A Framework for Controller-Based Multi-Flow Routing in MANETs

Klement Maria Hagenhoff
geb. Streit

Vollständiger Abdruck der von der Fakultät für Informatik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

genehmigten Dissertation.

Gutachter/Gutachterin:

1. Prof. Dr. rer. nat. Gabi Dreo Rodosek
2. Prof. Dr. rer. nat. Gunnar Teege

der Bundeswehr
Universität München

Die Dissertation wurde am 19.01.2024 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 16.03.2024 angenommen. Die mündliche Prüfung fand am 21.03.2024 statt.

Acknowledgments

Looking back, I am going to remember my Ph.D. as an exciting journey coming along with ups and downs as well as unexpected scientific insights. Reaching the finish line wouldn't have been possible without the help and support of the right people around me during that time.

I'd like to begin with my doctorate supervisor, Prof. Dr. Gabi Dreo Rodošek, who I am grateful to for giving me the chance to reach this accomplishment, providing me with the freedom I needed, doing my research but also on the other hand continuously pushing me with her high demands and expectations. I would also like to thank my co-advisor Prof. Dr. Gunnar Teege for spending hours of time in multiple meetings on discussions and critical questions.

A big thank you goes to all my colleagues, especially to Nils, Raphael, Florian, and Julius. Going through the "Ph.D. life" as a team has given greater significance to the journey. Sharing our thoughts and experiences has helped me grow professionally and fostered further development as a person. I also thank Eike for working with me during his time at UniBw M.

A special thank you goes to my wife Anna, for being such a good listener and motivating me constantly over the last years. And let's not forget the scientific publications, whose grammar you greatly improved over and over again. Thank you for that!

I would also like to thank all proofreaders of this thesis, especially Regina, whose endurance and diligence are remarkable!

Finally, I would like to thank my parents who have created the environment and conditions that have made it possible for me to complete this journey.

Abstract

A FRAMEWORK FOR CONTROLLER-BASED MULTI-FLOW ROUTING IN MANETS

Mobile Ad-Hoc Networks (MANETs) exhibit the exceptional functionality of routing and data delivery being carried out by the clients, distinguishing them from network architectures, where dedicated entities, such as routers, exist especially designed for path construction. MANET participants set up a wireless network, construct paths self-organized, and forward the upcoming data streams hop-by-hop to the desired destinations. This architecture does not rely on any cellular network to provide connectivity, which makes it attractive for the military.

However, routing decisions of participants are often based on obsolete or incomplete network topology knowledge, since keeping routing tables consistent and up-to-date is highly intensive regarding control message overhead. In addition, participants of these networks are moving continuously and their connections are therefore subject to change. These temporary connections are restricted in terms of transmission capacities, thus provoking bottlenecks often resulting in Quality of Service (QoS) violations. Due to these mentioned characteristics, reliable data transfer is a challenge in MANETs.

This thesis proposes a framework addressing the mentioned challenges, such as over-utilized network segments and constantly changing network topology, through a controlled distribution of all flows, also aiming for long connection lifetimes while considering the data rate demands. This framework comprises new techniques to quickly gather a complete and up-to-date network topology overview when a route is requested in a MANET. This can be achieved by establishing a controller outside the network or on any of the participants. In order to meet transmission capacity constraints and to accurately monitor and compute the network's data load, a transmission utilization model is presented and established in the controller's capabilities, thus enabling network load-aware routing. Based on up-to-date topology information, a robust and utilization-efficient routing tech-

nique is proposed, searching for long-living and capacity-conform paths of multiple flows while utilizing the MANETs as little as possible.

We propose and compare several self-designed, extended, and existing pathfinding techniques comprising heuristic and exact methods and investigate beside the mentioned goals the runtimes of all proposed approaches. In order to tackle the constantly changing topology in MANETs, we define and apply a link quality metric for each connection, distinguishing robust from fragile connections aiming for minimum link breaks and longer path lifetimes when computing paths with each proposed pathfinding technique.

To conclude, we propose potential application scenarios and provide recommendations, which pathfinding technique to apply in which use case considering their characteristics, the requirements, and the obtained results. This allows us to determine, which framework configuration is best suited to the respective application scenario in order to calculate long-lasting data transmissions in the shortest possible time, taking into account the capacity limits of the selected connections.

Zusammenfassung

In Mobilen Ad-Hoc Netzen (MANETs) werden die Wegfindung und die Datenübermittlung von den Endgeräten selbst durchgeführt, was diese Netze von herkömmlichen Netzarchitekturen unterscheidet. Die MANET Teilnehmer bauen hierfür eigenständig ein Funknetz auf, um die benötigten Pfade von den Sendern zu den Empfängern zu berechnen. Auch die darauf folgenden Datenübertragungen erfolgen durch die Teilnehmer selbst. Folgerichtig ist diese Netzarchitektur nicht von einer zentralen Instanz abhängig und im Zuge dessen unter anderem auch interessant für militärische Operationen.

Die Routenentscheidungen der einzelnen Teilnehmer beruhen jedoch in vielen Fällen auf veralteten und unvollständigen Netztopologieinformationen. Der Grund hierfür ist zum einen die hohe Mobilität der Teilnehmer, weshalb existierende Verbindungen abrupt getrennt und neue Verbindungen aufgebaut werden. Zum anderen verfügen die Teilnehmer lediglich über eine lokale Sicht des eigentlichen Netzes. Darüber hinaus sind die Übertragungskapazitäten der zu dem Zeitpunkt bestehenden Verbindungen begrenzt, was zu Engpässen und möglichen Verletzungen der Dienstgüte führen kann.

In dieser Dissertation wird ein Framework vorgestellt, welches die genannten Anforderungen, wie die überlasteten Netzsegmente und die sich ständig ändernde Netztopologie adressiert, um langlebige und qualitätsgerechte Verbindungen für die bestehenden Datenübertragungen zu gewährleisten. Dieses Framework beinhaltet eigens entwickelte Algorithmen, welche in kürzester Zeit die vollständige und aktuelle Netztopologie des MANETs zu einem Controller übermitteln. Zwei diesbezüglich angepasste Netzarchitekturen gewährleisten hierfür eine kontinuierliche Verbindungsgarantie zwischen jedem MANET Teilnehmer und dem Controller. Die Erste stellt eine dedizierte Funkverbindung bereit, wenn der Controller auf externer Hardware installiert ist. Die zweite Netzarchitektur installiert den Controller auf einem beliebigen MANET Teilnehmer. Letzteres beinhaltet ein selbstorganisiertes Controllerauswahlverfahren, um sicherzustellen,

dass lediglich ein Controller in einem MANET existiert, sowie eine kontinuierliche Routenpflege für jeden Teilnehmer zum Controller.

Um den beschränkten Übertragungskapazitäten der Verbindungen gerecht zu werden sowie die Datenlast des Netzes genau überwachen und berechnen zu können, wird ein Übertragungsauslastungsmodell vorgestellt und in die Kontrollfunktion des Controllers integriert, um ein auslastungsabhängiges Routing zu ermöglichen. Aufbauend auf den Topologieinformationen des Controllers stellen wir ein robustes und auslastungseffizientes Routing vor, welches potentiell langlebige Routen für mehrere Übertragungen berechnet. Dieser Routingansatz verfolgt das Ziel, die Übertragungslast des Netzes so gering wie möglich zu gestalten. Wir vergleichen und bewerten hierbei selbst entwickelte und bereits existierende heuristische und exakte Pfadfindungstechniken und untersuchen neben den genannten Zielen deren Laufzeiten. Angesichts der sich ständig ändernden Netztopologie stellen wir eine Routingmetrik vor, welche die Verbindungsqualität zwischen Teilnehmern in Hinsicht Stabilität beschreibt, um Verbindungen für Übertragungen mit vermeintlich langen Laufzeiten auszuwählen.

Abschließend werden die vorgestellten Pfadfindungstechniken, welche ebenfalls Bestandteil des Frameworks sind, hinsichtlich ihrer gewonnenen Ergebnisse in eigens definierte Anwendungsszenarien eingruppiert, um deren Vor- und Nachteile geeignet darzustellen. Hiermit können wir eine Aussage darüber treffen, welche Frameworkkonfiguration am besten zu dem jeweiligen Anwendungsszenario passt, um langlebige Datenübertragungen, unter Berücksichtigung der Kapazitätsgrenzen der ausgewählten Verbindungen, in kürzester Zeit zu berechnen.

Contents

CHAPTER 1

Introduction

Communication and computer technology have seen tremendous advancements and growth in the last decade. Business activities as well as social interactions nowadays rely on digital infrastructures. Digitalization is advancing, automating, and speeding up our activities and communications, which is why availability of digital services is vital. *"Being connected"*, e.g., having access to digital services, such as public transport information, government institutions, and doctor appointment platforms among others is more important than ever before.

According to the latest report of Global System for Mobile Communications (GSM) Association, parts of the Third World countries still lack Internet access. The largest coverage gap has been observed in Sub-Saharan Africa, where almost 20% of the area lacks connectivity [DB21]. Alternative network structures are discussed and applied to digitalize these dead zones. For instance, endpoints, such as smartphones can be used to form a single-layer network architecture in which each device is wirelessly connected and each participant has equal capabilities and functions. Such a union of devices forms a network topology, in which everyone is reachable as long as connectivity to one network participant for each node exists.

Mobile Ad-Hoc Networks (MANETs) include these features, having no need of external infrastructure connectivity, such as dedicated routers, base stations, or even cables [Gio02]. Their routing and forwarding behavior characterizes these networks as self-organized and autonomous, as each node is responsible for routing and data delivery, respectively [Toh96]. This means that routing, data forwarding, and delivery are offloaded onto all endpoints participating in the MANET. In addition, participants are wirelessly connected and are moving permanently leading to a continuously changing network topology. Hence, participants move out of each other's communication range forming a new topology. Connectivity to Internet services is provided if at least one participant is within coverage range of a

mobile carrier's base station. If the network is fully autonomous and self-organized, meaning no dedicated link to a base station exists, data exchange is nonetheless available within the network. In that case, participants are forwarding data to directly connected neighbors, using MANET tailored routing protocols to reach the desired destination.

This distributed and self-organized behavior characterizes MANETs as robust, as a certain number of node failures can be covered by the collective. Furthermore, it is more challenging for intruders to eavesdrop or jam ongoing transmissions, as routes change dynamically due to mobility.

MANETs are frequently discussed to be used in military operations since this architecture does not rely on a single point of failure [Bar13]. Furthermore, Hong Kong's citizens took advantage of these characteristics and formed a MANET to organize demonstrations [Wak19]. According to simulations, MANETs are also frequently debated to be a promising alternative in catastrophic situations, such as forest fires or flash floods, where power supply of base stations is prone to fail at any time [Jah+18].

Despite the aforementioned positive aspects, the mobility, the lack of a central routing instance, and wireless connections make the management of these networks more challenging for a number of reasons: (i) Keeping routing tables of all participants up-to-date and complete is complex and costly in terms of routing load, as nodes must exchange their sub-topology with all MANET members. In addition, the topology changes continuously as nodes are moving, forcing participants to update their routing tables all the time. (ii) Wireless channels experience data rate restrictions especially in the military sector, reaching $0.5\ Mbps$ [LAM12] and up to $2.0\ Mbps$ [MKL15] depending on the waveform. Participants must keep track of available resources to not overload network segments, especially if a new transmission is intended. This poses even more of a challenge, as neighbors interfere with each other if both forward data streams.

As a result of inconsistent routing table entries, nodes, for instance, route data to dead ends. In that case, all traffic addressed to this destination is dropped as long as rerouting is ongoing. Maintaining consistency and up-to-date topology information for all participants is only achievable at the expense of increased routing overhead. Both additional rerouting and routing overhead drain the MANET's transmission capacities. It is obvious that data rates should be used wisely so that the objective is to find routes that utilize the MANET the least and to exploit the available resources as efficiently as possible. Also, flows should exhibit minimum mutual interference with other transmissions to provide sufficient capacities for future

flows.

1.1 PROBLEM STATEMENT

In recent years, we have observed a continuously increasing demand for mobile data and in the future, it is expected to supersede the amount of wired traffic [Ericcson]. A major share, approximately 79%, of this traffic is multimedia and streaming content [Cisco]. This real-time traffic has individual Quality of Service (QoS) requirements. They differ with respect to delay, throughput and loss rate. Voice over IP (VoIP), for instance, relies on a delay of less than $150\ msec$ and recommends a data loss rate of less than 2% [SH04]. Video streaming requires a data rate between $800 - 1200\ kbit/sec$ to display an acceptable quality with a resolution of $720\ px \times 486\ px$ [Adobe]. Transferring a file, in contrast, requires on the one hand 100% correct data delivery but on the other hand sticks to best effort services with respect to delay and data rate demand.

These real-time services also require attention in MANETs and in turn create new challenges. MANETs are characterized by their distributed and self-organized network architecture in which mobile participants act as clients and routers, respectively. However, the wireless channel only provides restricted data rates that are easily exhausted, especially when transmitting real-time data, such as video streams.

A network transmission capacity measurement concept could avoid over-utilized situations, by monitoring the residual transmission capacity of each MANET member. QoS supported routing protocols for MANETs have been researched to guarantee data rate and delay-confirm delivery [HT07]. The complexity of measurement and computation increases in multi-hop wireless networks especially when reaching capacity boundaries of connections [NL07; RGM18]. Here, computations must be accurate to cover uncertain factors, such as randomness and unforeseen waiting periods for members to avoid interferences with another transmissions. It must be taken into account that flows, which are routed across multiple members utilize connections even more, as relay nodes must receive and forward data [Bia00]. This often results in over-utilized network segments and traffic congestions, as links for routes are chosen although their transmission capacities are insufficient. Consequences are delayed data delivery, stressed network segments, and in the worst case dropped data. From the users' perspective, parts of real-time data transmissions like video calls would appear choppy and parts of sentences would be omitted. To put it concisely,

monitoring the accurate capacity of a MANET when transmitting a flow via multiple route participants is very complex and challenging especially if the data rate demand of the flow reaches the transmission capacities of path members.

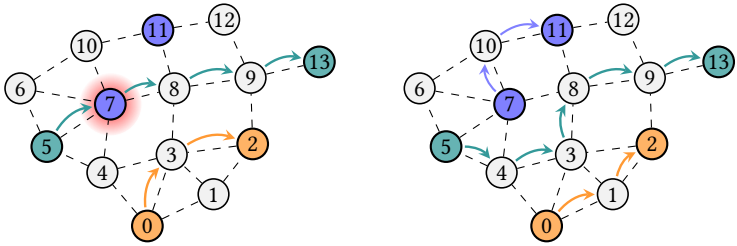
Monitoring and determining accurate utilization is even more challenging if MANETs are used to transport multiple flows. In this case, mutual interference of different transmissions increases. Wireless interfaces are more often occupied as multiple flows are routed in the MANET, increasing the probability that route participants of different flows must share the channel capacity as they are in each others coverage range, resulting in reduced transmission capacity per node. As a consequence, retransmissions caused by interference as well as competitive Medium Access Control (MAC) methods lead to longer sensing and contention periods, also decreasing transmission capacities.

Routing must also take into account transmission capacities of already deployed flows, when searching for a path for an upcoming flow. The data rate of an upcoming flow drains transmission capacities of connections either because of the nodes that are chosen as path participants or due to interference. Keeping that in mind, a potential path of an upcoming flow must be aware of its occurring interference and the residual data rates of surrounding links to not over-utilize currently transmitting links.

The combination of restricted transmission resources of wireless connections in MANETs and data rate demanding real-time services, such as video calls, makes it necessary to exploit available capacities most efficiently to be able to deploy as many flows as possible. This, despite an accurate transmission utilization model, requires an accurate and reliable overview of all flows, their demands, and the current topology including transmission capacities of links.

However, the decentralized routing behavior of MANETs complicates the mentioned requirements. Nodes make individual forwarding decisions based on their routing table entries, referring to common protocols [Cla+14; Neu+08; DPB03; HMJ07]. Hence, generated paths more likely disturb each other as decentralized routing does not change routes of existing transmissions to free capacities for an additional flow. As a consequence, achieving a common routing objective in decentralized network structures, like utilizing the MANET transmission capacity the least, is challenging since path construction is not centrally managed.

Figure 1.1 introduces the routing problem visually. There, utilization is related to nodes. We aim to explain efficient routing in terms of minimum



(a) Two flows $f_{(0,2)}$ and $f_{(5,13)}$ are deployed each utilizing their paths entirely. Flow $f_{(7,11)}$ must be deployed.

(b) Rerouting of flow $f_{(0,2)}$ and $f_{(5,13)}$ to free capacities for $f_{(7,11)}$.

Figure 1.1: Example MANET topology to illustrate an occurring over-utilized situation. Black dashed lines illustrate connections (links) between nodes. Solid arcs represent deployed paths from origin to target, generating utilization.

transmission capacity utilization in a simplified way. It does not coincide with the defined utilization model of this thesis. According to that, we specify with respect to this example that any node is only able to carry a single flow to not be over-utilized. As depicted in Figure 1.1a, flows $f_{(0,2)}$ and $f_{(5,13)}$ are deployed, utilizing each active path participant completely. An additional flow from 7 to 11 must be deployed, leading to an over-utilized situation as node 7 would have to carry two flows. Acting collectively to identify sufficient transmission capacity and reroute already deployed flows is not addressed in present routing protocols. The lack of a central routing instance performing path computation and defining forwarding rules makes it complex to deploy all flows in a capacity-conform way. Also, distributed routing protocols do not provide tailored route computation concepts, focusing on minimum utilization during route construction to deploy as many flows as possible. Hence, flow $f_{(7,11)}$ would either not be deployed or over-utilize the MANET.

A possible rerouting approach is depicted in Figure 1.1b. New paths are defined for $f_{(0,2)}$ and $f_{(5,13)}$ to free transmission capacities for the additional flow.

Applying a centralized routing concept focusing on utilization-efficient path computations requires reliable routing information, that is, complete and up-to-date topology information. Complete means that all nodes and each physically existing connection are represented in the topology snap-

shot before routing takes place. Up-to-date means that no obsolete links exist in the topology representation. Heading for these requirements is challenging as nodes must exchange their topology information with each other. The mobility of nodes and the ever-changing topology lead to incomplete and outdated routing information, resulting in inconsistent routing entries of nodes. Neighborhood information must periodically be flooded across the MANET, delaying topology changes. The delay depends on a configured interval. Reducing this interval increases the reliability of information but on the other hand, increases the routing overhead, which must be kept minimal as transmission capacities are restricted. Node failures and lost connections remain in routing tables until update messages are carried via multiple hops along the entire MANET. These obsolete routing tables result in outdated paths and connections, leading to rerouting approaches and, in the worst case, to dead ends.

To sum up, data rate demanding real-time traffic often overloads the restricted transmission capacities of MANETs especially if multiple flows must be delivered. MANETs are currently lacking an accurate utilization model to determine transmission capacities of links in highly utilized situations. Also, decentralized routing complicates utilization-efficient routing, leading to unnecessarily occupied segments, as no central routing concept is present that takes over path computations. In addition, MANETs lack uniform and up-to-date topology knowledge, which is a necessary prerequirement for building a centralized routing concept.

1.2 RESEARCH QUESTIONS

Taking the aspects mentioned in the previous section into account, we conclude that current MANETs routing approaches are not designed to face multiple flows carrying real-time traffic. This is mainly due to the self-organized network characteristic, which keeps routing within the collective strength of all members. The following MANET characteristics highlight once more the mentioned challenges of Section 1.1: (i) The restricted transmission capacities, (ii) the incomplete and outdated topology knowledge, and (iii) the isolated routing behavior of network participants. Based on these, we connect the dots and formulate the following Research Questions (RQs) to tackle the challenges:

RQ 1 *How to record and compute accurate connection utilization during capacity-demanding transmissions?*

RQ 2 *Which controller-equipped MANET modifications are required to provide a complete topology representation for an outsourced routing instance?*

RQ 3 *How to provide and access complete topology knowledge when the controller is deployed on a MANET member?*

RQ 4 *To which extent can time-sensitive routing compute robust paths of multiple flows while not over-utilizing the network?*

With RQ 1, we study the development of network utilization of single nodes as well as of routes carrying traffic. This includes monitoring the occupied states and time spans of the nodes' wireless interfaces among others. Using this knowledge allows us to determine how the injection of flows in MANETs influences chosen connections and additionally constructed routes. Thereby, we answer the question to which extent an accurate link utilization can be computed especially during high network load. Addressing RQ 1, we also introduce a self-developed, extended MANET architecture, see Section 4.2.1, having a central routing instance, named the controller, which is externally reachable for each MANET member. MANET members exchange management information with the controller via a dedicated channel while keeping all payload data within the MANET. We integrate our utilization model which benefits from the uplink, as capacity measurements and computations are outsourced to the controller. This Software-defined Networking (SDN)-aligned approach for mobile wireless networks is extended with further capabilities throughout the thesis.

Topology knowledge including nodes and links forming a graph must be consistent and provided to a dedicated instance responsible for path computation. This avoids decentralized routing behavior and prevents isolated decisions, aggravating the routing situation. Having this routing instance facilitates the collective objective to distribute flows with minimum utilization overhead. Heading for the mentioned goal, RQ 2 proposes a topology update concept to transfer this knowledge to an outsourced controller reachable via a dedicated wireless channel. Guaranteeing an up-to-date and complete snapshot requires an almost interference-free topology data aggregation and delivery. Otherwise routing knowledge on the controller would be more likely outdated and incomplete. Hence, we propose a reactive topology update concept focusing on the least response time, which starts conveying topology information to the controller immediately before routing takes place. This concept is applied to the controller-equipped MANET architecture and extends it with roles and processes to

handle different states and situations at runtime. In particular, we present MAC modifications for the MANET architecture managed by this external controller, which leads to reliably managed scheduling cycles to better control an interference-free topology update mechanism.

The previously introduced architecture comprises a MANET member and maintenance concept, a renewed topology update technique, and a flow computation and deployment strategy. Evaluations compare this approach with the proactive topology update technique and also investigate the newly defined routing strategy applying Ad-Hoc on-demand Distance Vector (AODV) as a counterpart.

Assuming solid availability of an outsourced routing instance is risky, as obstacles, interference, and further environmental influences disturb connectivity between MANET members and the controller. Also, nodes move out of the controller's coverage range and in the worst case, controller outages can occur. RQ 3 tackles situations in which the dedicated controller channel is not available. We introduce an algorithm, gathering the topology knowledge to an arbitrary node, guaranteeing similar capabilities compared to a controller reachable via a dedicated channel. This approach partitions controller and MANET only logically, meaning that the controller is also a MANET participant. All members of the MANET use one channel for data and management overhead. The topology-gathering process focuses on completeness in terms of nodes and connections and also provides minimum runtime to perform routing on an up-to-date network representation.

RQ 3 also includes a controller self-management concept tackling the following capabilities: (i) Path knowledge and maintenance for each member to reach the controller in case a route must be requested (ii) Minimum control data overhead to provide maximum MANET transmission capacity for data delivery. (iii) A self-managed controller election guaranteeing a single controller per MANET in case of controller outages and MANET merge situations. Nodes maintain a tree having the controller as root, previously constructed during complete topology gathering. Control message overhead between an arbitrary node and the controller is relayed along the associated branch that connects participant and controller. Quality parameters characterize the controller and are key decision makers in case multiple MANETs merge or in case of current controller outages.

Combining the utilization model and one of the controller-equipped MANET architectures allows us to investigate if and to which extent the MANET capacity can be exploited to compute routes that are robust and do not over-utilize the network. We thereby place a strong focus on rout-

ing in highly-utilized situations. There, complexity and difficulty of finding capacity-conform paths for multiple flows increase since fewer possible solutions exist. In doing so, we at first compute over-utilized situations generated with common shortest path algorithms. Based on that, we compute paths for the same flows (origin-target pairs) that do not over-utilize any link within the network.

Keeping in mind the mobility of members forces path computation to respond quickly as nodes move out of each other's communication ranges, resulting in topology changes. Routing concepts must take into account such behavior and adapt their path computations to long-living connections. In doing so, path techniques focus on strong reception powers, moving directions, and speed of members to find robust routes to participants. Keeping that in mind, RQ 4 develops, extends, and applies exact and heuristic pathfinding approaches focusing on fast response time and long-living links in terms of connectivity. A custom link quality metric identifies robust among unstable ones while optimizing the result. We discuss characteristics of all computation methods and give recommendations on which to apply based on the use case. Since this thesis examines two approaches having the controller either deployed within the MANET on an arbitrary node or reachable via an out-of-band channel, each path computation technique has its pros and cons regarding both architectures.

To sum up, we provide a centralized routing framework for MANETs with the focus on exploiting restricted transmission data rates of nodes in the most efficient way. We therefore first propose an accurate utilization model to monitor and compute the MANET's capacity, especially in heavily loaded situations. Secondly, we develop a topology update concept having the controller reachable via a dedicated channel, providing the possibility to route based on up-to-date and complete topology knowledge. Thirdly, we also provide full and up-to-date topology knowledge for routing purposes stored on an arbitrary MANET member. This concept also proposes a tailored concept providing and maintaining a route for each participant to the controller. Participants leverage this route for path requests in order to receive a route to their requested destination. Lastly, several pathfinding approaches are applied to evaluate to which extent available capacity can be exploited most efficiently.

1.3 CONTRIBUTIONS

In this section, we list and summarize all peer-reviewed conference papers and posters directly related to this thesis.

Conference papers directly associated with the thesis

This thesis identifies, examines and closes several gaps in the research area of MANETs. In particular, we published scientific articles affecting the areas of wireless transmission capacity measurements, network routing aspects, and multi-flow route optimizations. In total, this thesis contributes eight peer-reviewed full papers, one short paper, and one poster publication.

- Jörg Hähner, **Klement Streit** and Sven Tomforde. "Cellular Traffic Offloading Through Network-Assisted Ad-Hoc Routing in Cellular Networks". In: *2016 IEEE Symposium on Computers and Communication (ISCC)*. 2016, pp. 469–476

  Summary: This publication proposes foundations of the controller-equipped MANET architecture having an outsourced routing instance reachable via an out-of-band channel. The presented evaluation discusses performance parameters, such as routing load, throughput, and packet delivery rate, among others. Results are promising, as the designed architecture can cope with the well-known routing protocol AODV.

- **Klement Streit**, Nils Rodday and Gabi Dreo Rodosek. "AODV-CBR: Capacity-based Path Discovery Algorithm for MANETs with High Utilization". In: *2018 Advances in Wireless and Optical Communications (RTUWO)*. 2018, pp. 234–239

  Summary: This publication proposes a transmission utilization model suitable for self-organized and distributed routing in MANETs. The objective is to examine the accuracy of the model and to determine whether evenly distributed flows across the network increase the throughput. Results show that distributing paths transmission overhead-wise slightly exploits available MANET capacities if heading for minimum utilization.

- **Klement Streit**, Nils Rodday, Florian Steuber, Corinna Schmitt and Gabi Dreo Rodosek. "Wireless SDN for Highly Utilized MANETs". In: *2019 International Conference on Wireless and Mobile Computing,*

Networking and Communications (WiMob). 2019, pp. 226–234

SUMMARY: This publication extends the controller-equipped MANET concept, which connects nodes and a controller via a dedicated channel with a tailored utilization model. We present a measurement method for Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) and also propose computation approaches regarding entire routes to predetermine required capacities of upcoming flows. Evaluations focus on the accuracy of the measurement and computation methods. We compare the computed MANET transmission overhead with the measured one and experience reliable computation prediction if actual arising network utilization is below 90%.

- **Klement Streit** and Gabi Dreo Rodosek. "Cetup: Controller-equipped Topology Update Process for Tactical Ad-hoc Networks". In: *Proceedings of the 17th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*. 2020, pp. 57–66

  SUMMARY: The authors present a centrally managed Time-division Multiple Access (TDMA) method to support the MANET topology update process, including multi-route adaption. This contribution also proposes a participant management concept which is, as well as the topology update process integrated into the centralized controller-equipped MANET architecture. Results verify improvements compared to the initially designed architecture, lacking an utilization and reactive topology update concept.

- **Klement Streit**, Eike Viehmann, Florian Steuber and Gabi Dreo Rodosek. "Improving Routing with Up-to-date and Full Topology Knowledge in MANETs". In: *2020 Military Communications and Information Systems Conference (MilCIS)*. 2020, pp. 1–8

  SUMMARY: This publication introduces a topology update algorithm having the controller deployed on an arbitrary node in the MANET. The algorithm constructs a tree while gathering topology information, to later transport all local link knowledge from the leaves toward the root aka. controller. Requirements are fast response time and completeness of the reported sub-topologies. The evaluation covers an implementation on microcontrollers and a simulation to test the algorithm with a large number of nodes.

- **Klement Streit**, Corinna Schmitt and Carlo Giannelli. "SDN-Based

Regulated Flow Routing in MANETs". In: *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2020, pp. 73–80

SUMMARY: This publication covers two research areas. First, we propose a link quality metric applicable to shortest path algorithms to detect robust paths in terms of connectivity lifetime. Second, the publication proposes a flow data rate regulation to reduce transmission rates if connections lack the flow data rate requirements. We experience longer path connectivity lifetime applying the link quality metric.

- **Klement Hagenhoff**, Maximilian Tränkler, Corinna Schmitt and Gabi Dreo Rodosek. "RTC: Route to Controller Algorithm Providing SDN Capabilities in MANETs". In: *MILCOM 2022 Track 2 - Networking Protocols and Performance (MILCOM 2022 Track 2)*. Rockville, USA, Nov. 2022

  SUMMARY: The authors present a lightweight route maintenance algorithm to permanently provide a route to the controller for each MANET member. Having a controller with topology knowledge deployed on a participant requires to provide a route to send route requests to. Requirements are permanent reachability and minimum routing overhead to provide the most of the restricted capacity for data transmissions. The contribution covers the conceptual design and preliminary simulation results of the route-to-controller concept.

- **Klement Hagenhoff**, Eike Viehmann and Gabi Dreo Rodosek. "Time-sensitive Multi-Flow Routing in Highly Utilized MANETs". In: *18th International Conference on Network and Service Management (CNSM)*. Thessaloniki, Greece, Oct. 2022

  SUMMARY: The publication presents several path computation algorithms facing capacity conform routes. Since two up-to-date and complete topology update concepts that focus on minimum runtime are proposed [Str+20; SD20], path computation also requires responding in the least time. Also, we answer the question to which extent over-utilized MANET segments are resolved by the presented pathfinding techniques. We tackle this challenge on the basis of the previously proposed utilization model to determine network capacity [SRR18; Str+19b], combined with the robust link quality metric, published with publication [SSG20]. Results provide an indication

that applying the link quality metric extends the lifetime of computed routes. Beside the proposed pathfinding techniques, we contribute a framework embedding MANET specific characteristics, like realistic radio wave propagation, node mobility patterns, all pathfinding techniques, utilization model, and full topology knowledge, offering the opportunity for testing further pathfinding techniques.

Conference poster and short papers directly associated with the thesis

- **Klement Streit**, Raphael Labaca Castro, Nils Rodday and Gabi Dreo Rodosek. "Topology Update Algorithm for Wireless Networks". In: *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*. 2019, pp. 184–185

  Summary: The poster proposes first insights in the centralized controller update algorithm. The concept describes message types, their sequences and user roles for the controller-equipped MANET architecture. The authors highlight challenges and first results concerning the outsource controller-equipped neighbor update process.

