network. Prominent examples of topology routing protocols include Optimized Link State Routing (OLSR), Better Approach To Mobile Adhoc Networking (B.A.T.M.A.N.), and Babel (proactive) as well as Dynamic Source Routing (DSR), Ad-Hoc On-Demand Distance Vector Routing (AODV) and the IEEE 802.11s standard (reactive) (For more information on the routing protocols, please see appendix A). Several of these protocols have been designed for and are often deployed in static environments. The Babel protocol was specified by Chroboczek in [4] for the use in unstable wired or wireless networks. OLSR and B.A.T.M.A.N. are used by the German Freifunk community to build city-wide mesh networks with off-the-shelf routers and custom software[1]. Likewise, many of the experiments in research papers, which evaluate the performance of

[1] See `https://freifunk.net/en/` for more information on the Freifunk initiative

individual protocols, were conducted in static environments as seen in [5], [6], or [7].

The intended deployment scenario in swarm exploration on the other hand is highly dynamic and fast-changing since all nodes in the network are mobile. This places entirely different demands on the protocols than in a static environment. Nodes need to exchange crucial information, like their position or intended destination vector, in a timely manner, which makes prolonged interruptions unacceptable. In [5], Abolhasan et al. evaluated the convergence time of Babel, B.A.T.M.A.N., and OLSR after a node failure. The experiment yielded average recovery times of 14.4 s for Babel, 31.5 s for B.A.T.M.A.N., and 61.8 s for OLSR. While these interruptions may be acceptable for a community network, a swarm of quadcopters cannot wait for such an amount of time before a route is reestablished. In [8], Zeiger et al. examined the dynamic route adaption capabilities of AODV, DSR, OLSR, and B.A.T.M.A.N.. While B.A.T.M.A.N. was unable to detect routes with more than one intermediate node, AODV and OLSR needed more than 30 s to reestablish a route after the original route had degraded. Only DSR produced acceptable results with a maximum rerouting time of 2.7 s. A similar experiment with multiple mobile scenarios was conducted in [9] by Liechti. The different scenarios evaluated the reaction time of B.A.T.M.A.N. and OLSR to changes in the topology, which in some cases spiked to more than 20 s for both protocols. Our own experiments with B.A.T.M.A.N., Babel and the IEEE 802.11s standard in [10] yielded a very similar result. The protocols were not able to react to changes in the network topology in a timely manner, which resulted in frequent route changes or temporary (and in some cases permanent) connection drops (see section 4.3.2 for a summary of the data presented in [10]).

One possible explanation for this unstable behavior can be found when looking at the metric employed by the different protocols. The metric serves as a rating for the links to other nodes in the network and indicates their stability as well as their quality. Babel, B.A.T.M.A.N. and OLSR all utilize a metric that rates the link quality by periodically sending small *hello* frames to neighboring nodes (see [11] for Babel and OLSR as well as [3, pp. 73-79] for B.A.T.M.A.N.). By observing the network for a certain amount of time, the nodes can estimate the quality of the links based on the percentage of received *hello* packets. The accuracy of the measurement increases the longer the network is observed. In a dynamic network however, the topology changes too quickly for the measurement to provide a meaningful result, and its significance is therefore rather limited.

Additionally, the *hello* frames, which are broadcasted by the nodes, are relatively small in size compared to a full data frame of about 1500 Byte (see [1, p. 556]). In [12], Lee et

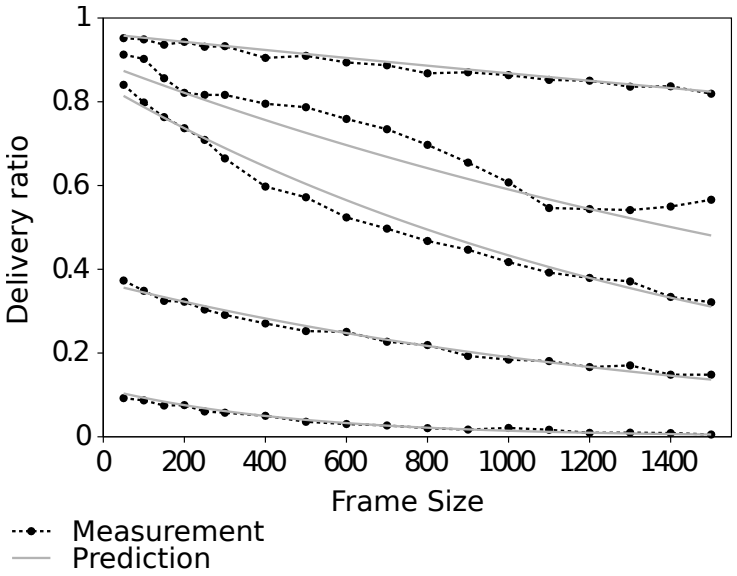


Figure 2.2: Packet delivery rate at varying frame sizes for different routes, as measured in [11]. The black dotted lines depict the measured values for each route, whereas the overlaid thin gray lines illustrate the predicted packet delivery rate for the route.

al. demonstrated that the size of a frame has a significant impact on its successful delivery. According to Couto et al. in [11], the error probability scales exponentially with the frame size as shown in figure 2.2. This bias towards smaller packets distorts the link quality measurement. A link could possibly have degraded to a point where a data frame cannot be successfully transmitted but the link is still open for smaller *hello* frames. This behavior can be observed in [10], where protocols frequently select a route which is stable enough for *hello* frames but not for data frames. This selection of routes subsequently leads to a significant packet loss, which in turn has a great impact on higher layer protocols such as the Transmission Control Protocol (TCP). TCP was originally designed to function in large-scale wired networks such as the Internet (see [1, p. 552]). When a route experiences packet loss, TCP interprets this as a sign of network congestion and reduces the transmission rate to match the maximum available bandwidth. In wireless networks however, packet loss is more likely to be caused by unstable links than by a router buffer congestion. As a consequence, the correct reaction to high packet loss in a wireless network would be to immediately resend lost frames. Yet, TCP reacts to this packet loss by drastically reducing the bandwidth. In [13], Cheng Peng Fu and Liew evaluate the performance of two different TCP congestion control algorithms under random packet loss situations. The results show that the data throughput degrades drastically

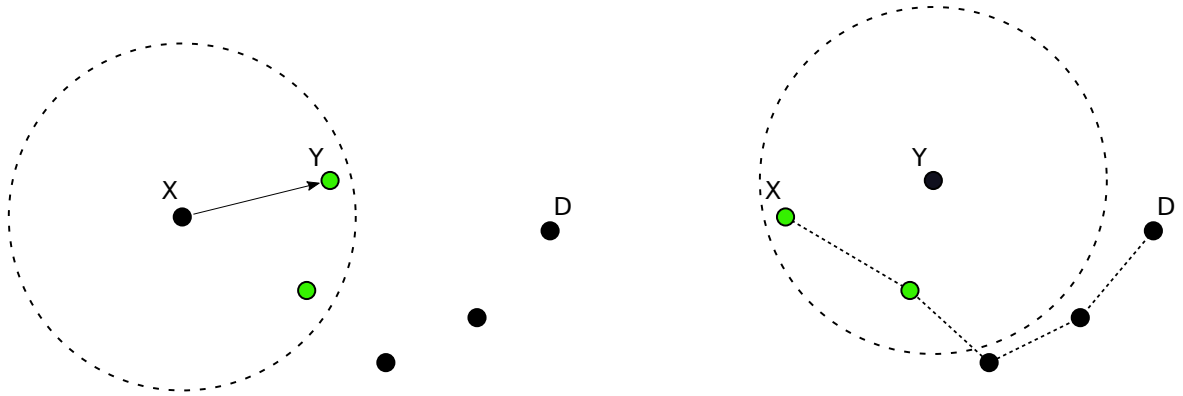
for both algorithms at a random packet loss of 10 % or more. However, many applications rely on TCP as transport protocol to reliably transmit data over the network. In order to ensure the correct functioning of higher level applications over TCP, a routing protocol must thus be capable of finding the most stable route and reduce the packet loss.

As outlined above, topology routing protocols experience many issues in dynamic environments due to the high mobility. These problems make a different category of ad-hoc routing protocols, called geographic routing, appealing for dynamic environments. In geographic routing, it is assumed that all nodes in a network are aware of their geographic position and utilize this information to forward messages to their destination. The next section outlines the different approaches to geographic routing and explores their suitability for dynamic environments.

2.1.2 Geographic Routing Protocols

Geographic routing protocols, which are also referred to as location-based or position-based protocols, rely on a positioning system like GPS to determine the location of all nodes in the network. The position information is then used by the protocol to forward packets in the direction of the destination. Geographic routing algorithms generally require all nodes to be located in a two-dimensional plane, and follow one of two different approaches: greedy forwarding or face routing (see [2, p. 622]). Both approaches require each node to maintain a table of its local neighbors which are positioned in its radio reception area (see [14, p. 52]). Nodes learn the position of their local neighbors by the means of periodic beacon packets, which are broadcasted by each node and carry its position (see [15] and [16]). The position of other nodes which are located outside a node's radio range can be queried through a location service that is maintained by the network (see [17] for an exemplary implementation).

Greedy forwarding is a very simple approach to geographic routing that tries to minimize the distance to the destination with each step (see [2, p. 623]). For that purpose, every packet is stamped with the position of the destination to which it needs to be delivered. When a node in the network receives such a packet, it consults its local neighbor table to determine the neighbor which is located closest to the destination and forwards the packet to that node, as shown in figure 2.3a. This process is repeated until the packet either reaches its destination or is caught in a local optimum (see [14, p. 55]). This is the case if the node that holds the packet does not know of any neighbors located closer to the destination than itself, as shown in figure 2.3b. In such a situation, the node will by default drop the packet, since forwarding the packet to a different node could possibly lead to the creation of loops (see [14, p. 55]).



(a) Neighbor Selection **(b)** Local Optimum

Figure 2.3: Depiction of greedy neighbor selection. In **(a)**, Node X means to deliver a packet to D by forwarding it to its neighbor Y, which is located closest to D. After the packet has been forwarded, Y performs the same neighbor selection in **(b)**. Since none of its neighbors are located closer to D than itself, a local optimum has been reached, and Y must drop the packet in order to avoid possible loops. The optimal route that the packet could have taken is shown as a dashed line. Figure taken from [16] with minor modifications.

As a consequence of this rule, node X in figure 2.3 would be unable to communicate with D if the nodes were to remain in their depicted positions. For this reason, routing protocols which follow the greedy approach are generally designed to avoid a local optimum or employ a fallback solution. One possible fallback solution is to switch to a face routing approach, which will be explained later in this section.

The greedy approach is very efficient because it only requires local communication between neighboring nodes after the position of the destination has been acquired. Contrary to many other topology routing protocols, the required communication overhead thus only scales with the local node density instead of the global size of the network (see [16, p. 245]). Yet, besides the local optimum problem, the greedy algorithm also favors unstable routes through its neighbor selection. Since it forwards packets to the neighbor that is located closest to the destination, it implicitly maximizes the transmission distance between the current and the next hop (see [12, p. 231]). This is illustrated in figure 2.3a, where node X forwards the message to Y as next hop and thus selects the node which is located furthest away. The rule leads to a high path loss and subsequently to a higher error rate. The same issue also arises with topology based routing protocols that try to find routes by minimizing the number of hops, as explained in [11]. In [12], Lee et al. propose a relativized distance metric which

incorporates the link quality to favor stable, closer links over more distant, unstable links. While this approach seems promising, the publication only features an implementation for the network simulator *ns-2*. A refitted implementation of the proposed approach which is suited for the intended experiments would exceed the time frame of this thesis.

The second approach to geographic routing, generally called face routing, is normally used as fallback solution in case the greedy algorithm reaches a local optimum and would have to drop the packet. If the nodes in scenario 2.3b employed face routing as fallback, node *Y* would not drop the packet but instead try to forward the packet by using a face routing algorithm. Contrary to greedy forwarding, face routing guarantees delivery of a packet as long as the destination is connected to the network (see [14]).

Face routing algorithms operate on a planar graph of the network. Such a graph does not contain any intersecting edges and is composed of polygonal regions which are connected by shared edges as shown in figure 2.4 (see [18, p. 64]). The polygonal areas bounded by the different edges are also known as faces of the graph.

To obtain a planar graph of the network, face routing algorithms first need to create a network graph from the position of all nodes in the network. In [19], Bose et al. first applied the concept of Unit Disk Graphs (UDG) to derive a network graph from the spatial positions. In such a graph, two nodes A and B with $\boldsymbol{p}_a, \boldsymbol{p}_b \in \mathbb{R}^2$ are connected by an edge if their disks intersect, i.e. $\|\boldsymbol{p}_a - \boldsymbol{p}_b\|_2 \leq 2 \cdot r_{disk}$ (see [20]). This approach can be easily implemented in simulations but leads to an oversimplified model of the network that does not match the physical reality as demonstrated by Kim et al. in [21]. Other publications, namely [22] and [23], have developed a modified version of the Unit Disk Graph, which Kuhn et al. refers to as Quasi Unit Disk Graph. In it, the same nodes A and B are connected if their mutual distance $d = \|\boldsymbol{p}_a - \boldsymbol{p}_b\|_2$ is smaller than a minimum threshold $d \leq r$, and are disconnected if their distance is greater than a maximum threshold $d > R$. If $r < d \leq R$, both nodes may or may not be able to communicate directly. Still, the new approach does not cope with disconnected or unidirectional links, which may occur if the radio path is blocked by an obstacle.

After the graph has been created, it may still contain intersecting edges and therefore needs to be planarized. Planarization is normally performed by an appropriate algorithm such as the Gabriel Graph [24] or the Relative Neighborhood Graph [25]. It is important to notice that this planarization does not always yield an optimal solution in a real testbed. In [21], Kim et al. also demonstrate that the planarization can lead to disconnected subgraphs, unidirectional links between nodes, or non-planar graphs if the conditions in the testbed do not meet the

ideal assumptions of the model.

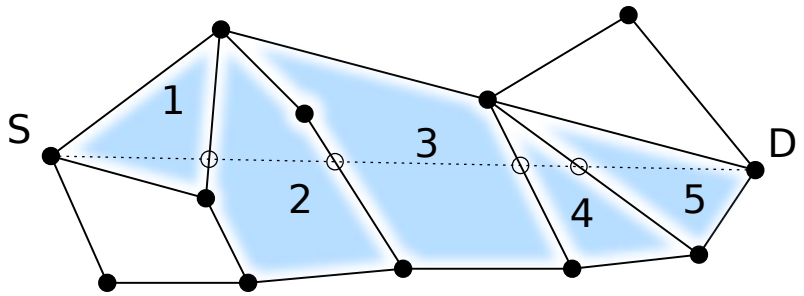


Figure 2.4: Face Routing performed on a planar network graph from **S** to **D**. The faces are numbered in the order in which they are traversed by the algorithm. Figure taken from [12] with minor modifications.

Once a planar graph has been constructed, the algorithm traverses the edges of the graph from source to destination. The following explanation of the face routing algorithm is based on [2, p. 625] and [26, pp. 218 sq.]. The algorithm uses two primitive operations: the *right-hand rule* and the *face change* primitive. The right-hand rule is used to traverse the edges of a face until the algorithm has reached a node that is positioned closer to the destination. It then performs a face change to the neighboring face, which is also traversed using the right-hand rule. In figure 2.4, the algorithm starts at node S and tours face 1 until it reaches an edge that crosses the imaginary line $\overline{SD}$ in a point closer to the destination than the node at which it entered the face. It then performs a face change and traverses face 2 following the same criteria. This process is repeated for face 3, 4, and 5 until the algorithm reaches D. Newer algorithms proposed in literature, such as the Greedy Perimeter Stateless Routing (GPSR, see [16]) or Greedy Other Adaptive Face Routing (GOAFR, see [27]) combine both geographic routing approaches: The algorithms first employ greedy forwarding and fall back to face routing if greedy forwarding fails.

Publications in the field of geographic routing mostly choose simulations as a means of algorithmic verification. While this simplifies comparing different algorithms, it does not promote verification in practical networks since the routing algorithms need to be refitted for a real world system. The only practical implementation, known to the author, was done by Kim et al. in [21] for the GPSR algorithm. The paper demonstrates the difficulties associated with implementing such an algorithm for use in a realistic testbed and outlines the overhead which is required to refit the algorithm for deployment on a real device. One of the issues that need to be tackled in such a practical testbed is to find an appropriate model for the radio

communication between the nodes. The UDG does not suffice as it does not capture obstacles or obstructions due to its mere reliance on the node positions. An alternate option would be to use a signal propagation model for a more accurate estimation of the radio range of the different nodes in the network. The following section 2.2 provides an overview of the different options to model the signal propagation of radio waves.

2.2 Radio Signal Propagation Models

Radio channels obey complex physical rules and, unlike wired channels, wireless channels are extremely random and do not offer easy analysis. In order to correctly predict the propagation of a wireless signal, an in-depth knowledge of the environment, through which the signal travels, is required (see [28, p. 29]). While this approach may be an option for small-scale indoor applications, it is not feasible for large-scale outdoor applications as the environment cannot be modeled accurately or the calculations become too complex. For this reason, the underlying physical rules are approximated through simplified models that estimate the physical conditions sufficiently for the intended applications. One physical characteristic which is often sought for is the power loss that a signal experiences while propagating. This so called *path loss* P_L is defined as the ratio of transmit power P_t to receive power P_r (see [28, p. 30]):

$$P_L = \frac{P_t}{P_r}. \tag{2.1}$$

In electronics, the ratio of powers is often expressed in decibel, which allows the definition of P_L to be expanded as follows:

$$P_{L,\text{dB}} = 10 \log_{10} \frac{P_t}{P_r} \,\text{dB} = P_{t,\text{dBm}} - P_{r,\text{dBm}}. \tag{2.2}$$

Since a regular channel does not contain any active elements, the dB path loss is a nonnegative number. From this point onward, unless otherwise noted, all powers ratios such as P_L or P_G are expressed in dB, and all absolute power values such as P_t or P_r are given in dBm.

In literature, multiple propagation models exist which estimate the path loss under different conditions and for various environments. The following section gives a brief introduction of the different approaches and proposes a simple model to estimate the expected path loss for radio signals as a function of the distance between transmitter and receiver. This model is

then used in chapter 3 as part of the metric function to approximate the link quality between different nodes in the network.

2.2.1 Propagation in Free Space

In free space, i.e. remote from earth, radio signals can propagate freely without interference by obstacles. Under these conditions, the path loss of a signal that travels a distance d from transmitter to receiver is given by the Friis free space equation (see [28, p. 32] and [29, p. 16]):

$$P_L = -10 \log_{10} \left[G_T G_R \left(\frac{\lambda}{4\pi d} \right)^2 \right] \text{dB} \tag{2.3}$$

where G_T and G_R are the antenna gains of transmitter and receiver, and λ is the wavelength in meters. The equation illustrates that the received power falls off with the square of the distance between receiver and transmitter.

In a practical scenario however, the propagation of radio signals is greatly affected by the environment. Deviations from the free space propagation can generally be attributed to reflection, absorption, scattering and diffraction of the signal, which are encountered when the signal interacts with its environment (see [28, p. 27]). For this reason, new models have been developed that better estimate the propagation in a heterogeneous environment. These models can be categorized as either analytical or empirical (see [28, p. 27] and [30, p. 55]). The former are based on the physical rules regarding signal propagation, require a precise knowledge of the environment such as terrain profile or the location of obstacles, and are generally more computationally intensive. The latter category of models is based on empirical measurements and requires less computational power. Ray-tracing models represent a prominent example of the analytical category. They approximate the wave propagation according to Maxwell's equations and require a detailed knowledge of the geometry and dielectric properties of the target region (see [28, p. 27]). While this approach seems viable for static set-ups like a cellular telephone array, it is not feasible for dynamic environments where transmitter and receiver constantly move and the computational capabilities are limited (see [28, p. 33]). For such scenarios with high uncertainty, empirical models provide a better alternative. These models are based on empirical measurements conducted in the target region to derive a correlation between distance and path loss.

2.2.2 Empirical Signal Propagation Models

A common application of empirical models is the prediction of radio wave propagation for television, radio or cellular telephones (see [31, p. 134]). The Okumura Model is a commonly applied empirical estimation of signal propagation in large urban macrocells. It was devised from in-depth measurements between a base station and a mobile receiver at frequencies from 150 MHz to 1500 MHz throughout Tokyo (see [28, p. 42]). The model classifies target regions as either open area, suburban area or urban area and applies correction factors depending on the classification (see [31, p. 147]). The model is often used as benchmark for other models and was also used as foundation for the Hata model, which simplifies its use. Both models are however not directly applicable for the intended use as prediction method for WLAN ad-hoc networks which operate at frequencies of 2.4 GHz and 5 GHz. In [32], Green and Obaidat propose an alternate model that provides a more accurate approximation of signal propagation in WLAN networks with small antennas of 1 m to 2.5 m in height. In dynamic ad-hoc networks however, the antenna height can still be considerably smaller than 1 m due to the mobile nature of the nodes. For this reason, a different model is required that captures the characteristics better and can be dynamically trained through parameter optimization. A possible solution is the log-distance or simplified path loss model, which will be described in the following section.