- **Klement Hagenhoff**, Eike Viehmann and Gabi Dreo Rodosek. "Eliminating Bottlenecks in MANETs". In: *NOMS 2024-2024 IEEE/IFIP Network Operations and Management Symposium*. 2024, pp. 1–5

  Summary: The short paper introduces additional pathfinding algorithms facing capacity-conform routes in minimum runtime. To the end, the proposed simulation model of [HVD22] is used to implement and evaluate all techniques. We extend the pathfinding approach of Akin et al. [AK19] with respect to our objective and also to consider shared channel access of MANET participants. In the end, we compare the results with a self-developed pathdinfing technique.

1.4 FIELDS OF APPLICATION

MANETs are said to be the generic term of various network types adopting basic characteristics. This section introduces specific subtypes of MANETs that our capacity-aware multi-flow routing concept is applicable for. We elaborate on the specific network architectures, their technical specifications, and also propose a concrete use case, respectively.

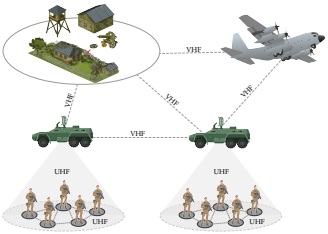
1.4.1 Mobile Tactical Networks

Due to the robust and non-transparent nature of MANETs, military and civil protection organizations invest in communication equipment to improve and facilitate their interactions. The Chilean fire company together with computer science departments, belonging to local universities, invested in hardware and research to simplify communication and organization of firefighters in action. A handheld device that provides infrastructure and adhoc mode is used, allowing firemen to ask for locations of specific streets or fire trucks and also to track each other. The devices automatically create a MANET if infrastructure service is unavailable, providing alarms and voice messages. [Mon+11]

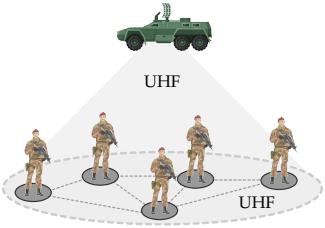
With regard to the military, the United States (US) Army for instance plan in the future to support and facilitate observations with camera-equipped drones connected to MANETs [Ham20]. Likewise, Germany and the Netherlands are catching up in shipping Internet Protocol (IP) radios for grounded soldiers and armored vehicles, forming MANETs [Dor19; BWI20].

Today's armed forces organize their communication through tactical heterogeneous network architectures in which participants e.g. headquarters, tanks, units, and soldiers are hierarchically connected through several wireless channels [NY18]. This defacto designed and generalized architecture has not become established in daily missions. There exist specific standard radio communication systems, interfaces, and sensors providing certain requirements. Hence, military communication architectures vary depending on the mission's objective.

Figure 1.2a introduces a possible military communication architecture.



(a) Communication architecture during military missions using different communication networks.



(b) Parts of the Mobile Tactical Network (MTN) architecture is applicable for the use case.

Figure 1.2: Example military communication architecture.

Satellite Communication (SATCOM) is applied for wide-range connectivity, enabling comprehensive communication between for instance Unmanned Aerial Vehicles (UAVs) and the headquarter. Dismounted troops, divided into units, are equipped with handheld radios providing Ultra High Frequency (UHF) for communication within the group. Soldiers operating on the ground are commonly mapped to an upper tier unit like sections or company command vehicles, also using UHF. Narrowband waveforms are used to enable communication between operation vehicles, which are also connected to the headquarter using Very High Frequency (VHF) channels. [Sur+16]

Data rates are limited according to existing specifications and security constraints. Speaking of North Atlantic Treaty Organization (NATO), their upcoming operational Narrowband Waveform (NBWF) utilizing a $25\ kHz$ channel shrinks margins of high throughput modes, having trouble providing data rates beyond $0.5\ Mbit$ [LAM12]. Dismounted soldiers carrying radios can draw from a little more capacity but also cope with low specifications. The Soldier Radio Waveform (SRW) designed for handhelds of US Army running CSMA/CA on MAC layer reaches peak data rates of $2\ Mbit$ [MKL15].

Switching manually between different communication networks during military operations to relay orders and instructions is time-consuming and error-prone. This old-fashioned composition of different communication types is well-known, which is why researchers try to exchange complexity with software-controlled approaches. Such techniques often follow a hierarchical structure to automatically and centrally manage network transitions. Such SDN-equipped military network architectures enable unprecedented programmability of physical network parts promising improvements in routing regarding restricted data rates and policy-based delivery in order to guarantee need-to-know constraints [VDC16].

Hence, several architectures, also known in the research community as SDN-equipped Tactical Ad-Hoc Networks and MTNs are proposed, investigating placement strategies of SDN controllers in heterogeneous military architectures. These architectures take into account resilience, reliability, reachability, and further military requirements. Published work reaches from hybrid controller approaches covering parts or the complete network architecture on one or several controllers [Phe+16; PIT18; Nob+16; Pou+19a]. These approaches aim, among others, to simplify routing and to design data delivery more reliable. Research often leverages SDN characteristics, such as scalability and programmability of devices in order to cope

with the restricted data rates of transmissions.

Inspired by related work and their described SDN-equipped Tactical Ad-Hoc Networks and MTNs, we propose two use cases, based on the military architecture in Figure 1.2b, where a controller, equipped with routing capabilities could simplify data delivery. Their structure comprises an upper tier command or operation vehicle providing long range, low transmission capacity connectivity for a dismounted soldier troop that form a MANET. For this use case, we deploy a controller, providing routing capabilities on the command vehicle. Grounded soldiers request routes via the uplink channel from the controller to deliver their data within their troop. The MANET channel is used for data delivery.

For the second use case, the controller is deployed on an arbitrary node of the MANET, in case connectivity to the upper tier vehicle is not present. This is likely to happen as the controller is a single point of failure. The controller takes over the same functionality, meaning routing and path deployment on the nodes. Soldiers could transfer surveillance videos of suspicious persons or better provide first aid following a doctor's instructions while streaming the injuries.

1.4.2 Flying Ad-Hoc Networks

Flying Ad-Hoc Networks (FANETs) are a variation of MANETs. Unlike MANETs, FANETs assume UAVs as nodes interacting with each others [WJ22]. Their use cases range from extending existing wireless networks and base stations [CFZ18; PAR05; GJV16] to support search and rescue teams in catastrophic situations [Mic+18; AM18]. FANETs also act self-organized in terms of routing and data delivery, constructing a semi or fully decentralized network structure, equal to MANETs. FANETs architectures are often predefined according to the mission. The major network architectures comprise several UAVs, a ground base station, or a satellite. Either the satellite connects each member with the infrastructure or a single member acts as a gateway between the UAVs and the station [KQK19]. Network structures having no central instance in place are also considered, referring to UAV swarm structures. Those behave completely autonomously regarding routing [BST13; Shu+20].

A further key difference compared to traditional MANETs are the preconfigured paths, that FANET members follow. These paths are related to mission objectives. The waypoints of nodes are applied accordingly. Drones must often deviate from routes due to environmental changes [BST13],

which is why commonly used MANET mobility models, such as Random Waypoint (RWP) [AWS06], are not preferred [Shu+20]. Tailored mobility models for FANETs are researched to cope with these special environmental conditions [Shu+20]. Transmission rates of UAVs are not a major concern compared to military networks. Most drones provide Institute of Electrical and Electronics Engineers (IEEE) 802.11 standards applicable for infrastructure and ad-hoc mode. Thus, according to evaluations, UAVs reach data rates up to $7.5\ Mbit$ in ad-hoc mode [Gui+16].

Both network architectures, UAVs connected to a base station or satellite and swarm structures, are suitable for the controller-equipped routing concept, referring to mentioned architectures and communication requirements. Utilization-efficient route distribution promises a valuable impact on for instance public safety applications. Referring to flash floods or forest fires, recorded live streams from multiple drones capturing different angles of devastated areas could help civil protection organizations to respond quickly.

To sum up, our controller-equipped routing framework is generally applicable in the following network architectures: Traditional MANETs and also in FANETs, containing an additional channel used for infrastructure connectivity. It is required that the wide range low data rate uplink does not provide sufficient capacity to deliver payload to be forced to offload data delivery onto the FANET. Therefore, we presented two application scenarios, which could be well-suited for our concept. Their network architecture, data rates, and characteristics fit the mentioned requirements, respectively.

1.5 PRELIMINARIES AND ASSUMPTIONS

We assume some conditions and preliminaries throughout this thesis. This concerns controller specifications, network constellations, wireless definitions, mathematical notations, and the used simulation environment.

Controller Specifications In case the controller is deployed externally, the following assumptions and limitations are present: The uplink provides an insufficient data rate to transmit application data. Referring to the military application scenario in Section 1.4.1, NBWFs are applied for uplink connectivity, providing approximately $0.5\ Mbps$. The controller can be embedded in infrastructures, like data centers, where computation power is not an issue. Hence, the controller's hardware equipment provides high performance computing power, guaranteeing fast response times regarding multi-path computations for MANETs.

In case the controller is deployed on a MANET member, hardware resources are limited. Smartphones, IP-based radios, and further mobile devices are assumed as devices. Computation times concerning multi-path routing in MANETs are an issue and must be taken into account, as pathfinding takes place at runtime. Nodes in MANETs are moving and change the topology continuously. Topology situation before and after path computation should be the same to avoid link breaks at the beginning of transmissions. The controller has the same abilities compared to the physically partitioned controller. Also, the controller node has the same capabilities as all other MANET members.

Network Constellations Unlike wired networks, MANETs unpredictably change their topology due to mobility. Mobility models like RWP and especially Gauss-Markov (GM) Mobility Models [AWS06] are often applied to MANET to simulate pedestrians. Both models are driven by randomness, which makes it difficult to predict future positions. It is not unlikely that members lose connectivity due to mobility. Also, MANETs may split into two smaller and more MANETs. For meaningful evaluations of algorithms and protocols, MANET topologies should stick together. Guo et al. [GHX10] propose an equation to compute the probability of isolated nodes. Isolation occurrence depends on network size, number of nodes, and transmission ranges of nodes. Our initial MANET topologies generated in RQ 2 and RQ 3 proportionally align to their findings to avoid generating isolated node situations.

Also, simulations do not consider any obstacles placed on the playground that decrease transmission powers and disturb data exchange. In addition, no further environmental influences that change connection quality, like weather conditions, are assumed.

Wireless and Channel Access Definitions Throughout this thesis, the propagation of radio signals appears omnidirectional. Therefore, MANET members are equipped with a single antenna used for transmission and reception. Nodes can either receive or transmit at one given point in time. Interference appears if nodes receive two signals from different sources at the same time. When speaking about interference, we do assume co-channel interference.

CSMA/CA is mainly applied for channel access throughout this thesis, relying on acknowledged unicast data frames. Bidirectional connections must be present between two nodes to accomplish data exchange as sink nodes must receive the data frame and source nodes the acknowledgment. For this reason, data exchange between nodes is only present if connections

in both directions are active.

Technically speaking, a connection aka. link $(n, m) \in \mathcal{L}$ exists if the reception power at m of any arbitrary frame sent by n is above the sensitivity threshold. This, however, does not ensure two-side connectivity. If so, link $(m, n) \in \mathcal{L}$ must also exist, meaning the reception power of any arbitrary frame at n must also be above the reception threshold.

Throughout this thesis, we only speak of connectivity between two nodes if both sides are able to decode frames of each other. Otherwise, no link in both directions exists in $\mathcal{L}$.

Links between two nodes in almost all MANET figures are illustrated with dashed lines. These represent bidirectional connections meaning an edge from n to m $\big((n, m) \in \mathcal{L}\big)$ and also the other way round $\big((m, n) \in \mathcal{L}\big)$. Due to aesthetics and clarity, we decided to design two-way links between two nodes as a single link since a connection in only one direction is not considered anyway.

Notations and Definitions This paragraph highlights special definitions required to read and understand the mathematical definitions properly. We highlight special notations like parameter spellings not commonly used. Well-known mathematical definitions, such as set notations, to describe our data structure are not discussed in the Preliminaries and Assumptions. Parameters and variables, which are defined and stored in commonly used data structures are introduced in their corresponding chapter.

This thesis consecutively constructs a framework from Chapter 4 until 7. Inside these sections, various variables and parameters are defined representing states and conditions of the framework. Each of these chapters comprises the Section Framework Components and sometimes also the Section Further Framework Components. All parameters and variables, defined in these sections are valid and used in the current and further chapters. Any other definitions not included in the Sections Framework Components and Further Framework Components are only valid in the current chapter, where they are defined. For instance, parameter declarations, such as Section 4.1.2 of Chapter 4 or Section 6.4.2 of Chapter 6 are not part of the framework definitions as these properties are defined outside of the Sections Framework Components or Further Framework Components. Those parameters and variables are only valid for the current RQ but not beyond.

Several parameters, such as xy_n, where $n \in \mathcal{N}$ are used, comprising two or more letters and a different font. These parameters are identical to commonly used parameters having a single letter, such as z_n, where $n \in \mathcal{N}$ and have no further characteristic. We use this notation to better distin-

guish all parameters from each others'.

Sets and sequences are defined in capital letters. We define a sequence of elements, such as $E = \langle e_0, \ldots, e_k \rangle$ to store primitive parameters and objects, where $k = |E|-1$. Sequences comprising these brackets have equal rules of basic set theory [Hal60], meaning that union and intersections, as well as complements can be applied to these sequences. The sequences only differ in the ordering. Sets do not follow an order whereas our defined sequences do not change, meaning that a newly inserted element in the sequence is positioned behind the last element.

Throughout this thesis, multiple flows are routed based on the topology of the current MANET. A flow is identified by its origin and target nodes. Suppose node x is the origin and y is the target. In that context, this flow is defined as $f_{(x,y)}$. Each origin-target pair is unique. This means that we define that neither an origin nor a target node can be chosen as an additional origin or target for another flow. In our case, origin x can not be defined as origin or target of another flow, nor y. The nodes x and y are part of the set $\mathcal{N}$ and a flow $f_{(x,y)}$ is part of the set of flows $\mathcal{F}$. The definitions of flows, nodes, and additional data structures are discussed in Section 4.2.1 in detail.

A flow is also defined as f_n, where $f_n \in \mathcal{F}$ and $0 \leq n < \mathcal{F}$. Hence, f_n is another spelling for $f_{(x,y)}$ and refers to a flow in $\mathcal{F}$. More specifically, with respect to flow f_n, this flow has the same characteristics as $f_{(x,y)}$ and as a consequence also the same origin and target nodes.

Evaluation Environment All simulated evaluations in this thesis are carried out with OMNet++[1] [Var10], version 5.4 if not further specified. We extended OMNet++ with the INETMANET framework, comprising MANET routing protocols [2].

The network simulator comprises a variety of protocols, ranging across each Open Systems Interconnection (OSI) layer. The modular structure facilitates specific network customizations from private networks to data centers to global Autonomous System (AS) connections, which designs the simulator as scalable and expendable. Each entity, like routers, endpoints among others is configurable with standardized components, such as network interfaces, routing and transport protocols, packet queuing logics, and a lot more. Each component is extensible and own components can be implemented since the network simulator is written in C++[3]. Despite the INETMANET framework, OMNet++ provides further extensions, such

[1] https://omnetpp.org/
[2] https://github.com/aarizaq/inetmanet-3.x
[3] https://cplusplus.com/

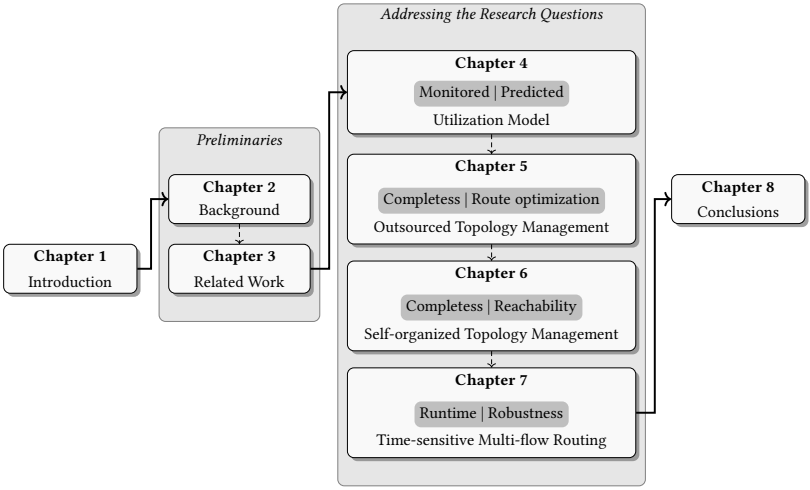


Figure 1.3: Thesis outline consisting of four RQs, starting at Chapter 4. The introduction outlines the motivation behind the topic of this thesis followed by the preliminaries Background and Related Work. Chapter 8 summarizes all RQs and discusses future work.

as Veins, INET, and SIMU5G.[4]

1.6 THESIS STRUCTURE

The Section Thesis Structure connects all parts logically starting with the Chapter Introduction. This chapter motivates the topic and points out potential research opportunities and resulting challenges discussed in Section 1.1 (Problem Statement). The introduced RQs in Section 1.2 cover all parts of the desired objectives to address the drawbacks introduced in Section 1.1. We briefly discuss the contributions in Section 1.3, tackling all RQs. In order to bridge the gap between our research and reality, we mentioned in Section 1.4 several use cases and application scenarios in which a centralized routing technique for MANETs is applicable and promising.

Chapter 2 introduces protocols and network principles relevant for communication networks and this thesis. Next, in Chapter 3, we discuss and present previous work concerning each RQ. Furthermore, we summarize

[4]https://omnetpp.org/download/models-and-tools

and compare their contributions with the objectives in this thesis and highlight similarities and differences.

Afterward, we deep dive into all RQs which have briefly been introduced in Section 1.2. With respect to the thesis structure in Figure 1.3, each Chapter comprises the chapter number, the heading, and two key characteristics (grey-filled rectangles), which are based on the main requirements of the corresponding RQ. The outcome of each RQ is embedded in a network architecture, continuously extended with further components throughout this thesis constructing a routing framework tailored for multi-flow distribution of highly data rate-demanding situations in MANETs.

In Chapter 4, we address the impact of wireless transmission in MANETs and propose a distributed and centralized utilization recording concept coupled with a computation technique to predict upcoming transmission demands generated by additional flows.

Chapter 5 addresses the importance of up-to-date MANET topology knowledge for routing purposes and proposes a topology update algorithm for MANETs containing an outsourced controller.

Chapter 6 aims to provide similar functionality deploying the controller on an arbitrary MANET participant. We highlight drawbacks of current topology reporting concepts and introduce a network topology reporting algorithm tailored for traditional MANETs. Furthermore, we present a self-organized controller management to provide connectivity for each participant to the controller and to manage the controller role in MANETs autonomously.

Lastly, we discuss the Multi-flow Routing Framework in Chapter 7, combining all findings of RQ 1 to RQ 3 in order to compare different pathfinding techniques. To begin, this chapter introduces an evaluation workflow constructing realistic MANETs. Next, we deep dive into five multi-flow path computation techniques and evaluate each approach to highlight advantages and disadvantages of them. Lastly, we propose a classification based on several application scenarios to highlight which approach is best suited for different situations.

To the end, we connect the dots of all contributions, revisit all RQs, and provide an outlook to future research related to this topic in Chapter 8.

CHAPTER 2

Background

This chapter discusses concepts related to networking which are required to understand this thesis. At first, we start with MANETs and commonly known routing protocols. Next, we introduce a controller-assisted network architecture, followed by channel access techniques. Lastly, this chapter introduces techniques to compute one or multiple routes from source to destination.

2.1 MOBILE AD-HOC NETWORKS

The history of MANETs starts several decades ago around the $1970s$, focusing on fast build-up possibilities also known as ad-hoc systems gathered under the term Packet Radio System (PRS). These networks comprise terminals, repeaters, and stations, constructing communication channels, as well as switching capabilities to provide data delivery between members. End users use terminals with limited power and short range to be portable which cover routing and data delivery. Repeaters function as relays for wide ranges, connecting several terminals executing routing instructions given by terminals. Gateways connect the mobile stations with infrastructure [FG75; GM78]. The research was mainly driven and supported by Defense Advanced Research Projects Agency (DARPA) to enable packet-switched communications between terminals [Kah+78]. The flexible network architecture was used for military scenarios. With the invention of notebook computers in 1990 and onwards, ad-hoc networks came to commercial applications and also gained attention in research forming the from now on commonly known term MANETs.

The field of MANETs has thereupon widely been researched over decades as represented by the number of published articles related to MANETs in

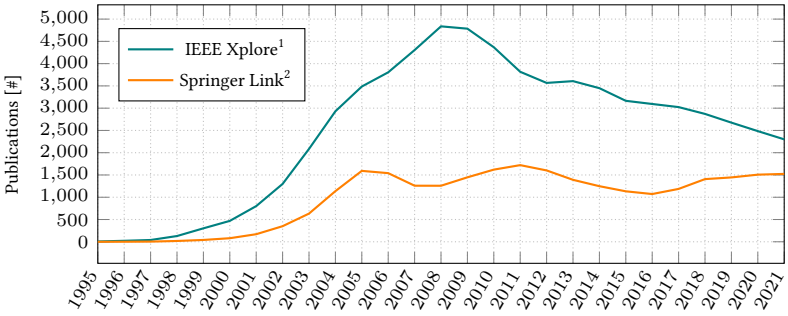


Figure 2.1: Number of scientific articles on IEEE Xplore and Springer Digital Library focusing on MANETs and their routing strategies.

IEEE Xplore[1] and Springer Link[2], see Figure 2.1 providing various routing approaches [Bou+11; HSS13; HXG02; LK05; MWH01; Raj20; RT99; KK00] and protocols [DPB03; Cla+14; CS21; HMJ07], to mention a few.

MANETs define several hosts, connected via a wireless link, forming an autonomous communication network [DZ10, Chapter 13] [LMO16, Section 1.3]. These hosts, also named nodes, serve in their function as clients and routers, meaning routes are defined and data flows are forwarded by those, respectively. These characteristics are the key difference compared to common cellular networks. There, base stations serve as access points supporting the needs of wireless communications by providing wide coverage range and wired connected infrastructure in the backend for routing and data delivery, for instance between two wireless hosts. On the subject of this use case, this service is self-managed by MANET nodes without additional network components. Further, network nodes are free to move in any direction. This leads to disconnections but also to new network links depending on the coverage ranges of hosts. Nodes are equipped with transmitter and receiver having an appropriate antenna whose reception threshold and transmission power determines the connectivity between participants. Speaking of network characteristics, the ever-changing topology also distinguishes MANETs from infrastructure networks whose key entities, like routers and storage units are static. MANETs operate locally which means that the areas are spatially limited. The basic form of MA-

[1]https://ieeexplore.ieee.org/Xplore/home.jsp
[2]https://link.springer.com/

NETs is isolated, meaning no gateway entity exists connecting the MANET to additional infrastructure. Origins and destinations of data traffic in this constellation are members of the MANET.

A variety of network architectures exist, which descended from MANETs, such as Wireless Sensor Networks (WSNs) (WSNs) and Vehicular Ad-Hoc Networks (VANETs). The first comprises sensors, wirelessly connected, using collective strength to monitor environmental conditions. The second comprises several network entities, like Road Side Units (RSUs), smart traffic lights, and vehicles forming a VANET to make road traffic more intelligent and safer [DKS16]. Further subgroups of ad-hoc networks are introduced in the following subsections.

2.1.1 Flying Ad-Hoc Networks

FANETs specify their participants more specifically. Members are meant to be UAVs forming ad-hoc networks in the air, wirelessly connected with each other. Compared to MANETs, participants behave the same in terms of routing and data forwarding [WJ22]. From the communication architecture perspective, FANET members provide data exchange among UAVs. Furthermore at least one UAV serves as gateway between the FANET and a ground station or satellite [KQK19] in case the network is meant to be connected to the infrastructure. FANETs are also supposed to behave completely autonomously lacking any channel beyond in-group connectivity [BST13]. Unlike MANETs, FANETs are built up for a specific use case following a configured objective. UAVs, for instance, can be used as bridge networks to cover areas that lack cellular coverage, for instance, in catastrophic scenarios, like flash floods [HYM16]. Tailored waypoints are pre-configured on UAVs to route the network to the desired area and also to keep a certain distance between nodes [Shu+20].

2.1.2 Mobile Tactical Networks

MTNs arose on the basis of MANETs due to their autonomous, self-organized, and decentralized characteristics. This, as a positive side effect, renders these networks opaque, as data routes are not forced to traverse via a static central routing instance. Routes are defined dynamically based on topology constellation and transmission overhead. Military and public organizations adopt these characteristics in their specific network architectures. These MTNs basically consist of several MANETs interconnected with each

other via different channels. In particular, at least one MTN member functions as gateway to a different MANET, each organized hierarchically to maintain the layered architecture of the military [NY18]. The architecture is heterogeneous compared to traditional MANETs. Hosts are supposed to be soldiers, military vehicles, and also UAVs, wirelessly connected providing different connection characteristics, such as communication range and throughput [LAM12; MKL15]. Participants of the network are mobile although waypoints can be pre-defined, based on mission instructions.

2.2 ROUTING ESSENTIALS

The term routing generally covers multiple functionalities. The most important are pathfinding techniques and network topology gathering, in case the network is not configured manually. The first functionality computes the routes from the origins to their requested destinations [TW11, Section 5.2.2]. The latter functionality collects and maintains network topology connections via so-called link state packets, which are the basis of shortest path algorithms to compute routes [TW11, Section 5.2.5]. The most naive and trivial approach is to periodically signal hello messages to all directly connected neighbors. Participants maintain adjacency lists which are picky-bagged across the MANET. Those periodically broadcasted link state packets are forwarded by all recipients to ensure that the information reaches all participants. For simplicity, mandatory fields like sequence numbers, unique address identifiers, as well as link costs are included in link state packets but not further discussed, as their usage is irrelevant for this thesis. Nodes form a graph based on the collected information, representing the network topology stored as adjacency lists. This information is updated each time a new link state packet arrives, containing new topology information.

Shortest path algorithms, such as Dijkstra [Dij59] or Bellman Ford [Bel58] are applied to search for paths to all participants in the current network. Routers, therefore, make use of the network graph to compute routes to desired destinations. The results are stored in the routers' routing tables containing the destinations, the next hops, and the costs to reach them.

2.2.1 Mobile Ad-Hoc Network Routing Strategies

MANETs distinguish between different routing categories. Categories are Flat routing, Hierarchical routing, and Geographic position-assisted rout-

ing. Flat routing is further divided into the routing techniques proactive and reactive. [HXG02]

Reactive protocols, also called on-demand routing protocols, start the routing process if one requests a path to any destination. Origins then start the protocol-specific pathfinding process by applying tailored flooding techniques to reach the desired destination. Reactive protocols set the focus rather on on-demand route discovery than on broad topology knowledge. Origins broadcast route requests instantly before data transmission is intended. Recipients relay the messages if no route to the desired destination is known [LMO16, Section 2.2]. Consequently, routing information beyond active routes is not provided. However, if no route is requested, no routing overhead is produced.

Participants of proactive routing continuously collect topology information using signaling and link state packets. Origins use the gathered information for routing. Referring to routing essentials in Section 2.2, the mentioned behavior is applied if routing is organized proactively, meaning topology information is exchanged periodically between all participants. The limited battery lifetimes and data rates force members to improve the topology-gathering process with the least possible message overhead while trying to provide similar up-to-date information in terms of topology representation. For instance, specific nodes are elected to broadcast topology information of direct neighbors while surrounding nodes stay idle. Further approaches delay received link state packets depending on how often these packets have been retransmitted so far. Equal to common routing techniques, MANET participants use the gathered topology information to fill their routing tables accordingly.

Routing protocols also combine reactive and proactive techniques to so-called hybrid approaches [LMO16, Section 2.4]. Using this strategy, network segments form interconnected nodes to clusters that route proactively. Reactive routing is applied between clusters.

Hierarchical routing organizes nodes in groups assigning them different functionalities inside and outside of these groups. This reduces the routing table size, as members of groups only store information about group participants. Further, members also store the gateway node to reach destinations beyond the group. MANET members in near vicinity are therefore often grouped into clusters containing cluster heads which provide gateway functionalities.

Routing in MANETs also integrates the Global Positioning System (GPS) to deliver data to destinations. Therefore, current devices continuously

track their own positions and provide these GPS coordinates for localization within meter range. Locations of reachable MANET participants are gathered via location services. All members are aware of the geographical positions of all destinations. Traffic routes are dynamically defined, as routing protocols define paths with forward strategies. [MD05]

For instance, relays pursue finding neighbors of which the geographic position is closer to the destination's position compared to their own. This process is repeated till the traffic reaches the destination. Geographic routing does not need any explicit route discovery except for gathering knowledge of node locations.

The following subsection introduces routing protocols used in this thesis.

2.2.2 Ad-Hoc on-demand Distance Vector

AODV [DPB03] is a well known and frequently used reactive routing protocol in the field of MANETs. Provided functionalities are route construction, maintenance, and route error handling. Pathfinding is primarily organized with Route Requests (RREQs) and Route Replys (RREPs).

The first message type is broadcasted by an originator node if no route toward the target node is known initiating the pathfinding process. In its basic form, RREQs are relayed from each receiver till the message reaches the destination. A RREP is generated and sent, if the receiver is the desired destination or a node that received the RREQ has a valid route towards the destination in its routing table. In this case, members send the RREP along the reverse route previously generated by the RREQ to the originator.

In this basic form, a so-called broadcast storm would be generated by each originator, as participants rebroadcast previously received RREQs. Sequence numbers combined with originator ids and Time to Live (TTL) constraints reduce the message overhead. If a member receives a RREQ with equal identification pair, meaning the same RREQ id and originator id within the path discovery timespan, the message is recognized as duplicate and will be discarded. RREQ ids are incremented by originators each time a RREQ is created. This also applies to multiple requests having equal origin and destination ids. Path discovery time defines a period an originator waits until pathfinding process for the same destination is initiated again. This for instance happens if a route for the previously requested destination has not been found. In that case, AODV sets TTL to 2 to also reduce rebroadcast overhead. This field is incremented by 2 if no RREP arrives within path

discovery time.

Actual route construction is transposed with RREPs initiated by target nodes or members having valid routes to the requested destinations. The destinations of RREPs specify the MANET member that previously sent the corresponding RREQ as the next hop for the current RREP. Intermediates follow the same procedure and send a RREP along the reverse route till the originator of the previously broadcasted RREQ is reached.

AODV provides route maintenance using hello messages. Hello messages are broadcasted in configured intervals of MANET members which are active path participants in order to offer connectivity information. Route Error (RERR) processes are initiated, if route participants do not receive hello messages after the defined hello period. Hello messages can be turned on or off.

The RERR is a message type used to inform MANET members about lost connections. According to protocol, RERRs are sent if data delivery interrupts. More precisely, if a route participant detects a link break to the next hop or the destination while trying to deliver payload. RERRs are also sent if a member receives a data packet addressed to a node for which it does not have an active route in the routing table. Those RERRs are relayed by MANET members.

Participants insert or update routes of next hops, originator, and target nodes during pathfinding process. MANET members which use AODV maintain a sequence number representing the actuality of routing information. These sequence numbers are shared with all AODV message types. Intermediate nodes that process and forward both messages also insert their sequence numbers. Nodes insert or update routing information of previous, next hops, and of originator and target nodes if the associated sequence number in the message is equal or higher compared to stored ones. That ensures loopless routes and also guarantees that information in routing tables are only updated with even more up-to-date information.

2.2.3 Optimized Link State Routing

Optimized Link State Routing (OLSR) [Cla+14] with the current version 2 is a proactive link state routing protocol for MANETs. The functionality of link state routing is to maintain a list of directly connected neighbors and to determine the cost to them [TW11, Section 5.2.5]. The cost is a numerical representation of a connection characteristic, such as transmission capacity and latency among others. This neighborhood information is distributed

across the network, as described in Sections 2.2 and 2.2.1.

OLSR comprises topology gathering and route construction applying appropriate shortest path algorithms. We introduce how OLSR obtains topology information on each participant which is the basis for the actual routing. In general, the wireless channel of MANETs only provides restricted data rates. OLSR defines Multi-Point Relays (MPRs) to limit the amount of routing load. All nodes periodically broadcast hello messages. Thereby, participants elect neighboring nodes as MPRs. The selection of MPRs depends on their connections to their direct neighbors.

Each member pursues the goal to select the minimum possible number of MPRs to each of its current 2-hop neighbors. The fewer the number of MPRs per node, the lesser the total routing load. The set of MPR of each node is the subset of the set of 1-hop neighbors.

Each node periodically sends hello messages and also processes hello messages of their neighbors. These messages store neighbor information, such as interface address which identifies each direct neighbor and the link code which represents additional information about this neighbor, such as if this node is elected as MPR by the sender. Hello messages comprise additional fields, not mentioned in this thesis as the information has no relevance so far.

Receivers of hello messages maintain their neighbor set, storing all current 1-hop nodes and also their 2-hop neighbors. More precisely, nodes store the sender of these messages along with the costs to reach them in their local link information base. MPRs also process these messages and maintain a MPR selector set, containing all neighbors which selected this node as MPR. These nodes leverage the neighborhood information in generating the topology information base containing all nodes having selected this node as MPR.

Furthermore, MPRs periodically broadcast topology control messages containing the current topology information base to all direct neighbors. All MANET participants populate their topology table based on this information which is the data basis for upcoming routing. Qayyum et al. deep dive into MPR selection and propose an algorithm heading for the least broadcast overhead.

OLSR proactively provides a complete topology view of the MANET for each participant. The actuality of the topology representation depends on the interval of hello and topology control messages. Characteristics, such as reliability and the objective to minimize routing load while keeping full topology knowledge makes this routing protocol interesting in the military

sector [Bar+16; Mar+17].

2.3 SOFTWARE-DEFINED NETWORKING

This section briefly introduces traditional network architectures to better highlight software-based topology construction, adaption, and flow management that all come with SDN.

Traditional networks run distributed protocols like Open Shortest Path First (OSPF) [Moy98] for routing. Their advantages over static routing are scalability, reliability, and less management through network operators. Protocols, such as OSPF detect topology changes dynamically and adapt routes and their data transmission accordingly. However, routing is not designed to determine a dedicated path for each transmission. Routers rather select outgoing interfaces for packets individually based on their routing table entries [TW11, Section 5.1.3].

SDN-managed networks extract the logic from routers and switches and outsource this ability to a central instance. Routing devices report current connections and further information, such as diagnostics to the central instance. This topology information is used to define paths for all requested flows. In particular the flow tables of devices are filled accordingly.

Over the past decades, networks increased continuously in size, connecting host and routing devices continuously. Routing and topology management requires more hardware resources and connectivity between different network operating system and protocols. Network operators increasingly lose visibility due to the large amount of heterogeneous physical devices, routing packets through their networks. Managing these networks to provide controlled routing, reliability, and security is complex and unrealistic considering the continuous network growth.

SDN is so far a well-known paradigm in the field of routing in large-scale network architectures by centralizing the routing and network management to a so-called Control Plane (CP), and exploiting it as a management entity for the entire network [Hal+15]. The networking appliances, such as switches, routers, and middlboxes, summarized to the term Data Plane (DP), instead forward data packets based on defined paths [Kre+15]. Switches thereby manage multiple data transmissions through different protocols. There exist several SDN-enabled switch implementations, such as Open vSwitch [3], OpenFlow Software Switch [4], among others [Kre+15]. The

[3] https://www.openvswitch.org/
[4] https://cpqd.github.io/ofsoftswitch13/

CP makes decisions if and where traffic is sent and also specifies the amount and sequence. Providing a centralized, in software implemented CP, allows dynamic access and administration without touching any switch or router individually.

Clear distinctions are generated through so-called southbound and northbound interfaces enabling scalability in terms of configuration and extensibility between switches and controller and also between controller and applications.

Unified and vendor-agnostic control interfaces, such as OpenFlow exist to configure the heterogeneous DP topology via the southbound interface [McK+08]. This standard allows to dynamically adapt routes and parameters according to real-time demands. The northbound instead stands for automation and agility in providing a variety of design and behavior configuration possibilities for how the controller manages the underlining network.

Network operators build applications storing network design strategies that fit their purpose. These applications range from routing, firewall configurations, load balancers, monitoring, intrusion detection system, Daniel of Service (DoS) detection, QoS monitoring and so forth [Jar+14; KWT16; CED19; Yoo+15; Shi+13]. As SDN stands for minimizing and centralizing the configuration of networks, scalability is one of its strengths. This is why SDN is mainly applied in data centers, core networks, enterprise networks, and nowadays in Wide Area Networks (WANs) [Jar+14; MK17].

The idea of controller centralization and fully-controlled routing comprise overlaps compared to our in this thesis proposed centralized path computation approach for MANETs with SDN. However, it is worth mentioning that SDN comprises a variety of functions and techniques, mentioned in this section, to manage the underlining data plane. Also, the field of applications is focused on cable-connected large-scale networks whereas MANETs comprise mobile nodes equipped with one or more wireless interfaces. These distinctions predominate and rather distance our centralized routing approach for MANETs from SDN.

2.4 WIRELESS CHANNEL ACCESS

This section introduces wireless channel access methods applied and extended in this thesis.

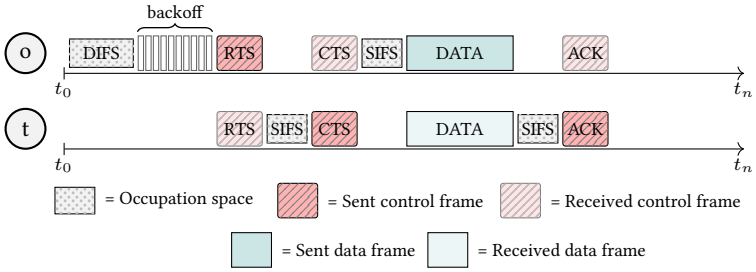


Figure 2.2: IEEE 802.11 conform frame sequence between nodes o and t without interference.

2.4.1 Carrier Sense Multiple Access / Collision Avoidance

CSMA/CA [IEEE802.11] is a widely used wireless channel access control protocol adopted by the IEEE 802.11 standard. The sublayer of CSMA/CA applies the Distributed Coordination Function (DCF) to organize and control successive channel access. IEEE 802.11 is mainly applied in small infrastructure networks having one or multiple access points providing connectivity for their clients. The standard also facilitates ad-hoc communication mode which allows device-to-device connections without the need for infrastructure support [Sch04, Chapter 12]. CSMA/CA is a contention-based access method focusing on collision avoidance. Nodes sense the common channel for ongoing transmissions before starting their own transmission. This technique also makes use of randomness to provide fairness.

Figure 2.2 elaborates on an exemplary frame sequence with CSMA/CA carrying out physical carrier-sense and virtual carrier-sense mechanisms. Node o represents the originator of the desired frame transmission and t the target. t_0 represents the start time of the frame sequence and t_n, where $n > 0$, the consecutive time steps. Propagation times are not visualized in this figure.

At the beginning node o first physically senses the channel according to the Distributed Interframe Space (DIFS) timespan. Short Interframe Spaces (SIFSs) and DIFSs are time periods in which the radio is occupied due to specified waiting times. The time periods of both are specified in their IEEE-802.11 standards. For instance, IEEE 802.11n operating in the $2.4GHz$ define a SIFS period of $10\mu sec$. DIFS timespan comprises SIFS plus two times the configured slot time which according to IEEE 802.11n is either

$9\mu sec$ or $20\mu sec$.

Suppose o detects a transmission during DIFS. Node o defers its desired transmission and starts attempting to access the channel later again. In case no transmission is sensed, node o starts with the backoff period which comprises a multiple of the configured slot time. The backoff represents a waiting time containing the Contention Window (CW) times the slot time where the CW per default contains integers ranging from 0 to 31. Before transmission starts, each node generally waits according to the computed backoff after DIFS period is finished. It likely happens that node o detects a transmission during backoff period. Node o would wait until the ongoing data transmission is complete and thereafter starts DIFS period again. To provide fairness, o would start decreasing the backoff at the point it previously stopped when data transmission has been sensed. This increases the probability the node o accesses the channel in near future compared to another node that desires to start a transmission at the same time.

After backoff, node o starts the virtual carrier-sense mechanism which is a control frame handshake between sender and receiver. The objective is to reach out to neighbors of the current sender and the potential receiver of the upcoming frame to stay idle during desired frame sequence. Both frames, named Request-to-Send (RTS) and Clear-to-Send (CTS) include the duration of the desired frame sequence which defines the period of time all neighbors in communication range will stay idle. Receivers complete the handshake if they are not occupied by an ongoing transmission which grants the data transfer.

In doing so, node o sends a RTS frame to t that waits the SIFS period before the receiver of this frame replies with a CTS frame. SIFSs represent the processing time of the current received frame up to the MAC layer. This time duration plus the transmission time of the subsequent frame are less than the DIFS interval to avoid channel access by multiple nodes at the same time.

Node o then waits a SIFS time period before data delivery takes place. After another SIFS timespan, the corresponding Acknowledgment (ACK) frame is sent by t which completes the frame sequence. Thereafter, all nodes that have desired to start transmission during the finished frame sequence again start the DIFS period and thereafter compute their backoff periods.

2.4.2 Time-division Multiple Access

The MAC technique TDMA is a managed access method where the entire available frequency is divided into multiple time ranges. The TDMA method often is centrally managed, meaning a central instance allocates individual time slots for all participants of a network.

It was previously applied by the European, North American, and Japanese mobile carriers in radio cell infrastructures [Pra13]. TDMA nowadays is applied in network architectures, coping with limited computing powers and data rate restrictions, such as Industrial networks, Internet of Things (IoT), and WSNs [SMM20; Khe+22; Sam20].

The general purpose of TDMA is to assign a dedicated time unit, so-called slot to each potential transmitter in the network. Multiple participants can share the same transmission medium while using only a part of the channel capacity. Data is transmitted in bursts, meaning that network nodes transmit data in sequences. TDMA includes guard times between slots to avoid overlapping signals, as synchronized clocks still lack marginal time deviations.

Partition of slots in terms of size depends on the use case and objective of networks. GSM, for instance, uses multi-frames comprising 26 frames. 24 of 26 frames are reserved for data. The last 2 frames are used for control information. Frames are further divided into 8 time slots.

The North American Digital Cellular provided only frames of 40 $msec$, each divided into 6 time slots. [Goo91]. TDMA is also combined with other well-known multiplexing methods, such as Frequency-Division Multiple Access (FDMA) [FAG95]. There, each individual frequency is further divided into time ranges.

Applications using TDMA techniques have also been researched in the field of MANETs. Due to the distributed nature and spatial reuse of time slots, the challenge focuses on how to manage and assign slots to participants. Assignment methods are either centralized or distributed, depending on the network structure and node capabilities of the MANET [XZ15]. The first method instantiates a master scheduler responsible for allocation and assignment of slots to all members. The second method manages slot allocation and distribution self-organized. Nodes, for instance, exploit messages of routing protocols and further design additional messages to detect free time periods and to assign free slots to each other. Such techniques are more robust as no central point of failure exists. However, the distributed slot management technique increases the control message overhead and also increases the access delay as slot information must be delivered across

multiple hops.

2.5 SHORTEST PATH ALGORITHMS

The shortest path and the shortest paths tree routing problems are well-known optimization problems in finding a path from one origin to a destination or to all targets. For that, the underlining real-world scenario is abstracted to a graph containing vertices and edges. The latter connects the vertices with each other. Referring to a computer network, vertices are routing entities and edges represent the connections between them. In the basic composition, edges have a numerical value, called the weight which is often referred to as the cost of the edge. In computer networks, the weight represents, for instance, the provided data rate, the delay, among other performance parameters. This abstract structure is the basis, shortest path algorithms are applied on. There, the objective is to find the best path concerning the defined weights from a given origin vertex to all or a particular target. Their applications range from transportation and navigation using highways, rail networks, and airline routes [FSR06; ZC09; Pat+21]. Also, hydraulic systems computing water, gas, oil, or water flow are fields of applications [BIO17; CC12]. Further well-known use cases are communication systems, such as telephone connection scheduling [AMO93, Section 4.2] and like computer networks using fiber optic links or cables, to transmit various types of data, such as voice video and files [Jia+14]. We introduce Dijkstra [Dij59] and Yen's K-Shortest Paths algorithm [Yen71] which have been applied throughout this thesis.

2.5.1 Dijkstra Algorithm

Dijkstra [Dij59] [Cor+09, Section 24.3] is one of the most often used shortest path algorithms. It was designed and investigated by Edsger W. Dijkstra in 1956. Dijkstra operates only on non-negative edge weights which designs this algorithm loopless. Each vertex is assigned a distance, representing the cost to reach this vertex at this point. This cost is an aggregated weight between this vertex and the origin. To begin, this distance is set to ∞ except the origin's distance which is set to 0. Also, the algorithm maintains a list of visited vertices containing those where the shortest path in this graph has already been found.

When actively searching for the best path for each participant in the graph, Dijkstra repeats the following routine. The algorithm picks the node

with minimum distance and inserts this node in the list of visited vertices. At this point the shortest path to this vertex has been found and will not further be considered during runtime. To begin, the vertex having distance 0 (the origin) is inserted in the list of visited vertices. Next, all distances to all neighbor vertices are updated if the current distance is higher. The potentially new distance to a vertex is the distance to the vertex previously inserted in the list of visited vertices plus the edge weight between these two nodes. If the last visited node is the origin, the new distance is 0 plus the edge weight between the origin and its chosen neighbor as the result is less than ∞.

Dijkstra repeats this process till each node of the graph is stored in the list of visited vertices. This includes picking the vertex with minimum distance, inserting this vertex in the list of visited vertices, and updating the distances to each neighbor vertex if new distances are less. Furthermore, the algorithm can be aborted if the target vertex has been inserted in this list. In addition, Dijkstra maintains the predecessor of each vertex which is updated when the distance changes. The predecessor information is used to later construct the paths.

Several extensions are proposed in the literature focusing on quality assurance in bounding the link weight to a constant threshold [Bem+19]. For runtime improvements, preprocessing techniques are discussed in literature where pruning the search space based on weight baselines reduces the search spaces and in addition reduces the runtime [Hil+06; Gut04]. Further extensions, having found more attraction are A* [HNR68] and Yen's K-shortest Path algorithm [Yen71]. The former computes paths from one origin to explicitly one target including a heuristic to the edge weight reinforcing the direction at runtime. The latter searches for alternative shortest paths from a source to a destination using Dijkstra. Previous research also extended Yen's shortest path algorithm to, for instance, define the degree of path similarity.

2.5.2 K-Shortest Paths

Finding several paths from the same origin to the same target in a graph is the objective when applying k-shortest paths algorithms. This differs from the single shortest path algorithms, such as Dijkstra, where $k = 1$. Previous research proposed two prominent variants, named k-shortest non-loopless shortest paths [HP59; Epp98] and k-shortest loopless path [Yen71; Law72]. Non-loopless paths do not constrain whether the same node is

part of a computed path multiple times or not. Speaking of loopless paths, each selected route participant appears only once in the route. The number of loopless paths in a graph is often dominated by the number of paths containing loops. Because of that, finding k-shortest paths not containing loops is harder to solve [NAP05].

Additional k-shortest paths computation algorithms have been proposed by Katoh et al. [KIM82] and by Hoffman et al. [HP59]. Their execution times and shortest path function calls are compared by Brander et al. [BS96].

In this thesis, we extend Yen's k-shortest path [Yen71] with the objective to find paths having the least similarities. Yen's k-shortest path uses a single shortest path algorithm, like Dijkstra as a subroutine. Further paths built on top of the first, successively exclude an edge of the shortest path. Each generated path is a deviation of the shortest path 1 through $k - 1$.

Regarding potential use-cases, k-shortest path algorithms find applications in social networks, investigating connections between users [Leb+17].

Also, a well-known and large research field for k-shortest path algorithms is traffic route calculation [KG10] and general routing in computer networks. The first application area generates alternative routes having different metrics for shortest path computation configured. Traffic conditions, distances to targets, or approximated gas savings are criteria for alternative routes. If the first route suggestion proposes the least cost with respect to traveling time, all alternative paths will be longer if one compares both paths with respect to their traveling times. Previous research also applied k-shortest path algorithms in MANETs heading for multi-path routing of data flows [Yin+14].

2.6 GENETIC PROGRAMMING

Genetic Programming (GP) is a meta-heuristic and a type of machine learning algorithm that explores the algorithmic search space. The evolutionary method is inspired by natural selection processes. [Hol92] There, only the individuals that are the fittest or closest to being the fittest are able to survive. These surviving individuals then reproduce to create new offspring with the expectation, that the offspring will also possess the necessary quality to survive.

More generally, the objective of GP is to evolve a population of individuals through genetic-mimicked operations called mutation, crossover, and fitness to converge towards a solution [Koz92, Chapter 5]. Applied to a computer application, GP converts populations of candidates, represented

as a string of codes (e.g. binary) towards an algorithmic solution of a previously defined problem statement.

The essential element of GP is the fitness function. This function determines the quality of individuals and how to proceed with each of them. Worse rated individuals are more likely not considered in further generations while higher rated are more likely selected for crossover techniques.

However, due to its stochastic and random nature, it is not guaranteed, that GP will generate a solution to a problem [PLM08]. Referring to the literature, GP has been successful in various application domains and has been applied to generate programs that outperform those which have been written by human programmers [Ban+98]

To begin, GP creates an initial population containing potential solutions to the defined problem. Next, GP constructs generation-wise new individuals with crossover and mutation techniques out of the stored population aiming for quality improvements with respect to the defined objective. During each generation, GP evaluates the fitness of each individual to determine if an acceptable solution to the problem has been found. Next, a selection process determines specific individuals for crossover which construct a new chromosome out of two. After crossover operation, mutation takes place to maintain diversity. Both, offspring and mutated individuals are inserted into the population and considered in further generations. A termination condition verifies if an individual exists, meeting the criteria to finish the algorithm. If false, GP continues starting in evaluating the fitness of each individual. The following paragraphs introduce the general process of GP.

Initial Population Before iterating through multiple generations, GP generates an initial population, also called generation zero. These populations are created with specific methods fitting for individual purposes, such as using full, grow, and tree methods. The objective is to avoid duplicates during the generation of the individuals to expand the range of potential solutions and generated ones.

Fitness Function To compare and evaluate individuals in GP, a metric is needed to determine their level of approximation to an ideal solution. Fitness functions are used to create a rating for each individual, which enables the algorithm to compare each potential solution with the others'. In other words, fitness functions are used to determine how close or far an individual is from a perfect solution. It is important to accurately define the function to avoid confusing individuals that represent weaker areas of

the program space for stronger ones and vice versa.

Selection Methods Selection methods are applied to select the so-called parents to create new offspring. Well-known methods are tournament selection and fitness proportionate selection [Lan+08].

Crossover The crossover operation generates two new offspring by combining genetic material from two parents [Ban+98, Section 5.4.1], [Lan+08]. To accomplish this, GP selects two parents from the current population using a specified selection method. Several crossover techniques exist, such as single point, N point, and uniform crossover, among others [KY17]. One of the most significant applied techniques is the One Point Crossover [LP02, Section 4.3]. This technique works by selecting a random point in each of the parents and exchanging the genetic material beyond that point to produce two new offspring. In other words, this crossover technique connects the first part of the first parent with the second part of the second parent. The other two parts of the two parents are treated the same way. We extended this technique to our needs which we introduce in Section 7.4.5.

Mutation Mutation is another fundamental operation in GP, which involves randomly changing a part of an individual in the population [Ban+98, Section 5.4.2]. Mutation operates on a single program, and it is used to introduce new genetic material into the population. The mutation operator is applied to an individual by randomly selecting one or more genes of the chromosome and changing the genetic elements at those points. The changes can be as simple as flipping a single bit in a binary string. We extended the classic mutation technique where a single gene is exchanged which we introduce in Section 7.4.5.

Termination The termination condition in GP is a stopping condition to determine when the evolutionary process should end. It is essential to set the termination condition appropriately to ensure that the evolutionary process is stopped when it has produced a satisfactory solution or, for instance, has reached the limit of available computational resources. There exist some predefined termination conditions, such as maximum generations, maximum fitness threshold, among others [Lan+08]. Similar to mutation and crossover techniques we defined and implemented a termination condition fitting for our purpose. We introduced this termination condition in Section 7.4.5.

CHAPTER 3

Related Work

This chapter lists and categorizes previous work relevant for all RQs. First, recent work concerning recording and computation techniques to accomplish accurate utilization measurements and predictions is summarized, referring to RQ 1. Second, we introduce related research, tackling a controller-equipped MANET architecture in which the controller covers all nodes via an out-of-band channel. Third, we list recent articles in which the authors propose architectures containing a routing instance, which is included inside the MANET. Therefore, we investigate how these approaches provide the routing information for the controller. Finally, this chapter comprises related articles proposing multi-flow routing strategies in multi-hop wireless networks and MANETs.

This thesis introduces two extended MANET architectures tailored for high-utilized multi-flow routing. We, therefore, categorize recent research according to constraints and requirements of the corresponding architecture to highlight similarities and differences between their work. The placement of the routing controller distinguishes both MANET architectures from each other. The controller is either placed on a participant of the MANET or outsourced and reachable via a dedicated channel. Referring to both concepts, the network topology information for routing of related research must be complete, up-to-date, and the routing engine must have full authority. Furthermore, MANET participants must switch MAC channels dynamically, in case the controller is reachable via a dedicated channel for each node. In case, the controller is placed on a MANET node, we require the characteristics role control and path control. All mentioned requirements of both network architectures are discussed in detail in Section 3.5.

3.1 DISTRIBUTED ROUTING STRATEGIES

During the last decade, many researchers have striven to improve the performance of MANETs with their routing decisions in including parameters, representing the networks' and participants' conditions. Similar to our approach in Section 4.1, previous work focuses on extending existing routing algorithms with those parameters gathered from the network's and nodes' observed conditions. Our capacity-based routing algorithm presents a transmission overhead measurement technique and a route construction concept also relying on such parameters.

Because of that, we first present previous research focusing on extended route construction techniques of reactive routing protocols. Secondly, related work regarding routing in MANET including capacity-based pathfinding and even route distribution is discussed.

3.1.1 Objective-driven Route Construction

Haitham Y. Adarbah et al. [Y+15] extend a pathfinding algorithm with decision-based forwarding constraints. Nodes relay a route request depending on their local neighborhood knowledge and the Signal-to-Noise Ratio (SNR) of the received request. In more detail, their approach derives a potential reception error rate from the SNR value. Overall, the reception quality and the number of directly connected neighbors generate a probability to forward the packet. At this point, a self-generated random number is compared with the computed probability. The request will be forwarded if the probability is higher. If not, the request will be dropped. This minimizes the number of broadcast frames and also focuses on stable connections as high-quality links are chosen for data delivery.

Sumit Kumar et al. [KM16] try to reduce the number of collisions during the route-finding process because the more members broadcast route requests the more collisions of frames are likely to arise. Reducing the number of broadcast frames makes route creation more efficient. Members forward received requests depending on the node density, the own battery level, and the available transmission capacity. These decision variables are input parameters for an extended fuzzy logic that forecasts node density, remaining energy, and available transmission rate. The parameters are used to make a decision based on a situation in the future that is likely to occur with the general objective to minimize routing overhead. Route requests are either dropped or forwarded depending on the result. Evaluations show that the algorithm performs well in crowded simulation environments.

Shobha Tyagi et al. [TSR15] use the MANET member's data rates and transmission delays as QoS-decision indicators during path-discovery process. The speed of nodes is an important decision parameter as the data rates and transmission delays of potential participants are only taken into account if their measured speed is below a determined threshold. QoS parameters examine each link of the route and pick the one with the best quality in case several paths are proposed.

Deepti Badal and Rajendra Singh Kushwah [BK16] extend the node selection of the path-discovery algorithm of Dynamic Source Routing (DSR) [HMJ07] with the remaining battery level of nodes. The remaining energy level is determined according to the utilization of a node. The probability that the node is longer available is higher if the measured utilization is low. The long-term goal is to increase the overall network lifetime by using nodes that have more energy left.

In general, several papers have been published and summarized as references [AKS17; WS10; Liu+12; ZW13; LDP06] that deal with modifications of path-discovery algorithms. They all satisfy the requirements for their specific problem. The following section introduces related work concerning routing protocol extensions to introduce and improve transmission capacity-based routing in MANETs.

3.1.2 Capacity-based Route Construction

Improving routing in MANETs by introducing capacity information has already been tackled 20 years apart by Cansever et al. [CML99]. The conceptual approach proposes a resource allocation technique for reactive routing protocols, maintaining transmission capacities proactively on the MAC layer. The authors present early insights into resource allocation during reactive path construction.

Kazantzidis et al. [KGL01] design and also evaluate a capacity-based path construction. The authors include network overhead measurements in the route reply procedure of AODV to detect bottlenecks and trigger route replies if capacity measurements during transmissions decrease below a defined threshold.

In 2005 a QoS-aligned routing approach was introduced by Yihai et al. [ZG05]. In their approach, the data rate requirement is verified during route construction. MANET nodes are excluded if their capacities are insufficient for the requested flow. Nodes not providing data rate requirements discard the route request.

There is no individual treatment implemented that focuses on the degree of the current transmission overhead, such as classifying the utilization in several stages to behave accordingly. Requests are forwarded instantly, even if transmission capacity is almost reached. Their evaluation shows promising results, as the AODV extension delivers more data compared to standard AODV. This gap increases even more with increasing transmission overhead.

In the same year, Lei Chen et al. [CH05] propose two utilization recording concepts. The first approach takes advantage of the Network Allocation Vector (NAV) value whereas the second technique gathers individual utilization recordings of neighboring nodes. These parameters are applied to enrich AODV during pathfinding.

In 2006, Kuei-Ping [Shi+06] designed TDMA slots based on the demanded data rates of flows. The time a slot lasts depends on the data rate demand of the flow. Therefore, the authors propose several data rate allocation techniques implemented on each node. Slot policies verify the QoS requirements and determine if available slots of a potential link can be occupied for data transfer. Route requests are only forwarded if required data rates can be guaranteed.

Su et al. [SSJ07] also extend AODV with capacity-based routing techniques and either forward or drop the route request depending on the transmission capacities of each node. Additional fields extend route requests, informing network members about capacity conditions. MAC layer measurements determine available data rates of potential route participants. The AODV extension reaches the highest throughput rates compared to other capacity-based routing protocols.

Liu et al. [Liu+12] also propose a QoS routing extension for AODV focusing on providing a sufficient transmission rate for the flow demand and on providing the required delay considering multi-hop routes. The authors introduce their MAC channel measurement method and further elaborate on how route discovery takes place considering the requirements. Delay and data rate restrictions are included as parameters in RREQs evaluated by each node before retransmissions are attended. In case a node already stores a route to the desired destination when receiving a RREQ for the same target for the first time, no further checks are performed to guarantee QoS requirements. The authors experience throughput increases and minor delays compared to traditional AODV but also detected a small increase in control overhead.

Kalpana et al. [KK18] also include delay estimations to their capacity-

based routing technique when applying AODV. Furthermore, they also drop route request packets if data rate demands are not met. The protocol picks the route with minimum hops and best capacity conditions for data delivery.

Perkins et al. proposed a draft that extends AODV with QoS requirements [PB01]. The authors present a conceptual approach including message fields applicable during path construction.

The majority of previous research extends AODV with QoS extensions to propose tailored routing extensions for their specific application.

3.2 CENTRALLY ROUTED WIRELESS NETWORKS

Research articles related to controller-equipped MANETs have gained attraction, especially as the architecture that distinguishes between the two layers, namely routing and data forwarding, has shown noteworthy performance improvements regarding controlled and managed routing [Kaf+22]. Complete knowledge of network devices and connections, their capabilities in terms of data rates, arising latencies, QoS performance classifications, as well as runtime information are several of a variety of applications provided by SDN. Runtime information, for instance, provides knowledge about the capacity utilization of connections and the actual services producing current overhead.

Subfields of such two-layered architectures comprise wireless service providers extending the communication architecture with so-called Device-to-Device (D2D) communication capabilities where mobile subscribers take over data delivery to relieve the base station. Data exchange between two directly connected devices using their built-in wireless adapters is thereby limited to single-hop paths. Otherwise, research refers to the term Multi-hop Cellular Device-to-Device (MhCD2D) communication if data relying through devices is desired. The purpose is to enrich cellular infrastructure networks with additional transmission capabilities on the subscriber layer.

The reverse approach is to extend MANETs with centralized routing capabilities that attempt to better provide SDN capabilities, like routing, application design, hardware resource allocation, dynamically logical network structure changes, and individual security levels definitions [Hal+15]. Here the intention is to better control and manage mentioned SDN characteristics. That is why research in this field often refers to the term Software-defined Mobile Ad-Hoc Network (SDMANET).

Our research in this thesis focuses exclusively on routing and path con-

struction in MANETs with a controller. This is the reason why to stick to the term controller-equipped MANET architecture. We introduce this architecture and all message types in Section 4.2. Both, MhCD2D networks and SDMANETs have the central instance managed control station in common that covers mobile nodes. Similar to our introduced research, both approaches require an uninterrupted dedicated channel to a controller that takes over routing.

The following sections summarize related work, of network architectures comprising outsourced network management capabilities covering multiple mobile devices via a dedicated out-of-band channel. Therefore, mobile devices possess two channels, one for in-layer communication, the other for reaching an outsourced entity, equipped with a routing instance. The first section elaborates on mobile devices connected to cellular networks, aiming for data offloading onto devices. The second section discusses previous research having an outsourced controller connected to a MANET. With respect to both architectures, we highlight recent work focusing on complete topology knowledge.

3.2.1 Multi-hop Cellular Device-to-Device Networks

The MhCD2D networks, first proposed by Lin et al. [LH00], comprises at least one cellular base station providing uplink connectivity to mobile devices, thus expanding infrastructure with the flexibility of ad-hoc networking.

MhCD2D communication is applied to, for instance, relieve base stations in data rate-demanding situations, reduce delays, balance traffic load, enlarge the system coverage area, or is used for privacy reasons [LSC08]. Lei et al. [Lei+12] elaborate on two D2D modes, typically advertised in this field. The D2D-controlled mode provides full operator control. The controller manages authentication, connection control, among others, where the combination of connection control combined with a controller-equipped MANET architecture fits the network design and the objective of this thesis the most. The loosely controlled D2D mode rather focuses on self-managed devices establishing connections with each other without the help of a central instance. In this section, we introduce previous work in this area and highlight how their research relates to this thesis.

In 2001, Ananthapadmanabha et al. [AMM01] elaborate on the network architecture of MhCD2D networks in general and highlight the advantages of multi-hop data delivery over single-hop transmissions. Ananthapadman-

abha et al. also argue with pure proactive topology update techniques in these network constellations as scalability is not provided especially when network topology is dense. The authors define a registration step for each node before participating in the network. Also, the D2D channel is divided into channels providing exclusive time slots for each registered node. Nodes broadcast beacons in their slots allocated by the controller.

Interference and the lack of scalability are not present as the topology constellation has no influence on the discovery process. Nodes switch to CSMA/CA [IEEE802.11] for data transmission between participants. Routes are requested from the base station which is common in MhCD2D networks.

The research of Ananthapadmanabha et al. overlaps with the research of this thesis as the topology update technique, proposed in Section 5.3 also applies a slotted channel access for the neighbor discovery and topology reporting phases which are allocated by the controller. However, we designed the technique reactively to have more control over the actuality of the topology representation.