2.2.3 Log-Distance Path Loss Model

The log-distance model is a rather simple model to estimate signal propagation in an unknown environment. It is based upon the empirical observation that the mean path loss, which a signal experiences, scales exponentially with distance (see [33, p. 209]). This correlation can be expressed using the following formula (see [28, p. 46] and [34, p. 102]):

$$PL(d) = PL_0 + 10\,n \log_{10}\left(\frac{d}{d_0}\right) \text{dB} \tag{2.4}$$

where n denotes the path loss exponent of the model. It indicates how fast the path loss increases with distance d. PL_0 denotes the path loss at a reference distance d_0 which mainly depends upon the characteristics of the employed antennas. Both parameters are normally trained using existing data from the target environment, as demonstrated by Abbas et al. in [35]. The value of the reference distance d_0 is chosen to fit the intended propagation environment. For microcellular systems, a value of 1 m to 100 m is commonly used (see [34,

p. 104]). Figure 2.5 depicts an exemplary path loss measurement. The data points represent the measured path for a given distance, and the curve illustrates the corresponding fitted log-distance model with a path loss exponent $n = 4$. The figure also shows that the data

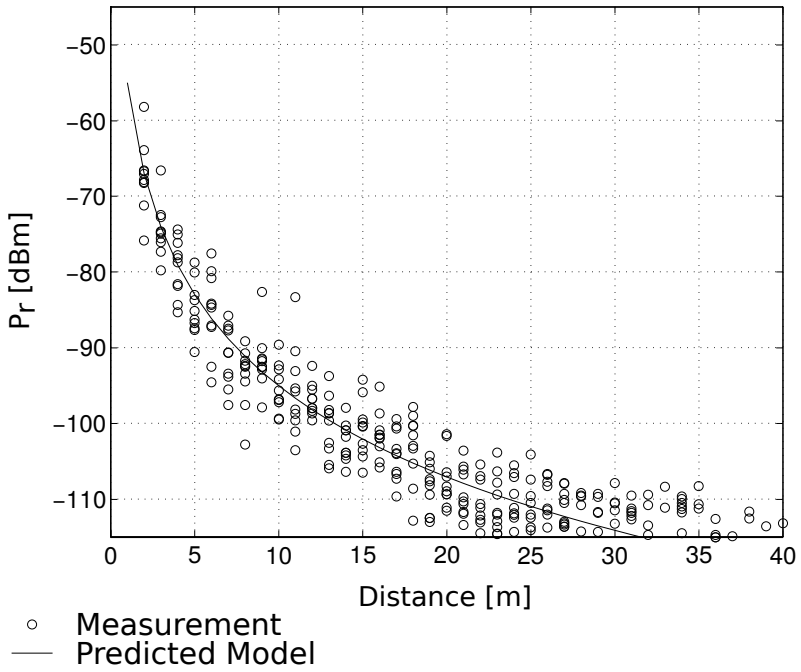


Figure 2.5: Path loss measurements with corresponding log-distance model. The circles represent individual data points, and the curve illustrates the estimated log-distance model with $n = 4$, $PL_0 = 55$dB, $\sigma = 4$ and $P_t = 0$ dBm. Figure taken from [36] with minor modifications.

points deviate from the estimated model to a certain degree. This scattering around the mean value is caused by large-scale fading effects due to obstacles or clutter in the propagation path. Additional measurements have shown that the values are log-normally distributed about the mean value of the model (see [34, p. 104]). To account for this deviation, a zero-mean Gaussian distributed random variable $X_\sigma \sim \mathcal{N}(0; \sigma^2)$ (in dB) with standard deviation σ is added to the equation (see [34, p. 104] and [28, p. 51]):

$$PL(d) = PL_0 + 10\, n \log_{10}\left(\frac{d}{d_0}\right) + X_\sigma \tag{2.5}$$

where σ indicates how strongly the measured values fluctuate around the path loss estimated by the model. It is important to understand that the parameters n, and PL_0 are specific to the target environment. Table 2.1 lists values of the exponent n in different environments.

Environment	Path Loss Exponent, n
Free space	2
Urban area cellular radio	2.7 to 3.5
In building, line-of-sight	1.6 to 1.8
In building, obstructed	4 to 6

Table 2.1: Log-distance model path loss exponents obtained in different environments [34]

The validity of the model for use with 2.4 GHz WLAN systems has been demonstrated by several publications. In [37], Ali et al. use the log-distance path loss model to estimate the signal propagation in an indoor environment with path-loss exponents between 2.68 and 4. In [38], Liechty et al. employ the same model as basis for a more sophisticated estimation technique which also accounts for obstructing buildings and obstacles in order to find suitable locations for wireless access points. Although this approach provides a better result, it is not feasible in an a priori unknown environment. The authors of [36] on the other hand show that the log-distance model can also be used to find a more accurate estimation of the behavior of wireless links than unit disk graphs. Specifically, they demonstrate that the link between two nodes experiences a transitional region with increasing distance, where the link is neither entirely stable nor unstable. The size of this region depends on the characteristics of the environment, namely n and σ, as well as the signal to noise ratio. The validity of their approach is verified using several empirical measurements in indoor and outdoor environments.

The presented log-distance path loss model is utilized in chapter 3 as a metric function to rate the link between nodes in a wireless network. In section 3.3, a regularized least squares regression is proposed to calculate the parameters n and PL_0 using data measured on-the-fly.

3 Proposed approach

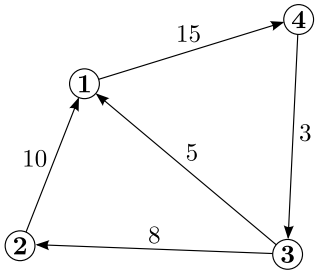
The following chapter describes the theoretical concept of the proposed routing algorithm. It is grouped into several sections which outline different aspects of the algorithm. The first section (3.1) defines the weighted directed graph, which is used by every node to store its perceived network topology. The algorithm itself is defined in section 3.2. It exchanges information about its neighborhood with other nodes to fill the network graph and calculate to routes to other nodes. The links between nodes in the network are rated based on their link quality and stored as weighted edges in the graph. The algorithm employs a metric function to rate the links and calculate the edge weights, which is outlined in section 3.3.

3.1 Network Graph Definition

The network graph, which forms the basis of the routing algorithm, is modeled as a weighted directed graph $G = (\mathbb{V}, \mathbb{E})$ that consists of a set of vertices $\mathbb{V} = \{ v_1, \ldots, v_N \}$ representing the nodes of the network numbered from 1 to N, and a set of edges $\mathbb{E} \subseteq \{ (v_i, v_j) \mid v_i, v_j \in \mathbb{V}, i \neq j \}$ which denotes the directed edges of the graph. The weight function $w : E \mapsto [0, \infty[$ assigns each edge in the graph a positive weight. The adjacency matrix $\boldsymbol{A} = [a_{i,j}]$ of the graph is a quadratic $N \times N$ matrix which holds the edge weight for all edges in the graph (see [39, p. 101]). Its elements $a_{i,j}$ are defined such that:

$$a_{i,j} = \begin{cases} w(v_i, v_j) & (v_i, v_j) \in \mathbb{E} \\ \infty & (v_i, v_j) \notin \mathbb{E}. \end{cases} \tag{3.1}$$

This definition implies that $a_{i,j} = \infty$ for a non-existing edge, which is a convenient characteristic for shortest-path algorithms that operate on the adjacency matrix, as traversing a non-existing edge would lead to an unbounded cost. Figure 3.1 depicts an exemplary graph G_s and its corresponding adjacency matrix $\boldsymbol{A}_s$.



(a) Graph G_s with four vertices

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \infty & \infty & \infty & 15 \\ 10 & \infty & \infty & \infty \\ 5 & 8 & \infty & \infty \\ \infty & \infty & 3 & \infty \end{pmatrix} \end{array}$$

(b) Adjacency matrix $\boldsymbol{A}_s$ of the graph

Figure 3.1: Weighted directed graph G_s and its corresponding adjacency matrix $\boldsymbol{A}_s$

With this graph definition, the weight can only be imposed on the edges of the graph but not on the vertices themselves. If however the vertices were assigned a weight, traversing a vertex from one edge to another would increase the overall cost of the path by the vertex weight. This property of the graph coincides with the physical characteristics of a wireless network, where traversing a node always incurs an additional transmission delay and should therefore be punished by the shortest-path algorithm. For this reason, an extended network graph $\widetilde{G} = (\widetilde{\mathbb{V}}, \widetilde{\mathbb{E}})$, which allows a vertex weight w_n to be set, is derived from the original model. For every vertex $v_i \in \mathbb{V}$, the set $\widetilde{\mathbb{V}}$ contains two vertices $v_{i,\text{in}}$ and $v_{i,\text{out}}$ such that

$$\widetilde{\mathbb{V}} = \{\, v_{i,\text{in}}, v_{i,\text{out}} \mid v_i \in \mathbb{V} \,\} \tag{3.2}$$

where $v_{i,\text{in}}$ denotes the ingoing vertex of a network node and $v_{i,\text{out}}$ denotes the outgoing vertex of the same node. Hence, every node in the network is modeled through two vertices. Edges between different nodes v_i,v_j in the network always span from $v_{i,\text{out}}$ to $v_{j,\text{in}}$. Additionally, for every node $v_i \in \mathbb{V}$, an internal edge $(v_{i,\text{in}}, v_{i,\text{out}})$, which carries the node traversal weight w_n such that $w(v_{i,\text{in}}, v_{i,\text{out}}) = w_n$, is added. The entire edge set $\widetilde{\mathbb{E}}$ can be derived from $\mathbb{V}$ and $\mathbb{E}$ as follows:

$$\widetilde{\mathbb{E}} = \{\, (v_{i,\text{in}}, v_{i,\text{out}}) \mid v_i \in \mathbb{V} \,\} \cup \{\, (v_{i,\text{out}}, v_{j,\text{in}}) \mid (v_i, v_j) \in \mathbb{E} \,\}. \tag{3.3}$$

The corresponding adjacency matrix $\tilde{\boldsymbol{A}}$ of the graph $\widetilde{G}$ has a dimension of $2N \times 2N$ and can

be obtained by expanding $\boldsymbol{A}$ as follows:

$$\tilde{\boldsymbol{A}} = \begin{array}{c|cccc|ccc} & v_{1,\text{in}} & \cdots & v_{N,\text{in}} & v_{1,\text{out}} & \cdots & v_{N,\text{out}} \\ \hline v_{1,\text{in}} & & & & & & \\ \vdots & & \infty & & & I_N \cdot w_n & \\ v_{N,\text{in}} & & & & & & \\ \hline v_{1,\text{out}} & & & & & & \\ \vdots & & \boldsymbol{A} & & & \infty & \\ v_{N,\text{out}} & & & & & & \end{array} \tag{3.4}$$

where $I_N \cdot w_n$ represents the internal edges with weight w_n, which always span from $v_{i,\text{in}}$ to $v_{i,\text{out}}$. The resulting graph $\widetilde{G}$ is equivalent to G with respect to the network but allows a common node weight to be added. Figure 3.2 depicts the node-weighted equivalent of graph G_s from figure 3.1.

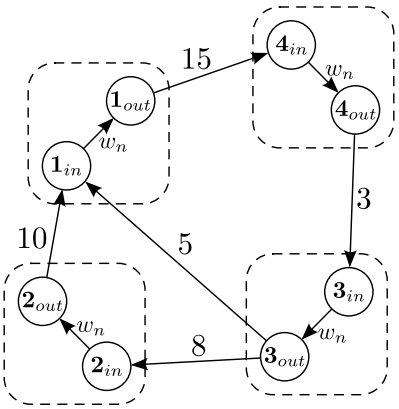


Figure 3.2: Weighted directed graph $\widetilde{G}_s$ with node weight w_n. The graph contains an ingoing vertex $v_{i,\text{in}}$ as well as an outgoing vertex $v_{i,\text{out}}$ for each node v_i.

3.2 Routing Algorithm

The following section explains the theoretical concept of the proposed routing algorithm. It is inspired by the original Link State Routing (see [1, pp. 373 sqq.]) but additionally utilizes the planar position of all nodes in the network to estimate the link quality between them and incorporate this information into its routing decisions. In order to make these decisions, each

node maintains its own view of the network, which it stores in a weighted graph as defined in section 3.1 above. The graph is used locally by the algorithm to find the shortest paths in the network and to derive its routing decisions. Instead of a source routing approach, where the sender predetermines the route a packet will take all the way to the destination (see [40] for a protocol that employs source routing), each node in the network which employs the proposed algorithm makes its own routing decision based on its local knowledge and forwards a packet to the next node (hop-by-hop). As a prerequisite for its operation, the algorithm makes the assumption that each node $v_i \in \mathbb{V}$ in the network:

(i) knows its own planar position $\boldsymbol{p}_i$ and periodically discloses it to all other nodes and

(ii) maintains a metric function $f_{v_j} : [0, \infty[\mapsto [0, \infty[, d \mapsto f_{v_j}(d)$ with which it estimates the link quality w_j to all other nodes $v_j \in \mathbb{V}$ in the network based on their Euclidean distance $d = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2$.

The planar position of each node can be acquired from a positioning system like GPS for outdoor applications. In order to disclose the position to other nodes, the information needs to be transmitted over the network. This leads to a circular dependency since the position information is required in order find routes to other nodes. Hence, the information must either be delivered in a different way such as a broadcasting mechanism, or it must initially be estimated through a different metric until a communication link has been established. While the theoretical definition of the metric function in (ii) suffices for the illustration of the algorithm, the actual function adopted for the implementation is explained in further detail in section 3.3.

The algorithm itself works in a loop which periodically executes several actions to update and maintain the network graph of each node. At any moment, the local graph of a node represents its estimated view of the global network topology and is used by the node to determine the shortest paths to other nodes in the network. In order keep the graph up to date, each node periodically calculates an estimation of its local neighborhood based on the planar position $\boldsymbol{p}_j$ of each node v_j and the corresponding metric function f_{v_j} from (ii). The neighborhood table represents all nodes which are located in its direct communication range. The table is then distributed among all other nodes in order for them to incorporate this information into their local network graphs. Each node then calculates the shortest paths to all other nodes in the network and uses these paths to derive its routing decisions. This process is repeated with each iteration of the algorithm. The individual steps of the algorithm are explained in greater detail in the following sections below.

3.2.1 Neighborhood Table Calculation

With each round r of the algorithm, every node v_i first determines its own neighborhood table $H_i[r]$. This table contains a tuple (v_j, w_j) for all other nodes v_j in the network from which a directed link with metric value w_j exists to node v_i. The link quality is approximated using the metric function f_{v_j}. A link is considered stable and part of the neighborhood table $H_i[r]$ if the metric value $w_j = f_{v_j}(d)$ is below a critical threshold w_{max}. The threshold parameter is specific to a particular metric function. If the metric value of a peer fluctuates around the value of the threshold parameter, this can lead to an oscillating behavior. For this reason, a hysteresis parameter Δw_h is introduced and the neighborhood table is built as follows:

$$\begin{aligned} H_i[r] = \{\, (v_j, w_j) \mid (w_j \leq w_{max} + \Delta w_h \wedge v_j \in H_i[r-1]) \vee \\ (w_j \leq w_{max} - \Delta w_h \wedge v_j \notin H_i[r-1]) \,\}. \end{aligned} \tag{3.5}$$

This definition ensures that a peer node is removed from the neighborhood table only if its metric value is above $w_{max} + \Delta w_h$, and reincluded in the neighborhood table if its metric value drops below $w_{max} - \Delta w_h$. The neighborhood table is then forwarded to all peers in the network.

3.2.2 Network Graph Update

As a second step, each node updates its own network graph $\widetilde{G}$ with the neighborhood tables it received from the other nodes since the last round as well as its own table which it built in the previous step. If it receives a neighborhood table from node v_i, it updates the corresponding elements in the column $v_{i,\mathrm{in}}$ of the adjacency matrix as follows:

$$a_{j_{out},i_{in}} = \begin{cases} w_j & (v_j, w_j) \in H_i[r] \\ \infty & v_j \notin H_i[r] \end{cases} \tag{3.6}$$

where v_j represents any other node in the network. This rule updates the weight of the edge from v_j to v_i with the new weight w_j for every node v_j in the neighborhood table received from v_i.

3.2.3 Route Update

In a third step, each node determines the shortest paths to all other nodes in the network by consulting its network graph. Since the graph only contains positive weights, the shortest paths can be determined with Dijkstra's algorithm (for a definition of the algorithm see [39, p. 49]). The algorithm yields a tuple $\Psi_{i,j} = (v_{i,\text{out}} \mid v_{k,\text{in}} \mid v_{k,\text{out}} \mid \cdots \mid v_{j,\text{in}})$ for every other reachable node v_j in the network. Each tuple contains the shortest path to a reachable node, with v_k being the first intermediate node in the path and $w(\Psi_{i,j})$ being the overall weight of the path. The paths determined by Dijkstra's algorithm are then installed as routing directives in the routing table. A routing directive consists of a destination as well as the next hop towards that destination. This implies that each node only makes a decision about the next hop in the route towards a certain destination. This process is repeated by all nodes until the packet reaches its destination.

A path $\Psi_{i,j}$ is installed as new route by setting the first intermediate node v_k of $\Psi_{i,j}$ as the next hop towards v_j in the routing table. If two paths in the graph have similar overall weights, this can lead to frequent route changes. For this reason, a second hysteresis parameter Δw_p is introduced, and a path $\Psi_{i,j}$ is only installed as new route to v_j if one of the following conditions applies:

a) the weight of the new path $\Psi_{i,j}$ is sufficiently lower than the current weight of the path $\widetilde{\Psi}_{i,j}$, which was installed as route to v_j in a previous round, i.e. $w(\Psi_{i,j})[r] + \Delta w_p < w(\widetilde{\Psi}_{i,j})[r]$, or

b) currently no route exists to v_j.

This route selection avoids frequent route changes, since a new route is only chosen if its overall metric value exceeds the path of the current route by at least Δw_p. At the same time the selection ensures that a node will switch to a different route if the path currently set as route has degraded since the time it was installed.

The routes installed in the routing table of a node are used to forward received packets to their destination. If a node v_i were to receive a packet destined for node v_j, it would search its routing table for a route to v_j and transmit the packet to the next hop v_k stored in the route. If no route existed it would drop the packet.

3.2.4 Summary

The routing algorithm performs the different steps explained above in a periodic manner. In each round, it exchanges data with each other node in the network as well as the operating system. The overall cycle of the data flow can be summarized as 5 distinct steps as shown in the list below:

1. Retrieve its own position and collect the positions of all other nodes in the network
2. Build the neighborhood table using the positions and disclose it to all other nodes
3. Collect all neighborhood tables received from other nodes and use them along with its own neighborhood table to update the network graph
4. Use the network graph to determine the shortest paths with Dijkstra's algorithm and update the routing table with the shortest paths
5. Pause for the specified wait time and continue with step 1.

This view of the algorithm is additionally depicted as data flow diagram for a network of two nodes in the following figure 3.3 below. The activities in the diagram are performed by each node with every round from left to right.

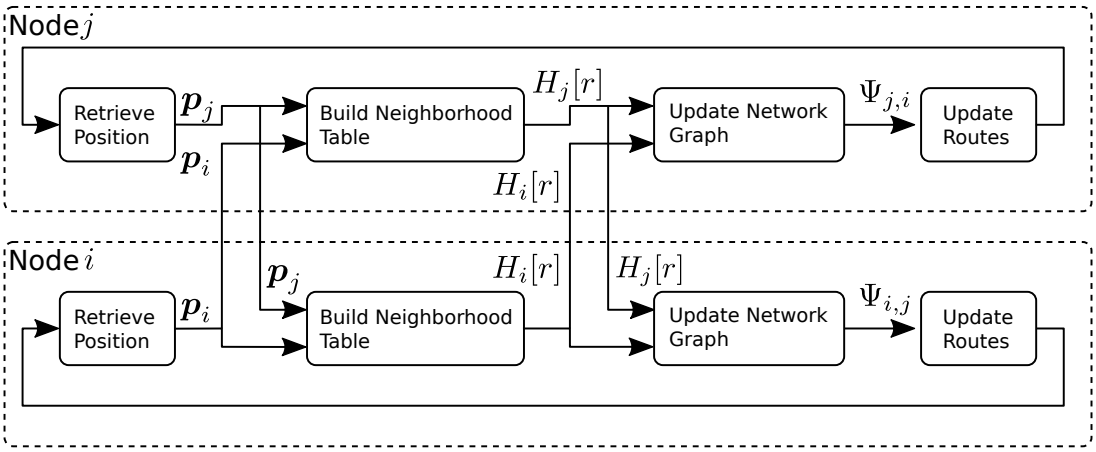


Figure 3.3: Data flow diagram depicting the exchange of information between two nodes v_i and v_j in the network. Every node performs the activities from left to right in a periodic manner. Lines which cross the boundary of a node represent information that is transmitted over the network. The process scales accordingly for more than two nodes.

While this section provided a detailed explanation of the routing algorithm, it only gave a theoretical definition of the metric function, which is used to rate the quality of a link between

two nodes in the network. Hence, the following section 3.3 introduces a metric function which rates the link quality based on the estimated path loss that the signal experiences.