Li et al. [Li+02] extend the common MhCD2D architecture with so-called multi-hop nodes mostly aiming to relay data transmissions between devices. These multi-hop nodes increase the coverage of access points. So far, multi-hop nodes are assumed to be placed at fixed predefined locations connected to a power supply. Devices periodically broadcast beacons updating directly connected participants. Sub-topologies are also periodically reported to the base station. A controller, deployed at the base station only receives route requests via an uplink channel if origins have no routing entry to the desired destinations in their tables. Results show that high performance can be reached which highly depends on the placement of the multi-hop nodes. Li et al [LYC03] adopt the proposed architecture, described in [Li+02] assuming a similar proactive update behavior to report the topology to an outsource controller instance.

Fodor et al. [Fod+12] address general challenges and communication techniques of D2D architectures to better exploit the potential. They identify challenges and requirements concerning device-to-device discovery and route computation processes. The authors discuss the functionality of several device discovery techniques focusing on whether to include or exclude a dedicated server and also highlight their potential for different use cases.

Considerations and challenges of Fodor et al. propose insights, advantages, and disadvantages of different device discovery techniques which are helpful to design the topology update technique in this thesis.

Lue et al. [Luo+03] propose a MhCD2D architecture supplying devices with proxy clients relaying data to eliminate insufficient link quality between devices and base stations. Each proxy client acts as a relay node monitoring the quality between the base station and surrounding devices. Clients use this relay in case no or only a weak and low data connection to the base station is available. Their approach targets delay and data rate improvements and also aims to indirectly enlarge coverage ranges of base stations through electing proxy clients. Lue et al. also investigate techniques to discover the proxy nodes proposing two routing approaches for complete path construction between devices and their base station using proxies as relays. They discuss proactive and reactive path constructions when connecting to proxy clients. Channel data rates of neighbors are interval-wise received. Participants forward route requests to next hops determining the best capacity towards proxies nodes in order to achieve higher throughput. Devices broadcast route requests, discovering proxies when experiencing low-quality connections to the base station when using reactive proxy discovery technique. Neighbors with high data rate connections to the base station contend for proxy roles. Reactive proxy discovery reaches higher throughput compared to the proactive approach.

The research of Lue et al. provides insights and experiences concerning topology detection of devices to increase connectivity to the base station. Participants are also connected via their wireless channels with each other. However, the technique does not focus on a complete topology representation on the controller. Nevertheless, their research provides important insights and experiences as we also investigate advantages and disadvantages of reactive and proactive routing in similar network architectures.

Yu-Cing et al. [HL02] also focus on centralizing topology knowledge at the base station. Like the proactive topology update technique of controller-equipped MANETs, introduced in Section 4.2, nodes periodically exchange neighborhood information. Due to restricted coverage ranges, multi-hop paths to the base station are present. Participants report their local connections periodically to the controller. Base stations periodically deliver routing information to all participants as paths to destinations will be requested by them. Yu-Cing et al. [HL02] apply a dedicated message to distribute the route before starting with the data transmission.

Abolhasan et al. [Abo+15] tackle operability and scalability in MhCD2D networks. Their work also proposes a network architecture assuming two physically separated channels. One channel is used for participants to stay connected to each other requiring multi-hop connections. The second chan-

nel is exclusively defined as a direct connection between devices and the controller. All nodes are registered at the controller using their uplink connection for routing and further protocol-related messages, such as link state information. Routing and forwarding decisions are shared between the controller and each device. The controller receives link information of each node and computes weights represented as decision units whether to be elected as route participant or not. These weights are reported back to the devices enabling the participants to discover the route in a distributed manner while balancing the traffic. Nodes are encouraged to report changes concerning topology and link conditions not periodically. They rather report information on demand. Case studies depict minimum routing overhead compared to MANET protocols AODV [DPB03] and OLSR [Cla+14].

Their research proposes a similar architecture also providing full-topology knowledge designing a reliable route distribution approach realizable. However, Abolhasan et al. also offload the routing load despite data on nodes which differs from our controller-equipped MANET architecture.

The authors Abolhasan et al. [Abo+18] propose another architecture built on top of their previous research [Abo+15]. There, an advanced Long Term Evolution (LTE) architecture comprising base stations, device tracking entity, and authorization services, among others is proposed aiming for lower routing overhead, higher scalability, and better security compared to traditional MANETs. Several SDN controllers are planned to scale mobile devices down to multiple coverage cells, each managed by a controller. These sub-controllers are connected to a main controller providing connectivity between all sub-controllers, which realizes device handovers between base stations' cells. In this work, traffic is also offloaded onto mobile devices.

Complete routing information is kept on the controller requiring devices to report each topology of each cell to the corresponding controller. Hello messages on the device channel keep neighbors informed about their one-hop connections. Adjacency lists must be reported to each sub-controller which rely on the experienced changes of various kinds of each device. These changes are, for instance, low battery lifetimes, and deviations in the information base tables.

The authors compare their proposed architecture with their previous work [Abo+15], offloading route computation and data forwarding on devices. Their results show better performance and higher scalability, speaking of minor routing overhead, and consumed energy of devices, among other indicators.

This work of Abolhasan et al. shows scientific overlap only in sub-areas

of the proposed architecture with respect to the controller-equipped routing architecture, proposed in Section 5.3. Their research rather considers the scalability with respect to a potential management overhead in case participants increase instead of designing reliable processes to accomplish an up-to-date topology representation on the controller.

Their work also comprises sub-controllers and directly connected devices. However, this structure is embedded in a bigger architecture on which emphasis is placed. Nevertheless, their topology reporting technique showed overlaps and enriched our idea generation.

Yuan et al. [YGW14] study routing techniques between participants in MhCD2D architectures and propose three self-defined routing strategies. Their research also considers device populations and topology densities when comparing their routing techniques. Routing is assumed to suffer co-channel interference of D2D and uplink channel, developing tailored routing techniques. Each routing strategy is designed to experience the least interference with surrounding transmissions each following a different approach.

Andreev et al. [And+14] tackle to which extent data offloading to devices in MhCD2D architectures contributes to more efficient transmission utilization of traffic. They motivate their work with the claim that service requests and the corresponding data rates through base stations, continuously increase. Infrastructure must be adapted to decrease the number of devices per base station with pico and nano cells.

In the long term, this requires a continuous investment in the infrastructure. Because of that, the authors investigate the potential of offloading data to the subscribers. Successful offloading requires adequate connections between devices that meet the service's QoS demands. Noteworthy delays and battery drains of devices have been observed if QoS can not be met.

In that context, the authors also discuss and propose neighbor discovery techniques to reduce battery usage. The use of client locations, leveraged by the controller lets control neighbor discovery by informing nodes about surrounding devices in direct coverage range if present.

The authors highlight additional advantages of infrastructure-assisted client discovery to reduce the base station's utilization share and also to increase overall performance in terms of service usage and data throughput. Results highlight, that offloading is promising and provides higher throughput. Also, their research shows, that effect of signal propagation during transmission and data offloading reduces channel occupations which increases the throughput between devices and base stations.

The use cases comprising a controller-equipped network covering multiple wireless devices via an out-of-band channel in combination with QoS when offloading data to devices provide meaningful experiences in terms of reliability and functionality. Andreev et al. provide an architecture concept and results that make it realizable to provide real-time services in controller-equipped MANET architectures.

3.2.2 Software-defined Mobile Ad-Hoc Networks

This section highlights related state-of-the-art research of SDMANETs. The articles introduce outsourced controllers in this network constellation responsible for routing and further SDN capabilities to improve data delivery in MANETs in terms of throughput, latency, among others.

Research in this section focuses on network architectures, where participants have two channels providing connectivity with peers and also to the controller. SDN controllers in multi-hop wireless networks, such as MANETs are promising in reaching QoS requirements, reacting better to specific service demands, and also more easily computing alternative paths in case of capacity bottlenecks due to profound topology knowledge [BDG18]. However, SDN targets mainly wired networks, such as data centers, and has barely been researched in the wireless sector [Kre+15]. Keeping that in mind, we present previous work dealing with an outsourced SDN instance exclusively responsible for managing MANETs, providing an uplink channel to realize communication.

Dely et al. [DKB11] introduced a SDN architecture for WSN using OpenFlow [McK+08] to enable flow-based routing for these networks. Nodes are equipped with multiple physical wireless interfaces split into two virtual ones for control and data traffic, respectively. Topology information about the WSN is exchanged via the control interfaces. This interface is used to accomplish multi-hop connectivity between participants. The data interface connects nodes to the controller via the OpenFlow protocol. A so-called monitoring and control server queries topology and connectivity information from WSN nodes to construct the network graph. The authors use OLSR [Cla+14] on the control interface of mesh devices and leverage the topology information of devices and report connections and their characteristics to the monitoring and control server. The controller constructs all paths based on the reported topology. A demo implementation considering mobility of nodes evaluates their proposed architecture. Results show the potential of integrating controller-based routing in wireless multi-hop

networks in comparing the proposed approach with mesh routing protocols.

In 2014 De Gante A. et al. [DAM14] proposed an SDN architecture tailored for WSNs focusing on extended battery life, lower convergence time for pathfinding algorithms, and less management overhead. The authors use monitoring messages, sent by the controller comprising various requests, triggering participants to report the requested local network information. The content of these messages comprises, for instance, energy level, distance to base stations, and neighbor lists, among others. This location information is used to define routing rules for requested application data that are deployed in the WSN nodes.

Their theoretical concept of De Gante A. et al. proposes reactive message exchange between nodes and the controller. The idea of triggering nodes for information has similarities to our approach although the content of the request differs.

Ku et al. [KLG14] also introduced a similar network architecture, which comprises a MANET, connected to a SDN-based mobile cloud. The research proposes architectures and processes, comprising OpenFlow [McK+08] to reinforce the strength of SDN in MANETs. All participants periodically exchange hello messages to keep track of directly connected nodes and report this information via OpenFlow-designed processes to the controller which is reachable via an uplink channel. Furthermore, a locally deployed SDN controller serves as a fallback if the uplink connection is not available. In addition, MANET participants have implemented MANET routing protocols, such as AODV [DPB03] and OLSR [Cla+14] to allow the SDN network to revert back to ad-hoc network mode. For channel access, Ku et al. introduce wireless network virtualization in which SDN controllers determine different frequencies for flows and also for the network. Networks can be sliced using different frequencies allocated by the controller and changed depending on the use case, such as frequency reservation for priority traffic and frequency hopping.

Labraoui et al. [LBF16] apply the most common MANET topology update approach where nodes periodically broadcast neighbor updates to direct connected nodes, which keep their neighborhood information based on the received message up-to-date. Furthermore, the list of directly connected nodes is also periodically sent to the controller via an uplink channel. Routes are requested from the controller in case the routing table of the nodes has no entry to the desired target.

In the year 2017, C. Yu H. et al. [YQR17] also designed and implemented

a SDMANET architecture and evaluated their approach with real devices. Thereby, the authors focused on the feasibility of their SDMANET architecture in terms of data delivery in real-world conditions. Following the SDN principle, the authors separate the MANET and the outsourced controller in DP and CP. Routing decisions are made by the controller and flows are forwarded by network nodes. The controller requests all switches to exchange their neighborhood information and report back their local view to the controller by applying the OpenFlow Discovery Protocol (OFDP). This process leverages the Link Layer Discovery Protocol (LLDP) triggering data plane devices to report topology information in the form of unidirectional links between switches to the controller [APF15]. The authors defined several multi-hop scenarios and compared their proposed architecture with OLSR [Cla+14].

C. Yu. et al. used multiple Raspberry Pi Model B+ chipsets with OpenFlow and Open Network Operating System (ONOS) [Onos]. According to the results, their concept performed better with respect to throughput measurements and successful rerouting after link failures.

Abdolmaleki et al.[Abd+17] complain about significant increases of control messages in highly dynamic networks when trying to provide topology knowledge to an outsourced controller. Therefore, the authors propose a fuzzy topology discovery protocol to tackle the mentioned drawback. Their approach is built in a SDN software solution [Gal+15] providing the tailored architecture for their algorithm. The authors define sink nodes used as gateway nodes between the participants and the controller. Nodes maintain a path to sinks to report the topology and also to request routes. Parameters of beacons received from directly connected nodes are processed through a fuzzy function, characterizing the connection and the sender of the beacon. The result of this function decides whether the node selects the origin as the next hop towards a sink node. Results show improvements compared to [Gal+15] especially when focusing on decision parameters of the fuzzy logic which are battery lifetime, and connection strength, among others.

The objective of Abdolmaleki et al. is to improve connectivity to sink nodes and also to reduce delays when reporting the topology to the controller. Their technique highlights important experiences for our approach as we also tackle to report an up-to-date topology representation to the controller.

In 2017 Poularakis K. et al. [Pou+17] developed and propose a hybrid SDMANET architecture. The objective is to find the most efficient balance between fully centralized and decentralized network architectures in order

to improve reliability and performance, as even a fully centralized network architecture, such as the SDMANET has disadvantages. The authors focus on a hybrid network architecture to combine the advantages of traditional MANETs with SDN architectures. Therefore, the authors introduced network segment clustering, a technique to perform dynamic switching between pre-configured MANET routing protocols and forwarding rules determined by the controller. This also includes the migration of backup rules. AODV [DPB03] and OLSR [Cla+14] are proposed in highly mobile MANET segments. The cluster-based technique also uses traditional MANET routing and outsources routing to the controller in case origin and target nodes are in different cluster cells. Further, participants have stored SDN backup rules, instructing them how to behave, for instance, if an active link breaks. Their proof-of-concept implementation shows the correct functionality of their proposed methods of the hybrid approach as results are similar to a traditional SDMANET architecture.

Besides the similar architecture compared to our controller-equipped MANET architecture in Section 5.3, the proposed work of Poularakis et al. shows overlaps by having a look at the big picture. Their methods are tailored for situations in which no central instance is available which we also tackle in the course of this thesis.

Nguyen et al. [NY18] propose a three-layered SDN-equipped MTN architecture to decentralize critical routing decisions from the upper layer to the mid-tier layer. The authors investigate MANETs, in which vehicles provide multiple routes to connect to other mid-tier networks and to also serve as gateway to the lower tier. The SDN controller is also deployed on MANET vehicles collecting topology information from both the mid-tier and lower-tier areas via specific agents, equipped with multiple interfaces. The controller periodically transmits hello messages and receives the current network information including routes by the agents. This information is inserted or updated and used to define new routes.

Kadhim et al. [KHS18] partition the MANET in cluster cells. Agents in clusters serve as local controllers responsible for establishing connections to nearby nodes. Cluster heads comprise two interfaces to provide connectivity to the outsourced SDN controller and MANET participants. MANET nodes are equipped with a single wireless interface. Cluster heads receive topology knowledge and further information, such as link characteristics, battery lifetimes, among others of cluster participants and only transmit new information concerning these parameters to the controller if the local cluster controller experiences changes. Route requests are either solved

by local cluster heads or forwarded to the controller if complete topology knowledge is required.

Introducing an additional layer between the controller and MANET nodes designs their architecture scalable if the number of participants increases. On the other side, cluster heads must maintain their local participants which increases the control overhead, especially when nodes switch cluster cells.

Bellavista et al. [BDG18] elaborate on the advantages and chances when leveraging SDN capabilities in MANETs to reach QoS requirements of flows. They address possible solutions on how to deal with specific situations reaching transmission capacity boundaries. Their research proposes concepts considering flow-prioritization, delaying flows, or rerouting transmissions around stressed segments. Further, Bellavista et al., elaborate on possible challenges and requirements related to the dynamically changing topology and mutual disruption due to QoS when applying a SDN controller in multi-hop wireless networks. Service-based prioritization of data flows is manageable in wireless networks according to their results in case the SDN controller takes over routing and deploys the flows in the MANET.

The research of Bellavista et al. overlaps with respect to our desired objective in leveraging full topology knowledge and routing control to improve path distribution based on specific needs.

Shaikh et al. [SW17] describe a hybrid topology update technique for MhCD2D communication that also uses a central instance for routing purposes. The authors present a generalized concept that focuses on a proactive and a reactive update technique as well as a suitable point of time to switch between these two schemes. A controller triggers all nodes to report their direct neighbors. Thereafter, the controller is able to compute a path with the previously received topology knowledge. However, their reactive topology update process is described only briefly, since they focus on a combination of a reactive and proactive update process and also on an appropriate situation to switch between this two update techniques.

The research of Shaikh et al. provides important insights when designing the reactive topology update algorithm in Section 5.3 as their research overlaps with our reactive topology update technique at this point.

Rabia et al. [SI21] also propose an SDMANET architecture with ONOS and OpenFlow [McK+08] in order to test the feasibility of SDN in a wireless network comprising mobile nodes as the continuously changing topology is assumed to be challenging. Throughput measurements with real-world scenarios have been carried out. Results are promising for future work as

they prove the feasibility of SDN in wireless networks.

Reaching promising results in a real-world scenario in which SDN manages wireless nodes is important at this point as parts of this thesis elaborate on improvements, such as reliable topology update concepts for controller-equipped MANETs.

3.3 SELF-ORGANIZED FULL TOPOLOGY KNOWLEDGE

In this section, we first review all existing routing protocols regarding full topology knowledge which are all based on the link state technique. Secondly, we discuss previous research focusing exactly on full topology knowledge where the MANET has a controller deployed on an arbitrary node participating in this network.

3.3.1 Proactive Link State Routing Protocols

A variety of routing protocols for MANETs have been developed and tested so far. Xiaoyan et al. [HXG02] and Boukerche et al. [Bou+11] summarize and categorize all protocols based on their functionalities. In this part of the related work section, we only focus on routing protocols providing full topology knowledge. Hong et al. differentiate between flat, hierarchical, and location-based protocols.

Starting with location-based routing, this technique explicitly uses geographic data as addressing scheme. So-called destination areas are defined to deliver data. Routing pursues the goal to forward data towards the destination area. In general, location-based routing is rather designed for multicast routing where several target nodes are positioned in the same destination area. Full topology knowledge is not necessary and not considered.

Hierarchical routing partitions the network in several areas, making routing scalable. The network is partitioned into clusters defining and assigning roles and functionalities to nodes in each cluster area. Geographical clustering is the most common approach to group members. Hierarchical routing is mostly applied in large networks in order to reduce control message exchange as protocol messages are not forwarded beyond clusters. This enables an individual routing behavior in each area but also requires routing to transfer data between clusters. Full topology knowledge can be accomplished inside clusters but not beyond, making this routing category useless for our needs.

Reactive and hybrid protocols, as described in Section 2.2.1 are also not suitable when aiming for full topology knowledge. Reactive protocols generate partial topology information only if a route is requested and sometimes during data delivery. The sub-topology information per node depends on the number of transmission requests. Hybrid protocols behave similarly to hierarchical protocols, defining clusters and connecting them reactively.

Boukerche et al. also partition routing protocols in multi-path, multicast, and power-aware. The first technique constructs several paths from source to destination mostly heading for reliability in terms of data delivery. Approaches are built on existing reactive routing protocols, such as AODV [DPB03] and DSR [HMJ07].

Multicast protocols are used to deliver data to multiple destinations. This routing scheme is suitable for multimedia traffic, such as video streams and audio or video teleconferencing where nodes subscribe to the specific service. Some reactive multicast protocols exist whereas the majority of multicast protocols construct and maintain a tree storing their subscribers. Full topology knowledge is not the focus for multicast protocols but rather electing core nodes (tree members) having promising connection characteristics for long-living tree structures.

A key characteristic of MANETs is that participants only have limited power supply as they are not connected to the infrastructure. Therefore, power-aware routing protocols focus on the least battery usage during routing. All routing protocols reviewed by Boukerche et al. apply reactive route construction to better include the current battery usage of route participants.

Based on these findings, most of the proposed routing protocols, aiming for full topology knowledge are of proactive fashion and all apply link state routing. However, the standard proactive link state technique, where each connection change results in retransmissions of control messages by participants is inefficient and does not scale with increasing number of participants [SRS01].

Chen et al [CG98] propose modifications of basic link state routing, called Global State Routing Protocol (GSR). Unlike classic link state routing, nodes do not forward link state packets beyond the first hop. This means that a node receives link state information of a participant after $n-1$ intervals, where n represents the number of hops both nodes are apart of each other. Full topology knowledge for each participant is accomplished in the worst case after $\mathcal{N}-1$ link state packets, where $\mathcal{N}$ represents the set of all MANET nodes. This would be the case if all nodes of a MANET

are arranged as a chain having connections to their predecessors, respectively. Similar to link state routing, participants populate a distance table with shortest paths to all MANET members which are computed based on the link state information.

Santiváñez et al. [SRS01] also propose an extension to disseminate link state messages in MANETs, named Fuzzy Sighted Link State (FSLS). This approach limits the number of nodes, to which link states are transmitted to and also limits the time between link state dissemination. Observations showed that changes of distant nodes have little impact on a local node's next hop decision. Nodes increase transmission intervals depending on the defined TTL of their link state packets. The higher the configured TTL, the higher the waiting period of link state packets between two transmission times. This makes control message overhead more scalable with increasing number of MANET participants but also decreases the actuality of the topology.

Pei et al. [PGC00] propose a similar approach also minimizing control message overhead in increasing topology update intervals depending on the hop count. The proactive routing protocol, called Fisheye State Routing (FSR) shares topology information with nodes within near vicinity more frequently compared to distant nodes. Nodes adapt message content and transmission frequency according to self-defined scopes. Each node generates from its point of view scopes, representing the TTL to other participants. MANET members assign nodes to their scope 1 if TTL equals 1 as well. Routing entries corresponding to nodes within the smaller scope are propagated to the neighbors with the highest frequency. With respect to scope range 1, topology updates of each node contain only link state information of its directly connected neighbors. This link state information will not be forwarded beyond scope 1. Topology updates of scope 2 only contain link state information of all 2-hop neighbors and are broadcasted less frequently.

OLSR [Cla+14], which is introduced in detail in Section 2.2.3 also provides full topology knowledge while focusing on minimum control message overhead and up-to-date information giving the opportunity to disseminate lost connections immediately. Participants elect neighbors as MPRs, functioning as topology transmitters for their direct neighbors. MPRs send their own topology control messages and also forward received topology control messages from other MPRs. The idea is to reduce the number of MPRs in order to reduce the control message overhead to a minimum. Dissemination of topology control messages provides a complete topology knowledge

on nodes. Flooding of these messages in addition accelerates the update information.

Garcia-Luna–Aceves and Spohn propose Source-tree Adaptive Routing (STAR) [GS99]. The routing fashion resembles multicast as each node maintains a tree reaching each participant. Therefore, nodes periodically exchange their sub-trees on the basis of their 1-hop knowledge. Each node sets itself as the root of the sub-tree and disseminates this branch to its neighbors. In general, the aggregation of adjacent links and source trees reported by neighbors constitutes the partial topology known by a node. The topology construction technique of STAR is not further considered as this protocol routes based on partial topology knowledge.

3.3.2 Self-organized Full Topology Control

This section introduces previous research concerning topology update algorithms in MANETs. Thereby, we focus especially on situations where the controller, responsible for routing is deployed on a node participating in the MANET. This related work section refers to Section 6.2.

Fotouhi et al. [Fot+16] complain about the inefficient and unreliable multi-hop data delivery in wireless networks in terms of distributed traffic control. In the course of that they propose an SDN-enriched routing approach including a self-organized topology discovery concept. The authors define an adapter node, used as a bridge between the controller and participants, starting a topology discovery phase. This node broadcasts a beacon containing the hop count parameter. In case the received signal strength of the beacon is above a defined threshold, receivers add the origin of the frame to their parent list and also update their hop count parameter by incrementing the value by one. Nodes rebroadcast the beacon with this incremented hop count. Each node, receiving a beacon is supposed to rebroadcast this message to complete the parent list. The parent list of each node is periodically sent to the controller.

This discovery technique is performed on a regular basis. The controller creates a topology map based on the parent lists and defines routes for requested transmissions. The authors also introduce a network operation phase comprising traffic monitoring and load balancing algorithms, leveraging the topology map.

The self-organized topology update algorithm, introduced in Section 6.2 aims to gather all connections in MANETs in minimum runtime. The approach of Fotouhi et al. shows overlaps with our proposed topology update

algorithm in this thesis as they also propose an update technique having the controller placed within the network. However, their technique is of proactive nature, not focusing on a topology snapshot at the controller at a specific time.

In 2018, the authors of [MDS18] and [DMS18] also propose a concept of deploying the controller on a MANET participant using a single channel. In more detail, connections between the controller and the nodes are formed as a tree in which nodes act as relays for other participants to reach the controller node. The approach covers both topology discovery and route maintenance to the controller to be able to route the received sub-topology to the controller node. In doing so, each participant periodically broadcasts neighbor discovery messages storing its hops to the controller node in the MANET. A node sets a node to its next hop towards the controller in case the hop count of the node that sent the neighbor discovery message is below its own hop count towards the controller. In addition, receivers add the origin of this message to their list of neighbors.

This technique is similar to the gradient routing scheme introduced by Ye et al. [Ye+05]. Furthermore, participants periodically send a neighbor information message storing all directly connected nodes to the controller. This information is used to construct the topology. Route requests are sent to the controller which computes the paths and deploys the next hops in the forwarding tables of the route participants. The authors evaluate their concept against OLSR with a different number of nodes and different velocities. Thereby, they obtain promising results until a density of up to 50 nodes.

A. Dusia et al. [DRS19] extended the approach of [MDS18; DMS18] with the objective to reduce the routing overhead especially when gathering the network topology. Thereby, the MANET detects critical nodes when exchanging hello messages. Each non-critical node must directly be connected to a critical node to provide the functionality of the algorithm. Based on that, it is sufficient to only report neighbors of all directly connected critical nodes to the controller in order to construct the entire MANET topology. Similar to the initial proposed process, introduced by [MDS18; DMS18], critical node election also takes place periodically, since the controller starts the hello message flooding after a defined interval. In case data delivery is intended, a node sends a transmission request message to the controller which computes the path based on the reported topology. The route is deployed by the controller sending a reply message storing the requested path on the MANET.

However, their SDN-enabled MANET concept is not designed to report the topology in minimum time when needed. The focus of Dusia et al. is to reduce the control message overhead during topology detection. Nevertheless, their approach provided an interesting technique to keep track of the route to the controller for each participant which this thesis also investigates in RQ 3 with Section 6.4.

Zabian et al. [Zab06; Zab07] propose a topology discovery algorithm for mobile wireless networks with the objective to reduce the number of hops when computing paths from origin to target nodes. The main focus of their approach is to reduce the power consumption of participants which depends on the length of computed paths and the sizes (bit) of frames. The algorithm is designed generic making it possible to run the discovery algorithm on top of routing protocols. Participants of the MANET form a dynamic tree comprising all nodes starting at the origin. Therefore, each node broadcasts hello messages waiting for responses creating a parent-child connection. Encountered hello messages are answered with respond messages setting the node that previously sent the hello message as its child. The new child starts broadcasting hello messages when the connection is established. Furthermore, already existing and detected nodes can respond to a hello message in also sending a response. The authors sent the number of children per node to 5. This process is named the discovery phase which is applied by each participant. Further, a monitoring phase maintains the constructed tree to ensure that no node loses connectivity to the MANET. There, nodes send ACK messages to MANET participants that previously sent hello messages to inform them about their presence. However, their algorithm is not aiming for an up-to-date snapshot of the current MANET topology. Inspired by the number of allowed children per parent, we also restricted the number of potential parents when designing the topology update algorithm, introduced in Section 6.2. Defining the number of potential upstream parents generated faster parent-child connection and increased the runtime of our approach.

Thongthavorn et al. [TP21] align the topology detection technique to traditional SDN functionality comprising CP and DP implemented in MANETs. Therefore, the controller, which is placed on a MANET participant, periodically transmits advertisement messages ensuring on the one hand the existence of the controller role and on the other hand the placement and the corresponding reachability. Nodes in close vicinity insert or update their route to the controller and rebroadcast these messages. All children behave the same when receiving these messages. Additionally, participants

also maintain and forward their neighbor list to the upstream node towards the controller as long as they receive the interval-based advertisement messages. Paths to target nodes are either constructed locally in case an entry in the routing table is present or a route request to the controller is delivered to receive a route later. Results outperform AODV [DPB03] in terms of required routing load, and delivered data, among other performance parameters.

3.4 MULTI-COMMODITY NETWORK FLOW ROUTING

This section introduces the Multi-commodity Network Flow Problem (MCNFP) which is applied when searching for paths or routes of multiple commodities. Furthermore, we highlight related research, also applying similar solving techniques for the given problem.

The MCNFP has firstly been introduced by Ford and Fulkerson [FF04] and has widely been researched over decades, see [Ken78; Ass78] as well as nowadays [SB20]. The MCNFP is often applied in transportation, production, and in network communications. A conventional approach is to formulate the objective as a Linear Programming (LP) model, also applied in this thesis in Section 7.4.4. The MCNFPs describes optimization problems where multiple paths with certain constraints have to be found focusing on the same optimization criteria described either by a minimization or a maximization function. The optimization technique is also used to split multiple flows in several paths aiming for the desired optimization objective. Previous research has shown, that splitting a flow into multiple paths in wireless networks achieves only negligible performance increase [HT18]. Because of that, we formulate the MCNFP as a model containing only unsplittable paths.

The MCNFP applied in wireless networks such as MANET has barely been researched so far as the up-to-date survey of K. Salimifard et al. [SB20] is dealing mostly with wired network applications where multiple commodities are routed. There exist issues, such as mutual interference and coverage ranges, among others that distinguish wireless from wired networks. Furthermore, the difference becomes even more apparent when mobility of participants comes into play.

Conventional solving techniques of solvers, applied to compute a MCNFP are exact methods like Branch & Cut and Branch & Bound [Mit09]. These solvers guarantee the optimal solution if the runtime to find this result is not a major concern. As our research in the following section shows,

the majority of previous research tackles the MCNFP with exact methods because of its reliability.

3.4.1 Exact Solving Methods

Exact solving methods summarize related approaches, aiming for the global optimum, which can be achieved by using a solver, such as IBM ILOG Cplex Optimizer (CPLEX), provided by IBM [1]. Research in this field often assumes central topology knowledge, deterministic channel characteristic, and optimal routing and packet transmission scheduling policy when designing the path distribution technique.

Kolar et al. [KA06] design a LP model maximizing the routing performance, assuming the global routing knowledge. As required in wireless networks the model considers link-based interference to not exceed transmission capacity constraints. Multiple extensions of the optimization function are shown to reach desired goals even closer. The authors compare their model with DSR [HMJ07] showing performance impacts in terms of delivered data, end-to-end delay, and general throughput. Unlike our approach, their formulation allows splittable paths of multiple flows.

Capone et al. [CM07] propose a formalization of the MCNFP for Wireless Mesh Networks (WMNs) (WMNs) assuming a fixed network structure in which the topology does not change over time. Path computation aims for maximum flow that can be deployed in the wireless mesh network. Their formalization also accepts multiple paths per flow, similar to the work of Kolar et al. [KA06]. Furthermore, the authors propose a traffic scheduling scheme using a graph coloring technique for slot computation which depends on the link capacity and the data rate demands of flows. Their numerical examples comprise grid and traditional mesh networks focusing on the total accepted traffic, forwarded through both networks.

Ciciriello et al. [CMP07] compare the optimal solution when computing paths for multiple flows with a distributed route generation approach. The solver searches for a path setting with the least number of transmitting links. The distributed approach assumes an up-to-date tree comprising all nodes in which the root of the tree is the node actually maintaining the tree. Participants implement the same function determining the quality of each neighbor which decides whether to select a node as the parent in the tree and as a consequence as the next hop toward the target or not. Changing the current parent to a different neighbor occurs when the current quality

[1] https://www.ibm.com/de-de/products/ilog-cplex-optimization-studio

of a neighbor, which is not the parent, is the best compared to all other current neighbors including the current parent. As assumed, results of the distributed approach are worse but in some cases close compared to the optimal results, generated by the solver.

Our last RQ also compares various pathfinding techniques with a solver aiming towards the optimal solution. Results are pioneering as approaches such as the distributed approach of Ciciriello et al. can be customized heading for minimum runtime while still computing reasonable results.

Szymanski et al. [Szy12] minimize the routing costs in distributing high-bandwidth backhaul flows via shorter paths to minimize the utilization and transmission power in wireless networks. Their LP formalization focuses on flow maximization whereas a subgraph of the main topology is used to enforce the distance constraints for high data rate demanding flows. Subgraphgs for each node in the network are generated applying Dijkstra [Dij59] as this algorithm returns a path comprising minimum edges to each destination in the graph. Next, distance constraints shorten the subgraphs based on the commodities. The LP, aiming for minimum routing cost is formulated for these subgraphs. According to the results, distance constraints improve execution times noteworthy.

Extracting criteria-based subgraphs out of the network and consequently generating a smaller search space is an interesting and promising approach especially when aiming for minimum runtimes. However, solutions become rare in highly-utilized capacity constellations designing it complex to shrink the initial graph as removed edges and vertices might be necessary to identify any solution.

Bazan et al. [BJ08] also aim for maximal throughput in wireless networks. In contrast to the majority of MCNFP-solving strategies, the authors use smart antenna configurations that representing realistic antenna models and compare the results with an ideal radio model carrying out numerical results. Smart antennas represent directional radiations and receptions of energy distinguishing them from omni-directional antennas, where energy radiation in each direction the same in theory. Both configurations are evaluated with the same formalized model aiming for optimal throughput while considering the interference with network participants when determining the paths.

The results of Bazan et al. are important to figure out whether simulations can produce realistic results when using radio models not assuming any environmental influences except of mutual interference due to simultaneous transmissions.

3.4.2 Meta-Heuristic Pathfinding Techniques

Nature-inspired Meta-Heuristics, such as GP [Ban01] and Ant Colony Optimization (ACO) [DMC96] pursue more unsupervised approaches. This generates more diversity leading to results probably not being chosen by a practical solution, such as embedding a shortest path algorithm in a routing engine searching for a single path of each flow.

Genetic Algorithm (GA) converts a population of individuals with their respective fitness through operations including crossover and mutation resembling the Darwinian evolution. As described in Section 2.6, GP is a nature-inspired approach mimicking evolution in computer science. Researchers developed and extended this Meta-Heuristic extensively [Spe+99; MR19; PS06]. GP is promising in solving the MCNFP since the generation of the initial population can be customized to cover fast and efficient solution creation and high diversity regarding various paths. At first, the following work focuses on computing paths for multiple flows in wired networks applying GP. Secondly, we present previous research tackling path computations with GP of a single flow in wireless multi-hop networks such as MANETs.

Wired Network Architectures

El-Alfy et al. [ESM07] apply GP to route data aiming for minimum routing cost and minimum link utilization. They also use LP to enhance generating the initial population focusing thereby on maximum hop count per path, the number of virtual paths, and the link capacities. Some chromosomes become infeasible at runtime through modifications by GP, forcing them to repair also applying LP. The evaluation compares the proposed approach with a so-called simple GP technique exclusively generating initial populations randomly and penalizing chromosomes exceeding link capacities and maximum number of hops. Comparing the results of both fitness functions, one can highlight better performance of the hybrid approach (GP combined with LP) leading to the least utilization and shorter paths.

Results highlight to which extent different generation methods of the initial population affect the outcome of GP. So far, relying on only random paths reduces the probability to generate satisfying results. Diversity in terms of initial population generation methods obviously increases the probability of high quality results.

Yu et al. [YK18] propose a GP-aligned routing strategy for SDN. GP-enriched routing takes place when standard shortest path computation is

not satisfying the flows' data rate demands. The authors emulate their network comprising 5 switches using Mininet [2]. In addition, they configure a bottleneck connection in this network in order to shrink the quality of data transmission. Results demonstrate the agility of GP as routes are adapted to the transmission volume and network area capacities. To be more concrete bottlenecks are identified and detoured if connections are present, providing sufficient transmission capacities.

Similar to Yu et al. [YK18], Farrugia et al. [FBB18] also apply GP in SDN. Their practical motivation is to compute the minimum number of multi-paths for each flow as reordering of Transmission Control Protocol (TCP) segments at the target generates additional overhead and increases with the number of multi-paths per flow. The fitness function aims to maximize the number of flows, minimize network costs, and also minimize the number of paths per flow. Furthermore, chosen connections must meet their transmission capacity boundaries.

Results compare the GP approach, LP, OSPF [Moy98], and Equal Cost Multipath (ECMP) [HT00; Hop00] with each other regarding the throughput reached by each approach. It turns out that the fewer paths used per flow the higher network load is reached where GP returns best results, followed by LP, ECMP, and OSPF.

Wireless Multi-Hop Networks

Magaia et al. [Mag+15] tackle the multi-objective routing problem applying GP in WSNs. Thereby, the authors identify the latency of data transfer and Expected Transmission Count (ETX) during QoS transmissions as critical. The latter represents the number of transmissions including retransmission required to deliver a packet to the target node. Authors define the number of successfully delivered packets in contrast to failed delivery attempts from source to destination and vice-versa as an indicator of whether a connection is expected to have plenty of retransmissions or not.

DSR [HMJ07] and an extension of this routing protocol [MPG15], providing multi-path and ETX-based metric routing are used to compare their performance with the proposed GP approach. GP allows to concentrate on both objectives using a customized fitness function focusing on minimum delay and ETX. The DSR routing technique focuses on minimum hops per path not considering parameters, such as ETX or delay. Their proposed DSR extension generates multiple paths from an origin to a target focusing

[2] http://mininet.org/

on minimum cross-interference of these paths.

Results show that GP outperforms the DSR extension regarding ETX and also generates less delay with respect to plain DSR and the extension of this routing protocol.

Camelo et al. [COC10] also apply GP to find QoS conform paths in WSNs. Their individual population generation combines randomness and search algorithms, such as Breadth First Search (BFS) and Depth First Search (DFS). The evaluation focuses on several scenarios comprising different multi-objective parameters. Similar to the research of Magaia et al. [Mag+15] the GP approach outperforms shortest path algorithms not capable of aiming for multiple objectives.

To sum up, GP is well suited to distribute multiple flows in wireless multi-hop networks, such as MANETs when aiming for multiple objectives which is why we selected this meta-heuristic as pathfinding technique among other approaches.

3.4.3 Practical Algorithmic Solutions

Adaptation and customization of heuristics solutions, such as combining shortest path algorithms deterministically, have high potential as these approaches can be extended and modified based on the specific need. To be more concrete, one can wrap different shortest path algorithms and apply them in a deterministic scheme in order to aim for a pre-defined routing objective. The speed of execution could, for instance, be decreased with search space boundaries like defining maximum path costs of connections [Bem+19]. Furthermore, one can search for multiple paths in the first place having minimum path similarity [Cho+15] and use these paths as search space for further computations to also reduce execution times.

Krzysztof [Wal06] propose two Greedy-based algorithms both focusing on distributing multiple flows in wired networks. His approaches focus on optimizing found connections in the network. Specific edges of the graph, representing the topology are removed to select other edges for successive flows and to minimize the search space for the objective function. The first algorithm processes connections, one by one sorted based on a defined criteria which can be configured beforehand.

The second algorithm already starts with an already computed potential solution. Furthermore, this technique also searches after routes, which comprise the most connections and removes those edges. This technique aims to identify and eliminate connections having routes using the most

congested arcs. The obtained results of the second Greedy extension outperform compared to previous research with the expense of computation times as decisions are more time-consuming. This designs this technique unflattering when applying this approach to MANETs as results must be returned quickly due to the mobility of nodes.

Katayama [Kat19] also presents a Greedy-based approach that search for multiple paths from an origin to a target node. Their approach operates on a reduced edge set of the graph and applies Branch & Bound [Mit09] to improve the result computation process. The algorithm computes solid results and also reduces the runtimes. The fast execution times are mainly due to the removal of specific edges in the graph, representing the network.

Applying the Greedy algorithm to solve the MCNFP seems to be promising concerning found paths and measured execution times which is why we also decided to develop a Greedy-based approach as pathfinding technique. Our approach is described in Section 7.4.2.

Akin et al. [AK19] search for edges in a graph high-potentially being chosen by multiple flows. The authors apply their approach in wired networks. The objective is to avoid to utilize a link with more than one flow. This distributes paths of flows more evenly. Therefore, the authors apply a Breadth-First Dijkstra combination to at first detect these links potentially being chosen by multiple flows. Next, the algorithm penalizes these connections by increasing their costs. Dijkstra [Dij59] is started again for each flow to dynamically reinforce the objective of adapting link costs of utilized edges. Transmission resources can be exploited more efficiently which is proven by their evaluations when comparing their approach with similar pathfinding algorithms.

The proposed technique of Akin et al. seems to be a promising approach with respect to the routing objective which we tackle with RQ 4. There, the objective is to find capacity-conform and robust paths of multiple flows when their currently deployed paths over-utilize the MANET. Enforcing the number of flows per connection to one reduces the probability to over-utilize connections and also distributes the transmission even more. We extended the research of Akin et al. tailored for wireless multi-hop networks, such as MANETs since the approach guarantees fast execution times and avoids link utilization. Our approach is introduced in Section 7.4.3.

3.5 SUMMARY

We classify related work with respect to the framework which we develop throughout this thesis. In the end, this framework contributes an autonomous MANET routing engine focusing on high data rate-demanding multi-flow distributions. Therefore, up-to-date topology knowledge (nodes and connections) builds the basis to compute routes based on that knowledge. Furthermore, transmission overhead recordings and MANET-tailored pathfinding techniques support link utilization aware multi-flow path distribution also included in our framework. We present two network architectures making up this framework. The first introduces a MANET equipped with a controller reachable via an out-of-band channel, called uplink. The controller is equipped with transmission utilization, full topology knowledge, and pathfinding techniques ready to compute paths for multiple flows.

The second architecture purposes similar characteristics. In contrast to the first architecture, we place the controller on an arbitrary node in the MANET utilizing a single channel for control overhead and data. This architecture has a different topology refresh technique to keep the controller up-to-date with topology knowledge. In addition, this architecture comprises a process to manage the controller role self-organized within the MANET and also provides a route to the controller algorithm to guarantee controller reachability for each node. The latter is mandatory as no direct connection to the controller, deployed on an arbitrary MANET node, exists in contrast to the first architecture. The utilization recording and computation model and the developed pathfinding techniques can be applied to both architectures to reach the desired routing goal.

The following Tables 3.1 and 3.2 list previous research, introduced in this chapter, dealing with wireless multi-hop networks having a central routing instance as these are key components to provide up-to-date and centralized controller routing in MANETs.

Table 3.1 lists related work discussing similar network architectures, such as SDMANET and MhCD2D providing processes, message types, and algorithms, among others, to propose an outsourced routing instance for multi-hop wireless networks.

The definition Complete characterizes the topology representation. This requirement is fulfilled if the controller receives topology information (connections and participants) to build up a network graph representation of the multi-hop wireless network via an uplink. The topology is partially complete if the corresponding concept does not desire to report the entire

Table 3.1: Literature review of controller-equipped wireless multi-hop networks having an out-of-band channel to provide direct connectivity between the controller and mobile devices. The symbols are defined as follows: ✓ = include, ❍ = include with limitations, and ✗ = do not include the characteristic.

Related Work	Topology		Controller		N-type
	Complete	Up-to date	Dynamic Channel	Routing Authority	
Abdolmaleki et al.[Abd+17]	✓	✗	✗	✓	SDMANET
Ananthapadmanabha et al. [AMM01]	❍	✗	✗	✓	MhCD2D
Shaikh et al. [SW17]	✓	❍	✗	❍	SDMANET
Li et al. [Li+02]	❍	✗	✗	❍	MhCD2D
Nguyen et al. [NY18]	✓	✗	✗	❍	MTN
Poularakis et al. [Pou+17]	✓	✗	✗	✗	SDMANET
Kadhim et al. [KHS18]	✓	❍	✗	✗	SDMANET
Lue et al. [Luo+03]	❍	✗	✗	✗	MhCD2D
Ku et al. [KLG14]	✓	❍	✓	❍	SDMANET
Rabia et al. [SI21]	✓	✗	✗	✓	SDMANET
Abolhasan et al. [Abo+15]	✓	✗	✗	❍	MhCD2D
Bellavista et al. [BDG18]	✓	✗	✗	✓	SDMANET
Poularakis et al. [Pou+19b]	✓	✗	✗	❍	MTN
Yu-Cing et al. [HL02]	✓	✗	✗	❍	MhCD2D
De Gante et al. [DAM14]	✓	❍	✗	✓	SDMANET
Abolhasan et al. [Abo+18]	✓	✗	✗	❍	MhCD2D
Dely et al. [DKB11]	✓	❍	❍	✓	SDMANET
Labraoui et al. [LBF16]	✓	✗	✗	❍	SDMANET
Yu et al. [YQR17]	✓	✗	✗	✓	SDMANET
Cc-MANET	✓	✓	✓	✓	SDMANET

topology to the controller. The requirement Complete is not fulfilled in case the controller is not used to route traffic and as a consequence does not store a topology representation.

A topology is said to be Up-to-date if the update mechanism focuses on an instant response time to report the entire topology to the controller. A promising approach would be to design the topology update technique reactively. Partially up-to-date is, for instance, a technique in which the controller triggers the topology reporting. However, the adjacency lists of nodes are maintained proactively. A topology is not up-to-date if the algorithms do not focus on a fast response.

The N-Type (Network Type) defines the architecture the controller-equipped MANET is designed for.

Table 3.2: Literature review of controller-equipped wireless multi-hop networks having the controller deployed on an arbitrary participant. The network uses a single channel for routing load and data. The symbols are defined as follows: ✓ = include, ❍ = include with limitations, and ✗ = do not include the characteristic.

Related Work	Topology		Controller			N-type
	Complete	Up-to date	Role Control	Path Control	Routing Authority	
Fotouhi et al. [Fot+16]	✓	❍	✗	✗	✓	WSN
Dusia et al. [DMS18]	✓	✗	✗	✓	✓	MANET
Zabian et al. [Zab07]	✓	✗	✗	✓	❍	MANET
Thongthavorn et al. [TP21]	✓	✗	✗	✓	❍	MANET
Sc-MANET	✓	✓	✓	✓	✓	MANET

The Dynamic Channel functionality is fully supported in case the controller also controls frequency or time division to more efficiently achieve the objective of the research.

The characteristic Routing Authority is fulfilled if the controller is exclusively used as a routing engine and no other situation is designed in which routing is taken over by any other participant. An architecture provides partial routing authority if, for instance, nodes start a new data stream to the next hop when detecting an existing table entry to the destination and skip requesting a route from the controller as an entry in the routing table already exists. No control is set in case routing is, for instance, only partially computed by the controller.

This thesis also focuses on controller-equipped multi-hop wireless network architectures having the controller deployed on an arbitrary participant. Nodes utilize a single channel to exchange control overhead and data. This architecture shall provide similar functionality and must provide a management technique containing controller reachability and election, as already mentioned. We have identified and studied several scientific papers characterized in Table 3.2 regarding the defined requirements.

The requirements Complete, Up-to-date, and Routing Authority are identical to Table 3.1. The Role Control specifies if the network comprises a management process to maintain the controller role self-organized. Controllers in mobile wireless networks can experience outages or lose connectivity requiring to elect a new controller self-organized. This require-

ment is fulfilled if the authors propose controller role management processes.

The characteristic Path Control comprises a route to the controller concept to provide a path for each participant. This requirement is fulfilled if the topology update technique also maintains these paths proactively, reactively, or hybrid. N-Types, which introduce controller-equipped multi-hop wireless networks, are traditional MANETs or WSNs.

In the last row of both tables, we list our network architectures, comprising all mentioned requirements. Both architectures are discussed in detail from Chapter 4 to 7.

CHAPTER 4

The Utilization Model

This chapter presents two MANET utilization models both aiming at accurate overhead predictions, presenting different transmission recording and computation methods. The second concept proposes an extensive utilization recording concept, mainly inspired by the findings of the first approach. In this context, we also introduce network architecture extensions in which the utilization concept is implemented.

4.1 DISTRIBUTED MODEL

As mentioned in Section 2.2, various routing protocols for MANETs have been researched so far. Functionality and logic of proactive and reactive routing techniques have been extended with specific QoS support, fitting for specific applications [HT07]. Latency, data rates, and also capacities of links are used to enrich routing decisions. Guo Z. et al. [GM07], for instance, propose a link delay metric, obtained with neuronal networks using OLSR. Also, Jabbar et al. [JIN14] adapt the routing metric to elect nodes having sufficient energy in terms of battery.

Speaking of proactive routing protocols, network conditions are shared using protocol-specific control messages. These network conditions are mapped to numeric representations which increase or decrease the probability that a certain node is elected as route participant. Routing tables often store multiple next hops to reach the desired destination. The following statement applies to proactive routing in this context: Routing is fully based on parameters representing link and environmental characteristics that are decision-makers for routing engines.

Reactive routing designs pathfinding more interactive. This technique constructs routes based on current network conditions, such as transmission overhead and occurring delays. Routes are actively searched and constructed immediately before the desired data transmission starts. Nodes

broadcast route requests till the destination is found. Routing protocols, such as AODV [DPB03] and DSR [HMJ07] apply such routing strategies. Network conditions, such as transmission overhead and connection capacities, are in theory considered as nodes take more time to process and forward route requests if they are occupied, which avoids stressed segments automatically.

The following sections introduce the distributed utilization recording model based on the reactive routing protocol AODV [DPB03]. The objective is to successively inject low-demanding data flows to provoke an even transmission utilization across the MANET until each participant is almost fully utilized. We extend AODV during path construction to instantly forward, delay, or drop RREQs depending on the utilization of the node. The extension should distribute all flows more evenly and generate more throughput of the utilization measurement concept is accurate.

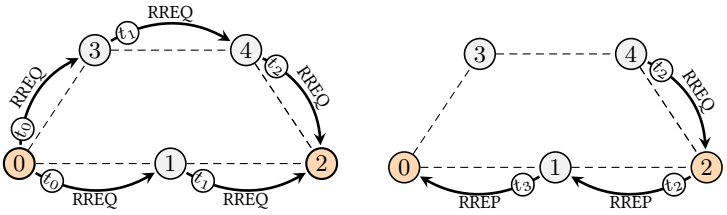
First, the traditional path construction process of AODV is introduced. Secondly, we present our capacity-based path construction including the utilization measurement concept, followed by an evaluation, comparing standard AODV with Capacity-based Routing (AODV-CBR) [SRR18].

4.1.1 AODV Path Construction

Paths are created only if needed, which is the identifying feature of reactive routing protocols. There, origins start the route discovery process, broadcasting route request messages that contain the destination. Nodes forward the route request if they are not the desired destinations or the destination is unknown so far. In case none of the two mentioned conditions are true, a route response is sent back along the reverse path the route request came from. The first received route response decoded by the origin will be used as path for the desired data transfer.

Several reactive routing protocols for MANETs, such as AODV [DPB03] and DSR [HMJ07] apply this process. In theory, this routing strategy is inclined to avoid utilized network segments, as nodes automatically take more time for data processing due to occupied queues and occurring contention overhead on the MAC layer. One suspects an evenly overhead distribution, as route request forwarding takes longer on occupied nodes.

Figure 4.1 introduces the path finding process with standard AODV form origin 0 to target 2. Like in previous figures, circles having numbers and dashed lines between them represent MANET members and their connection with each other. Arrows show AODV message types named by



(a) AODV path construction process broadcasting RREQs to find route from 0 to 2.

(b) RREP process using reverse path previously constructed with broadcasted RREQs.

Figure 4.1: Exemplary AODV route construction using RREQs and RREPs to find path from node 0 to 2.

their labels. Orange-colored nodes depict origin and targets. In this figure, node 0 is the origin and 2 is set to the target. (t_n), with $n \in \mathbb{N} \cup \{0\}$, represents the timestamp. According to Section 2.2.2, AODV uses the messages RREQ and RREP for path construction between origin and target nodes. We assume node 0 has no valid route stored for target 2 in the routing table.

To follow the protocol, 0 broadcasts a RREQ (Figure 4.1a) containing message fields target address, origin address, and the sequence numbers of origin and target. RREQs carry further message fields, such as Join Flag, Repair Flag, among others which are not further discussed here as they are not relevant for this routing approach. Nodes 1 and 3 receive the RREQ at time t_0, respectively. Both perform actuality checks. In particular, 1 and 3 verify if they previously received a RREQ having equal RREQ id and originator address. Both neither have received this request before nor have a corresponding valid entry pointing to the target in their routing table.

At t_1, after updating routing tables with the precessing node, both participants rebroadcast the RREQ, having included their sequence numbers. Node 4 behaves similarly compared to their predecessors when receiving the RREQ but in addition also inserts an entry pointing to the origin in the routing table. The forwarded RREQ by 1 reaches the target 2 which at first inserts two routes pointing to 1 and 0 in the routing table and also responds to node 1 with a RREP. Node 2 receives the RREQ of 4 at the same time but discards the message, as the same request (equal RREQ id and originator) has been received within path discovery time. The RREP of 2 is addressed to the predecessor of the first received RREQ which according to Figure 4.1b is node 1. All nodes of the shortest path to the target would receive a RREP,

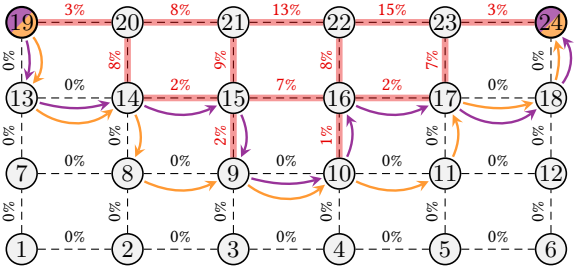


Figure 4.2: Desired route construction using reactive routing technique to evenly distribute traffic across the MANET. Colored edges (teal and violet) represent two desired routes for a data stream between 19 and 24. Percentages next to dashed links illustrate current utilization. The currently occupied connections are highlighted with red lines between nodes.

if the path were longer as they are part of the reverse route. Node 1 receives the RREP, updates the routing table, and forwards the message to the origin which also inserts the new route towards the target. Thereafter, at t_3 node 0 would start with the first data packet.

We evaluate AODV to verify if reactive route construction takes network overhead into account and distributes paths evenly on its own. Figure 4.2 introduces a grid network, presenting two possible paths in order to deliver the data of the traffic flow between 19 and 24. Red lines represent links occupied by already deployed data flows. Transmissions between 19 and 24 would most likely be routed via 20, 21, 22, and 23, in case no data flow occupies these nodes. Considering the aforementioned situation, reactive routing should detour any flow between origin and destination due to longer delay periods on occupied nodes. In doing so, this figure illustrates two possible paths for flows between 19 and 24.

We assume that RREQs reach the destination more often, if one continuously injects low data rate flows. This should lead to an even distributed network utilization resulting in more throughput, as occupied nodes take more time for queuing and processing of received frames. We investigated exactly this assumption and simulated route construction when continuously injecting low data rate flows in the network. Data flows increase the network load of shortest path between origin and target more and more, prompting AODV to construct routes around this occupied network segment.

Table 4.1: Simulation parameters

Parameter	Value
Simulation time	1000 sec
Bitrate of nodes	2.0 $Mbit/sec$
Network interface	IEEE 802.11
Sending node	19
Sending rate each flow	$32.76Kbps$
Scenario 1 (3 Flows): Destination nodes	$20, 21, 24$
Scenario 2 (5 Flows): Destination nodes	$20, 21, 22, 23, 24$

We therefore constructed the grid network of Figure 4.2 with configuration shown in Table 4.1, describing two scenarios. The first delivers 3 flows, the second 5 flows. According to the Request for Comments (RFCs) of AODV, origins only start searching for destinations if their routing tables do not store the destinations or the routes toward the destinations are flagged as inactive. Node 19 delivers data to different destinations arranged after each others to guarantee a utilized network segment and also to force AODV to start the route-finding process for each flow. Each data flow starts $10sec$ after the previous leading to an ever-increasing connection utilization, especially for nodes 20, 21, 22, 23, and 24. To recapitulate, with increasing network overhead, RREQs forwarded by neighboring nodes 14, 15, 16, 17, 8, 9, and 10 should arrive at 24 earlier leading to an evenly distributed traffic overhead. RREQ along the shortest path, via 20, 21, 22, and 23, to 24 should take longer as nodes are more utilized and as a consequence take more time to process and forward RREQs.

Figures 4.3 and 4.4 show the relation of sent to delivered payload and sent data frames of both scenarios (3 and 5 flows) in Mbit of each MANET member. Taking a closer look at Scenario 1, almost all data is delivered to the destinations. One identifies a slight decrease as more relaying nodes are positioned between source and destination. Interference is common and cannot be excluded in wireless networks which is problematic for broadcast and multicast transmissions since no acknowledgment is considered from the receiver. Because of that, the origin starts an additional RREQ after a configured timeout if the retry limit is not reached. We experienced several interfered Address Resolution Protocol (ARP) requests, resulting in dropped data frames as MAC queues drop frames if the limit, which is set to 15 entries per default is reached. The transport layer generates and passes

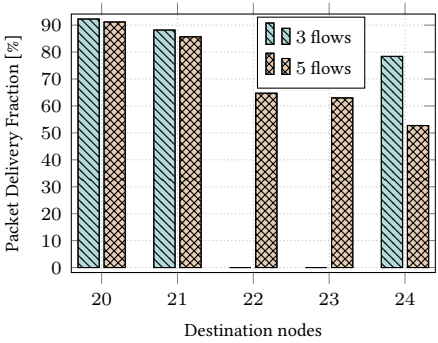


Figure 4.3: Relation of sent compared to received data of Scenario 1 (3 flows) and 2 (5 flows).

datagrams down to the network layer also forwarding the packet to the MAC layer which decreases the amount of successfully delivered data.

Throughput measurements in Figure 4.4 of each node in the MANET allow for deducing the route of all three packet flows. Nodes 20, 21, 22, and 23 carry the majority of the entire traffic, especially of Scenario 1. These nodes show the most throughput. This proves that AODV mainly constructed the shortest path towards all three destinations. We observe some detours via nodes 13, 14, and 15, as these nodes show a slight throughput increase.

The observed throughput distribution of Scenario 2 is similar compared

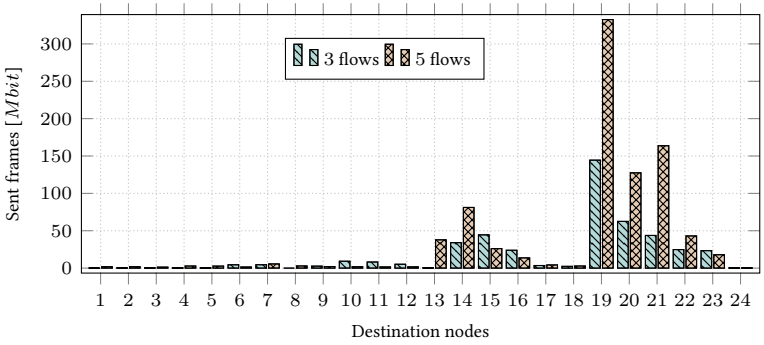


Figure 4.4: Send frames of each node in the MANET partitioned in Scenario 1 (3 flows) and 2 (5 flows).

to Scenario 1. As expected, the traffic carried by participants is higher since Scenario 2 transmits two more flows. Nevertheless, the majority of nodes in the MANET is not involved in traffic forwarding. In particular, the more links a node is separated from the shortest path of a flow, the more the probability shrinks that this node is involved in data forwarding.

Referring again to the delivered payload in Figure 4.3, Scenario 2 definitely drops more data along the path to the destinations compared to Scenario 1. Only half of the data stream addressed to node 24 has been delivered. The shortest route with respect to the number of hops, referring to both targets, has been constructed even though data has been dropped along the paths.

Taking both throughput and delivered packets into account, the grid MANET provides capacities for data delivery that are not used by AODV resulting in dropped frames. AODV still more often constructs the shortest path between origins and targets. Traffic should be detoured along unused connections to determine whether an even distribution leads to even more throughput. We therefore reinforce route construction of AODV to elect connections with minimum utilization. The following section introduces the proposed extension in detail.

4.1.2 AODV-CBR Path Construction

The objective is to manipulate route construction in a way that participants with less utilization will be preferred over highly utilized MANET members. Therefore, we modify the RREQ process. According to previous research, extending AODV shows promising results which is why we propose a capacity-based path construction process with this routing protocol. Our approach treats RREQ in a very specific way which differs from previous work discussed in Section 3.1.2. We therefore firstly introduce a transmission utilization technique and thereafter deep dive into the capacity-based path construction extension for AODV.

Utilization Monitoring

All packets traversing across a node must be recorded to determine the utilization of a MANET member. Therefore, nodes continuously monitor and record all traffic overhead that reaches their network layer to quantify their utilization at runtime. This affects mainly path participants relaying data but also MANET members that are integrated in the AODV processes. This approach uses header information to compute the current utilization

of nodes. Radio waves converted to symbols at the network interface are processed through all lower layers until they arrive as packet at the network layer. MAC addressing checks are performed before processing frames to the upper layer. Frames with destination MAC addresses that match the receivers addresses, broadcast addresses, and also multicast address matchings (not relevant for this approach) are processed to the network layer. The network layer of all nodes that start a transmission constructs the IPv4 header while encapsulating User Datagram Protocol (UDP) Datagrams to packets. This includes origin and target addresses and also data-related information such as packet length and Differential Services Code Point (DSCP) for QoS information, among others [Alm92]. When destinations or forwarding nodes receive data, they extract destination addresses and packet lengths, containing byte representation from IPv4 header [Pos81].

We therefore define a set of nodes $\mathcal{N}$, each referring to a MANET member. Each node maintains a set $N(m) = \{n_0, \ldots, n_i\}$, where $m \in \mathcal{N}$, representing all current neighbors gathered through AODV control messages, where $0 \leq i < |N(m)|$. Each node m stores a time-related queue $O_m = \langle o_0, \ldots, o_j \rangle$ representing the overhead of node m. Values in O_m are stored for a defined time span t and popped thereafter. We set t to one second. According to that, the entry o_j at index j represents the last element when the second is reached. Each element $o \in O_m$ represents the packet size in Bps. Considered packets are AODV routing management, ARP overhead, and data flows between participants. In case the addressing scheme of any packet is unicast, the packet size $o \in O_m$ is composed of the actual packet size including all headers, meaning UDP, IP, and MAC headers. The measured overhead o of broadcast packets is the packet size in bytes including all headers. Utilization recording also considers AODV specific routing overhead.

To compute the current utilization U_m, where $m \in \mathcal{N}$, m adds up all overhead in this timespan of the connections to all neighbors.

$$u_m = \Big(\sum_{n \in N(m)} \sum_{o \in O_n} o \Big) \tag{4.1}$$

The utilization represents the throughput during the timespan t. Given this value, the next step is to determine the residual capacities.

Capacity Estimation

Various circumstances influence the quality of connections when speaking of wireless transmissions. Communication via a single channel increases

radio wave overlaps, resulting in disrupted frames. Physical obstacles, such as rocks or walls among others, and other electromagnetic influences also reduce the connection strength. This mutual interference becomes even more relevant in multi-hop networks, such as MANETs. Mobile base stations, in contrast, cover a certain region providing connectivity for their end users. This infrastructure manages dedicated channel access per registered device providing methods, such as TDMA, FDMA [Ole16, Chapter 3], Orthogonal Frequency Division Multiple Access (OFDMA) [YA06] among others. The GSM networks used to provide connectivity in the past were based on TDMA and FDMA [Kuk18].