3.3 Path Loss Metric Function

The routing algorithm described in section 3.2 requires each node v_i to individually maintain a distinct metric function $f_{v_j}(d)$ for every other node v_j in the network in order to rate the link quality based on the Euclidean distance d between both nodes. This function should provide a more accurate estimation than the Unit Disk Graph (UDG) and adapt to a changing environment on-the-fly. Several publications (see [10] and [36]) have shown that the quality of a link correlates with the path loss that a signal experiences. For this reason, the empirical path loss model (2.4) is proposed as metric function for the routing algorithm:

$$f_{v_j}(d) = PL_0 + 10\,n \log_{10}\left(\frac{d}{d_0}\right) \tag{3.7}$$

where d denotes the distance $\|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2$ between node v_i and a peer v_j. As a metric function, $f_{v_j}(d)$ must be greater than zero over its entire image domain, which holds true for all $d \geq d_0$. To ensure that the required condition $d \geq d_0$ is fulfilled, d_0 is set to a value of 1 m which also represents a minimum safety distance for the agents in the network.

Since the metric function f_{v_j} is specific to a certain peer, every node must maintain a set of metric functions in order to rate the link quality to all of its peers. Every instance of the metric function is defined by the value of its parameters n and PL_0. The parameters are specific to a certain target environment and need to be estimated for every deployment before the metric function can be applied. The estimation is normally performed with measured data of the intended target environment. The following sections outline how the necessary measurements are gathered and how the parameters n and PL_0 are estimated.

3.3.1 Parameter Estimation

The parameters of the metric function (3.7) for a certain peer v_j can either be estimated through training with existing data from the intended environment or on-the-fly with live measurements. A live fitting of the parameters does not require any a priori knowledge of the environment and ensures that the model can adapt to changing weather conditions and other

effects which impact the signal propagation. The required measurements are gathered during regular operation by sensing the received signal strength P_r of a peer node v_j at an instant when its position $\boldsymbol{p}_j$ is known. The received signal strength is often provided by a link level API of the WLAN chipset[1]. The path loss can then be calculated using equation (2.2) if the transmit power P_t is known. Antenna gains do not need to be taken into account separately as they are part of the link and therefore implicitly included in the measurements. WLAN stations typically operate at $P_t = 20\,\text{dBm}$ which is the maximum allowed transmit power for private use in Germany (see [41]). For every measurement taken by node v_i about peer v_j, the node stores a tuple (t, d, P_L) containing the timestamp of the measurement t, the distance to the peer d as well as the path loss P_L at time t. As soon as it has gathered sufficient measurements about its peer v_j, the parameters of the metric function f_{v_j} are trained to best approximate the measured data. It is the goal of the parameter estimation to determine a set of parameters which best fits the measured data, i.e. minimizes the error between model and measurements. For functions such as (3.7) which are a linear combination of their parameters, the parameters can be fitted analytically using a least squares regression (see [42, p. 119]). The estimation of the parameters using a least squares regression is explained in the following section.

3.3.2 Least Squares Regression

The least squares regression is an analytic approach to find an optimal solution to an overdetermined system of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ by minimizing its quadratic error

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} E(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{3.8}$$

where $E(\boldsymbol{x})$ is defined as the error function of the system, and $\hat{x}$ represents its optimal solution (see [43, p. 115]). Given the Euclidean norm $\|\cdot\|_2$, the error function can be rewritten as follows:

$$E(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})^\intercal(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) \tag{3.9}$$

and the optimal solution $\boldsymbol{x}$ of the linear system can thus be found by analytically determining the absolute minimum of $E(\boldsymbol{x})$ (see [43, p. 116]). The least squares regression is often

[1] The radiotap standard provides additional information about received WLAN frames such as the signal strength or bit rate from the driver to user space applications on Unix systems. See `http://www.radiotap.org` for more information.

used to fit a linear model function to a set of uncertain measurements. In general, the number of measurements exceeds the number of adjustable parameters, which results in an overdetermined system of linear equations that cannot be solved directly. The least squares approach finds an optimal solution to the overdetermined system, which best fits the measured data, by minimizing the quadratic error between model function and measurements. In the case of function (3.7), the linear system of equations for a set of N measurements is given as follows:

$$\begin{aligned} y_1 &= f_{v_j}(d_1) \\ y_2 &= f_{v_j}(d_2) \\ &\vdots \\ y_N &= f_{v_j}(d_N) \end{aligned} \tag{3.10}$$

where y_i denotes the measured path loss at a distance d_i taken from the ith measurement. The linear equations in (3.10) can be rewritten in matrix form

$$\boldsymbol{A} = \begin{bmatrix} 1 & 10\log_{10}\left(\frac{d_1}{d_0}\right) \\ 1 & 10\log_{10}\left(\frac{d_2}{d_0}\right) \\ \vdots & \vdots \\ 1 & 10\log_{10}\left(\frac{d_N}{d_0}\right) \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} PL_0 \\ n \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{3.11}$$

to comply with the general definition of the error function in (3.8). The least squares error of the metric function for a set of N measurements can thus be expressed as the sum of the individual errors of each measurement:

$$E(n, PL_0) = \sum_{i=1}^{N} (f_{v_j}(d_i) - y_i)^2 = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \,. \tag{3.12}$$

Weighted Least Squares Regression

The definition of the error function in (3.12) takes all measurements in a changing environment into account equally. It is however desirable to give newer measurements a stronger weight in the optimization than older measurements. To account for the age of a measurement, the measurements are weighted by scaling the error, which each measurement introduces to $E(\boldsymbol{x})$,

with a scale factor α_i:

$$E(n, PL_0) = \sum_{i=1}^{N} \alpha_i (f_{v_j}(d_i) - y_i)^2 \,. \tag{3.13}$$

The scale factor α_i is chosen based on the age $e_i \in [0, 1]$ of a measurement relative to a certain maximum age. Measurements which are older than this maximum age e_{max} are not considered for the estimation[2]. The actual weight is introduced by a scale function $f_\alpha : [0, 1] \mapsto [0, 1]$ which maps the relative age of a measurement to a scale factor α_i. The weight of a measurement with age e_i is thus calculated as $\alpha_i = f_\alpha(e_i)$. Figure 3.4 depicts an exemplary scale function which favors measurements with a relative age $e_i < 0.5$.

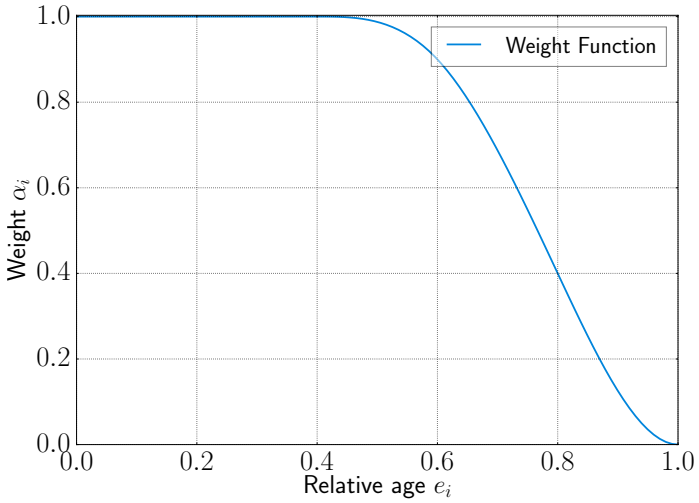


Figure 3.4: Exemplary scaling function f_α which prioritizes measurements i with a relative age $e_i < 0.5$.

In order to incorporate the scale factors into the definition (3.9), a diagonal scale matrix $\boldsymbol{Q}$ is introduced, which contains the individual scale factors for each measurement:

$$\boldsymbol{Q} = \frac{1}{\sum_{i=1}^{N} \alpha_i} \begin{bmatrix} \alpha_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_N \end{bmatrix} . \tag{3.14}$$

The weight matrix is normalized to ensure that the impact of the regularization, which is introduced in the following section, remains constant with an increasing number of measurements.

[2]The relative age can be calculated as follows: $e_i = \frac{t - t_i}{e_{max}}$, where t denotes the current time and t_i the time when the measurement was recorded.

The weighted error function can then be defined as follows (see [42, p. 123]):

$$E(\boldsymbol{x}) = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})^\intercal \cdot \boldsymbol{Q} \cdot (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}). \tag{3.15}$$

Least Squares with Regularization

For a small number of measurements, the uncertainty of $\boldsymbol{y}$ is amplified by the regression and can have a great impact on $\boldsymbol{x}$ (see [43, p. 124]). Additionally, at least two measurements are required to solve the equation and to determine $\boldsymbol{x}$. In such cases, it is desirable to condition $\boldsymbol{x}$ to a worst case default value $\boldsymbol{x}_0 = [PL_0, n]^\intercal$ by adding a regularization term as follows:

$$E(n, PL_0) = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})^\intercal \cdot \boldsymbol{Q} \cdot (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) + \gamma \|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2 \tag{3.16}$$

where γ denotes the importance of the regularization term. The regularization term allows the equation to be solved with less than two measurements and stabilizes the result by favoring a set $\boldsymbol{x}$ which is located close to $\boldsymbol{x}_0$ according to the Euclidean norm. The influence of the regularization can be adjusted by setting γ accordingly.

Putting it all together

Equation (3.16) depicts the final form of the error function. The optimal solution to $E(n, PL_0)$ can be found by determining the absolute minimum of (3.16). Again, using the alternate form of the Euclidean norm, the expression can be transformed into:

$$E(n, PL_0) = \boldsymbol{x}^\intercal \boldsymbol{A}^\intercal \boldsymbol{Q} \boldsymbol{A} \boldsymbol{x} - 2\boldsymbol{x}^\intercal \boldsymbol{A}^\intercal \boldsymbol{Q} \boldsymbol{y} + \boldsymbol{y}^\intercal \boldsymbol{Q} \boldsymbol{y} + \gamma(\boldsymbol{x}^\intercal \boldsymbol{x} - 2\gamma \boldsymbol{x}^\intercal \boldsymbol{x}_0 + \boldsymbol{x}_0^\intercal \boldsymbol{x}_0). \tag{3.17}$$

The minimum of (3.17) can be found by determining the critical points of the error function:

$$\frac{\partial}{\partial \boldsymbol{x}} E(n, PL_0) = \boldsymbol{A}^\intercal \boldsymbol{Q} \boldsymbol{A} \boldsymbol{x} - \boldsymbol{A}^\intercal \boldsymbol{Q} \boldsymbol{y} + \gamma \boldsymbol{x} - \gamma \boldsymbol{x}_0 \stackrel{!}{=} 0 \tag{3.18}$$

which yields the directly solvable system of equations:

$$(\boldsymbol{A}^\intercal \boldsymbol{Q} \boldsymbol{A} + \gamma \boldsymbol{I}) \boldsymbol{x} = \boldsymbol{A}^\intercal \boldsymbol{Q} \boldsymbol{y} + \gamma \boldsymbol{x}_0 \tag{3.19}$$

$$\boldsymbol{x} = (\boldsymbol{A}^\intercal \boldsymbol{Q} \boldsymbol{A} + \gamma \boldsymbol{I})^{-1} (\boldsymbol{A}^\intercal \boldsymbol{Q} \boldsymbol{y} + \gamma \boldsymbol{x}_0)\,. \tag{3.20}$$

Since $\boldsymbol{A}^\intercal \boldsymbol{Q} \boldsymbol{A}$ is always positive definite, the solution of (3.20) is necessarily a minimum of $E(n, PL_0)$.

3.3.3 Summary

In a live scenario, each node continuously updates the model function f_{v_i} with new measurements as it gathers them. In the remaining part of the thesis, this type of parameter estimation will be referred to as an on-the-fly fitting. It can be adjusted by changing the parameters e_{max}, γ, and $\boldsymbol{x}_0$ of the regression. e_{max} denotes the maximum age of a measurement to be considered for the regression in the on-the-fly fitting. γ represents the weight of the regularization term and determines how strongly the regression is drawn towards the default value $\boldsymbol{x}_0$. Please note that for a value of $\gamma \to 0$, the weight of the regularization is equal to zero and equation (3.20) can be simplified to a non-regularized regression by dropping the corresponding terms which include γ.

In contrast to a live, on-the-fly fitting of the model parameters, a static fitting refers to a non-weighted regression as defined in equation (3.8) in which all measurements are assigned an equal weight. The fitting is performed in post-processing with all gathered measurements as opposed to the on-the-fly fitting which only includes measurements newer than e_{max}.

4 Experiments

The following chapter evaluates the performance of the proposed routing algorithm through multiple experiments. The first section outlines the experimental platform which was used to conduct the experiments. This comprises the hardware of the agents as well as the deployed software. Section 4.2 evaluates the proposed path loss metric and details, based on which criteria the parameters of the metric were chosen for the experiments. Finally, section 4.3 discusses the performance of the routing algorithm itself through several experiments.

4.1 Setup

The experiments were conducted using multiple refitted quadcopter MAVs on which an implementation of the proposed routing algorithm was deployed. The following sections describe the hardware of the MAVs, the network configuration which was used during the experiments, and the software architecture of the routing algorithm implementation.

4.1.1 Experimental Platform

The experimental platform consists of a refitted Hummingbird quadcopter by Ascending Technologies, which is equipped with additional hardware to perform live calculations. The MAV itself features an onboard flight control unit, a GPS receiver, and an Inertial Measurement Unit (IMU) for position estimation. The fused position estimation has a horizontal accuracy of approximately 2 m. The quadcopter is powered by a three-cell lithium-polymer battery which lasts for a total flight duration of up to 20 minutes. The quadcopter can either be operated by remote control or autonomously navigate to custom waypoints. In order to communicate with the flight processor to query the current GPS position and send custom waypoints, the onboard processor provides a serial link interface (see [44]). Beyond this standard configuration, the MAV features a custom frame, which is mounted below the battery and holds a Raspberry Pi

computer, two WLAN antennas, and several sensors. The Raspberry Pi computer features a 900 MHz ARM quad-core processor, 1 GB of RAM, an SD card slot for the operating system, multiple USB ports, and a serial interface to communicate with the flight controller(see [45]). It does however not provide a real-time clock which is capable of keeping the time while the computer is powered off. Hence, the time of all agents must be synchronized prior to an experiment. The Raspberry Pi runs a Raspbian Linux operating system and serves as platform to deploy custom algorithms. It is equipped with a primary as well as a secondary WLAN antenna (the network configuration of the two antennas is outlined in section 4.1.2). Both antennas are fabricated by LogiLink and feature a Ralink RT5370 chipset as well as a 2 dBi detachable antenna (see [46]). The default Linux kernel uses the rt2800usb driver module to interface with the device. The entire MAV with all components is depicted in figure 4.1.

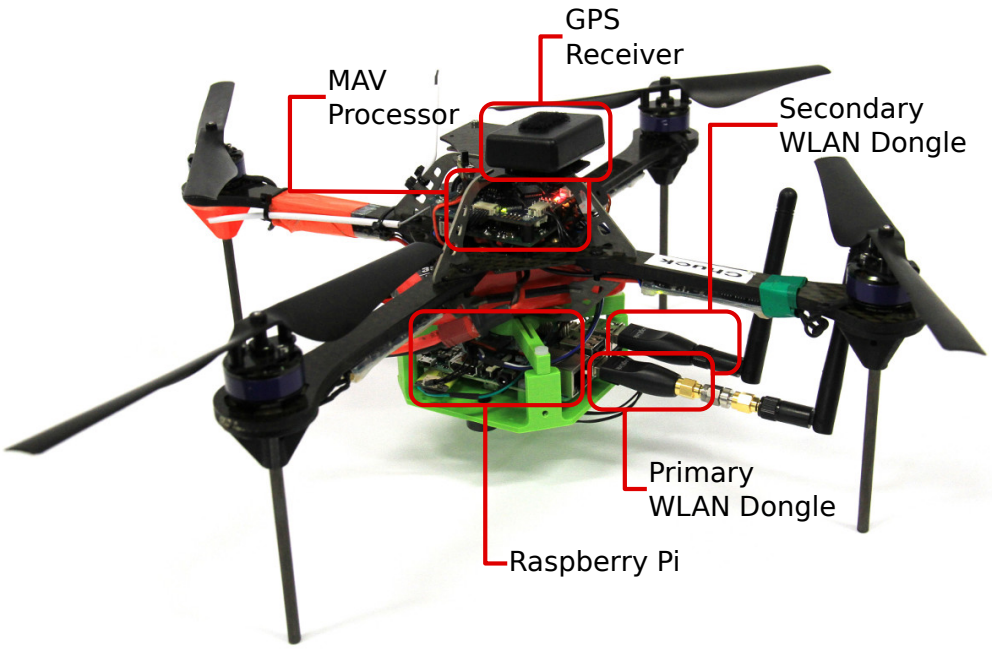


Figure 4.1: Experimental MAV Platform used for the experiments. The quadcopter features an onboard processor, a GPS receiver and a custom frame which holds a Raspberry Pi and two WLAN antennas.

4.1.2 WLAN Network Configuration

Each of the agents is equipped with two independent WLAN modules and corresponding antennas which represent the primary as well as the secondary network. Both modules operate

in the 2.4 GHz WLAN spectrum and support ad hoc mode with a transmission speed of up to 54 Mbit/s. The primary network represents the communication link that would be used by the agents in a swarm to communicate with each other during an exploration scenario. The communication among the agents is simulated by exchanging randomly generated data over the network. Under ideal conditions the radio range of the equipped antennas extends approximately 150 m. As this would result in unfeasible walking distances if a connection was to be triggered, the gain of the primary antenna is reduced with off-the-shelf 6 dB attenuators. Figure 4.1 illustrates the antenna of the primary network with an attenuator and two adapters. This configuration limits the radio range to approximately 50 m, depending on the environmental conditions.

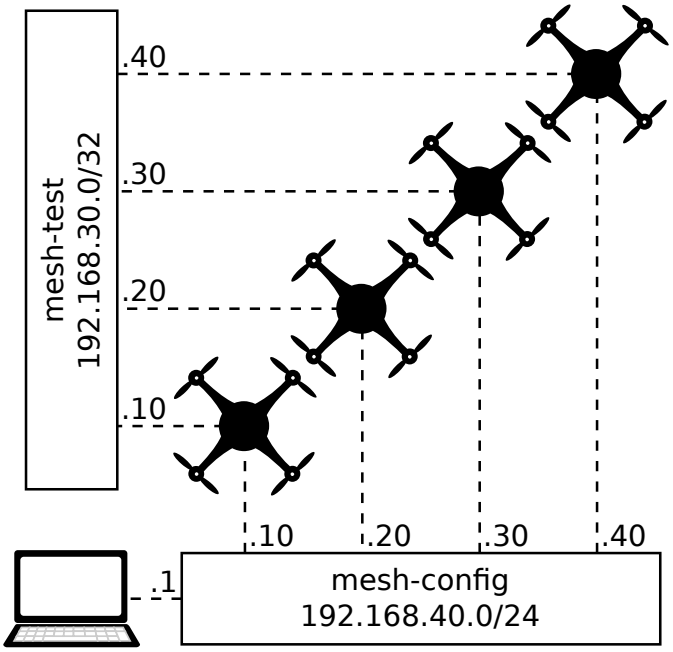


Figure 4.2: Network setup used for the experiments. Each agent participates in the *mesh-test* as well as the *mesh-config* network and is assigned a static IP address for each network. An operator computer is used to control the agents via the *mesh-config* network.

Due to the limited range of the primary network, a secondary network with an increased range is required in order to reliably control the agents during the experiments independent of their position with respect to the operator. Since the antenna of the secondary network is not attenuated, it provides a coverage radius of 150 m, which is sufficient for all experiments. The secondary network is used to synchronize the clocks of the agents, start and stop the experiments and to transmit live data to monitor their progress.

Both networks operate in ad hoc mode on different WLAN channels to avoid interference. The primary network is named *mesh-test*, while the secondary network is called *mesh-config*. All agents are assigned a unique static IPv4 address from a separate address range for each of the two networks. The address configuration is depicted in figure 4.2. It is important to notice that the subnet mask of the *mesh-test* network is set to 32. This implies that, by default, each agent only knows a route to its own address. Routes to other agents in the *mesh-test* network are added individually by the routing algorithm. Besides the agents themselves, an operator computer also participates in the *mesh-config* network in order to control the individual agents. It is assigned the first address in the subnet range and also serves as default gateway to provide an Internet connection to the agents for updates and maintenance.

As previously mentioned in section 3.2, the routing algorithm requires both the positions of each agent as well as the neighborhood tables to be transmitted over the network. This approach leads to a circular dependency since data can only be transmitted once routes to other nodes have been established but is also required to find these routes. One possible solution is to transmit this data by means of a broadcast mechanism that does not depend on individual routes. For the sake of simplicity and due to the limited time to conduct the experiments, the mentioned data is transmitted over the secondary network unless otherwise mentioned.

4.1.3 Algorithm Implementation

The algorithm, which is theoretically described in section 3.2, is implemented as a Python program for the experiments. The application uses the *NetworkX* graph library to map the network graph and perform the shortest path search using the Dijkstra implementation of the library (see [47]). In order to function, the algorithm requires input data, namely the position $\boldsymbol{p}_i$ of each agent, the neighborhood tables $H_j[r]$ of all other agents, and the path loss PL_j measured individually for each other agent. With each round, it yields its own neighborhood table $H_i[r]$ and a set of routes $\Psi_{i,j}$. This input/output data-centric view of the algorithm is depicted in figure 4.3. The required data is not gathered by the python implementation itself but provided by other programs. The python implementation of the algorithm uses the Robot Operating System (ROS) to communicate with the other programs and distribute the data among the agents.

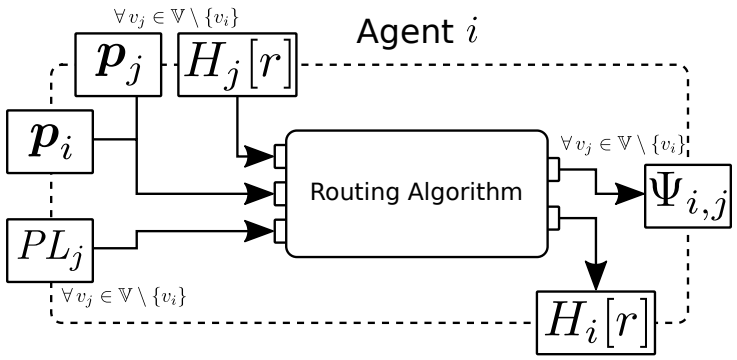


Figure 4.3: Input/Output view of the algorithm. It accepts the position $\boldsymbol{p}_i$ of each agent, the neighborhood tables $H_j[r]$ of all other agents, and the measured path loss PL_j of each agent as input data. The algorithm yields a set of routes $\Psi_{i,j}$ and its own neighborhood table $H_i[r]$ as output data.

ROS

ROS is an interprocess-communication system with programming APIs for Python, C++, and Java (see [48]). With it, different ROS nodes, which represent individual processes, communicate with each other by passing messages over an N-to-N publish-subscribe system. A single node can publish messages to a certain topic which represents a logical channel and is identified by a string name. Other interested nodes can subscribe to the same topic and will receive messages published by others. Every ROS system requires a central master node which keeps track of all publishers and subscribers for a certain topic. Contrary to other message passing systems, it does not function as a broker, but merely initiates communication between individual nodes, which thereafter exchange messages peer-to-peer. Nodes in a ROS system can be distributed among different computers as long as they are all connected to the same master node.

While this approach is very flexible, the reliance on a master node proves to be a weakness in a distributed system that communicates over an unreliable network. Nodes can only initiate a message exchange if they can both reach the master. If the connection is lossy, the message delivery cannot be initiated or runs into a timeout and possibly freezes the involved systems. To avoid these issues, each agent in the experiments runs its own master and creates a local ROS system that is isolated from the other agents. The topics which need to be exchanged among the nodes are transferred using a topic bridge node (see [49]). It functions as subscriber in the source ROS system and as a publisher in all target ROS systems. The topic messages are transferred over a custom connection that can quickly be reestablished if the network is

unstable. Since every node runs its own master, the different ROS systems are autonomous and not threatened by a lossy network. This configuration improves the stability of the agent systems while allowing to use ROS as message exchange API.

Position and Signal Strength Measurements

The position of each agent is calculated by the flight control unit of the quad. A special ROS node interfaces with the unit over the serial link and periodically queries the current position. The gathered information is then published over a ROS topic to be made available to the local algorithm and to the other agents over the topic bridge.

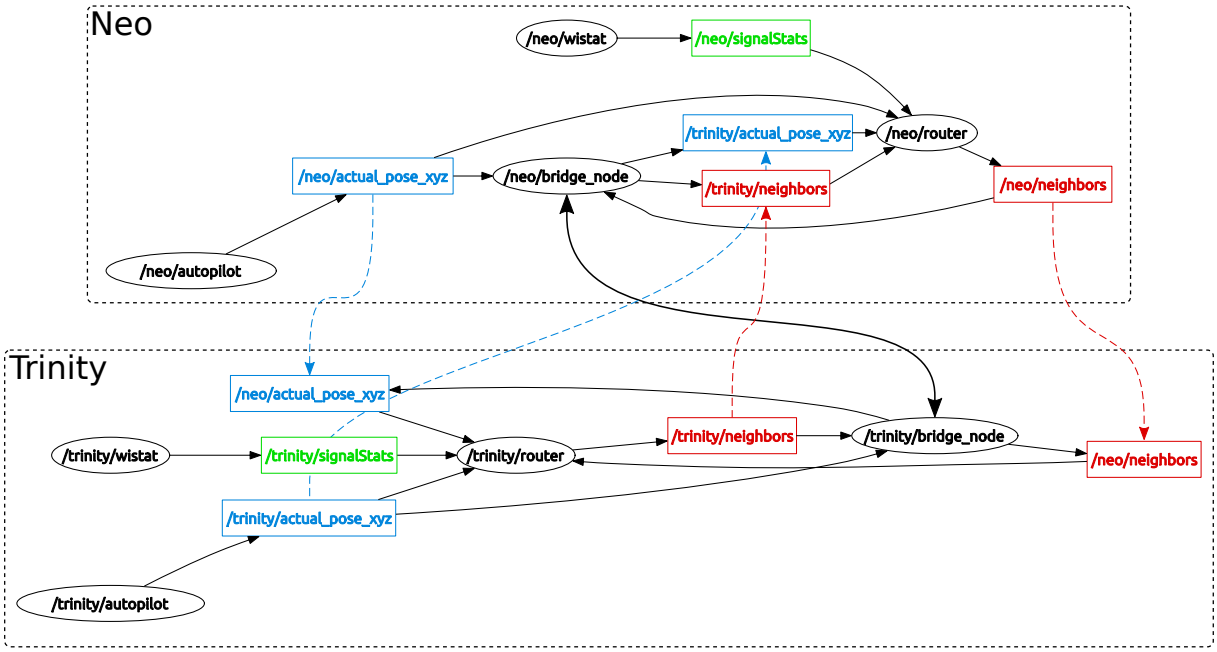


Figure 4.4: Graph of the local ROS systems of two agents, named *Neo* and *Trinity*, in the swarm. The position topics are highlighted in blue, the neighborhood table topics in red, and the topic distributing the signal strength P_r in green. The arrows which span the systems boundaries illustrate topics that are transferred by the topic bridge.

The received signal strength P_r is provided by the driver of the WLAN interface as meta information for every received frame. The different stations can be distinguished by their unique address which is specified in the frame. An independent ROS node stores the signal strength P_r for every remote peer station and periodically publishes a message which contains the last measured signal strength for each of them (see [10, p. 20]).

Figure 4.4 displays the local ROS systems of two agents and depicts how the different topics that carry the required input data are connected to the router nodes of the two agents. Position topics are colored in blue, topics for the neighborhood tables in red, and the topics which provide the measured signal strengths in green. The rectangular boxes around the graphs designate the boundaries of the two ROS systems. Topics which do not originate from a local ROS system are transferred by the topic bridge as illustrated by the colored arrows that connect the topics with equal names in the different ROS systems.

Routing

The algorithm implementation does not perform any routing itself. Instead, it manipulates the routing table of the underlying Linux kernel using the *pyroute2* package for Python (see [50]). With every iteration, the algorithm yields a set of shortest paths $\Psi_{i,j}$ to all other nodes v_j in the network, which must be translated into routing table entries. The routing table contains direct as well as indirect entries towards a destination v_j. A direct route only consists of the address of the destination and specifies that a packet should be forwarded directly to that destination. An indirect route on the other hand consists of the address of the destination as well as the address of the next hop towards that destination. A packet addressed to the destination of an indirect route is forwarded to the next hop of the route. This implies that of the entire path calculated by Dijkstra's algorithm, only the first intermediate node is stored in the indirect route.

A path $\Psi_{i,j} = (v_{i,\text{out}} \mid \cdots \mid v_{j,\text{in}})$ is translated into a route by first determining its length. A path with a length of 2 can only contain the origin and destination of the path and can therefore be translated into a direct route. In all other cases the path is translated into an indirect route. The next hop of the route is chosen to be the first intermediate node of the path.

#	Type	Destination	Prefix	Next Hop
1	Direct	192.168.30.20	32	-
2	Indirect	192.168.30.30	32	192.168.30.20
3	Indirect	192.168.30.40	32	192.168.30.20

Table 4.1: Exemplary routing table of node 192.168.30.10. The routing table contains one direct route and two indirect routes.

Table 4.1 depicts an exemplary routing table of a node with one direct and two indirect routes to showcase the difference. If the node were to receive a packet addressed to 192.168.30.40, it would forward it to the node with address 192.168.30.20 according to the third entry. All routes have a netmask prefix of 32 since they are only valid for a single node.

4.2 Evaluation of the Path Loss Metric

It is the goal of the path loss metric function to provide an approximation of the transmission path loss that is as accurate as possible but also stable in such a way that it smoothes the raw measurements. These properties are essential to the stability of the routing algorithm which relies on the metric. The behavior of the metric function can be adjusted greatly by varying its parameters e_{max}, $\boldsymbol{x}_0$, and γ, which need to be set correctly to meet the required properties (please refer to section 3.3.3 for a short summary of the parameters). In the following section, possible values for the parameters are discussed and the accuracy of the metric is evaluated. The function depicted in figure 3.4 is used as scale function f_α. Appendix table B.1 lists the support points that were used to generate the scale function using cubic splines.

4.2.1 Measurement Acquisition

In order to select appropriate parameters and evaluate the metric, several measurements were conducted using two agents in an outdoor environment. During the measurements, the primary network was saturated with generated traffic to simulate the exchange of real data. Simultaneously, the position of both agents as well as the signal strength of all received packets was recorded along with the corresponding timestamp. One of the agents was held in a static position approximately 1 m off the ground while the other was carried on a linear path at the same height. Figure 4.5a depicts the complete trajectories of both agents during such a measurement scenario. The position drift of the static agent can be attributed to the GPS accuracy. Unless otherwise noted, the mobile agent moved at a speed of approximately 1 m/s as can be seen in figure 4.5b. It illustrates the mutual distance between both agents during the measurement as a function of time. In this example, multiple breaks were taken in between. The breaks are recognizable as plateaus of constant distance in figure 4.5b.

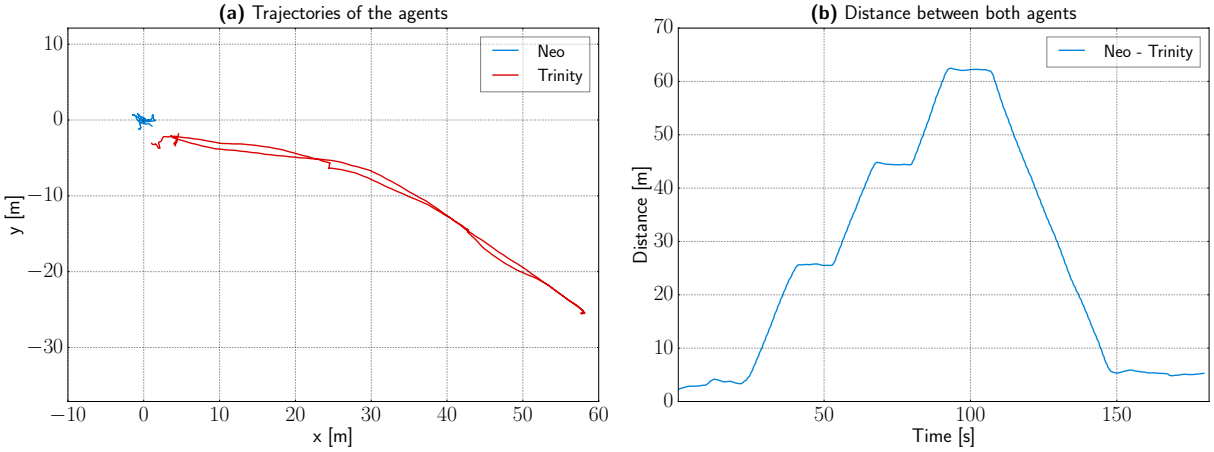


Figure 4.5: Trajectories of both agents **(a)** and the absolute distance **(b)** between them during a measurement to evaluate the path loss metric. Plot **(a)** illustrates the complete trajectory of each agent, which consists of all positions taken up by the agent throughout the measurement.

4.2.2 Maximum Age Parameter

The log-distance path loss model represents a compromise between simplicity and accuracy. While it is an efficient means to predict the path loss for a certain environment, it does not make any assumptions about the propagation environment and can therefore not fully capture its characteristics. In a changing environment this means that the more data from different points in time is used to fit the model the less accurate it is in regard to a single one of them. Figure 4.6a depicts the measurements taken in scenario 4.5 and the corresponding log-distance model with $n = 2$ and $PL_0 = 54\,\text{dB}$, estimated from the data points using a static model fitting according to equation (3.8). For better comparison with figure 2.5, plot 4.6a does not illustrate the path loss but instead shows the measured signal strength P_r which can be calculated from the path loss for a given transmission power $P_t = 20\,\text{dBm}$ according to equation (2.2).

While the static fit captures the general course of the path loss, the estimation is not entirely accurate in comparison to the measured path loss as seen in figure 4.6b. This becomes evident during the first 10 seconds of the measurements where the agents were still positioned on the ground and the signal propagation was attenuated. Since the measurements taken during this time only make up a small portion of the entire measurement, their impact on the error function is limited and the regression is biased towards the remaining measurements. As a consequence, the fitted model does not correctly reflect the signal propagation during this

period of time.

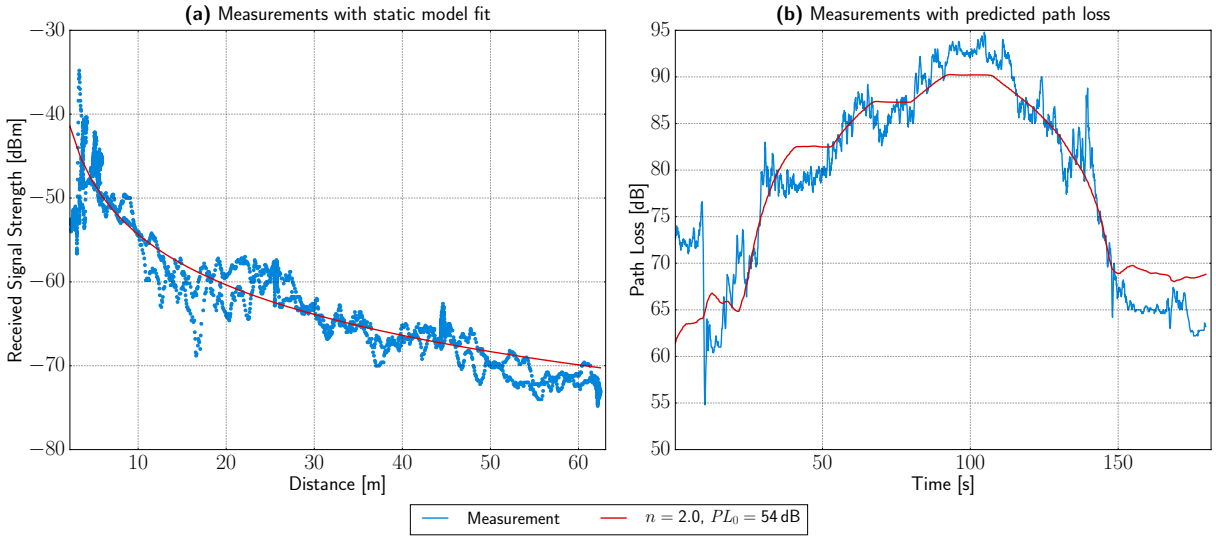


Figure 4.6: Measurements taken for the evaluation, depicted as function of the distance together with a static fit of the log-distance model in **(a)**, and over time in **(b)** with a prediction of the path loss using the same static fit.

The most straight-forward approach to improve the accuracy of the model is to limit the number of measurements that are considered for the model fitting. In the proposed regression, this property is controlled by the e_{max} parameter. It specifies the maximum age that a measurement can have to be considered for the on-the-fly model fitting. In order to evaluate the impact of the e_{max} parameter, the path loss was predicted for each measurement in the captured set with a rolling window of e_{max}. This means that to predict the path loss for a certain measurement i with timestamp t_i and the distance d_i, a model fitting is performed with all preceding measurements j in the set with a timestamp t_j that fulfill the condition $t_i > t_j \geq t_i - e_{max}$. The predicted path loss is then calculated using the fitted model $PL(d_i)$ according to equation (2.4). Figure 4.7 depicts the resulting path loss prediction as well as the values of n, and PL_0 over time for a weighted linear regression without regularization.

In comparison to figure 4.6b, the on-the-fly fitted values provide a much better estimation than the static fit but nevertheless smooth the raw measurements considerably. However, the predicted parameters become considerably more unstable with lower e_{max}, as can be seen in the two left parameter plots of figure 4.7. This is especially true for the behavior of n and PL_0 for $e_{max} = 20\,\text{s}$, which shows three significant peaks that coincide with the three breaks taken during the measurement. During these three breaks, the measured distance remained

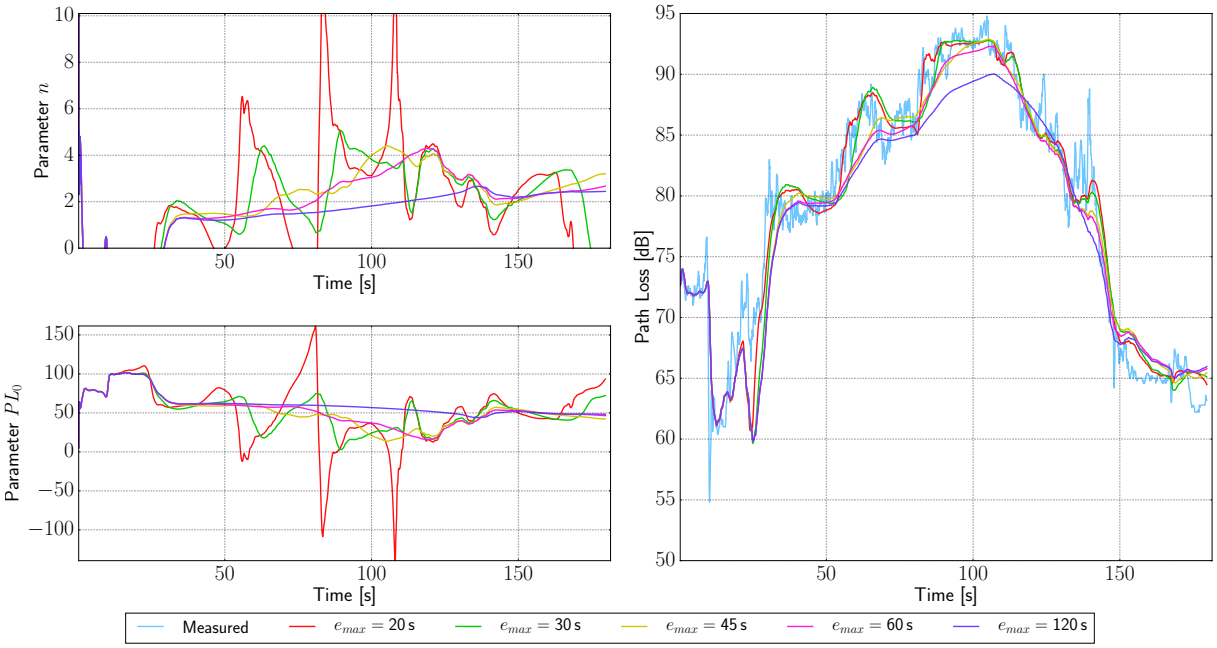


Figure 4.7: Path loss prediction with on-the-fly model fitting for different maximum ages e_{max} without regularization. The left plots illustrate how the parameters n and PL_0 of the prediction change over time. The right plot shows the predicted path loss for the different maximum ages. See figure D.1 for an enlarged view.

approximately constant for all measurements, and fluctuations in the path loss can mostly be attributed to noise. All measurements captured during these breaks describe the same physical situation and do not add any new information to the model other than the noise of the path loss measurement.

Of all the different values, an e_{max} between 30 s and 45 s presents a good compromise between parameter stability and accuracy. It is important to remember that for the route selection of the routing algorithm, it is better if the prediction overshoots the measured value than if it predicts a path loss which is lower than the real value. Nevertheless, the prediction can be further stabilized through the use of the regularization with weight γ and a default value $\boldsymbol{x}_0$. This is especially important if the distance between two agents remains constant for a period of time longer than e_{max}.

4.2.3 Regularization Parameter

In order for the regularization to actually stabilize the regression and provide an improved result, a suitable default value $\boldsymbol{x}_0 = [PL_0, n]^\mathsf{T}$ must be chosen. The free space propagation formula (2.3) specifies that the path loss exponent n be equal to 2 in an unobstructed propagation environment. The result of the static model fit in figure 4.6, which yields a value of $n = 2$, indicates that this is also true for a ground-based scenario. Additionally, the prediction depicted in figure 4.7 shows that PL_0 fluctuates around a default value of 50 dB, which approximately matches the static prediction with $PL_0 = 54\,\text{dB}$. For this reason, the default parameters PL_0 and n of $\boldsymbol{x}_0$ are set to 50 dB and 2, respectively. Figure 4.8 depicts the same path loss regression as figure 4.7 for different values of γ and a maximum age $e_{max} = 30\,\text{s}$. Due to the limited time, a gamma value of $1 \cdot 10^{-10}$ is used to approximate a non-regularized regression for comparison (please see section 3.3.3 for an explanation of the approximation).