MANETs must manage interference-free channel access in a self-organized way as no central instance takes over this task. Considering multi-hop transmissions, having one antenna for receptions and transmissions for each participant, intermediate route members require double the data rate as they are occupied twice. According to the research of Chung et al. [NL07], MANET nodes utilize 5 times more, depending on the number of route participants. These findings are obtained in placing eight nodes after another, each connected exclusively with their predecessor and successor, if participants are neither source nor destination of the route chain. Their results also show that with 5 and less participants the throughput of a path is $\frac{1}{l}$ where l is the number of nodes in a route. Ping Chung Ng used the network simulator NS2.1b9 as simulation environment [IH10]. We obtained similar data rate reductions with the same configuration using the network simulation environment OMNet++ [Var10].

The next step is to include the experienced overhead volume in the capacity-based routing strategy of AODV with the objective not to elect nodes as route participants if upcoming flows exceed their transmission capacities. As a recap, the desired outcome of our AODV-CBR should be a more well-distributed transmission utilization in the MANET if low data rate flows are continuously injected. For that, our extension reinforces the individual treatment of RREQs and RREPs. The more utilization is present on a node, the more delayed an arbitrary RREQ will be. The route lengths of future routes must be defined to determine if transmission capacity is present. Standard AODV increments path lengths during RREQs starting with the origin till the target node is reached. Path participants use this information to set the number of hops to the target in their routing tables when receiving the corresponding RREP.

AODV-CBR leverages this information in using a dedicated field for path lengths in their customized RREPs. Path participants maintain a vari-

able named path length $\texttt{pl}_n$ representing the number of hops of the longest path they are part of. A node n compares $\texttt{pl}_n$ with the length of a newly defined route when receiving a RREP and only updates $\texttt{pl}_n$ if the new route is longer. The configured data rate on the network interface as well as the data rate requirement of the upcoming flow are necessary parameters for determining the residual capacity of nodes. The following equation computes the residual data rate $\texttt{rd}_n$ of a node $n \in \mathcal{N}$ considering the capacity of c_n and the $\texttt{pl}_n$.

$$\texttt{rd}_n = \frac{c_n}{\texttt{pl}_n} \cdot 0.95 \tag{4.2}$$

The constant value 0.95 represents unforeseen environmental disturbances, such as bit errors, collisions, or ACK timeouts.

Suppose node n receives a RREQ, it first determines the demanded data rate of the upcoming flow `df`. We use the DSCP field of the IP header to provide this information for MANET members [Alm92].

$$\texttt{ad}_n = \frac{1}{\texttt{rd}_n} \cdot \Big(\texttt{rd}_n - u_n - \texttt{df}\Big) \tag{4.3}$$

The result $\texttt{ad}_n$ represents the current available data rate of node n, which is the relation between the theoretical capacity of n based on the path length and the residual capacity. Each node decides based on $\texttt{ad}_n$ how to treat a received RREQ. A RREQ can either be instantly *send*, *delay*, or *drop* which is determined with the following distinction:

$$0.99 \leq \texttt{ad}_n \leq 1.00 \implies \mathit{send} \tag{4.4}$$
$$0.99 > \texttt{ad}_n > 0.00 \implies \mathit{delay} \tag{4.5}$$
$$0.00 \geq \texttt{ad}_n < 0.00 \implies \mathit{drop} \tag{4.6}$$

Given this treatment, nodes forward RREQs instantly if almost no utilization is present (Equation 4.4). In case the overhead represented by $\texttt{ad}_n$ is between 0.99 and 0.00, RREQs are delayed according to Function 4.7. Otherwise, if node n is completely utilized, meaning $\texttt{ad}_n \leq 0.00$, all received RREQs are dropped.

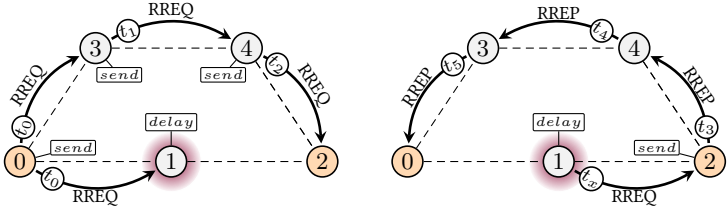
$$\mathit{delay}(n) = \texttt{ptt}_n \cdot \texttt{pl}_n \cdot 2 \cdot (1 - \texttt{ad}_n) \tag{4.7}$$

Equation 4.7 shows how node n computes the delay of a received RREQ. The packet traversal time $\texttt{ptt}_n$ represents the latency in $msec$ of a packet

traversing between two directly connected nodes. The constant $\texttt{ptt}_n$ represents the average one-hop traversing time of each received routing packet. Each node maintains a queue containing the traversing time of all received routing packets within the last second. The transmission time of the actual received request is used if the queue is empty. The delay of n represents the round trip time of the path which depends on $\texttt{ptt}_n$ and the path length. With $(1 - \texttt{ad}_n)$, the delay also depends on the current overhead of n and is weighted more heavily. It has to be mentioned that the result of Equation 4.7 can be higher than the defined rebroadcast timeout of AODV, resulting in a new RREQ by the origin before n restarts the RREQ of the ongoing pathfinding process. Node n would delete the newly received RREQ.

Revisiting the introduced route construction from origin 0 to target 2, previously introduced with standard AODV, we now discuss the pathfinding concept with Figure 4.5 applying AODV-CBR. Each node permanently tracks the current utilization. The utilization of nodes is illustrated with red-shaded surroundings. Rounded rectangles linked to nodes show the treatment of received RREQs.

The RREQ of node 0 reaches all direct neighbors which provides the required data rate. Both receiving nodes compute $\texttt{ad}_1$ and $\texttt{ad}_3$. Node 3 forwards the RREQ instantly as no utilization is present. However, node 1 is computing the delay based on Equation 4.7. The time t_x at node 1 when forwarding the received RREQ represents the result of Equation $delay(1)$, where $x \geq 2$. No further node is utilized except for 1, which is why each member decides to instantly forward the RREQ after computing the residual



(a) AODV-CBR path construction process from 0 to 2 delaying RREQ at 1 due to measured utilization.

(b) RREP process using reverse path with least utilizaiton via nodes 3 and 4.

Figure 4.5: Route construction with AODV-CBR considering utilization of node 1 (red-shaded).

capacity. The target receives the request from node 4 (Figure 4.5b) and responds as node 1 still delays the previously received RREQ. The RREP traverses along the reverse route the same way as defined in pure AODV.

This, in theory, provokes alternative routes and a more evenly distributed throughput, as utilization measurements avoid stressed network segments.

4.1.3 Evaluation

With this evaluation, we want to answer the question if even network load distribution in MANETs leads to more throughput compared to route constructions that focus on shortest paths. We define two similar scenarios where multiple MANET members transmit data. The data rates of transmissions are low to provoke an evenly network load distribution, as participants increase their utilization only slightly. The following throughput measurements compare AODV width AODV-CBR. Therefore, we randomly constructed 10 MANETs of 39 nodes, each equipped with a 2 $Mbps$ wireless interface using CSMA/CA [IEEE802.11]. These simulation parameters generate topologies where each node is connected to several neighbors in close vicinity. Tables 4.2a and 4.2b show applied parameters of both scenarios. Participants move between $[1.5\ m/sec, 3.0\ m/sec]$ based on the RWP [AWS06].

Scenario 1 injects 18 transmissions in the MANETs originated by 9 different nodes. Different source and target pairs are purposely selected to not overload sending and receiving nodes. Also, equal origin and targets would result in instant data delivery skipping the pathfinding process, as origins already store an entry towards the destination in their routing table. To reinforce path construction even more, origins start their transmission after the previous pathfinding process has been finished. Both AODV and AODV-CBR are evaluated under the same conditions meaning general data transmission starts at the same time based on equal MANET topologies. Therefore, random number generators are used to ensure that the simulation time a node starts transmitting data varies only slightly. To average out measurement errors, each run was executed 10 times with different seeds. We used OMNet++ [Var10] and extended the AODV implementation with all AODV-CBR properties.

Figures 4.6a and 4.6b show the generated throughput of both scenarios. Figure 4.6a shows throughput results with 18 successively injected transmissions. Simulation time between red dashed lines illustrates time periods where origins start flows with 10 seconds waiting time in between. Both

Table 4.2: Parameter setting of both scenarios.

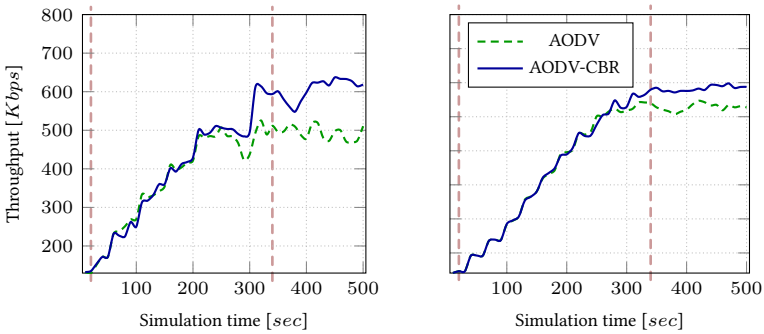
(a) Setting scenario 1

Parameter	Value
Origins	18
Targets	9
Bitrate flow 1-8	70 $Kbps$
Bitrate flow 9-18	40 $Kbps$

(b) Setting scenario 2

Parameter	Value
Origins	24
Targets	12
Bitrate flow 1-12	47 $Kbps$
Bitrate flow 13-24	26 $Kbps$

routing protocols increase the throughputs by almost the same amounts as transmissions start at almost the same times. This also verifies that path construction using AODV-CBR works out. Having a closer look at the simulation time 300 sec of Figure 4.6a, the throughput of AODV decreases and later oscillates around 500 $Kbps$ whereas the throughput of AODV-CBR remains the same at first and then exceeds 600 $Kbps$. Around simulation time 320 sec, two last transmissions are injected experiencing a noteworthy throughout increase. Thereafter, throughputs oscillate and also show slight decreases which might be caused by mutual interference. In total, AODV-CBR outperforms pure AODV, meaning alternative routes are found and



(a) Throughput of AODV and AODV-CBRs while successively injecting 18 transmissions.

(b) Throughput of AODV and AODV-CBRs while successively injecting 24 transmissions.

Figure 4.6: Comparing throughput of AODV and AODV-CBR while increasing network load with 18 transmissions (left) and 24 transmissions (right).

used, which results in more throughput.

Scenario 2 (Figure 4.6b) shows similar results compared to Scenario 1. Until simulation time 300 sec, AODV and AODV-CBR exhibit almost the same delivery rates. Both protocols increase throughputs equally and last injected transmissions also have a negligible impact as throughput of both protocols differs only slightly. AODV-CBR shows a slight increase with the last three transmissions, whereas AODV seems to have no capacities left to deliver additional flows. Thereafter until the end of the simulation, AODV-CBR shows a higher throughput compared to AODV.

Including participants' traffic overhead and transmission requirements during path construction increases the throughput. Our findings also show that even path distribution exploits available capacities more efficiently, which is promising, referring to RQ 4. There, several pathfinding techniques exclusively try to distribute multiple paths aiming among other for minimum transmission utilization.

4.1.4 Findings

During the simulations, we observed frequent replies, meaning MANET members who are not the requested destination responded with RREPs, as they store a routing entry that points to the requested target. AODV obtains topology knowledge and fills nodes' routing tables if available. For instance, routing entries pointing to originators of RREQ are generated if not present or if the sequence number of the originator in RREQ is higher compared to the entry in the routing table. A member also inserts a routing entry for each predecessor that sent a RREP if no entry exists or the destination sequence number of the routing entry is equal or lower compared to the sequence number of the RREP. Furthermore, the routing table will also be extended with the next hop of the RREP as destination is not present so far. The originator address of the RREP is also stored in the routing table if the corresponding sequence number is equal to or higher than the one in the RREP. All routing information increases the network knowledge and members more frequently store a route entry to a requested destination in their routing table. Participants of these constructed routes might already forward multiple transmissions.

MANET members manage channel access in a self-organized way applying CSMA/CA. Nodes might be occupied even though no active reception or transmission is ongoing because, for instance, an ongoing transmission is interfering. Also, neighboring nodes occupy the channel, which

forces surrounding nodes to wait until the channel is free. AODV-CBR only records layer 3 packets of the network stack and does not consider lower layers. Packets must be addressed to them, otherwise, membership checks on the MAC layer drop all frames not desired for this node. This occurs frequently as radio waves propagate omnidirectionally. Nodes do not record this type of utilization as these frames do not reach the network layer. Capacity computations do not consider passive occupation which leads to paths not being able to carry the data rate required for the new transmission. It is worth mentioning that we experienced a noticeable loss rate during runs in which layer 3 capacity measurement played a major role.

In total, utilization information of nodes and data rate demands of flows during path computation lead to more throughput of MANETs as routing can distribute paths more evenly. Path computation techniques, as introduced in RQ 4, can be applied to exploit the capacity more efficiently. We also detected that layer 3 only provides incomplete utilization information for members. This often resulted in inaccurately computed transmission capacities of nodes receiving a RREQ as passive interference is not tracked on the network layer. Even though transmission capacities of route participants seem sufficient for requested flows, surrounding path participants of different transmissions are disturbed which is known as mutual interference. This results in dropped data. MANET members do not know if their transmission disturbs an ongoing transmission of a neighboring node. The lack of a complete and up-to-date topology knowledge including ongoing transmissions results in mutual interference and congestions.

4.2 CONTROLLER-EQUIPPED MODEL

First, we present a SDN-aligned network architecture comprising data structures and formal definitions. This includes architecture components, message types, and their data fields. We extend this architecture framework and its definitions in each research question. We secondly analyze the impact of data transmissions in wireless multi-hop networks, such as MANETs, to provide the basic knowledge for the subsequent node-based utilization recording concept. A network overhead prediction concept is proposed, using recorded node utilization as a baseline. The proposed evaluations focus on the overhead and the computation accuracy.

Table 4.3: Global variables and parameters used in several RQs.

Properties	Belonging	Description
n	$n \in \mathcal{N}$	Unique id of a MANET participant.
o^f	$o \in \mathcal{N}, f \in \mathcal{F}$	Origin node of a flow.
t^f	$t \in \mathcal{N}, f \in \mathcal{F}$	Target node of a flow.
p^f	$p \subseteq \mathcal{N}, f \in \mathcal{F}$	Path as sequence of nodes of a flow.

4.2.1 Framework Components

This section maps the MANET to a directed graph and further introduces the Central-controlled MANET (Cc-MANET) architecture and the message types.

MANET Notations

The MANET is represented as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L}, \mathcal{F})$ comprising nodes, connections, and transmissions, also known as flows. Parameters and variables of this RQ and beyond (global) are listed in Table 4.3. Red-colored rows depict parameters, which are newly defined in the RQ of this chapter. Detailed formal definitions and constraints are introduced in this section.

Nodes ($\mathcal{N}$): All $n \in \mathcal{N}$ participating in the MANET. Each node n also serves as unique identification (id), making it possible to distinguish them from each other. IP addresses, for instance, could serve as identifiers. Each node is identified by natural numbers $\mathbb{N} \cup \{0\}$ throughout this thesis.

Links ($\mathcal{L}$): Tuples of nodes (n, m), where $n, m \in \mathcal{N}$. Any $(n, m) \in \mathcal{L}$ comprises a source node n and a sink m. The link (n, m) exists if the reception power of m, caused by the transmission of n, is higher compared to the sensitivity threshold of m. This verifies that m can receive and decode frames of n. Each $l \in \mathcal{L}$ has the constant parameter transmission capacity c_l and the variable utilization u_l. The first represents the amount of transmission overhead l can provide. Values above c_l result in incomplete data delivery due to dropped frames along the path. The latter represents all overhead comprising transmissions, receptions, interferences, and MAC-related occupations of l. A link is completely utilized and cannot serve any additional flow if $u_l = c_l$. The properties utilization u_l and link capacity c_l are introduced in detail in Section 4.2.4. $\mathcal{G}$ is defined as directed graph

without loops, meaning $\forall(n, m) \in \mathcal{L} : n \neq m$.

Flows ($\mathcal{F}$): Our research focuses on flow transmissions within the MANET. A flow characterizes a source and a destination address, a source and a destination port number, and the used transport protocol, in case of IPv4 is used. Using IPv6 defines a source address, a destination address, and a flow label field as key identifiers [DWJ15, Section 1.4]. We stick to the IPv6 definition as we only consider a single flow transmission between a source and a destination not focusing on a specific transport protocol.

All occurring transmissions are stored in the set of flows $\mathcal{F}$. Any flow f has an origin o^f and a target t^f node, defined as $f_{(o,t)}$, where $o, t \in \mathcal{N}$. $f_{(o,t)}$ is also the unique identifier of this flow (flow label), following that no flow with equal origin and target pair exists twice in $\mathcal{F}$. In subsequent sections f always represents an $f_{(o,t)}$ of $\mathcal{F}$. Each flow has the property path p^f. Each path is a loopless-ordered sequence $p^f = \langle n_0, \ldots, n_k \rangle$, where $k = |p^f| - 1$ and $n \in \mathcal{N}$. Each p^f is defined by the following constraints:

$$
\begin{aligned}
& n_0 = o^f \wedge n_k = t^f && , n_0, n_k \in p^f \\
& i \neq j \implies n_i \neq n_j && , \forall n_i, n_j \in p^f \\
& (n_i, n_{i+1}) \in \mathcal{L} && , \forall n_i \in p^f
\end{aligned}
$$

This defines that the first and last node of the path equals the origin and target, respectively. Also, each node is at maximum once included in a path. A path must be complete. This means that the node n_i and the directly following node n_{i+1} of p^f must be in the set of all links $\mathcal{L}$. A flow f also has the property demand d^f representing the required data rate which must meet the capacity of a chosen link, carrying the flow f.

Network Architecture

This chapter introduces the Cc-MANET architecture and embeds all mentioned notation in the architecture and message fields. This structure forms the basic architecture applied and extended throughout this thesis.

A bird's view comprising complete MANET topology knowledge, currently deployed flows as well as their required data rates are required to route multiple flows in this capacity-restricted network considering mutual interference of multiple flows. Having a controller reachable for each MANET member via a dedicated channel achieves these requirements. MANET participants could provide connection information and current data transmissions. Paths could be computed considering actual network overhead,

having the controller equipped with a routing application. There exist application scenarios in which the controller-equipped MANET architecture improves the conditions, referring to Section 1.4. Speaking of the military use case, introduced in Section 1.4.1, the controller could be placed on the headquater or on a command vehicle, providing a wide range low data rate link, covering all MANET members. Referring to the FANET scenario, the controller could be placed on the ground station to cover all UAVs. Exploiting this uplink connection for data delivery is not viable, keeping in mind the characteristics of these dedicated links, which is why data forwarding remains within the MANET. Overhead concerning route requests, responses, as well as topology information is transmitted via the uplink channel to the controller. We call this architecture Cc-MANET as a controller is reachable by each node via a dedicated link. MANET members continuously exchange neighborhood information and forward local topology knowledge to the controller via the so-called uplink channel. The controller computes and sends requested routes back to the MANET where data delivery physically takes place.

Figure 4.7 introduces the Cc-MANET architecture and messages, where the controller is externally deployed. Based on that, each MANET participant is equipped with a dedicated uplink to reach the controller. Gray, rounded rectangles represent entities (controller and MANET) that are physically partitioned. Rectangles with blue color illustrate message types. Dotted arrows represent message flows to the pointed direction, using the uplink if arrows connect the MANET with the controller.

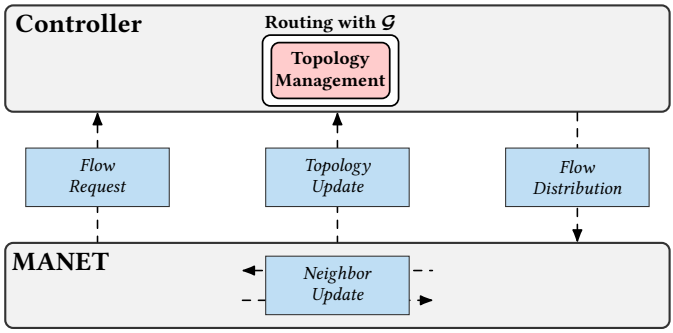


Figure 4.7: Cc-MANET architecture comprising controller and MANET entities as well as message types combined with arrows pointing to the transmission direction.

MANET members periodically broadcast *Neighbor Updates* within the MANET to convey their ids to all participants within coverage range. In particular, each $n \in \mathcal{N}$ broadcasts a *Neighbor Update* and starts counting down a configured timer to repeat this sequence when reaching zero. *Neighbor Updates* contain the id of the source that is used as identification for the receiver node. Participants store a set of links representing their neighbor connections. Each source of these links is a direct MANET neighbor within communication range. Formally described as, any node $n \in \mathcal{N}$ stores $L_n := \{(i,j) \in \mathcal{L} | j = n\}$ and associates each entry with a time stamp when a *Neighbor Update* was received. Ids and time stamps are either inserted if the id is still unknown or only the time stamp of the corresponding id is updated if an entry exists. The runtimes of the *Neighbor Update* intervals are equal on each node, and entries in the list of neighbors are dropped if entries exceed double the update time.

The connection list L_n is maintained continuously and reported to the controller at determined intervals as well. *Topology Update* messages are used to deliver this information via the uplink. Similar to *Neighbor Updates*, members transmit *Topology Updates* periodically to the controller, each having configured the same interval. In particular, each MANET member $n \in \mathcal{N}$ conveys the current set of neighbor connections L_n to the controller. Timestamps of each $(j,n) \in L_n$ are checked and links are removed from L_n if n has not received a *Neighbor Update* from j two interval times in a row. The controller constructs the graph $\mathcal{G}$ based on these lists. *Neighbor Updates* and *Topology Updates* are transmitted continuously, also if no data delivery is present.

The update concept for constructing a topology representation on the controller is called Proactive Topology Update Process (Pro-CeTUP) since nodes update topology information at determined intervals. This neighborhood information is used by the controller to construct the snapshot, representing the current MANET topology. The controller uses the graph to compute requested routes. Graph maintenance and shortest path routing are provided with the component **Topology Management**.

MANET members send *Flow Requests* to the controller if a data transmission is desired. This message includes o^f, which is the origin of the requested flow f and also the sender of the *Flow Request*. The target t^f represents the destination of the requested route. *Flow Requests* trigger the controller to compute paths from origins to their destinations with Dijkstra [Dij59] at runtime, applying an edge weight of 1 for $l \in \mathcal{L}$ at the moment. Computation for further network members is omitted when the

shortest path to the desired destination node has been found. Dijkstra computes the shortest path for each node if either the destination is the last visited node or no path in $\mathcal{G}$ exists. When route computation is finished, the controller sends a *Flow Distribution* containing the path p^f via the uplink. *Flow Distributions* are unicast messages addressed to the originator of the previously send *Flow Request*.

When receiving a *Flow Distribution* via the uplink channel, the originator of this flow first inserts the next hop to t^f in its routing table and thereafter attaches p^f to the first data frame. This frame is routed to the next hop according to p^f. Each path participant encapsulates this first data frame, inserts or updates the next hop towards the destination in its routing table based on the conveyed information and thereafter forwards the frame consisting of the path to the next hop. All subsequent data frames are routed based on the previously updated routing table information. The following sections introduce the utilization recording and computation concept which is embedded in the Cc-MANET architecture. The next section introduces how multi-hop networks influence wireless transmissions, which is considered by the utilization measuring concept.

Special Cases and Exceptions

It happens that control messages do not arrive at the desired destinations. An origin of the unicast *Flow Request* receives an ACK as a response for the successful message delivery to the controller. The same *Flow Request* is retransmitted in case no ACK has been received. The same happens if *Flow Distribution* is not delivered to the node that previously transmitted the *Flow Request*.

Nodes transmit a new *Flow Request* via the uplink to the controller in case path participants lose connections to the next hop or the target node. The controller computes the path to the destination. However, the broken link is still present as the corresponding *Topology Update* has not been delivered yet. This results in a remaining message loop as the controller computes the same path consisting of the broken link until nodes update the outdated network segment. This message overhead can be avoided if nodes, detect link breaks, delete the lost connection, and send the *Topology Update* via the uplink to the controller. Also, nodes that are not part of the route anymore remove the unused routing entry after 3 seconds.

4.2.2 Wireless Multi-hop Transmissions

Unlike commercial wireless architectures where mobile carrier networks provide one-hop base station connectivity for network access, MANETs construct multi-hop routes carrying data traffic to destinations. Paths often contain more than origin and target nodes. Referring to assumptions and preliminaries in Section 1.5, route participants must consider channel access of neighboring nodes also forwarding data for the same flow, as the single antenna is only capable of either transmitting or receiving a frame. The applied frequency band is entirely occupied for all neighbors that receive a transmission of a MANET member as no frequency or time multiplexing is assumed. Radio waves of simultaneously initiated transmissions are subject to overlap if nodes are within communication ranges. The represented symbols are disrupted and information is useless for decoding purposes. Even worse, two or more overlapping signals in theory extend their ranges as the amplitudes of overlapping radio waves increase [MW15]. Path participants in multi-hop routes must stay idle if their desired transmission would result in overlapping signals. This delays data delivery and decreases throughput rates of the flow they transmit.

The treatment of received radio waves depends on the reception power of signals. The power of signals decreases logarithmically related to the distance, meaning the wider the distance between sender and transmitter the lower the reception power. Antennas experience noise during reception, which is the sum of all signals observed on the channel in relation to the currently received one plus the receiver's noise [TMB01].

For successful reception, antennas must also be configured with appropriate sensitivities. The sensitivity represents a threshold, indicating whether reception is possible or not. Signal powers above sensitivity are receivable and further processed whereas reception powers below this threshold are classified as interference or noise depending on the energy detection. If reception is not possible, antennas further determine if signals are above energy detection, meaning signals are recognized and potentially interfere upcoming receptions or transmissions. If the signal is below energy detection, antennas classify radio waves as noise, otherwise as interfering signals. [Sey05]

Theoretical research related to wireless transmissions often classifies antenna behavior in different ranges [DLV04; XGB02]. The transmission range specifies the coverage range in which potential destinations are able to receive a frame. The interference range describes the area beyond the transmission range. Nodes, which are out of transmission range but inside

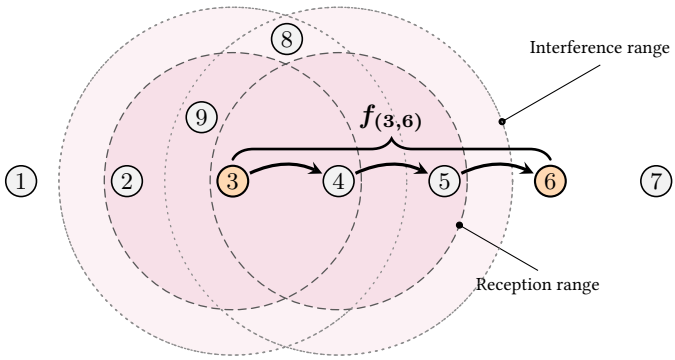


Figure 4.8: Radio signal propagation exemplary of nodes 3 and 4 including transmission and interference ranges when transmitting flow $f_{(3,6)}$ via nodes 4 and 5.

interference range of a sending node are not able to receive further frames at this time as overlapping signals distort the reception. The sensing range is understood as the area in which a node recognizes any signal in case this node desires to also start a transmission.

Since all MANET nodes have equal radio configurations, all MANET members theoretically use the same powers to radiate radio waves and all receivers are also set to the same energy and sensitivity thresholds. This concludes, that each node behaves the same when detecting a frame. This means that each node goes through the same process when deciding whether a frame can be received, is interfering, or is classified as noise, which depends on the distance between the sender and the receiver of a transmission. Because of that, transmission and interference ranges can be used to represent the constant parameters sensitivity threshold and energy detection.

Figure 4.8 shows these ranges during a data transmission between MANET nodes, arranged as a chain. We left out interference and transmission ranges of all nodes except of 3 and 4 as visualization of further rangers would appear confusing and unclear. At the beginning, the origin node 4 transmits the data of flow $f_{(3,6)}$ to the next hop. Nodes $2, 4, 9$, and 8 are inside transmission or interference ranges with respect to this transmission. All mentioned nodes are occupied when receiving frames of this flow. Nodes in interference range are not able to further process them. More concrete nodes within the interference range are utilized for the time the

frame occupies the channel and are not able to either receive or transmit another frame. Node 4 forwards this frame to 5 also occupying all nodes in transmission and interference range. This prevents node 3 from starting the next frame. Node 3 must wait until 4 has finished forwarding the previously received frame. Node 5 delivers the previously received frame to target 6, forcing 4 to delay the next frame to node 5. Also, 3 must stay idle for the transmission period of 5 since any further frame delivered to 4 would be received disrupted as, node 4 is within transmission range of 3 and 5, respectively.

These dead waiting periods decrease the throughput of the entire route which decreases even more with additional route participants. Simulations as well as constructed routes with laptops in real environmental conditions prove this behavior [RGM18]. Their setup comprises isolated routes of up to 6 participants forming a linear chain, in which each node is connected to its predecessor and successor except for the origin and target nodes. Nodes in the experimental measurement and simulation are equipped with equal wireless configurations with respect to bit rate, wireless coverage ranges, and hardware equipment, among other parameters. The measured throughput is shown in Plot 4.9. It clearly proves that the throughput decreases enormously in both scenarios, which is caused by the shared channel usage of all chain members. Chung et al. [NL07] also experienced this throughput behavior during their research.

We extended the chain scenario with an additional flow to analyze dropped frames since our research considers multiple transmissions in MANETs.

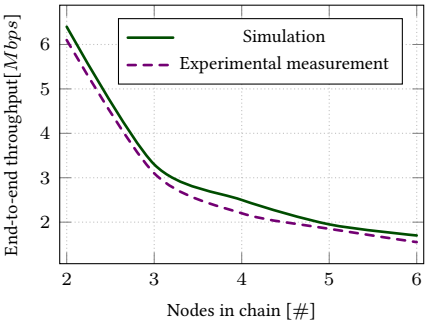


Figure 4.9: Measured end-to-end throughput of different chain lengths by Rezaei et al. [RGM18].

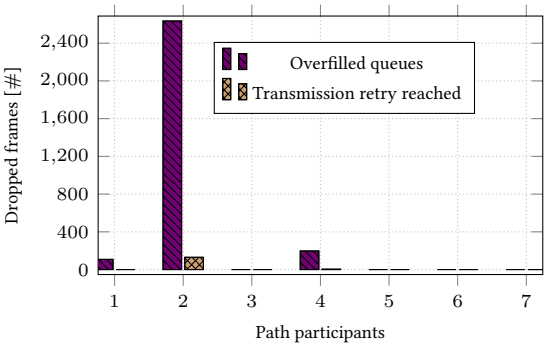


Figure 4.10: Dropped frames of active route participants during data delivery of flows $f_{(1,3)}$ and $f_{(4,7)}$ according to Figure 4.8.

Therefore, we constructed a MANET like in Figure 4.8. Two flows $f_{(1,3)}$ and $f_{(4,7)}$ are configured. MANET members are configured with an IEEE 802.11 MAC interface, operating in mode g with 2 $Mbps$. We first started each packet flow isolated while increasing the throughput to reach the transmission capacity boundaries of each path. Next, we investigate performance parameters while injecting both flows during one simulation scenario.