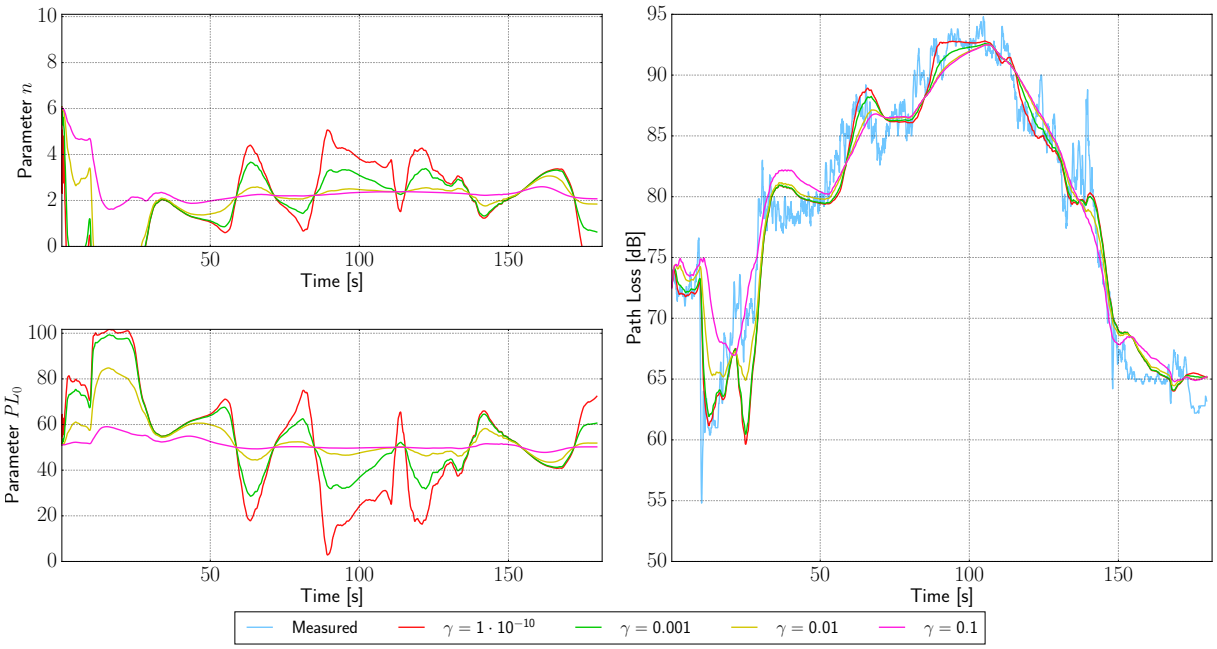


Figure 4.8: Path loss prediction with on-the-fly model fitting using a regularized least squares regression. The maximum age was set to 30 s and $\boldsymbol{x}_0 = [50\,\text{dB}, 2]^\mathsf{T}$. See figure D.2 for an enlarged view.

In contrast to the previous prediction without regularization, the fluctuation of the model parameters is reduced significantly. For a γ value of 0.01, the prediction remains as accurate as the non-regularized prediction and in some cases even provides an improved result. If however γ is set too high, the prediction is smoothed too much and converges to a static

prediction, which only provides a rough fit as explained in the previous section. It is important to realize that the regularization only affects the prediction if the overall error of the model is small in relation to the default value $\boldsymbol{x}_0$ of the regularization. If however the experienced path loss is significantly larger ($n \gg 2$), the regression abandons the default value in favor of the deviating measurements.

The smoothing effect has an even stronger impact on the parameters of the regression in a situation where the distance remains constant. Figure 4.9 depicts the model parameters which were fitted on-the-fly for a measurement where both agents were positioned statically and did not move for the duration of the experiment. The captured data only contains path loss measurements for a single input distance and is subject to a significant amount of noise from the path loss measurement equipment.

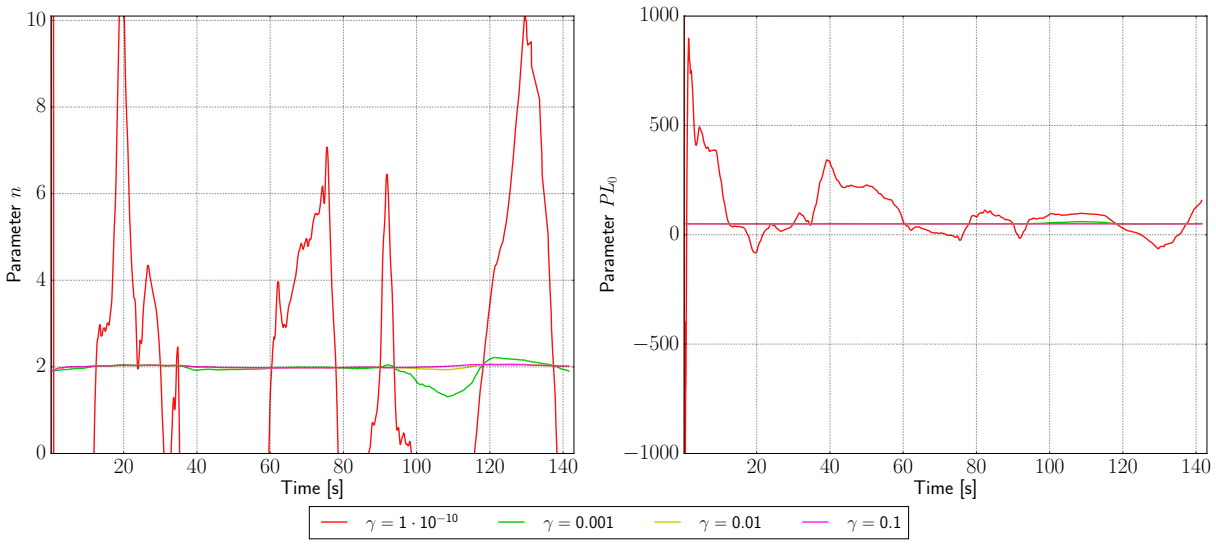


Figure 4.9: On-the-fly fitting of the model parameters using a regularized regression in a static scenario. The distance between the involved agents remained constant throughout the entire measurement. The regularization was performed with the same default value $\boldsymbol{x}_0 = [50\,\text{dB}, 2]^\intercal$. See figure D.3 for an enlarged view.

Again, the curve for a gamma value of $\gamma = 1 \cdot 10^{-10}$ represents a non-regularized fit of the parameters. The small value range of the input data leads to a seemingly underdetermined system of equations which describes the fluctuating path loss for a single input distance. The regression tries to find a solution that best approximates the noise but does not necessarily represent a realistic value. The resulting, non-regularized calculation is numerically unstable and fluctuates considerably over the course of the measurement. The curves with regularization

on the other hand are considerably more stable and fluctuate minimally. They are strongly influenced by the regularization and only differ slightly from the default value. This is due to the fact that for most of the measurement, the value range is so small that the quadratic error, which drives the regression, is insignificant in comparison to the regularization weight. As a result, the regression drifts towards the default value and yields a more stable result.

4.2.4 Summary

The stability of the metric is an essential property which considerably influences the routing algorithm. The discussion of the different regression parameters has shown that, with the use of the correct parameters, the path loss prediction can be stabilized considerably while providing an accurate estimation of the measured path loss. For the remaining experiments, unless noted otherwise, a maximum age of $e_{max} = 30\,\text{s}$ is used. The regularization is parameterized with a gamma of $\gamma = 0.01$ and a default value of $\boldsymbol{x}_0 = [50\,\text{dB}, 2]$.

4.3 Evaluation of the Routing Algorithm

The following part of the thesis evaluates the performance of the proposed routing algorithm in combination with the path loss metric through multiple outdoor experiments. The first section explains the different experiment scenarios that were conducted as well as the applied performance metric. In order to provide a reference for the experiments and put the evaluation of the proposed routing algorithm into perspective, section 4.3.2 first presents the evaluation results of other routing algorithms which were evaluated in [10] using the same scenarios. Finally, section 4.3.3 evaluates the results of the experiments for the proposed routing algorithm.

4.3.1 Experiment Scenarios

The conducted experiments aim at simulating use cases that are likely to arise in a real exploration scenario. In such a scenario, it is important that the agents can communicate reliably. Many applications, such as the Robot Operating System, rely on TCP to transmit information. However, due to its design, TCP is very prone to random packet loss which occurs over degrading wireless links. In order to measure how reliably the routing algorithm can support TCP transmissions and stable network links for routing in the network, two of the

agents which participate in the experiments transmit generated data over two separate TCP connections. The agents, which will hereafter be named *X* and *Y*, transmit data with a speed of 100 kbit/s and hence saturate the network with a total of 200 kbit/s. While this amount of traffic is relatively low compared to the maximum possible data rate, it represents a realistic scale for actual exploration algorithms and is meant to measure the reliability of the network. Besides the agents *X* and *Y*, two additional agents *A* and *B* are used for the experiments. They do not transmit any data themselves but act as regular nodes in the routing algorithm and forward data between *X* and *Y*. For evaluation purposes, the network graph $\widetilde{G}$ as well as the routing table are recorded with every iteration of the algorithm for all agents.

The evaluation of the routing algorithm makes extensive use of the words *path* and *route*. While these words are often used interchangeably in literature, they have a slightly different meaning in the context of this thesis. A path describes a set of vertices and edges that leads from one vertex in the graph to another. In a graph, often more than one path exists from one vertex to another. A route on the other hand is defined through a path and describes the set of vertices and edges that packets take while being routed through the network. Contrary to a path, only one route can exist for a certain pair of sender and receiver, and thus a route is a special unique path in the network.

During the experiments, a node weight $w_n = 50$ was used to ensure that the packets were forwarded with the minimum number of hops possible. The neighborhood hysteresis Δw_h was set to a value of 1 which was sufficient to prevent an oscillating behavior. The route hysteresis parameter Δw_p was set to a value of 5 dB. The metric threshold w_{max} was selected empirically by determining the highest possible path loss at which the link between two agents was stable. In our case w_{max} was set to 83 dB.

Scenario 1

Scenario 1 aims at testing how well the algorithm can handle a degrading network link. It involves three agents that are initially positioned in direct communication range. After the start, *X* and *Y* move away from each other until the direct link fails, while the support node remains in a static position. After a short break, they return to their initial positions.

Figure 4.10: Experiment scenario 1 of the routing algorithm evaluation

In order to maintain the connection, the routing algorithm must recognize the degrading link between X and Y and switch to a route over A before the link degrades. This decision must be made by both X and Y individually.

Scenario 2

In scenario 2, X and Y are placed out of direct radio range and are forced to communicate over A and B. X moves past the support nodes to trigger a route change when the link to A degrades and X approaches B.

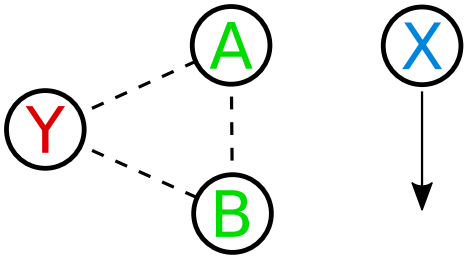


Figure 4.11: Experiment scenario 2 of the routing algorithm evaluation

The scenario evaluates the route stability of the routing algorithm. In a situation where more than one possible path exists, it must select the most appropriate one and only switch when necessary.

Scenario 3

In the last scenario, Y and the support nodes are placed in a line that only allows communication between neighboring agents. This means that Y and B cannot communicate directly. X then travels along and past the side of the line until it can only communicate with node B directly. Once this position has been reached, X returns to its starting point by traveling back the same path.

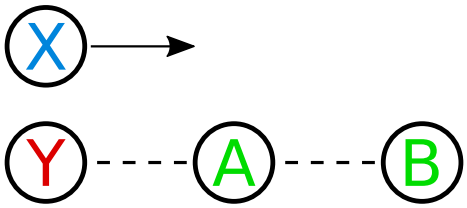


Figure 4.12: Experiment scenario 3 of the routing algorithm evaluation

This scenario evaluates the ability of the routing algorithm to detect routes that span up to two uninvolved nodes. The algorithm should not fluctuate between different paths and provide a stable connection between X and Y no matter the position of X.

4.3.2 Comparison to Babel and B.A.T.M.A.N.

In [10], the topology based routing protocols Babel and B.A.T.M.A.N. were evaluated using the experiment scenarios 1 and 3, as illustrated in the previous section. The following section presents the results of the evaluation of the two routing protocols and motivates the design of the proposed routing algorithm by pointing out the weaknesses of the two conventional protocols. For some of the experiments conducted in [10], a deviating transmission data rate of 150 kbit/s was used, however the results are still comparable.

The different scenarios are evaluated by rating the stability of the TCP connections between X and Y. Figure 4.13 illustrates such a plot of an experiment scenario. The two plots show the transmission data rate, at which both agents exchange the generated data. The left plot shows the data rate of the transmission from X to Y, and the right plot shows the data rate in the opposite direction. If the connection is stable, then the data rate should remain approximately constant as seen in the first 30 s of the experiment. If however the route is unstable or degraded, then the data rate will drop or the transmission will stop entirely. The background color of the plots depicts the number of hops that the packets take on their route from sender to receiver. Zero hops means a direct route (light blue), one hop a route over one support node regular blue) and two hops a route over both support nodes (dark blue). The dotted vertical lines which separate the different background colors represent route changes. The label next to a dotted line indicates the route that becomes active after the change.

Babel

As previously mentioned in chapter 2, the Babel protocol is a proactive topology based routing protocol. Contrary to geographic routing protocols, it does not require nor use the spatial position of the nodes in the network. The following figure presents the results of the first experiment scenario with Babel. It was created based on the data captured in [10].

The figure shows that, while at the beginning the transmission is stable, the direct link between X and Y gradually degrades and eventually fails. After approximately 45 s the TCP connection is interrupted in both directions due to the high packet loss. Approximately 20 s

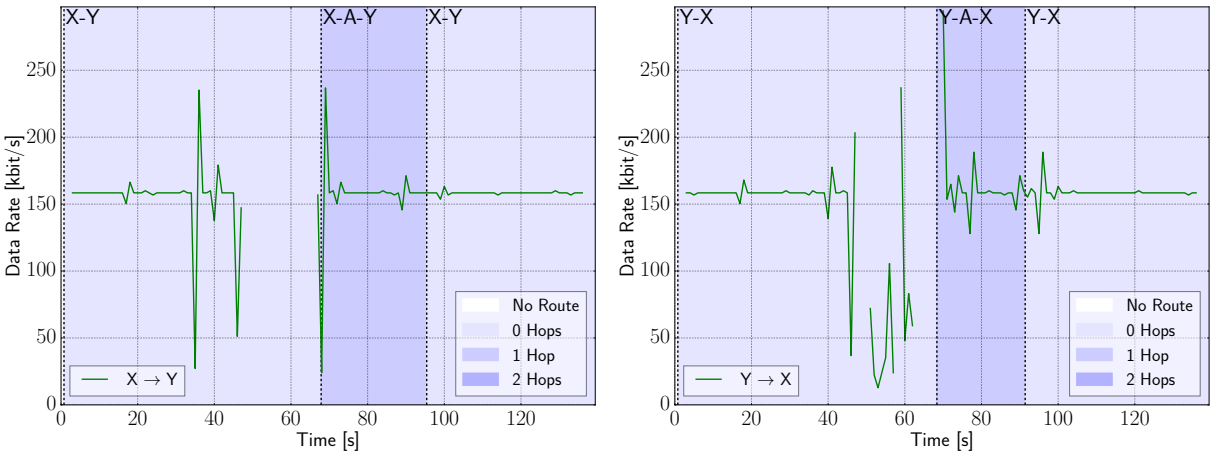


Figure 4.13: Transmission data rate for the first experiment scenario with the Babel protocol

later, Babel switches to an indirect path over A and the TCP connection resumes. The switch back to a direct route happens at an appropriate time and without any complications. Nevertheless, the scenario demonstrates that the protocol takes too much time to detect the failing link between X and Y.

Figure 4.14 illustrates the result of the third experiment scenario with Babel. Contrary to the formal definition of the third scenario, this experiment only recorded the first part of the scenario in which X moves towards B.

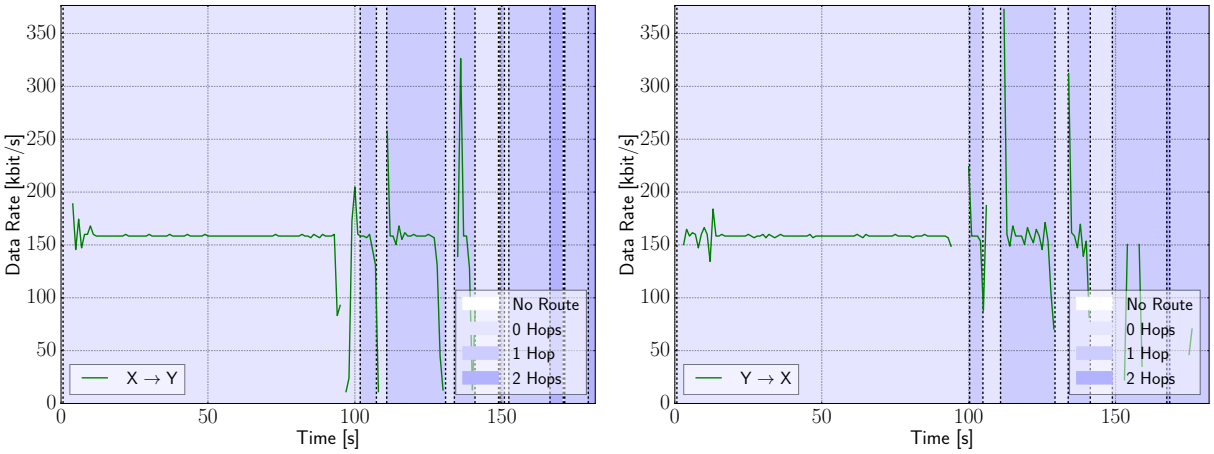


Figure 4.14: Transmission data rate for the third experiment scenario with the Babel protocol. Due to the high number of route switches the route labels were omitted to improve readability.

The plots show that in this experiment Babel takes a shorter amount of time to recognize the destabilizing link between X and Y. After about 100 s, the route switches to an indirect path over A. This switch however is not permanent, and the protocol attempts to return to a direct path multiple times although the link is already too weak to support the transmission of the TCP data packets. After approximately 150 s, the indirect route over A also fails, but the protocol does not permanently switch to a path over B, and the TCP connection is thus interrupted for the remainder of the experiment.

The two scenarios with the Babel protocol illustrate that detecting a failing link in a dynamic environment presents a difficult task for a routing protocol. This has a direct impact on the stability of the routes in the network, as they can only be considered stable if the underlying link rating is accurate.

B.A.T.M.A.N.

The B.A.T.M.A.N. protocol also is a topology based routing protocol, originally designed for the creation of city-wide mesh networks by the *Freifunk* community. For the evaluation, the B.A.T.M.A.N. *advanced* variant of the original B.A.T.M.A.N. protocol, which operates on ISO/OSI layer 2, was employed. Figure 4.15 illustrates the result of the first experiment

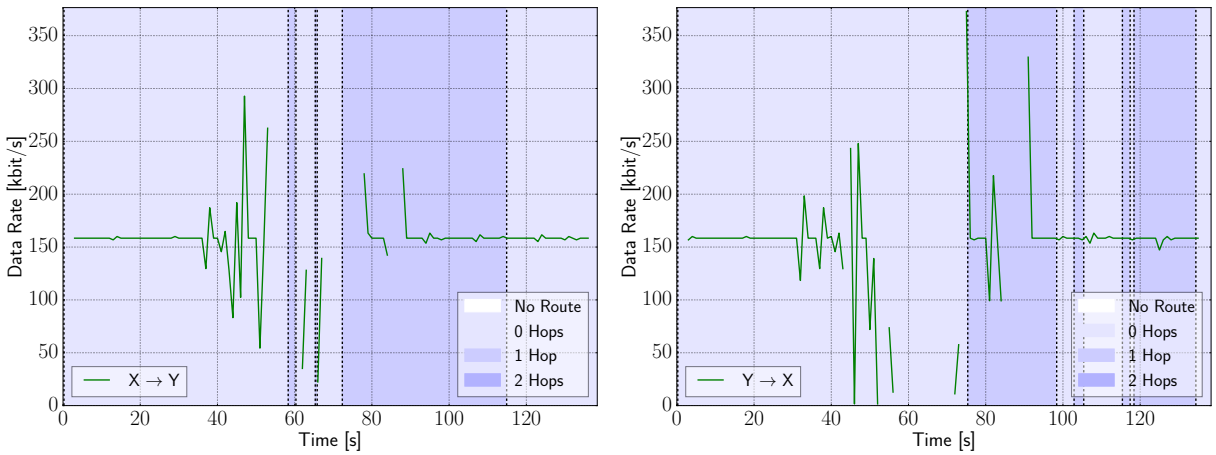


Figure 4.15: Transmission data rate for the first experiment scenario with the B.A.T.M.A.N. protocol. Due to the high number of route switches the route labels were omitted to improve readability.

scenario. The B.A.T.M.A.N. protocol experiences similar issues as the Babel protocol. It takes too long to detect the failing link between X and Y. The route from X to Y switches back

and forth multiple times before it stabilizes, although the direct link has already degraded too much to support the TCP connection. However, even after the two agents are again in direct communication range, only the route from X to Y switches back to a direct path. The B.A.T.M.A.N. protocol does not seem to correctly rate the direct link from Y to X and continues to use the indirect path over A for the remaining part of the experiment.

The experiment to evaluate to third scenario, which is depicted in figure 4.16, demonstrates quite similar issues. The protocol takes too long to detect the degrading link and to converge to a stable route over A. It does not seem to be capable of supporting a route over more than one hop in the last part of the experiment and switches back and forth multiple times.

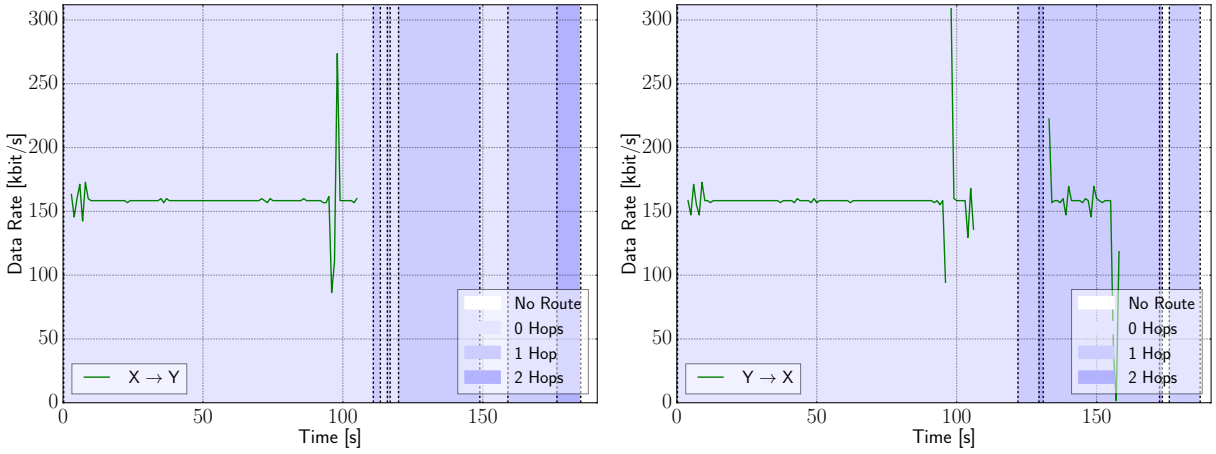


Figure 4.16: Transmission data rate of the third experiment scenario with the B.A.T.M.A.N. protocol.

This section does not mean to shed a bad light on the compared topology based protocols, but merely demonstrate the reasons which led to the development of the proposed routing algorithm. The results of the two routing protocols have shown that in order for a protocol to function reliably in a dynamic environment, an accurate link rating is essential. The mechanism which both of the routing protocols use to rate the links between nodes in the network relies on the transmission of periodic beacon packets. While this mechanism does not require any additional information such as the position of the nodes, it is too slow and not stable enough for such a dynamic environment. This realization led to the design of the routing algorithm as well as the path loss metric.

4.3.3 Evaluation of the Experiment Scenarios

The following section presents the results of the experiments, which were conducted to evaluate the presented scenarios. As with the measurements to evaluate the path loss metric, agents which were to remain static according to a scenario definition were positioned approximately 1 m off the ground. The mobile agents were manually carried with approximately 1 m/s at the same height, without obstructing the line of sight to the other agents. Appendix C illustrates the trajectories of the agents during the conducted experiments.

Scenario 1

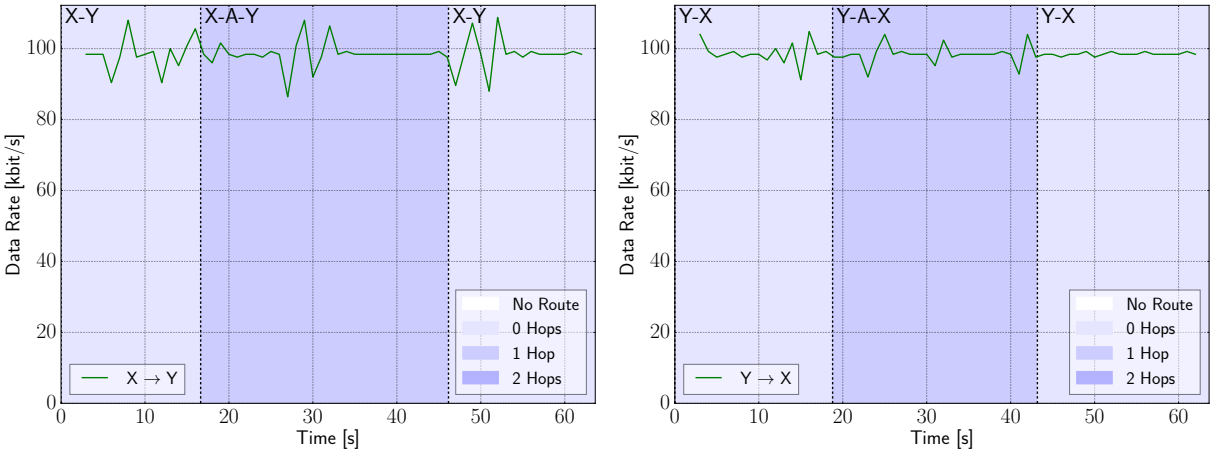
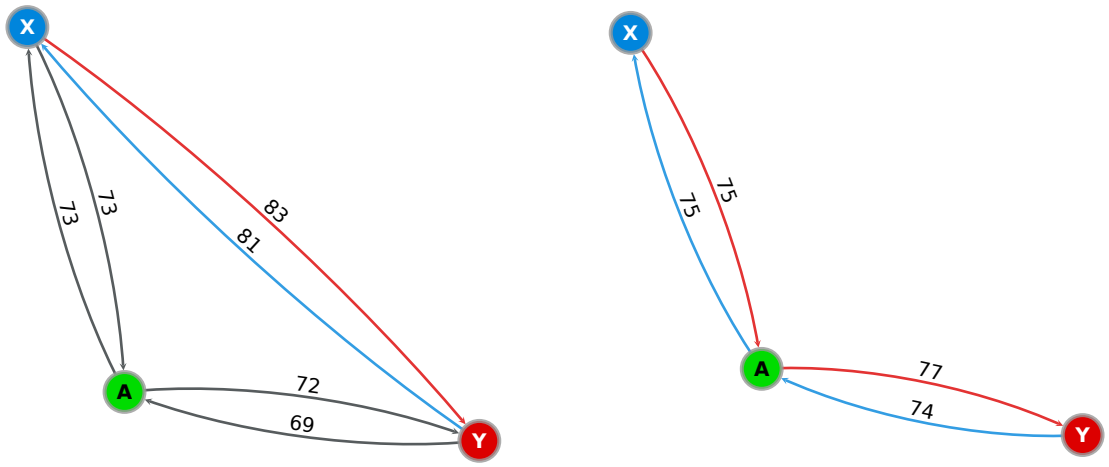


Figure 4.17: Data rate of the TCP connections in both directions for scenario 1. The left plot depicts the transmission from X to Y and the right plot the transmission in the opposite direction. The different background colors represent the number of hops that the data takes from sender to receiver.

Figure 4.17 illustrates the course of the first experiment scenario. At the beginning, both agents communicate directly. As they move away from each other the path loss increases until it reaches w_{max} and the direct link is removed from the graph. Interestingly, the routes do not switch at the same time but several seconds apart. This is due to the fact that a certain link between two agents is modeled as two independent edges in the graph which are rated separately. If the antenna gains or transmission powers are shifted asymmetrically, as in this scenario, then one direction of a link may have already been deemed unstable while the other direction experiences a lower path loss and is still stable. In this scenario, the routing algorithm switches routes in time to avoid an interruption of the transmission. The route

switches cleanly without any oscillation thanks to the hysteresis parameter. After the switch, the transmission rate remains stable without any interruption until the path loss of the direct link drops below w_{max} and both agents switch back to a direct route. This route change, too, happens at different times due to the antennas characteristics.

Figure 4.18 depicts the network graph of agent X at 15 s as well as 20 s into the experiment. The numbers on the edges of the graph represent the metric, which is assigned to them by the routing algorithm. The red edges represent the network route from X to Y, and the blue edges represent the network route from Y to X. They are calculated by simulating the transmission of a message from sender to receiver at the instant when the graph was captured. Starting at the sender, the message is forwarded from agent to agent according to the local routing table, and the edges which are traversed in the process are marked with the corresponding color (see section 4.1.3 for more information on the local routing table). Figure 4.18 depicts



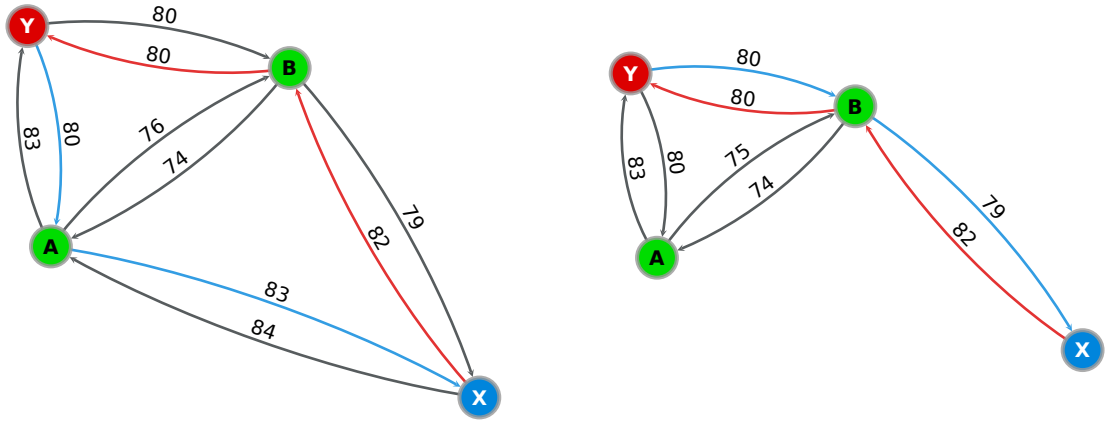
(a) Network graph at $t = 15\,\text{s}$ **(b)** Network graph at $t = 20\,\text{s}$

Figure 4.18: Network graph G of X at 15 s and 20 s into scenario 1. The numbers on the edges represent the metric which is calculated by the routing algorithm.

the local network graph of X at two different times during the experiment. The left figure 4.18a shows the graph before the direct link between X and Y has degraded and both agents still exchange data directly. In figure 4.18b, which depicts the network graph of X at $t = 20\,\text{s}$, the direct route has already failed and both agents exchange data over the support node A. The scenario shows that the routing algorithm is capable of handling a failing link and can switch to an alternative path if necessary.

Scenario 2

Scenario 2 aims at testing the route stability of the algorithm by offering more than one path to choose from. Initially, the agents are set up so that X only detects a stable route to Y over support node A. As X moves in the direction of B, it eventually establishes a link towards B as can be seen in figure 4.19a. It depicts the local network graph of X at 97 s into the experiment, where a mutual link between X and B has been established. In this situation, X and Y can choose between a path $\widetilde{\Psi}$ over support node A or a path Ψ over support node B. The red route has already switched to path Ψ over B since the overall weight of the new path is sufficiently lower than the weight of the path over A, i.e. $w(\Psi_{X,Y}) + \Delta w_p < w(\widetilde{\Psi}_{X,Y})$. This is not true for the opposite direction and the blue route thus still uses the path over A. As



(a) Network graph at $t = 97$ s **(b)** Network graph at $t = 110$ s

Figure 4.19: Network graph G of X at different times during scenario 2

X moves past B, the link to A gradually degrades and is eventually deemed unstable once its path loss rating reaches a value of $w_{max} + \Delta w_h$. This situation is illustrated in figure 4.19b, which depicts the local network graph of X at $t = 110$ s. Since the link to A has already failed, the global routes are forced to use the path over B.

Figure 4.20 depicts the data rate of both TCP connections for the duration of the scenario. The initial route over A is considered as a one hop route and colored accordingly. Ideally, the algorithm should only perform one necessary route change, as the path over A degrades, and not switch back afterwards. This route change occurs at approximately 100 s for both

directions and is illustrated by the dotted line. Since the chosen route in both direction always takes one hop, the background color does not change but remains the same throughout the experiment. After the change, the routes do not switch back and use the path over B until the end of the experiment. This stability can be attributed to the route hysteresis parameter Δw_p which was set to a value of 5 dB for the experiment. In order for the route to actually switch back, the weight of the path over A would have to improve by at least Δw_p as compared to the path over B. This is very unlikely since X moves away from A. In retrospective, a lower route hysteresis parameter would have been sufficient, as the metric function does not fluctuate rapidly and already provides a relatively stable prediction. The plots of the TCP data

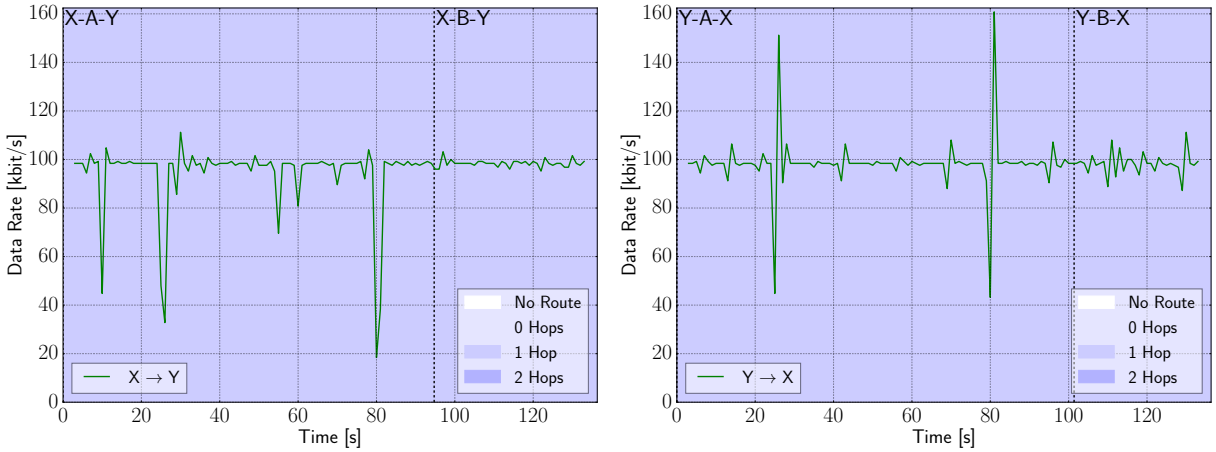


Figure 4.20: Data rate of the TCP connections for the second scenario. The switch from support node A to support node B is marked by the dotted lines in both plots at approximately 100 s.

rate also show that for the duration of the experiment, besides two small peaks at 23 and 80 seconds, the data rate remains stable and the connections stay active throughout the entire experiment. Even after the routes switch, there is no visible peak or instability in the data rate. This shows that the algorithm is capable of maintaining a stable TCP connection over one hop in a dynamic environment.

Scenario 3

Scenario 3 aims at testing the route stability over up to two intermediate hops. The result of the conducted experiment is illustrated as data rate plot in figure 4.21. Initially, X and Y communicate directly as illustrated in figure 4.22a, until X moves out of direct radio range. Apparently, in this experiment the antenna asymmetry between X and Y has a much

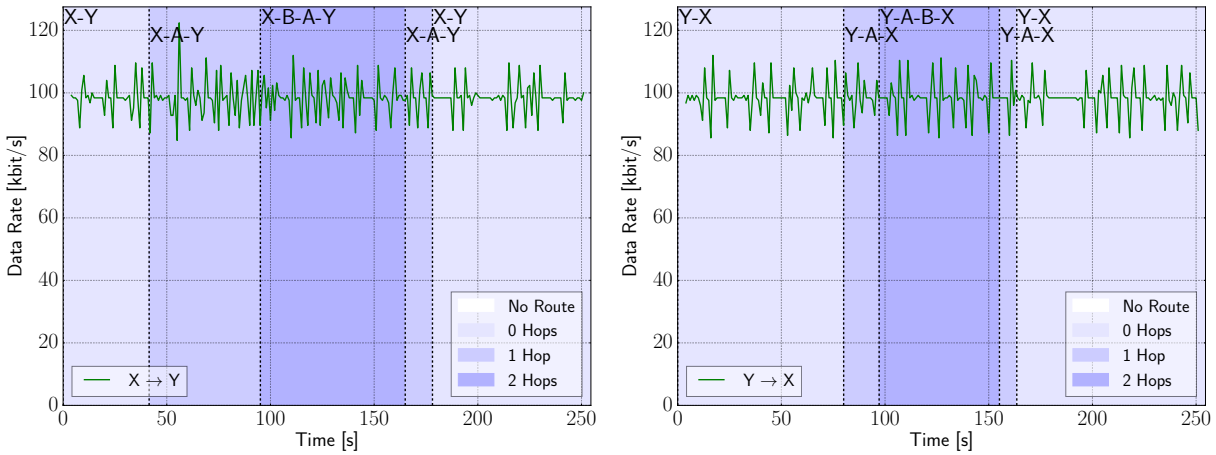
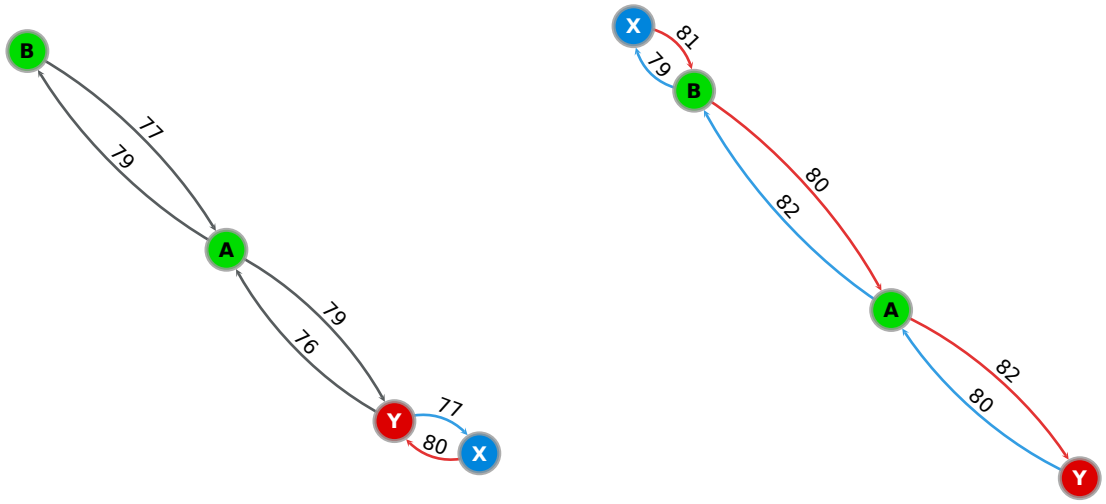


Figure 4.21: Data rate of the TCP connections for the third scenario

stronger impact as the routes switch to a path over A approximately 40 s apart. However, this emphasizes the necessity for an approach which does not only take into account the positions of the nodes in the network like many of the geographic routing protocols do but also considers the varying propagation conditions. Finally, at about 100 s into the experiment, X has reached B and left the radio range of all other agents. The only possible route leads over the two intermediate nodes as depicted in figure 4.22b. However, independent of the number of hops that the routes incorporate, the TCP connection remains stable. As X moves back



(a) Network graph at $t = 0$ s

(b) Network graph at $t = 124$ s

Figure 4.22: Network graph G of X at different times during scenario 3

to its starting position, the number of hops which are taken by the routes decreases. After about 170 s, a direct route between X and Y is reestablished. Again, the connection remains stable as the routes switch back, which implies that the algorithm only selects a shorter route when it is stable enough. The conducted experiment illustrates that the algorithm behaves as expected, and is capable of handling routes that span more than one hop.

Scenario 3 with Airborne Agent

In order to verify that the metric and the algorithm behave similarly in an actual flight situation, the third scenario was repeated with an airborne agent X. The other agents were positioned

Figure 4.23: Scenario 3 with an airborne agent X. The individual agents are marked according to their role.

at the same locations and X was remote-controlled to trace a path similar to the one used in the previous experiment. Figure 4.23 illustrates the positions of the ground-based agents as well as the flying altitude of the airborne agent during the experiment. Each of them is marked and labeled according to its role.