Results of Figure 4.10 focus on the packet drop rate on the MAC layer to investigate how multi-flow data delivery affects the participating members individually. In general, the number of delivered frames to targets 3 and 7 in relation to the entire sent payload drops significantly. Referring to flow $f_{(4,7)}$, 92.80% of all data transmissions reached the target. Even worse, the packet delivery fraction of $f_{(1,3)}$ only reaches 35.61%. This illustrates that transmission capacities of MANET participants are insufficient to carry both flows. The following results show this cause based on dropped frame rates.

Frame sequences between all intermediate route participants experience delays as nodes compete for free slots with predecessor and successor nodes. Especially between nodes 3 and 4, this causes disturbed signals and competing times caused by the MAC protocol. Each sequence starts with the RTS of the source followed by a CTS of the sink if reception was successful. Sources of links then transmit data frames to sinks and receive associated ACKs within a defined timeout period. Delivery attempts are repeated if frame sequences cannot be completed. Results show dropped frames due to overfilled MAC queue as well as dropped frames because of lost connec-

tions during data delivery of $f_{(1,3)}$ and $f_{(4,7)}$. Delivery attempts of RTS signals and data frames run into reached retry limits resulting in dropped data frames. These retries occur mainly at node 2. Frame reception of 2 and initiated transmissions of 4 cause mutual interference. Node 3, for instance, is forced to stay idle and leaves RTS-CTS handshake initiated by 2 unfinished as data transmission of 4 is ongoing. Because of that, the sink node 2 aborts ongoing frame sequences and drops data frames which have already been scheduled for transmissions, as shown in Figure 4.10. Also, further data frames from 1 to 3 are dropped due to the already overfilled MAC queue. Referring to node 4, the desired data transmission interval to 5 delays due to ACKs of 3, which also results in dropped frames, as maximum queue size is reached. Mutually used channel capacity by nodes 3 and 4 forces node 2 to wait until 3 is free for reception, resulting also in an overfilled MAC queue. Node 4 cannot last the desired transmission interval and also drops payload.

In general, route participants disturb data delivery of other flows if the sink of a transmitting connection is in the communication range of a node that transmits frames of another flow. Because of that, the transmission overhead of already deployed flows must be considered when searching for a path for an additional flow.

4.2.3 Measuring Traffic Utilization

This section introduces our applied MAC measurement technique to record accurate and meaningful connection utilization.

Generating usable knowledge about arising utilization in wireless networks requires covering, on the one hand, occupation times related to MAC protocol states and, on the other hand, each signal that corresponds to a protocol and data frame. Findings during Section 4.1.4 highlight how important it is to apply utilization measurements on the MACs layer as these protocols are designed to efficiently access the channel without interference. Thereby, these protocols transfer radios into different states to prevent simultaneous transmissions. Focusing on the MAC layer guarantees complete access to control and data frames forwarded from the upper and lower layers. Packets are handed from the network layer down to the MAC layer before the transmission of these frames is intended. This includes control packets of any type, such as routing or address resolution, and data packets of all applications currently running. All frames are decoded and further processed according to protocol, referring to the process from the

physical layer up to the MAC layer. Processing also includes frame destination checks on the MAC layer. Utilization must be recorded on the MAC layer, since this layer has knowledge of all data transfer and also manages channel access passes. However, the utilization concept must be adapted to each MAC protocol individually.

Applying the right MAC protocol for our utilization measurement purpose depends on multiple requirements. According to the Cc-MANET architecture, minimum configuration management and efficiency should be the major strength of the MAC protocol. CSMA/CA [IEEE802.11] fulfills the best out of both criteria. The contention-aware access method is self-managed, as participants organized interference-free access on their own. Control frame exchange cycles and inter frame time periods handle a controlled channel usage. Comparing slot and frequency multiplexed access methods like TDMA [Rod06, Section 14.7] and FDMA [Ole16, Chapter 3] with CSMA/CA, MANET members must be allocated a dedicated time slot or frequency range for channel access. This requires additional computation overhead and also reduces the throughput of participants proportionally with increasing number of nodes. Spatial reuse of already assigned frequencies and slots can be applied by the controller. On the one hand, this increases the efficiency resulting in higher throughputs per node but on the other hand increases the frequency and slot computations, since those must be recomputed at runtime depending on the mobility of nodes. Furthermore, spatial reuse is error-prone, as nodes having configured same slot times or frequency ranges might move in each other's coverage ranges. CSMA/CA provides spatial reuse by sensing the channel self-managed. In case no transmission is active, nodes start their frame sequence even though data delivery beyond the desired sender and receiver is ongoing.

In the following section, we present a traffic utilization measurement concept for nodes, which aggregates all node's utilizations based on the MAC technique CSMA/CA. Measurements, computations, and evaluations are carried out with the network simulator OMNet++ [Var10].

Node Utilization

Measurements carried out on nodes function as quality assurance for utilization predictions by the controller. The objective of utilization recording is to determine how much transmission capacity is left and if this node is able to carry an additional flow. We leverage the current measured utilization on MANET members to later compare the recorded utilization with the transmission overhead computed by the controller. The subsequent para-

graphs introduce the utilization model integrated into CSMA/CA.

In best-case situations, nodes compete for channel access and start the desired frame sequence if access is granted. If one does not consider DIFSs and backoff times, the NAV represents the total time nodes in transmission range must defer their desired transmissions until the current frame sequence is finished. CSMA/CA control frames include the NAV value in their headers to provide this information for nodes in close vicinity. In reality, nodes lose connection during transmission due to, for instance, mobility and other disturbing transmissions. According to CSMA/CA, introduced in Section 2.4.1, a node in action either transmits, receives, waits Interframe Spaces (IFSs), or backoff periods, which are all part of a frame sequence. Based on that, a node is utilized at any point in time when it is not idle. Idle means that the node is free for transmitting or receiving data. Otherwise, the node is utilized at this time and has no capacity for receiving or transmitting data.

We define u_m as the amount of utilization of node m during the time span s, where $m \in \mathcal{N}$. Measurement periods take place between two consecutive *Topology Updates*. Each node sends *Topology Updates* when the so-called update interval runs out. Members have equal interval durations but differ in their starting points. Because of that, each time span s is set to exactly the time between two *Topology Updates*. We define a sequence of time spans $S = \langle s_0, \ldots, s_{|S|-1} \rangle$ to associate each utilization recording to a certain time span. In the subsequent sections, s always represents a particular $s_n \in S$.

Our utilization recording technique are partitioned into the following types:

- $\mathtt{uo}_m^{k_s}$: Occupation time, which is split into sensing and competing.
- $\mathtt{ut}_m^{j_s}$: Active transmission of control or data frames.
- $\mathtt{ur}_m^{i_s}$: Reception of control or data frames, considering utilization due to interference.

Each recording of these utilization types is separately stored in an ordered sequence, containing all measurements of a time period s. The lengths of these sequences are independent of each other as the number of states and actions of each interface ranges depending on the traffic conditions. For instance $U_t^s(m) = \langle \mathtt{ut}_m^{0_s}, \ldots, \mathtt{ut}_m^{j_s} \rangle$, where $m \in \mathcal{N}$ stores all transmitting time spans between two consecutive *Topology Updates*. The sequences $U_r^s(m)$ and $U_o^s(m)$ store all reception times and occupation times during

a time period s, respectively. The indices i, j, and k uniquely identify each measured record of each utilization type during a time period s. At the beginning of each $s \in S$, each counter starts at 0 and increments for each new record until the next *Topology Update* message is sent. Because of that, each index ranges from 0 to the last entry before the next *Topology Update* is sent by the corresponding node. The counters of each set are defined as follows: $0 \leq i \leq |U_r^s| - 1$, $0 \leq j \leq |U_t^s| - 1$, and $0 \leq k \leq |U_o^s| - 1$.

Source Utilization Recording

This section demonstrates how utilization is recorded and associated with their utilization types if MANET members start a frame sequence. Frame sequences end with an associated ACK frame as a response to the previously sent data frame. A frame sequence refers to a CSMA/CA-defined data frame exchange comprising channel contention process, the frame transmission, and the corresponding ACK response. Because of that, frame sequences only provide unicast transmissions, as broadcast and multicast do not use ACKs.

RTS and CTS are not mandatory in frame sequences and are only applied if the size of the frame reaches a configured threshold [Cro+97]. Skipping this handshake reduces the control frame overhead. Smaller data frames are more often transmitted without RTS and CTS exchange as the probability decreases as signal propagation takes less time. Mutual interference increases during capacity-demanding transmissions as there is less time during which the radios stay idle. As a consequence, we applied RTS and CTS for each data frame independent of the size.

Figure 4.11 shows all recordings when node m starts a frame sequence to a destination node during the update time period $s \in S$. The first line shows the basic frame sequence, whereas successive lines demonstrate if CSMA/CA defined timeouts run out or frame fragmentation is ongoing. The x-axis represents the time between two consecutive *Topology Update* messages. Rectangles depict protocol frames or waiting states of CSMA/CA and are associated with the recorded utilization type via curly braces on top of these rectangles. In particular, rounded rectangles represent frames and angular rectangles waiting periods. The bulk, labelled with Elapsed time, represents the remaining time span of this period s in which CSMA/CA protocol data exchange happens when transmissions are desired.

Each frame sequence starts with the DIFS period followed by the backoff timespan, randomly chosen with the CW. Physical carrier-sense (DIFS) and contention (backoff) are preliminaries of CSMA/CA before accessing the

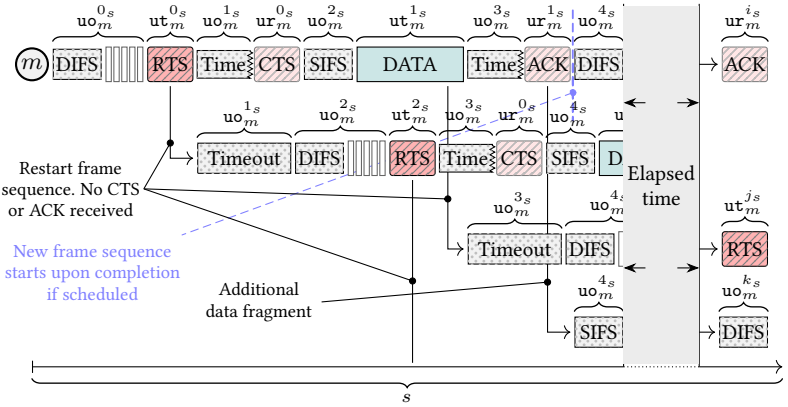


Figure 4.11: Exemplary utilization recording of node m during time span $s \in S$, when starting a unicast frame sequence to deliver data leveraging CSMA/CA.

channel. This is part of the current frame sequence. Receptions of other frames at this time would interrupt and delay the transmission attempt. Because of that, node m is utilized for the times DIFS and backoff, which are recorded in $\mathtt{uo}_m^{0_s}$. In the first line of this figure, no signal detection during sensing is assumed and m transmits a RTS and counts down the CTS timeout, which is defined as the sum of SIFS, slot times, and the duration of transmitting the physical header information, which is defined according to the specifications [IEEE802.11].

Timespans of RTS and the associated timeout are recorded in $\mathtt{ut}_m^{0_s}$ and $\mathtt{uo}_m^{1_s}$, respectively. The timeout measurement period stops when the corresponding CTS is received and decoded on the MAC layer. Otherwise, node m starts an additional transmission attempt with increased CW, as shown in the second line of this figure, which is introduced later. The received CTS time period is classified as reception utilization and stored in $\mathtt{ur}_m^{0_s}$. Based in CSMA/CA, m waits and records the SIFS period in $\mathtt{uo}_m^{2_s}$ and thereafter starts the data transmission and records the time period in $\mathtt{ut}_m^{1_s}$. The ACK timeout is started and the elapsing time is recorded in $\mathtt{uo}_m^{3_s}$ till the time span of this timeout is reached. This would happen if either the transmitted data frame or the awaiting ACK control frame were not received properly, as shown in the third line. Equal to the CTS timeout, node m would start an

additional transmission attempt. The number of retries for lost RTS and ACK frames is set to 7 per default. With respect to the first line, node m completes the frame sequence and records the received ACK frame. Frame sequences can be split into several fragments, which depend on the size. In that case, node m only counts down the SIFS period and thereafter starts with the first data fragment, shown with the fourth line. Going on with line 1, node m starts an additional transmission attempt immediately after the previous one has been completed. In doing so, node m starts the DIFS period and records the overhead in $\mathfrak{uo}_m^{4_s}$.

Suppose decoding of the RTS or data frame signal fails since the transmission of node m was interfered with another frame. Node m then runs into the CTS timeout as no response arrives, which is covered by the utilization recording technique, as shown with the second line. Next, node m behaves according to CSMA/CA protocol and restarts the frame sequence. In particular, after reaching the end of the CTS timeout m restarts DIFS and backoff time spans. However, CW is now doubled, which most likely increases the backoff period. Recording considers additional DIFS and backoff periods, increasing the overall recorded occupation utilization share. RTS must also be transmitted once more, resulting in a newly started CTS timeout period depicted with the vertical line below RTS in the second line. The following procedure equals the first line. Node m starts and records the CTS timer until the frame will successfully be decoded. After recording the reception overhead of the CTS frame, node m starts the SIFS period, followed by the data frame. Node m thereafter starts and records the ACK timer after finishing the data frame transmission, which is not shown.

The third line depicts the fragmentation procedure of CSMA/CA if the data must be split into several chunks. IPv4 packets arrive fragmented at the MAC layer if packets exceed a maximum size. The Maximum Transmission Unit (MTU) defines the threshold, which is set to $1500B$ per default. IEEE 802.11 also implements a fragmentation threshold, which applies if the MTU is higher [Kim+05]. Fragments of packets are identified in the IPv4 header, classifying fragment identification and position of the current fragment compared to the entire packet size [Pos81]. In this case, node m repetitively transmits frame fragments until all chunks have been delivered. With respect to the third line, utilization recoding is ongoing during the entire fragmentation process, starting with SIFS recorded in $\mathfrak{uo}_m^{4_s}$, followed by data, and ending with ACK, which is not further shown in this figure.

When a time period s elapses, m stores all occurred utilization periods associated with their types in $U_r^s(m)$ for recorded utilization, $U_t^s(m)$

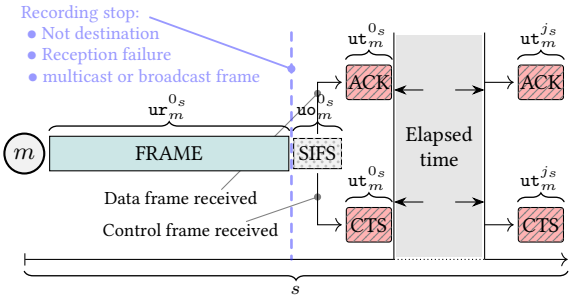


Figure 4.12: Exemplary utilization recording of node m when starting reception of a frame leveraging CSMA/CA.

for transmitted utilization, and $U_o^s(m)$ for occupied utilization, where $i = |U_r^s(m)| - 1$, $j = |U_t^s(m)| - 1$ and $k = |U_o^s(m)| - 1$.

Our recording strategy also supports multicast and broadcast transmissions. This reduces the data delivery process to competing and raw data propagation, skipping RTS-CTS handshake and ACK response. Suppose node m starts a multicast or broadcast transmission. CSMA/CA protocol overhead is then reduced to DIFS and backoff waiting times to provide fairness. Data frames start immediately after these competing periods end.

Sink Utilization Recording

Figure 4.12 exemplary shows all utilization recordings of node $m \in \mathcal{N}$ when receiving a frame during time period $s \in S$. According to this figure, node m records the time of reception and stores the time period in $\mathtt{ur}_m^{0_s}$. Reception ends if processing leads to disrupted data or if the MAC destination check fails, meaning m is not the desired receiver of this frame. Node m determines if the current frame is receivable. Several situations lead to a reception failure. The reception power is too low to decode a frame correctly. Sink nodes receive signals from multiple transmissions at the same time. Utilization recording stops in case of a reception failure.

Otherwise, the address resolver of the MAC protocol decides if a response is desired, resulting in ongoing utilization recording. multicast and broadcast addresses per definition do not follow a further response strategy, which is why utilization measurement stops at this point, as shown with the blue description in the figure.

SIFS recording in $\mathtt{uo}_m^{0_s}$ is performed if node m is the destination. The now decoded frame contains either control information (CTS) or data. Node m starts recording the response of either an ACK or CTS frame in $\mathtt{ut}_m^{0_s}$.

Utilization recordings from the sink's point of view follow this procedure until the update period s elapses. The index j stands for the size of the sequence $U_t^s(m)$, which represents the last utilization record of this sequence during the time period s.

At runtime, nodes transmit and receive multiple frames during one period. Recordings during sending and receiving take place based on proposed techniques during s. Furthermore, each node stores measurements in the corresponding sequence. For simplicity, the proposed examples in Figures 4.11 and 4.12 show recording techniques during a time period s, either explicitly during sending or receiving. In action, the utilization measurement concept inserts sending and recording entries in the sequences depending on the time-driven occurrences. For instance, node m attempts to transmit a frame and starts DIFS and backoff time span. Next, node m stores the recording in $\mathtt{uo}_m^{n_s}$ and inserts the result in $U_o^s(m)$. Suppose m receives a frame during backoff. Node m defers the desired transmission and starts receiving the current frame. Next, node m also stores the recording in $U_r^s(m)$. Suppose the received frame is control information and addressed to m. Node m starts recoding SIFS with $\mathtt{uo}_m^{n+1_s}$ and stores the result in $U_o^s(m)$. Other utilization types behave equally and chronologically store recordings either when sending or receiving in the sequences.

Recording Interruptions

Interference or reception of data also occurs during frame sequences. Suppose m recognizes a transmission during DIFS period. Node m starts the reception utilization recording process depicted in Figure 4.12, depending on the type of frame. Since the RTS of an own intended frame sequence has not been sent, m would respond with an ACK or CTS and record the occurred utilization in case m is the destination of the received frame. Thereafter, m starts the DIFS timespan again after going through the reception process. Node m behaves the same regarding the reception and recording process in case m senses transmission signals during backoff period. After reception recording, m restarts and records the DIFS period and afterward continues counting down the backoff starting at the same position when interrupting reception has been received.

Interruptions during CTS and ACK handshakes are rather rare, as MA-

NET members within communication ranges of source and sink stay idle for the duration of this frame sequence, according to NAV . If RTS or ACK timers run out, the handshake or the desired data exchange is repeated.

Gathering Utilization Records

Utilization entries must be aggregated for a defined time span to represent the utilization ratio to each other after having the recorded time periods of all utilization types at hand. Another motivation is to figure out which utilization types require which amount of capacity in different network load situations. And overall, utilization must represent the share of a MANET member's entire capacity to accomplish network overhead-aware routing by the controller.

MANET members periodically transmit their sub-topology via *Topology Updates* to the controller using the uplink channel. This topology update interval is set to the same length for each member, whereas the interval start time of each node differs. In order to separate and process all recorded utilization during s, we aggregate all recordings separately during this controller update period, as shown in Equations 4.8.

$$\mathtt{ut}_m^s = \sum_{j \in U_t^s(m)} j \quad \mathtt{ur}_m^s = \sum_{i \in U_r^s(m)} i \quad \mathtt{uo}_m^s = \sum_{k \in U_o^s(m)} k \quad , m \in \mathcal{N}, s \in S \tag{4.8}$$

The next step is to convey the recorded traffic utilization of each MANET member to the controller. The measured overhead is used to influence routing decisions. Our objective is to achieve a very accurate utilization prediction technique to determine for sure if an already utilized link can carry an additional flow. This situation is best evaluated in highly utilized network situations. The controller, therefore, only requires the utilization of nodes as a whole and does not rely on what causes the greatest share of a particular utilized link. All individual utilization types are later used during evaluations to view the share of each type during data transmissions.

$$u_m^s = \mathtt{ut}_m^s + \mathtt{ur}_m^s + \mathtt{uo}_m^s, m \in \mathcal{N}, s \in S \tag{4.9}$$

Because of that, each MANET node sums up all recorded utilization types during the elapsed time period s, as shown in Equation 4.9, immediately before transmission of the *Topology Update* message starts.

Table 4.4: Global parameters and variables used in multiple RQs.

Properties	Belonging	Description
n	$n \in \mathcal{N}$	Unique id of a MANET participant.
o^f	$o \in \mathcal{N}, f \in \mathcal{F}$	Origin node of a flow.
t^f	$t \in \mathcal{N}, f \in \mathcal{F}$	Target node of a flow.
p^f	$p \subseteq \mathcal{N}, f \in \mathcal{F}$	Path as sequence of nodes of a flow.
d^f	$f \in \mathcal{F}$	Demanded data rate of a flow.
u_l	$l \in \mathcal{L}$	Current utilization of a link.
c_l	$l \in \mathcal{L}$	Capacity of a link.

4.2.4 Further Framework Components

We enclose all variables affecting the utilization model in combination with the proactive Cc-MANET architecture. Section 4.2.3 introduces how utilization types are summarized and gathered resulting in compound utilization u_m^s, where $s \in S$ defines a recording time period representing a *Topology Update* period and $m \in \mathcal{N}$. Utilization from now on and further chapters will be link-wise, meaning the current transmission overhead is related to a link $l \in \mathcal{L}$ and is defined as u_l^s, where $s \in S$. Formally, $u_n^{s_i}$, where $n \in \mathcal{N}$ and $s_i \in S$ is now mapped to $u_{(m,n)}^{s_i}$, $\forall (m,n) \in \mathcal{L}$. Technically, the utilization u_n^s of an arbitrary node n decreases the transmission to any other neighboring node in communication range equally regarding the residual transmission capacity. In the subsequent sections of this thesis u_l, where $l \in \mathcal{L}$, is always associated with an update period $s \in S$, even if not explicitly mentioned.

Table 4.4 describes additional global properties and parameters defined for this RQ. Again, newly defined parameters are colored-red. The utilization result u_l, where $l \in \mathcal{L}$, carries the most valuable information and extends the *Topology Update* along with c_l, representing the transmission capacity of the link in $Mbps$. We also introduce the parameter d^f representing the required data rate in $Mbps$ of a flow $f \in \mathcal{F}$. This parameter is included in the *Flow Request* to determine which links have sufficient transmission capacities to carry this flow. Basically, the computed transmission rate of an active link must not exceed its capacity.

The Cc-MANET architecture, shown in Figure 4.13, now contains two additional entities to accomplish utilization-aware routing. Starting with the MANET, the element **Link Utilization** comprises traffic overhead mea-

surements introduced in Section 4.2.3. There, participants continuously record all incoming and outgoing traffic including occupied times to accurately determine their current utilization. Nodes gather utilization data, see Equations 4.8 and 4.9, and include it in the upcoming *Topology Update* immediately before nodes transmit these messages to the controller via the uplink channel.

Central routing empowers control of path construction compared to common distributed techniques in MANETs, as the controller has complete authority over path construction and can design traffic flows considering mutual interference. Global knowledge and the **Utilization Model** are used to eliminate over-utilized path settings in the first place. In particular, having current deployed paths and their transmission utilization at hand, allows us to identify and bypass stressed network segments. Therefore, required data rates of flows are included in *Flow Requests* and used for path constructions by the controller. The utilization-based path construction is introduced in the next Section.

4.2.5 Capacity-based Path Computation

The controller applies Dijkstra [Dij59] on graph $\mathcal{G}$ when searching for a path from o^f to t^f. This algorithm searches for a path from an origin node to all other participants in the graph based on the determined link weights. The link weights define the desired characteristic of paths. If one searches for the shortest paths, the link weights must be defined in a way to compute paths with the least number of links. Equal weights focus on the paths with minimum links resulting in the shortest.

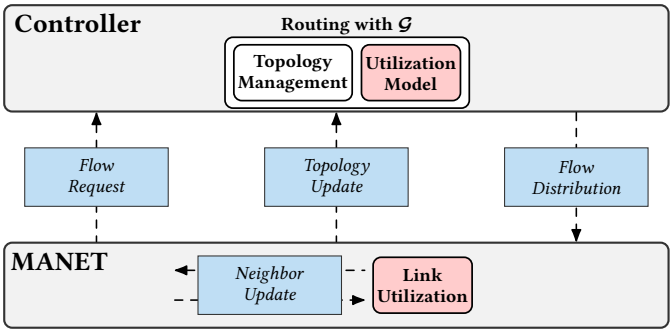


Figure 4.13: Cc-MANET architecture with utilization model extension.

```
1  begin computeDijkstra(o^f, t^f, d^f)
2      ⋮
       /* Investigating current node n and the direct neighbor m, both
          ∈ 𝒩. Computing n's upcoming demanded bandwidth based on d^f
          */
3      if m ≠ t^f then
4          lh = |p^f|
5          lh = lh > 4?4 : lh
6          rd^f = d^f · lh
7      end
8      oc_n = computeContention()
9      tu_(n,m) = u_(n,m) + rd^f + oc_n
       /* Computing edge weight w                                    */
10     if tu_(n,m) > c_(n,m) then
11         w = |ℒ| + tu_(n,m)
12     else
13         w+ = tu_(n,m) + 1
14     end
15 end
```

Algorithm 1: Dijkstra modifications when searching for a path from o^f to t^f taking into account the current utilization, the required flow's data rate and wireless channel complexity.

Our objective is to find the shortest paths from all origins to all targets while guaranteeing the capacity boundaries of selected links. Thereby wireless and multi-hop transmission behaviors, mentioned in Section 4.2.3, must be considered. In doing so, the flows and their data rate demands, the utilization of links, and all uncertainties, such as concurrent channel access, fragmentation, and interference, must be considered to not over-utilize the MANET with a new flow. Algorithm 1 introduces Dijkstra modifications to achieve the desired goal.

Dijkstra modifications take place when comparing link weights of directly connected MANET members not stored in the list of Dijkstra's visited nodes. Therefore, each link is evaluated according to its weight. Weights of sinks are updated if the newly computed distance is less than the currently stored distance. Next, the algorithm compares the distances to all nodes in the MANET with each other and picks the node with minimum distance.

Next, the algorithm repeats comparing all link weights of connected neighbors.

Line 3 shows Dijkstra modifications when recomputing the distance to node m if the new distance is lower. First, the standard route target check is performed to determine the potential next hop along the path. Before comparing the stored and computed distances of m, the number of route participants and the required data rate of this flow request is defined. Collision avoidance on the MAC layer detects ongoing transmissions, forcing transmission attempts to defer until the channel is free. In particular, RTS and CTS exchanges force nodes to stay idle for the frame sequence timespan. Utilization due to virtual carrier sense increases in relation to the number of route participants and stagnates when the number of route members exceeds 4.

Suppose a route consists of 5 members ordered as a chain with coverage ranges of each node reaching the predecessor and successor node and not further. A packet stream from 0 to 4 is ongoing. Node 2 must provide 3 times the required data rate d^f to relay the frames of the flow to node 4. The reduction is composed of receiving frames from 1 and forwarding frames to node 3. In addition, node 2 is virtually occupied when receiving CTS from 1 as node 0 transmits the data flow to the next hop. And lastly, node 3 also forwards the flow to the destination, which is why 3 only partially sends CTSs as response to the received RTSs. Lines 4 to 6 determine the required data rate $\mathtt{rd}^f$ for the upcoming flow, where $f \in \mathcal{F}$ and take into account the number of hops.

As mentioned earlier, CSMA/CA provides self-organized channel access starting with physical carrier sense (SIFS) and a subsequent backoff period. Access probability highly relies on the number of transmitting neighbors. Therefore, the next step is to predict the span node m might defer a frame, as surrounding nodes also try to access the channel (line 8).

$$\mathtt{oc}_n = \mathtt{bo}_n \cdot \mathtt{st} + \frac{(\frac{\mathtt{mtu}}{r}) \cdot L_n}{2} \tag{4.10}$$

Equation 4.10 gives an estimation of the mean occupation time $\mathtt{oc}_n$ of node n, where $n \in \mathcal{N}$ due to competing for channel access before it can start sending, given that the neighbors L_n also want to send and there is no collision. The left part of the equation describes the wait time due to backoff $\mathtt{bo}_n$ and the duration of a single slot $\mathtt{st}$. On the right side, additional time is being added for each of m's neighbors that have drawn a smaller backoff timer and thus are sending before n. Provided that, on average, they utilize

half of the MTU. The transmission duration dependents on the capacity of the link $c_{(n,m)}$ and the number of transmissions prior to m.

Having the current utilization $u_{(n,m)}$, the previously determined required data rate $\texttt{rd}^f$, where $f \in \mathcal{F}$ and the predicted occupation due to contention $\texttt{oc}_n$ lets us compute the upcoming total utilization $\texttt{tu}_{(n,m)}$, where $(n,m) \in \mathcal{L}$ (line 9). Next, we apply the link weight, ensuring capacity-conform path computation. The link weight increases by the total number of links in the MANET if the capacity of the link (n,m) is not sufficient, as shown in Line 11. Otherwise, the weight increases by 1 and the total utilization of the link, as described in Line 13. We include $\texttt{tu}_{(n,m)}$, where $(n,m) \in \mathcal{L}$ in the computation in order to choose the path with less demanded bandwidth if Dijkstra finds two paths with equal hop count.

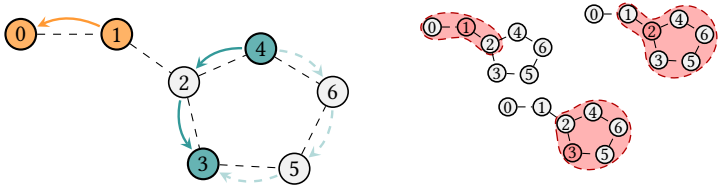
4.3 RESULTS

The following evaluation focuses on highly-utilized MANET transmission constellations. We present detailed results of two different evaluation scenarios. We first made a deep-dive into occurring and increasing utilization in a static-defined MANET. We compare the measured utilization with the utilization computed by the controller while increasing the flow's data rate. The second scenario concentrates on the elapsed time until the controller detects an over-utilized segment in the MANET. Both scenarios separate and analyze the share of each utilization type during different network loads.

We implemented our proposed concept with the network simulator OMNet++, version 4.5 [Var10]. MANET members are equipped with an IEEE 802.11 wireless interface configured in ad-hoc mode, using the $2.4\ GHz$ band for communication within the MANET. Each participant reaches transmission capacities up to $2\ Mbps$. The fragmentation threshold on the MAC layer is set to $2346\ B$ because higher frame chunks increase the throughput [Swe+10]. The RTS/CTS threshold is set to $0\ B$ to avoid collisions and interference. The uplink connection is also configured with an IEEE 802.11 interface providing $1\ Mbps$ of data rate.

4.3.1 Utilization Computation Accuracy

Figure 4.14 shows on the left side (Figure 4.14a) the MANET topology, connections, and routes of packet flows $f_{(1,0)}$ and $f_{(4,3)}$. The right side (Figure 4.14b) depicts which MANET members reach the radio waves of nodes



(a) MANET topology with flows $f_{(1,0)}$ and $f_{(4,3)}$ via intermediate node 2.

(b) Red surroundings illustrate nodes within communication range of node filled with red color.

Figure 4.14: MANET topology of static scenario evaluating utilization recording computation.

1, 2, and 3, filled with red color, respectively. Nodes within their transmission rages are symbolized by red surroundings.