Figure 4.24 depicts the data rate of the TCP connections between X and Y for the flight scenario. In comparison to the ground-based version of the third scenario (figure 4.21), the data rate fluctuates more. The fluctuation can have multiple reasons such as interference from the motor electricity, or obstructing objects between the agents on the ground and X,

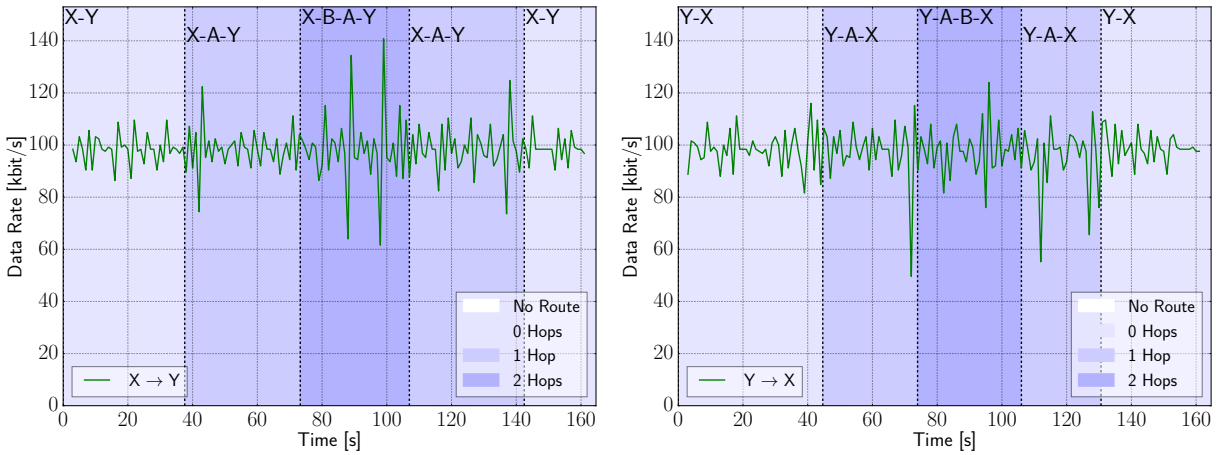


Figure 4.24: Data rate of the TCP connections for the third scenario with airborne agent *X*

like the vegetation in picture 4.23. Since these fluctuations in the data rate occur throughout the entire experiment, they do not seem to stem from the routing decisions made by the algorithm. The experiment shows that the algorithm can also function in a dynamic and partially, or potentially even fully airborne environment and provide reliable routes.

Scenario 3 without Secondary Network

During all previous experiments, the position as well as the neighborhood table updates were transmitted over the secondary network. In a realistic scenario however, a swarm of agents will only be equipped with a single communication link. For this reason, the following experiment examines the performance of the routing algorithm without secondary network. This is accomplished by rewiring the topic bridge to distribute the data topics over the primary network. This workaround does not present an optimal solution and is merely attempted to evaluate how the routing algorithm behaves if the data which it depends upon is also transmitted over the network which it controls. The topic bridge also uses TCP connections to relay the topic messages between the ROS systems. This means that the delivery of the relevant data is just as prone to network instabilities as the generated data. To avoid a TCP timeout after a network instability, the topic bridge was set up to reestablish a connection if it did not receive a new message for more than 5 seconds over that connection.

As previously explained in section 4.1.2, this configuration leads to a circular dependency, as the routing algorithm can only receive the required data once it has set up routes to all other agents. To circumvent this issue, the routing table is initially filled with a set of default routes.

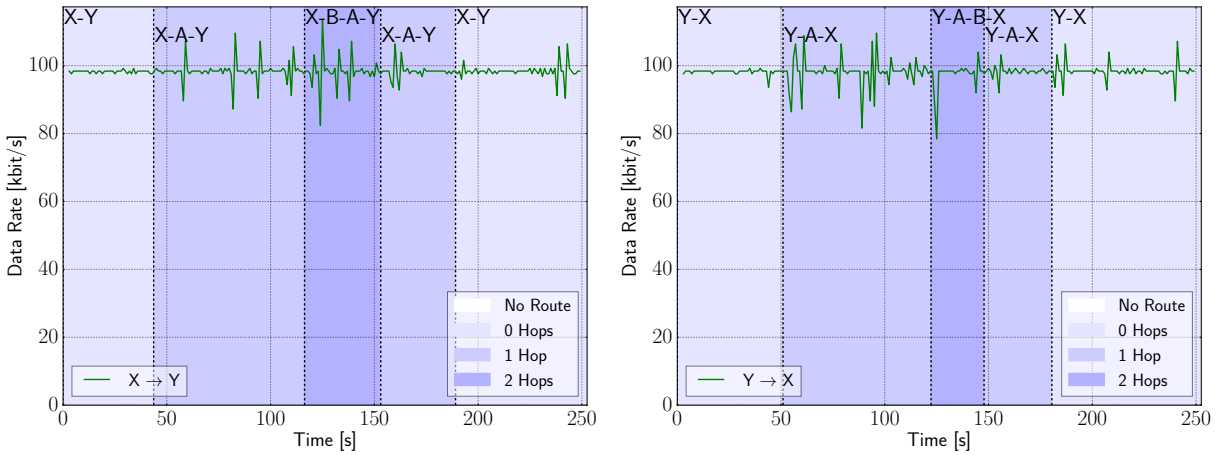


Figure 4.25: Data rate of the TCP connections for the third scenario without secondary network

Figure 4.25 illustrates the data rate plots for the third scenario without the secondary network. Similar to the regular evaluation of the third scenario, the routes change when necessary to maintain the end-to-end connections, and no temporary or permanent connection loss can be observed. The plots show that the algorithm also functions correctly if the information that it requires is transmitted over the primary network. Nevertheless, the rewiring of the topic bridge only represents a workaround. If the link to one of the agents were to fail entirely, there would be no way for it to reconnect to the network once the connections of the topic bridge have timed out. Thus for future experiments, a different mechanism to transmit the required data over the primary network is required.

4.3.4 Summary

The experiments conducted in [10] have shown that handling a dynamic environment with moving nodes represents a challenging task for a routing protocol. The issues that were documented along with the results for Babel and B.A.T.M.A.N. lead to the design of the presented routing algorithm. The experiments which were performed in this thesis demonstrate that the resulting prototype delivers considerably better results in the given scenarios and provides a stable foundation for application level services in a dynamic environment such as the swarm exploration.

5 Conclusion and Further Work

This thesis proposed a new routing algorithm as well as a new metric with the aim to provide good performance in dynamic environments with moving participants, such as a swarm of quadcopter MAVs. The presented metric rates the wireless links between different nodes in the network by predicting the path loss which the radio signals experience during propagation from sender to receiver. The expected path loss is estimated using an empirical log-distance propagation model. In order to fit this model to the current environment, its parameters are continuously updated with on-the-fly measurements using a regularized least squares regression. The resulting link ratings are used by each node, which employs the proposed routing algorithm, to create a local graph of the network. The proposed algorithm uses this graph to determine the most reliable routes in the network and forward data through the network accordingly.

In order to evaluate the accuracy of the metric and to find a suitable set of parameters for the model regression, several experiments were conducted in an outdoor environment. The presented path loss metric has proven to provide an accurate estimation of the signal propagation path loss in a dynamically changing environment, while at the same time stabilizing the noisy measurements of the equipment. The resulting prediction presents a reliable rating of the link quality and a stable foundation for the routing algorithm.

The routing algorithm itself was evaluated through multiple dynamic experiment scenarios that aimed at simulating realistic use cases. The results showed that the algorithm can quickly discover and correctly rate links between different nodes and use this information to route data through the network. It presents a stable foundation for application-level services which require a reliable communication infrastructure to function. Specifically, the experiments showed that even in a line-of-sight scenario, the signal propagation conditions can change considerably over the course of an experiment or from one day to another. This emphasizes the necessity for an adaptive approach which does not merely rely on the spatial position of the nodes in the network, such as several geographic routing protocols do. In comparison to conventional

topology routing protocols, the proposed algorithm provided a much better performance in the tested scenarios. However, it is intended to function in smaller networks with dynamic characteristics and does not aim at replacing these protocols, which were originally designed to function in large-scale or even city-wide networks.

Nevertheless, the implementation of the algorithm employed for the experiments is only meant as a first prototype to test the feasibility of the proposed concept. It still relies on a secondary network to distribute the necessary position and neighborhood updates for the algorithm. A possible next step will be to dispose of this network and distribute the required data over the primary network. The corresponding experiment has already demonstrated that the algorithm can function correctly without secondary network. However, it currently still requires the routing table to be filled according to the initial network configuration. In the future, the algorithm should be enabled to correctly detect a set of initial routes by itself, or the data should distribution be decoupled from the routing mechanism. This could be achieved by distributing the required data via a custom broadcasting mechanism.

Additionally, the algorithm is currently designed to function as isolated service for applications. In a realistic swarm scenario however, the path planning algorithm will require knowledge of the network topology to optimize its flight paths and avoid losing contact to the swarm. As part of future work, the routing algorithm could be enhanced to provide the necessary feedback to the path planning controller of the MAV. With this information, the controller could ensure that certain routes to other agents in the network are maintained and the agent is not isolated from the rest of the swarm.

References

[1] Andrew S. Tanenbaum. *Computer Networks*. 5th ed. Pearson Prentice Hall, 2010.

[2] Fraser Cadger et al. "A Survey of Geographical Routing in Wireless Ad-Hoc Networks". In: *IEEE Communications Surveys and Tutorials* 15.2 (2013), pp. 621–653.

[3] Corinna Aichele. *Mesh: drahtlose Ad-hoc-Netze*. Open Source Press, 2007.

[4] Juliusz Chroboczek. *The Babel Routing Protocol*. RFC 6126 (Experimental). Apr. 2011.

[5] Mehran Abolhasan, Brett Hagelstein, and J. C.-P. Wang. "Real-world performance of current proactive multi-hop mesh protocols". In: *2009 15th Asia-Pacific Conference on Communications*. November. IEEE, Oct. 2009, pp. 44–47.

[6] Rosario G. Garroppo, Stefano Giordano, and Luca Tavanti. "Experimental evaluation of two open source solutions for wireless mesh routing at layer two". In: *ISWPC 2010 - IEEE 5th International Symposium on Wireless Pervasive Computing 2010* (2010), pp. 232–237.

[7] David Murray, Michael Dixon, and Terry Koziniec. "An experimental comparison of routing protocols in multi hop ad hoc networks". In: *2010 Australasian Telecommunication Networks and Applications Conference*. IEEE, Oct. 2010, pp. 159–164.

[8] Florian Zeiger, Nikolaus Kraemer, and Klaus Schilling. "Commanding Mobile Robots via Wireless Ad-Hoc Networks - A Comparison of Four Ad-Hoc Routing Protocol Implementations". In: *2008 IEEE International Conference on Robotics and Automation*. IEEE, May 2008, pp. 590–595.

[9] Marc Liechti. *Real-World Evaluation of Ad Hoc Routing Algorithms*. Tech. rep. Eidgenössische Technische Hochschule Zürich, 2013.

[10] Lukas Magel. "Einrichtung und Evaluation von WLAN Mesh Netzwerken für die Kommunikation in einem mobilen Multiagentensystem". In: *Bericht zum Modul Praxis II* September (2015).

[11] Douglas S. J. De Couto et al. "A High-Throughput Path Metric for Multi-Hop Wireless Routing". In: *Wireless Networks* 11.4 (July 2005), pp. 419–434.

[12] Seungjoon Lee, Bobby Bhattacharjee, and Suman Banerjee. "Efficient Geographic Routing in Multihop Wireless Networks". In: *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing - MobiHoc '05*. New York, New York, USA: ACM Press, 2005, pp. 230–241.

[13] Cheng Peng Fu and Soung C. Liew. "TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks". In: *IEEE Journal on Selected Areas in Communications* 21.2 (Feb. 2003), pp. 216–228.

[14] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. "An Algorithmic Approach to Geographic Routing in Ad Hoc and Sensor Networks". In: *IEEE/ACM Transactions on Networking* 16.1 (Feb. 2008), pp. 51–62.

[15] Quanjun Chen et al. "Adaptive Position Update in Geographic Routing". In: *2006 IEEE International Conference on Communications*. Vol. 12. 3. IEEE, 2006, pp. 4046–4051.

[16] Brad Karp and H. T. Kung. "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks". In: *Proceedings of the 6th annual international conference on Mobile computing and networking - MobiCom '00*. New York, New York, USA: ACM Press, 2000, pp. 243–254.

[17] Jinyang Li et al. "A Scalable Location Service for Geographic Ad Hoc Routing". In: *Proceedings of the 6th annual international conference on Mobile computing and networking - MobiCom '00*. New York, New York, USA: ACM Press, 2000, pp. 120–130.

[18] Richard J. Trudeau. *Introduction to Graph theory*. Vol. 1. Dover Books on Mathematics. Dover Pub., 1993.

[19] Prosenjit Bose et al. "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks". In: *Wireless Networks* 7.6 (Jan. 2001), pp. 609–616.

[20] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. "Unit Disk Graphs". In: *Annals of Discrete Mathematics* 48.C (1991), pp. 165–177.

[21] Young-Jin Kim et al. "On the Pitfalls of Geographic Face Routing". In: *Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing (DIALM-POMC '05)* (2005), pp. 34–43.

[22] Lali Barrière et al. "Robust Position-Based Routing in Wireless Ad Hoc Networks with Unstable Transmission Ranges". In: *Wireless Communications and Mobile Computing* 3.2 (2003), pp. 141–153.

[23] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. "Ad Hoc Networks Beyond Unit Disk Graphs". In: *Wireless Networks* 14.5 (Oct. 2008), pp. 69–78.

[24] K. Ruben Gabriel and Robert R. Sokal. "A New Statistical Approach to Geographic Variation Analysis". In: *Systematic Zoology* 18.3 (1969), pp. 259–278.

[25] Godfried T. Toussaint. "The Relative Neighbourhood Graph of a Finite Planar Set". In: *Pattern Recognition* 12.4 (Jan. 1980), pp. 261–268.

[26] Young-Jin Kim et al. "Geographic Routing Made Practical". In: *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2 (NSDI '05)*. 2005, pp. 217–230.

[27] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. "Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing". In: *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing - MobiHoc '03*. New York, New York, USA: ACM Press, 2003, pp. 267–278.

[28] Andrea Goldsmith. *Wireless Communications*. Cambridge: Cambridge University Press, 2005.

[29] J. D. Parsons. *The Mobile Radio Propagation Channel*. Chichester, UK: John Wiley & Sons, Ltd, 2000.

[30] Tapan K. Sarkar et al. "A Survey of Various Propagation Models for Mobile Communication". In: *IEEE Antennas and Propagation Magazine* 45.3 (June 2003), pp. 51–82.

[31] John S. Seybold. *Introduction to RF Propagation*. Hoboken, NJ, USA: John Wiley & Sons, Inc., Sept. 2005, pp. 1–330.

[32] David B. Green and M.S. Obaidat. "An Accurate Line of Sight Propagation Performance Model for Ad-Hoc 802.11 Wireless LAN (WLAN) Devices". In: *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*. Vol. 5. IEEE, 2002, pp. 3424–3428.

[33] Scott Y. Seidel and Theodore S. Rappaport. "914 MHz Path Loss Prediction Models for Indoor Wireless Communications in Multifloored Buildings". In: *IEEE Transactions on Antennas and Propagation* 40.2 (1992), pp. 207–217.

[34] Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. Electrical engineering. Prentice Hall PTR, 1996.

[35] Taimoor Abbas, Carl Gustafson, and Fredrik Tufvesson. "Pathloss Estimation Techniques for Incomplete Channel Measurement Data". In: (2014).

[36] Marco Zuniga and Bhaskar Krishnamachari. "Analyzing the Transitional Region in Low Power Wireless Links". In: *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.* Vol. 00. c. IEEE, 2004, pp. 517–526.

[37] Abdul Halim Ali et al. "Investigation of Indoor WIFI Radio Signal Propagation". In: *2010 IEEE Symposium on Industrial Electronics and Applications (ISIEA)*. Isiea. IEEE, Oct. 2010, pp. 117–119.

[38] Lorne C. Liechty, Eric Reifsnider, and Greg Durgin. "Developing the Best 2.4 GHz Propagation Model from Active Network Measurements". In: *2007 IEEE 66th Vehicular Technology Conference*. 404. IEEE, Sept. 2007, pp. 894–896.

[39] Santanu Saha Ray. *Graph Theory with Algorithms and its Applications*. Vol. 1. India: Springer India, 2013.

[40] David B. Johnson, David A. Maltz, and Josh Broch. "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks". In: *Computer Science Department Carnegie Mellon University Pittsburgh, PA* (2001), pp. 1–25.

[41] Bundesnetzagentur. *Allgemeinzuteilung von Frequenzen für die Nutzung in lokalen Netzwerken; Wireless Local Area Networks (WLAN- Funkanwendungen)*. `http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Frequenzen/Allgemeinzuteilungen/2013_10_WLAN_2,4GHz_pdf.pdf?__blob=publicationFile&v=4` (visited on 08/22/2016).

[42] Dierk Schröder. *Intelligente Verfahren*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, 840 S.

[43] Thomas Huckle and Stefan Schneider. *Numerische Methoden: Eine Einführung für Informatiker, Naturwissenschaftler, Ingenieure und Mathematiker*. eXamen.press. Berlin/Heidelberg: Springer-Verlag, 2006.

[44] Ascending Technologies. *AscTec Hummingbird*. `http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-hummingbird/` (visited on 09/03/2016).

[45] www.raspberrypi.org. *Raspberry Pi FAQs*. `https://www.raspberrypi.org/help/faqs/` (visited on 09/04/2016).

[46] LogiLink. *Wireless N 150Mbps USB Adapter*. `http://www.logilink.com/media/datasheets/WL0151.pdf` (visited on 09/08/2016).

[47] networkx.github.io. *Overview - NetworkX*. `https://networkx.github.io/` (visited on 09/08/2016).

[48] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA* 3 (2009), p. 5.

[49] Lukas Magel. "Implementierung eines TCP basierten Kommunikations-Interfaces für ein Multi-Agent System". In: *Bericht zum Modul Praxis III* April (2016).

[50] docs.pyroute2.org. *pyroute2 netlink library*. `http://docs.pyroute2.org/` (visited on 09/15/2016).

[51] Ricardo C. Carrano et al. "IEEE 802.11s Multihop MAC: A Tutorial". In: *IEEE Communications Surveys & Tutorials* 13.1 (2011), pp. 52–67.

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

[53] T Clausen et al. *The Optimized Link State Routing Protocol Version 2*. RFC 7181 (Proposed Standard). Apr. 2014.

A Routing Protocols - Further Reading

Protocol	Publication	Website
IEEE 802.11s	[51]	https://github.com/o11s/open80211s/wiki/HOWTO
DSR	[40]	http://www.cs.cmu.edu/~dmaltz/dsr.html
AODV	[52]	http://moment.cs.ucsb.edu/AODV/aodv.html#Implementations
Babel	[4]	http://www.pps.univ-paris-diderot.fr/~jch/software/babel/
B.A.T.M.A.N.	[3]	http://www.open-mesh.org/projects/batman-adv/wiki/Wiki
OLSR	[53]	http://www.olsr.org/mediawiki/index.php/Main_Page

Table A.1: References containing further information regarding the routing protocols

B Cubic Spline Scale Function

The following table lists the support points which were used to generate the scale function 3.4 using cubic splines. Additionally, the first derivative at the first and last support point is defined to be zero.

#	1	2	3	4	5	6
x	0.0	0.2	0.4	0.6	0.8	1.0
y	1.0	1.0	1.0	0.9	0.4	0.0

Table B.1: Support points used to generate the scale function f_α for the linear regression.

C Agent Trajectories during the Evaluation Scenarios

The following figures illustrate the trajectories of the agents during the experiments that were conducted to evaluate the routing algorithm.

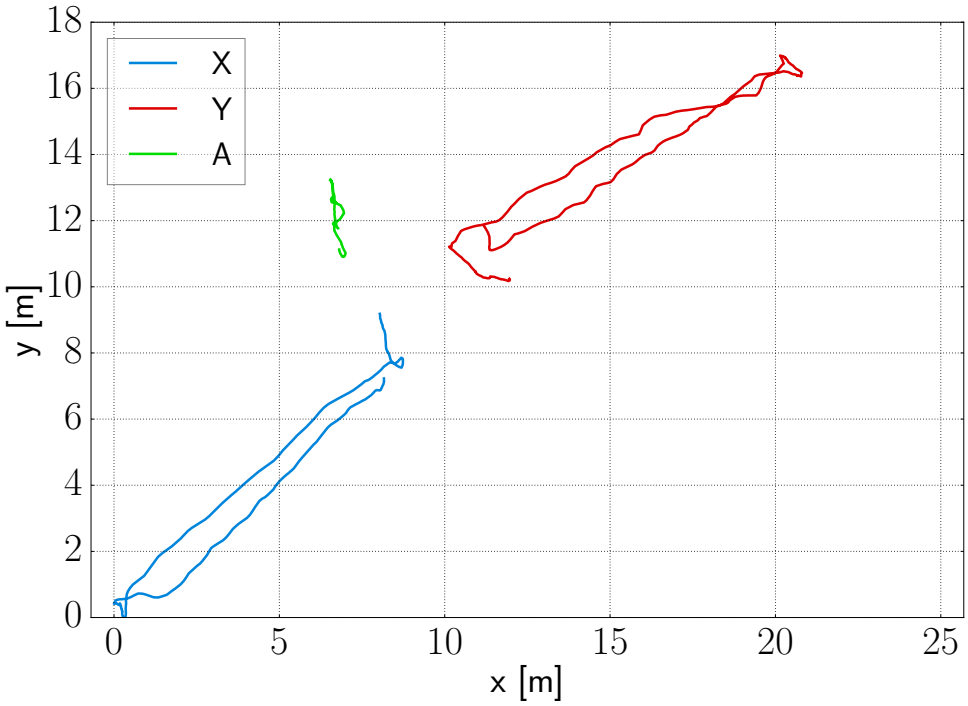


Figure C.1: Trajectories of the agents during scenario 1

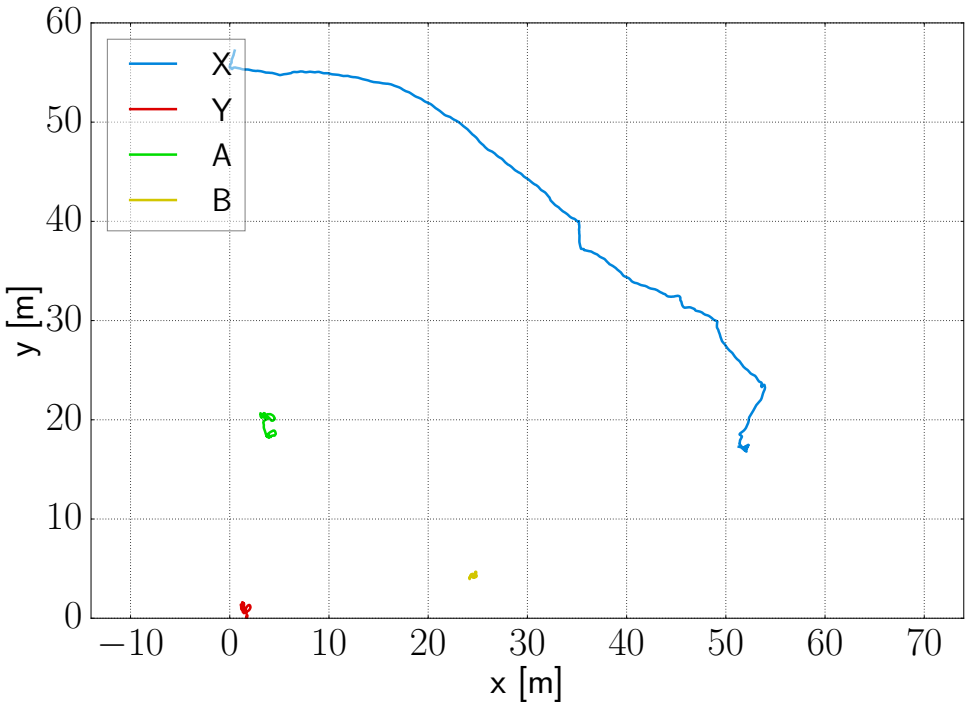


Figure C.2: Trajectories of the agents during scenario 2

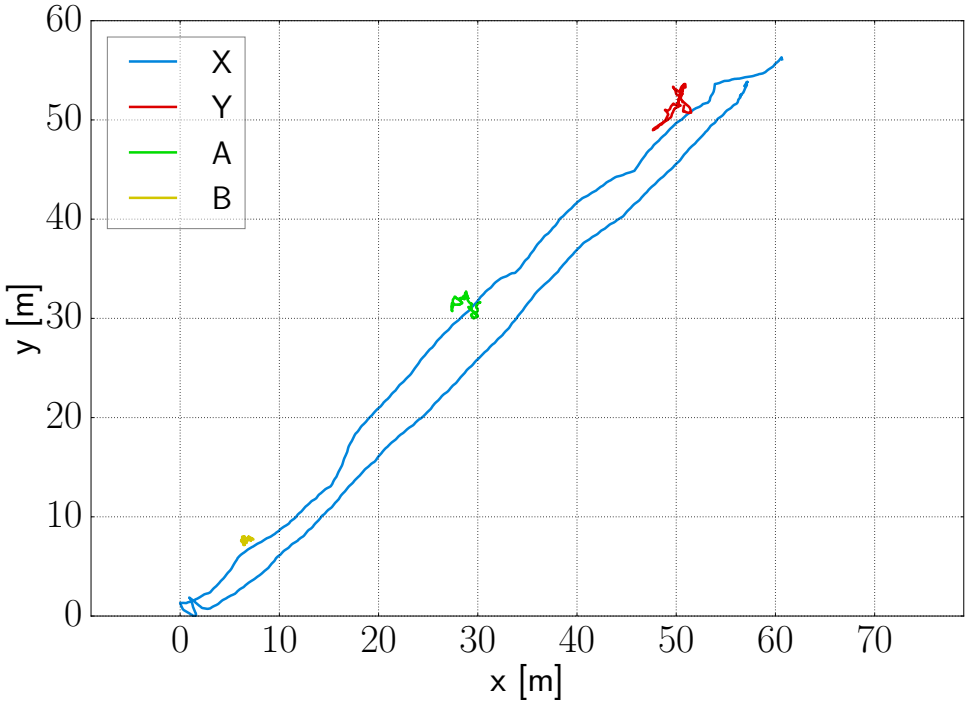


Figure C.3: Trajectories of the agents during scenario 3

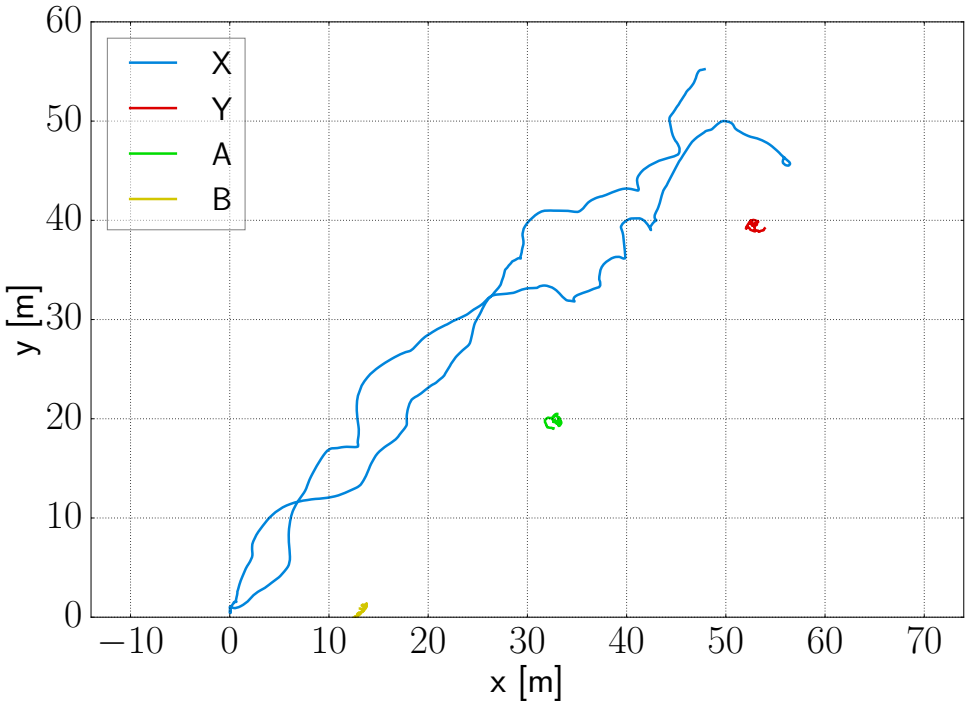


Figure C.4: Trajectories of the agents during scenario 3 with airborne agent

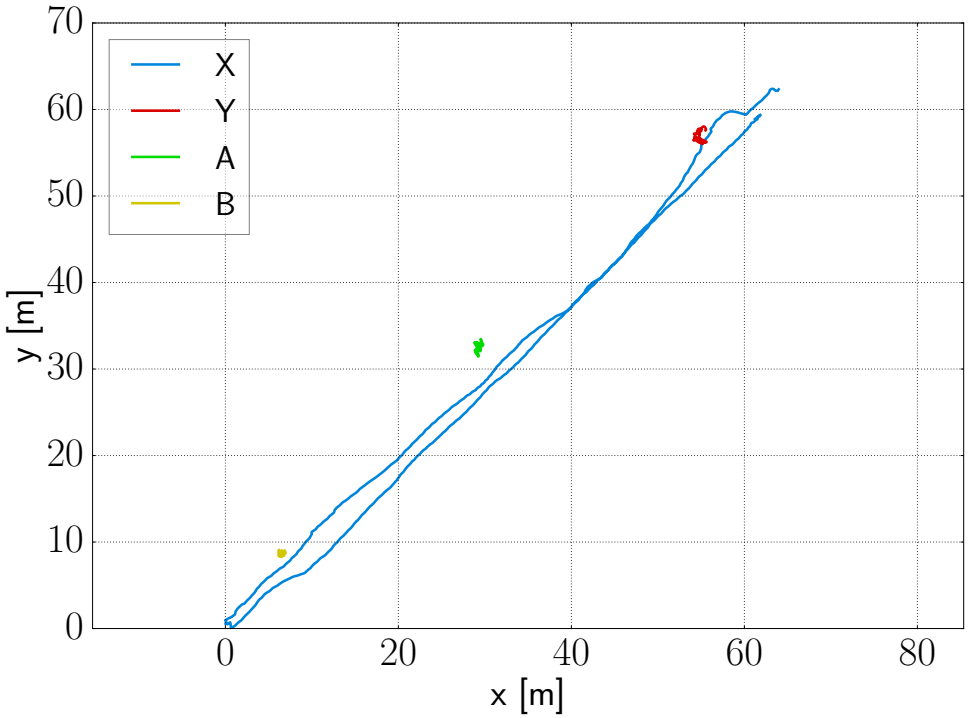


Figure C.5: Trajectories of the agents during scenario 3 without secondary network

D Enlarged Evaluation Figures

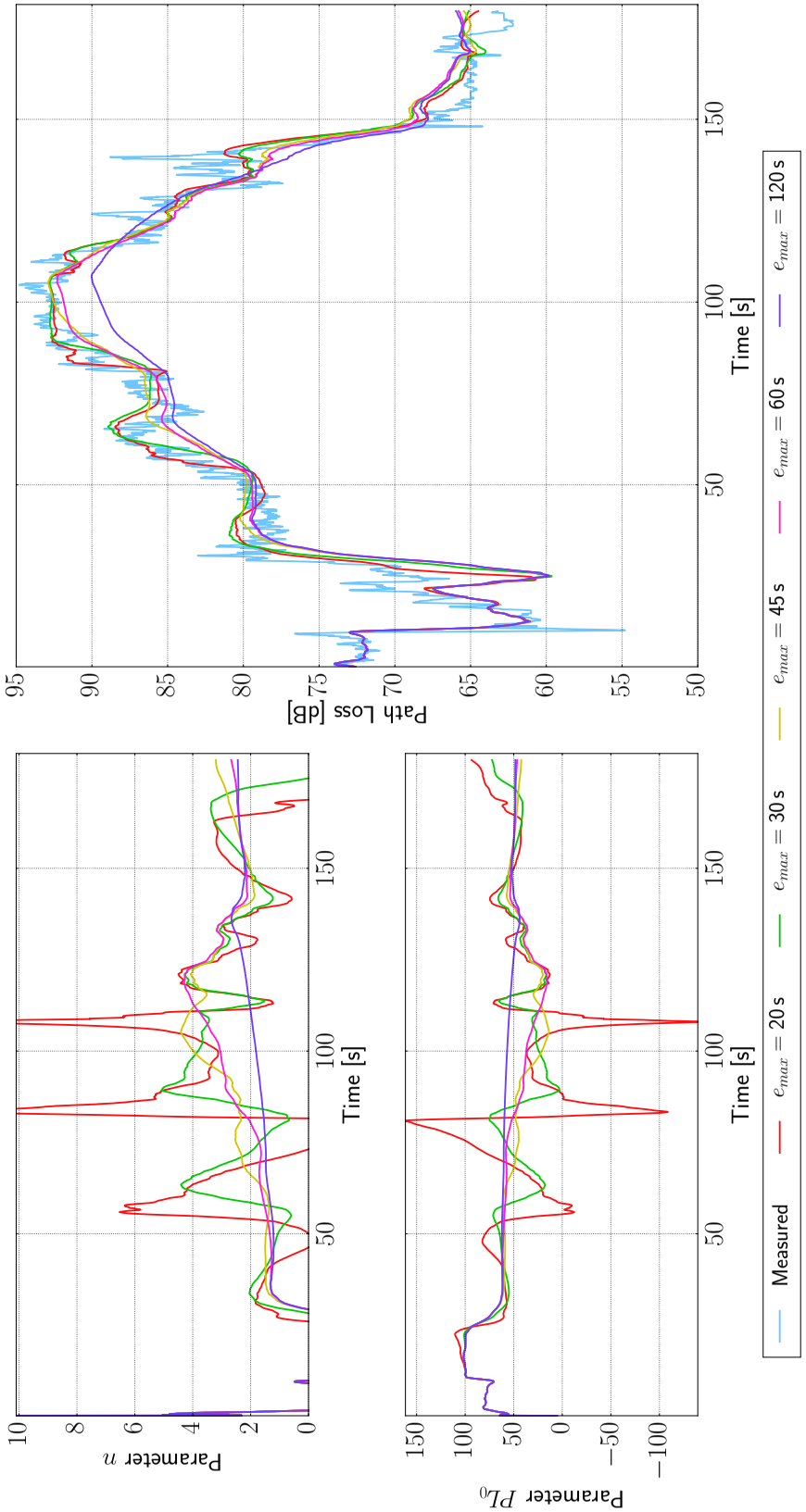


Figure D.1: Enlarged view of figure 4.7

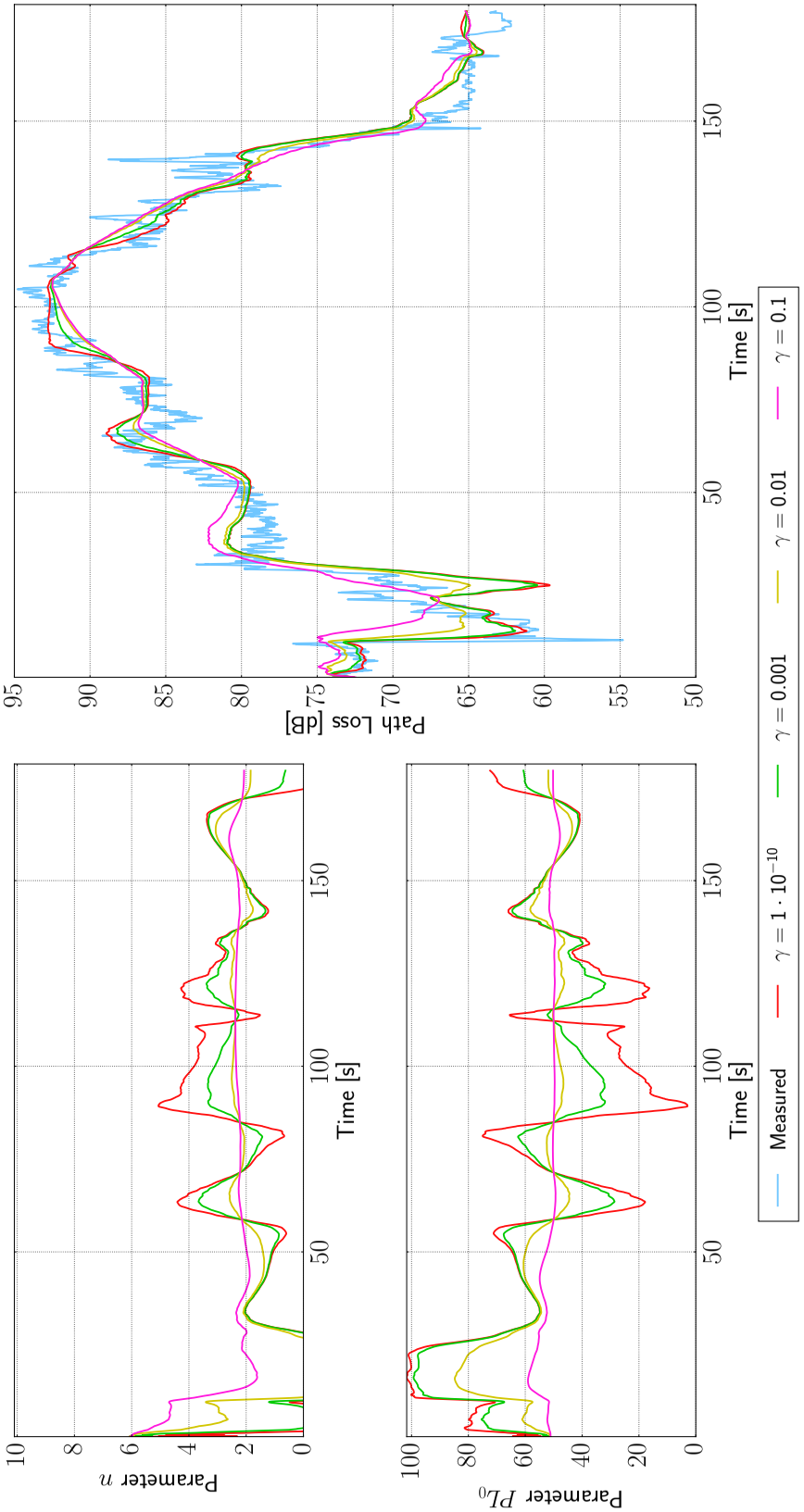


Figure D.2: Enlarged view of figure 4.8

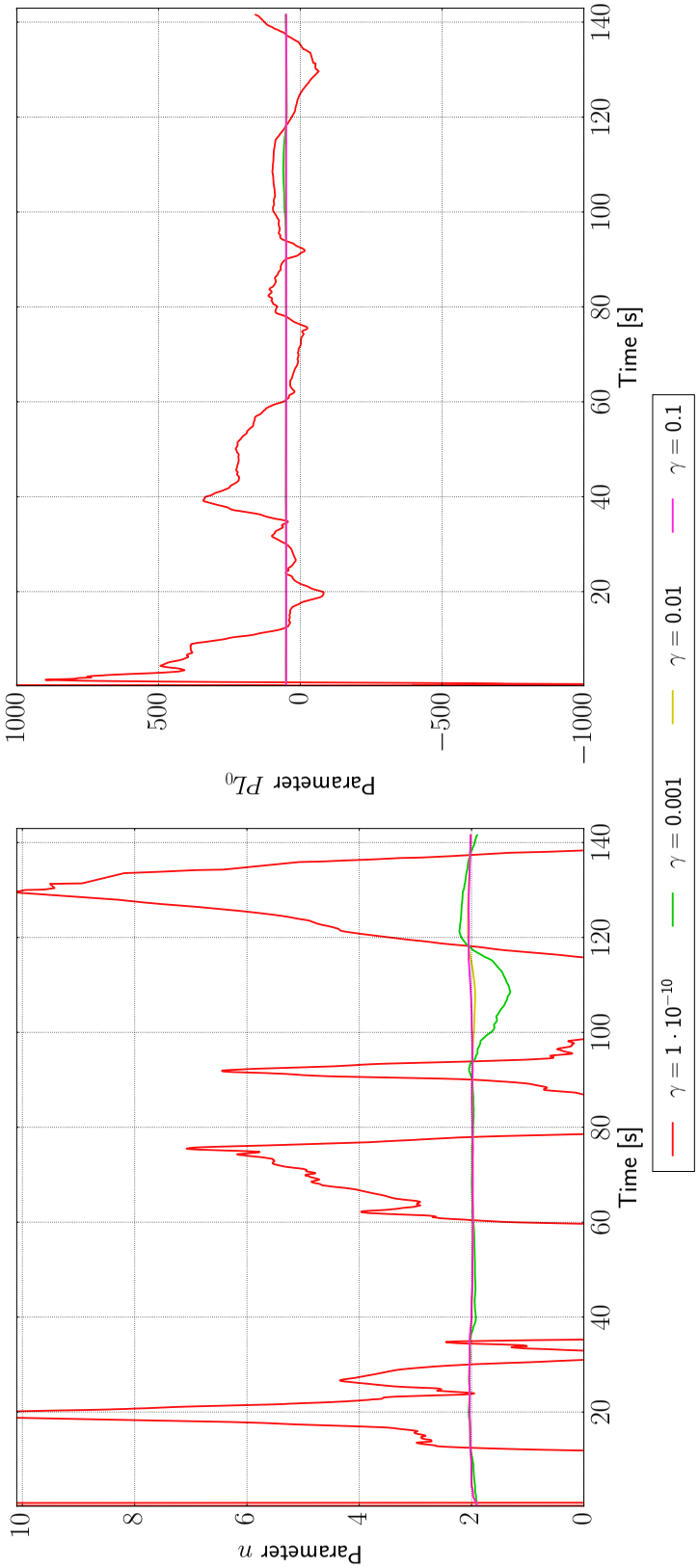


Figure D.3: Enlarged view of figure 4.9