Data rate prediction of route participants for requested flows is the major objective of this evaluation. Therefore, the computed utilization by the controller is compared with the actual utilization measured by the MANET members. Referring to the static scenario in Figure 4.14, the MANET delivers two flows. The flow $f_{(1,0)}$ is already ongoing and persistent for the entire simulation. The second flow $f_{(4,3)}$ stops and restarts several times along this scenario. Node 4 each time starts a new *Flow Request* using the uplink, triggering Dijkstra algorithm on the controller, see Algorithm 1 for path computation. Route participants update their forwarding tables each time they receive the path $p^{f_{(4,3)}}$ via the broadcasted *Flow Distribution*. Per default, route members flag forwarding table entries as invalid after an elapsed timeout period of 3 seconds if the entry is no longer needed. Each time, the packet flow $p^{f_{(4,3)}}$ demands more transmission capacity, which will over-utilize the MANET during simulation time. Thereby, we focus on the accuracy of the predicted utilization. The controller will reroute flow $p^{f_{(4,3)}}$ via node 6 and 5 to destination 3 in case the data rate of this flow exceeds the transmission capacity of either link $(4, 2)$ or $(2, 3)$. To be more concrete, flow $p^{f_{(1,0)}}$ already utilizes link $(1, 0)$ at the beginning of the simulation to 46%. At simulation time $10\ sec$ flow $f_{(4,3)}$ requests a route from the controller with a data rate demand of $5000\ Bps$. Node 4 then stops the transmission after $10\ sec$ and $10\ sec$ later requests an additional route and increases the data rate demand by $5000\ Bps$. Node 4 starts further *Flow Requests* following the described procedure while increasing the data rate accordingly every $10\ sec$ until the simulation stops.

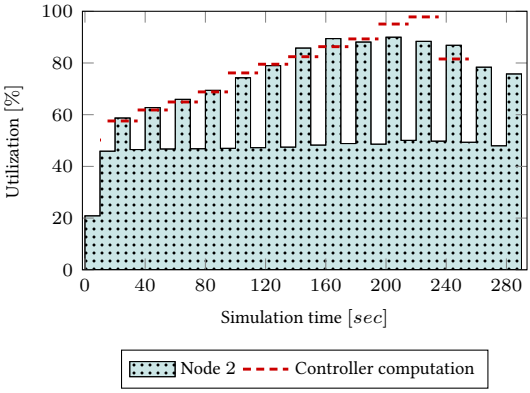


Figure 4.15: Comparing measured utilization of node 2 with controller computed overhead.

These data rates are chosen wisely to provoke an over-utilized situation while the simulation is ongoing. Thereby, each MANET member periodically reports the current utilization to the controller via *Topology Updates*. So far, $f_{(4,3)}$ is routed via intermediate node 2. Flow $f_{(4,3)}$ should be rerouted via 6 and 5 when over-utilization is detected by the controller. The data loss rate of both flows $f_{(1,0)}$ and $f_{(4,3)}$ should be negligible if the controller's utilization prediction concept is accurate. The following results show firstly the accuracy of upcoming utilization when the controller computes the path for flow $f_{(4,3)}$. Secondly, we deep-dive into all utilization types and analyze their share in relation to the total utilization. Thirdly, we show the occurred loss rate of $f_{(4,3)}$ to evaluate the controller's utilization predictions.

Plots 4.15 and 4.16 compare the recorded utilizations of nodes 2 and 5 with the computed controller overhead. Red lines show the controller's computed link with the highest utilization during path computation of flow $f_{(4,3)}$. Nodes 2 and 5 are chosen as they obtain the highest utilization of both, the initial and the bypassed route for $f_{(4,3)}$. Controller computations in Plot 4.15 range from simulation times 0 sec to 260 sec since node 2 obtains the highest utilization until that point. Thereafter, utilization of node 5 increases the most, as shown in Figure 4.16.

Node 2 in Plot 4.15 is constantly utilized at 46% as data and control frames of 1 reaches 2. Utilization of 2 increases each time the controller

deploys the path $p^{f_{(4,3)}}$ on the MANET. The controller computes $p^{f_{(4,3)}} = \langle 4, 2, 3 \rangle$ as long as this route provides sufficient capacity since this is the shortest path to the destination. Detouring $f_{(4,3)}$ at simulation time 260 sec increases the utilization of node 5 shown in Plot 4.16. The links of the initial route exceed their capacities. MANET members are occupied and increase their utilization if a node within communication range is transmitting, which is why measured and computed utilization of node 5 increases significantly at simulation time 260 sec. Nodes 4, 5, and 6 are within each other's communication range and have to stay idle during transmissions of their neighbors.

It turns out that the measured utilization of node 2 never reaches 100% whereas the utilization computed by the controller predicts a higher overhead. This can be explained by the extended backoff time period, which is introduced by the exposed terminal problem caused by node 1 [Bha+94]. Node 2 experiences a variety of interference as transmissions from nodes 1 and 4 frequently overlap, resulting in CTS and ACK timeouts at origin node 4. Longer backoff periods delay and also drop packets of flow $f_{(4,3)}$ resulting in misclassified idle times of node 2, which should rather be classified as occupied times. That is why the controller's overhead computation becomes inaccurate during these almost fully-utilized situations. The gap becomes visible when the predicted utilization is nearly exhausted at simulation time 200 sec. The controller at this point detects over-utilization and reroutes flow $f_{(4,3)}$. Referring to Plot 4.16, utilization recording of node 5 increases when rerouting took place as this node now forwards data to the

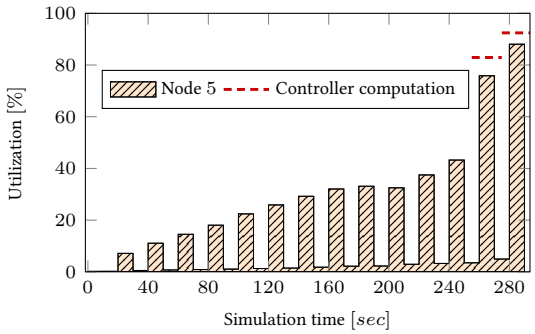


Figure 4.16: Comparing measured utilization of node 5 with controller computed overhead.

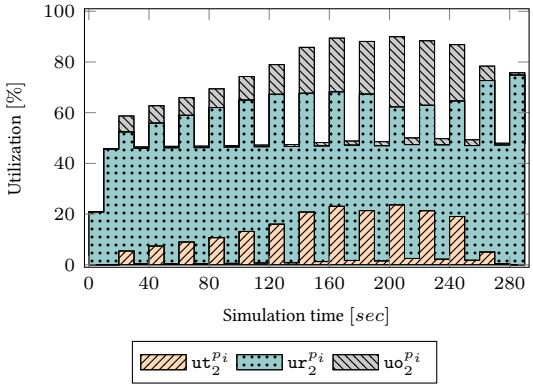


Figure 4.17: Share of recorded utilization types of node 2 during simulation of scenario 1.

target 3.

Figure 4.17 analyzes all measured utilization types of node 2, separated and stored in $\mathtt{ut}_2^{p_i}$, $\mathtt{ur}_2^{p_i}$, and $\mathtt{uo}_2^{p_i}$, respectively. Update periods p_i in this figure contains $P = \langle p_0, \ldots, p_{289} \rangle$ since MANET participants transmit *Topology Updates* via the uplink channel to the controller each second.

The share of utilization due to transmitting and receiving is as expected. Reception utilizes node 2 roughly twice compared to measured transmission overhead as node 2 decodes frames form $f_{(1,0)}$ and $f_{(4,3)}$. After simulation time 150 sec when reaching the capacity maximum, utilization due to occupation continuously increases whereas measured transmission and reception stay the same or even decrease slightly. Frames from nodes 1 and 4 disturb and delay transmission attempts more often. Also, interferences extend backoff periods, all resulting in higher occupation times.

Figure 4.18 shows the packet delivery fraction of both flows while transmission overhead continuously increases. Flow $f_{(1,0)}$ is ongoing during the entire simulation and does not experience any loss. The second flow from 4 to 3 stops and restarts after 10 seconds. The transmission lasts for 10 sec, explaining the gaps in which 4 does not send $f_{(4,3)}$. Flow $f_{(4,3)}$ starts losing frames from simulation time 200 sec and ongoing while the controller's predicted utilization is at 96%. Interfering frames and increasing backoff times result in idle times at node 2, which are not considered by the controller. This generates inaccurate computations especially when reaching utilization recordings above 90%.

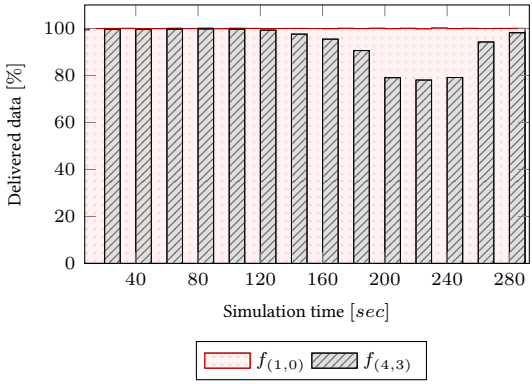
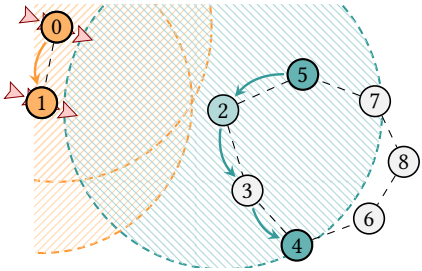


Figure 4.18: Received data in relation to sent data of both flows.

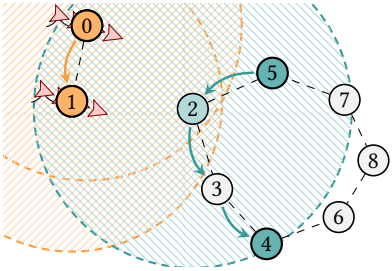
Summing up, utilization prediction computes upcoming utilization very accurately. Utilization due to occupation reaches a significant share with respect to the overall utilization which is also predicted accurately. However, utilization predictions become inaccurate when the measured utilization exceeds approximately 90%. This was to be expected as the channel access of participants also depends on randomness and node utilization computation becomes more complex with an increasing number of transmitting participants.

4.3.2 Over-Utilization Detection

The second scenario investigates the reaction time of the controller if over-utilization appears abruptly because transmitting nodes reach each other's coverage range. The objective is to measure the elapsed time and amount of lost data when the capacity of a path participant exceeds. The scenario depicted in Figure 4.19 generates this over-utilized situation. Nodes 0 and 1 move $3\ m/sec$ continuously south-east while delivering $f_{(0,1)}$. The scenario is configured so that node 2 has insufficient transmission capacity when communication ranges of nodes 2, 1, and 0 overlap while 0 and 1 should still deliver their flow without loss. Flow $f_{(0,1)}$ demands 77% of nodes 1 and 0 transmission capacities. Reception, forwarding and waiting states of node 2 increase the capacity demand of node 2, which results in an over-utilized situation when frames of $f_{(0,1)}$ reach node 2. Similar to scenario 1, the controller will reroute $f_{(5,4)}$ via 7, 8, and 6 when the initial



(a) Transmitting nodes 0 and 1 move south-east while delivering flow $f_{(0,1)}$. Communication ranges do not overlap yet.



(b) Frames of $f_{(0,1)}$ reaching active path participant 2 of $f_{(5,4)}$, exceeding transmission capacity.

Figure 4.19: MANET delivers flows $f_{(0,1)}$ and $f_{(5,4)}$, which do not interfere with each other at the beginning. Nodes 0 and 1 move south-east and over-utilize node 2, causing incomplete data delivery.

path is over-utilized.

We investigated the loss rate and also the recorded utilization of the detoured path when the controller's predicted utilization exceeds the MANET member's capacities. Different *Topology Update* intervals are applied to compare these performance parameters with each other.

Figure 4.20 shows the share of sent in relation to received data of $f_{(5,4)}$. This also demonstrates when the alternative route of this flow was computed and deployed on the MANET. Packet delivery of flow $f_{(5,4)}$ reaches almost 0% when both flows ($f_{(5,4)}$ and $f_{(0,1)}$) interfere with each other. Interference starts shortly after simulation time 37 sec. The controller's path computation and deployment via *Flow Distribution* message takes a neg-

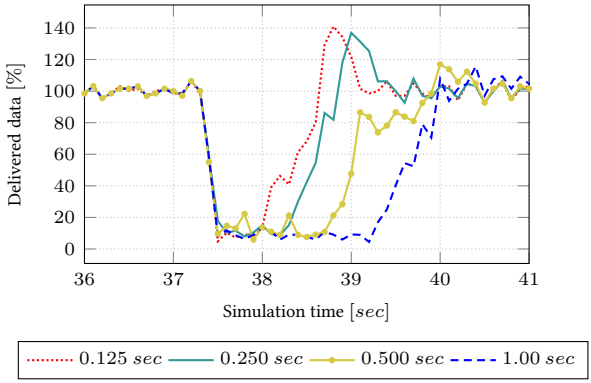


Figure 4.20: Delivered data of flow $f_{(5,4)}$ immediately before node 2 receives frames of $f_{(0,1)}$ with different configured *Topology Update* intervals, ranging from $0.125\ sec$ to $1.00\ sec$.

ligible share of time compared to the bottleneck detection. For instance, utilization of an active route participant will not reach 100% if 50% of the *Topology Update* interval has elapsed when interference starts. Several route participants will take two *Topology Update* interval periods until the controller predicts over-utilized network segments. However, the amount of delivered packets increases earlier when update interval time spans are shorter.

We compared the amount of sent with the amount of received data in $100\ msec$ steps. The MAC queue of the origin node 5 was full as long as frames have been routed via the congested path. When deploying the alternative path in the MANET, node 5 selects node 7 as the next hop but must queue and delay the frames since the MAC queue is still filled. These frames addressed to node 7 have often been sent in the previous recording step and are now being delivered within the next $100\ msec$ recording time. This explains why delivery rate reaches 140% when the detour path has been deployed.

4.4 SUMMARY

This chapter proposed two traffic utilization concepts. The distributed recording concept showed inaccuracies regarding the measurement method. Also, the lack of complete network topology knowledge makes it difficult

to ensure that new paths and their data rate demands do not interfere with existing flows. The second utilization model introduces the Cc-MANET architecture having the complete topology at hand. Thereafter, the challenges and considerations of delivering data in wireless multi-hop routes were discussed and addressed. We also presented an extended the utilization recording concept on the MAC layer using CSMA/CA. We used the measured overhead as baseline for further utilization predictions to answer the question if nodes have sufficient capacities left to carry additional flows. Finally, we demonstrated results of the proposed utilization measurement concept, presenting accurate utilization prediction by the controller if the measured transmission overhead is below 90%.

However, the interval-based *Topology Update* concept delays topology changes and measured over-utilization, which results in congested network segments for at least one *Topology Update* period. The following two chapters present two reactive topology update processes to avoid outdated and delayed topology and utilization information.

CHAPTER 5

Outsourced Topology Management

The results of the previous chapter proved reliable link utilization predictions until utilization reaches approximately 90%. Due to this, the controller can take transmission utilization into account and prevent links of being over-utilized. For instance, if an additional flow must be deployed and the preferred path lacks transmission capacity because selected links are already relaying another flow, the controller is able to detour the upcoming flow to ensure capacity-conform data delivery.

However, we also observed a sluggish reaction when the controller must detect over-utilization at runtime. Mobility, for instance, can cause transmission interferences and, as a consequence, lead to over-utilized connections. The previously introduced proactive topology update mechanism of Pro-CeTUP delays important network conditions of MANET participants, which could be avoided if the controller is provided with up-to-date topology knowledge. These network conditions include information about current MANET members, their connections, and transmission utilization statistics.

In this chapter, we introduce the Reactive Topology Update Process (Re-CeTUP), triggering all MANET members to report their connections and utilization information to the controller immediately before routing is desired. Also, the integration of Re-CeTUP in the Cc-MANET architecture is proposed.

5.1 VERIFY TOPOLOGY QUALITY

Before deep-diving into the reactive Cc-MANET architecture and the revised topology update process, we measure the actuality of the topology representation on the controller when using Pro-CeTUP, which was introduced in the previous chapter. We review the extent to which the representation of the MANET topology becomes outdated compared to the actual

MANET constellation when mobility comes into play. Finally, we evaluate the performance regarding topology actuality of Pro-CeTUP. As a recap of the Pro-CeTUP process, MANET members periodically broadcast *Neighbor Updates* and also periodically transmit *Topology Updates* via the uplink channel to the controller.

We transmit *Flow Requests* to the controller with randomly chosen origin and target pairs to evaluate the actuality of Pro-CeTUP. Having the proactively reported topology representation at hand, the controller computes all paths and deploys them in the forwarding tables of corresponding nodes via *Flow Distributions*. We evaluate the actuality of all paths computed by the controller with different mobility speeds ranging from $5\ km/h$ to $30\ km/h$. Routes exist if the target nodes of paths receive data from their origins, which proves that all connections along the paths are active. Different simulation settings are carried out, each running 200 times with different seeds. Therefore, we randomly generated MANETs and origin-target pairs.

We define a speed range, each spanning exactly $5\ km/h$ to evaluate actuality with different velocities. We apply the RWP [AWS06], in which each node randomly picks a speed from this defined range and moves towards a randomly chosen direction for between $20\ sec$ and $40\ sec$. Thereafter, nodes again pick a speed from the same speed range. To be more precisely, we define the speed ranges $\mathtt{sr}_5 = [0\ km/h, 5\ km/h]$, $\mathtt{sr}_{10} = [5\ km/h, 10\ km/h]$ to $\mathtt{sr}_{30} = [25\ km/h, 30\ km/h]$, each associated to a run. Each run is executed 200 times. Each range comprises real numbers from the defined start to the defined end. According to our definition, the following velocities $V = \{\mathtt{sr}_5, \mathtt{sr}_{10}, \mathtt{sr}_{15}, \mathtt{sr}_{20}, \mathtt{sr}_{25}, \mathtt{sr}_{30}\}$ are applied. The different speed ranges are each combined with 5, 30, or 60 MANET nodes. Further simulation parameters, that are applied in each run are shown in Table 4.4.

It is important that at least one connection to the MANET exists for each participating node so that routing can take all participants into account. We therefore pre-run each number of nodes and speed range combination to determine a proper playground size. Runs, for which no path from origin to target is found by the controller are aborted and not listed in the results. The evaluation only considers the current run if the origin starts the data transfer, which only happens if the controller sends a *Flow Distribution*.

Based on the introduced configurations each run proceeds as follows: The nodes start moving based on their configured speeds after having been placed randomly on the playground. Next, nodes start Pro-CeTUP, meaning

Table 5.1: Simulation parameters

Parameter	Value
MAC protocol	CSMA/CA
Communication range	150 m
Runs	200
Mobility model	Linear mobility
Data rate	2 $Mbps$
Topology Update interval	1 sec

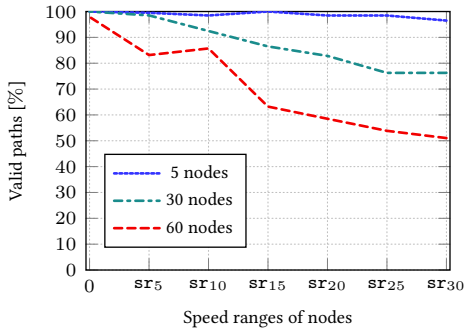


Figure 5.1: Computed up-to-date paths by the controller in percentage applying Pro-CeTUP with different node speeds.

that each participant periodically exchanges neighborhood information and also periodically transmits *Topology Updates* to the controller. After 60 sec, a randomly chosen node starts a *Flow Request* to the controller. After each path participant updated the forwarding table with the received routing information, the origin starts the data transfer. The simulation terminates either, if the data frame has been delivered to the target or any node along the path has experienced a lost connection.

Figure 5.1 depicts all valid computed paths in relation to all successfully terminated runs of all MANET sizes. The first x-tick (0) depicts found valid paths with no mobility. It turns out that the controller's topology representation becomes more inaccurate with increasing number of nodes. In fact, the number of correctly computed paths decreases to 50% with $\texttt{sr}_{30}$ and 60 MANET members. With 5 MANET participants, the controller com-

putes almost all paths correctly, with respect to all speed ranges. Several factors play into why the actuality of the topology representation decreases with increasing number of nodes. The most obvious reason is that the number of route participants increases, which is error-prone as the probability increases that outdated connections are delivered to the controller. Also, each MANET on average detects and maintains more neighbors, and nodes more often report connections to the controller which no longer exist. Neighbor maintenance of Pro-CeTUP removes connections if previously received *Neighbor Updates* of nodes have not been decoded for two consecutive *Neighbor Update* time periods. Removing connections after one *Neighbor Update* period improves the actuality of the topology representation, but may also lead to the erroneous removal of connections due to interference.

This evaluation proves that the Pro-CeTUP is sluggish in terms of keeping the topology representation on the controller up-to-date when the number of participants and their speeds increase. The controller more often computes and deploys paths, which are not present in the MANET. The topology representation must be more accurate and reliable considering the objective to route several data rate demanding flows while guaranteeing to not exceed the transmission capacity of any transmitting connection.

Now, we transform Pro-CeTUP to a reactive topology update technique. This approach is called Re-CeTUP$_{\mathrm{CSMA}}$ as the controller gathers topology information immediately before path computation starts. The uplink and the MANET channel are equipped with CSMA/CA [IEEE802.11]. Nodes only send *Neighbor Updates* and *Topology Updates* if the controller triggers them to update the topology which takes place immediately before routing starts. When the controller receives a *Flow Request* it broadcasts a so-called *Topology Refresh* via the uplink channel to the MANET, triggering all members to first exchange *Neighbor Updates* and secondly start updating the controller's topology snapshot with *Topology Updates* [Str+19a].

MANET members do not exchange any control data at runtime until a *Topology Update* is broadcasted by the controller. Path computation can now be based on the previously triggered topology information. This reactive update process is configured and evaluated in the same way as Pro-CeTUP.

Figure 5.2 depicts all valid computed paths in relation to all successfully terminated runs considering each speed range and each number of nodes configuration. Except for topologies with only 5 nodes, this evaluation shows that the controller computes fewer valid paths than the Pro-

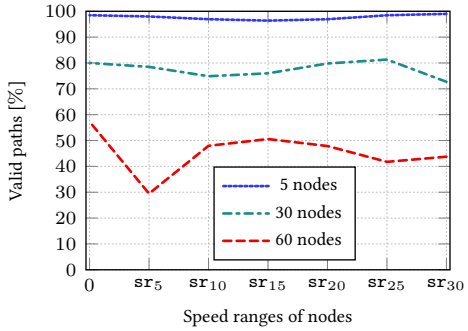


Figure 5.2: Computed up-to-date paths by the controller in percentage applying Re-CeTUP$_{\mathrm{CSMA}}$ with different node speeds.

CeTUP method, especially as the number of participants and their speeds increase. The results are rather remarkable considering that the number of valid paths appears to be similar with respect to runs, configured with 30 and 60 nodes combined with all speed ranges. Inconsistencies between real and represented topology are obviously not caused by node speeds.

Nodes start competing for channel access to broadcast *Neighbor Updates* almost simultaneously as the *Topology Refresh* reaches each MANET member almost at the same time. Messages overlap and arrive interfered if nodes start their transmission attempt at the same time. This happens more often with increasing number of nodes. We experienced the least simultaneous *Topology Updates* transmissions compared to *Neighbor Updates* during test runs. The impact of lost *Topology Updates* is enormous compared to lost *Neighbor Updates* as this results in lost sub-topologies.

We evaluated the *Neighbor Update* exchange of Re-CeTUP$_{\mathrm{CSMA}}$ with microcontrollers, using the IoT-Lab[1] in order to confirm the results. The platform provides various hardware boards to deploy `C` code on. Their founders are scientific and economic institutions, currently providing over 2700 wireless sensor nodes. We used the most powerful boards at this time, named A8 Open Node. Their wireless interfaces is equipped with the ZigBee 802.15.4 standard, utilizing the $2.4\,GHz$ band [IEEE802.15]. The nodes reach data rate peaks of $250\,Kbps$. As a processor, the boards use a $32\,bit$ Arm Cortex A8 microprocessor[2].

[1] https://www.iot-lab.info
[2] https://developer.arm.com/ip-products/processors/cortex-a/cortex-a8

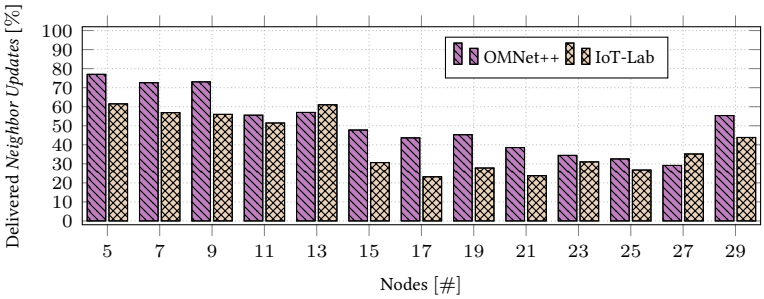


Figure 5.3: Comparison *Neighbor Update* message exchange of ReCeTUP$_{\text{CSMA}}$ between OMNet++ simulation and IoT-Lab (ZigBee) testbed with different number of nodes. Results show successfully transmitted *Neighbor Updates*.

For the results carried out with our setup, we configured nodes equipped with similar components and data rates. In particular, each member of the simulation is equipped with a ZigBee module, provided by the INET framework[3]. The size of a *Neighbor Update* in B compounds 25 B MAC header, 6 B frame overhead, which is based on IEEE 802.15.4 specifications [IEEE802.15], 20 B IPv4 header, 8 B UDP header, and 2 B node id. This *Neighbor Update* size is considered in both the hardware testbed and the simulation. Also, topologies are equally designed, meaning all hardware nodes and simulation nodes are fully meshed. We included randomness in the IoT testbed as nodes in the simulation are placed differently each time. In particular, various placements of simulations are imitated with short random waiting times before *Neighbor Update* transmission is intended by hardware nodes.

Based on the introduced configurations the simulation and the hardware testbed run as follows: The controller broadcasts a *Topology Refresh*, reaching each node. The *Neighbor Update* exchange starts thereafter. A *Neighbor Update* has been transmitted successfully if each participant successfully decoded the frame. Both simulation and hardware testbed scenarios ran 10 times.

The averaged results from 5 to 29 MANET participants are depicted in Figure 5.3. The results of the simulation and the hardware testbed appear to be similar from 5 to 29 nodes, whereas the simulation more frequently ex-

[3]https://inet.omnetpp.org

hibits better delivery rates. That was to be expected since the transmission power in the IoT-Lab shrinks due to walls and further obstacles. Overall, we can claim that the previously obtained results (Figure 5.2) are valuable as the results of the current comparison between simulation and hardware testbed are similar.

5.2 ASSESSING THE PROACTIVE TOPOLOGY UPDATE

In this section, we discuss Pro-CeTUP and highlight the advantages and disadvantages regarding the following evaluation criteria when using this topology update in the Cc-MANET architecture.

Message Overhead Cc-MANET was initially designed to be proactive. *Neighbor Updates* and *Topology Updates* serve as information delivery of topology knowledge to the controller. Each MANET participant repeats both messages periodically [HST16].

It follows that this control overhead is present even though no flow is actively delivered in the MANET. Message overhead can be decreased with increasing the transmission intervals of *Neighbor Updates* and *Topology Updates*. This decreases the accuracy of the topology, resulting in the controller computing wrong paths. Nodes repetitively send *Flow Requests* to the controller as long as no topology update, affecting the outdated network segment is transmitted to the controller. This results in wrongly computed paths by the controller, which are if course transmitted via *Flow Distributions* to the MANET.

Battery Usage MANET nodes do not have access to a power supply and depend on long-living batteries. Message transmissions consume a considerable amount of power to radiate radio waves which drains the batteries. Nodes keep the controller up-to-date by periodically transmitting *Neighbor Updates* and *Topology Updates*. Battery levels of nodes decrease in linear fashion also if the network is idle. This makes Pro-CeTUP inefficient regarding battery usage.

Topology Reliability and Quality MANET participants remove a node from their neighborhood tables if two consecutive *Neighbor Updates* have not been delivered by a neighbor. On the one hand, this considers interference as members broadcast *Neighbor Updates* not waiting for ACKs. Data leaks remain and are included in *Topology Updates* even though no new *Neighbor Update* in the previous round has been received. On the other hand, this results in an unreliable topology representation.

The objective of this thesis is to compute robust routes of multiple flows while not over-utilizing the MANET. A robust path requires an accurate connection quality metric, distinguishing stable links from unstable ones. The link quality changes during time since robustness also depends on distance between nodes and their speeds. The proactive update nature distorts the determined link qualities although the metric itself defines stable links very accurately as actuality depends on control message update periods. The robust link quality is discussed in Section 7.3.4.

This section pointed out drawbacks of current topology update concepts and highlights challenges of a reliable and up-to-date update concept introduced in the following section.

5.3 REACTIVE TOPOLOGY UPDATE PROCESS

This section presents the Re-CeTUP [SD20], reporting the entire MANET topology to a controller by providing wireless connectivity to each node via the uplink. This concept focuses on an up-to-date topology snapshot including all nodes and their connections to avoid path computations based on outdated topology knowledge. The controller therefore reactively triggers all MANET participants to report their direct neighbors immediately before routing takes place.

In that context, we introduce an adapted TDMA [Rod06, Section 14.7] scheme to guarantee unique channel usage. This also includes the definition of new messages types and data fields of Re-CeTUP for the Cc-MANET architecture. Re-CeTUP is a MANET topology gathering algorithm, designed to be embedded in the Cc-MANET architecture, alongside Pro-CeTUP. When speaking of Re-CeTUP, this technique employs TDMA throughout this thesis if not further abbreviated.

5.3.1 Framework Components

Re-CeTUP focuses on an up-to-date topology representation regarding MANET participants and their connections in order to build reliable routing comprising only of all actual existing connections and no obsolete ones. For that, topology information must be gathered right before the controller starts with the routing process. All *Neighbor Updates* and *Topology Updates* of each MANET participant must start exactly after a *Flow Request* has been received by the controller, as briefly proposed in Section 5.1. Evaluations of that context showed drawbacks, such as mutual interference during *Neigh-*

bor Update exchange as nodes start their transmissions almost at the same time.

Because of that, we apply TDMA [Rod06, Section 14.7], adapted for the Re-CeTUP to guarantee exclusive channel access. Also, occupation times due to channel access competitions by nodes, using CSMA/CA, do not longer utilize nodes as access time periods for each MANET participant are defined in advance. This requires detailed information about participants' wireless interfaces, which is why a sophisticated access management is needed for Re-CeTUP. So far, the topology management of the proactive topology update fashion (Pro-CeTUP) has no registration or access method that allows for further wireless interface configurations to be extended. As a consequence, nodes that are transmitting *Topology Updates* to the controller automatically take part in the Cc-MANET as they are included in path computations by the controller.

The registration process of Re-CeTUP requires two additional messages, named *Registration Request* and *Registration Confirmation*. A node willing to take part in the MANET sends the unicast message *Registration Request* to the controller which stores the radio configuration of a participant, necessary for slot computations. The controller stores these parameters and replies with a *Registration Confirmation* including a unique id. *Registration Confirmation* and *Registration Request* are unicast messages and the controller and all MANET nodes access the channel with CSMA/CA for them. At this point in time, no slot has been allocated for the new node so far, which is why CSMA/CA is used. Furthermore, participant management must be reliable, which is why message acknowledgments are necessary. The registration handshake is successful in case the MANET member received the *Registration Confirmation* message.

Figure 5.4 introduces Re-CeTUP components and the associated new message types. MANET nodes send *Flow Requests* to start a transmission or to find a new route for an ongoing transmission if the connection to the next hop is not available anymore. *Flow Requests* still contain origin and target ids, as well as, the data rate of each requested flow as parameters. The controller comprises the new module **Participant Management** also including two new components. **Access Management** maintains all MANET members including registering and removing of nodes. This component goes along with the **Slot Allocation** as channel access times for nodes are only assigned if nodes are registered. **Slot Allocation** computes and allocates slots of the uplink and the MANET channel based on the radio configurations of both wireless interfaces of each participant.

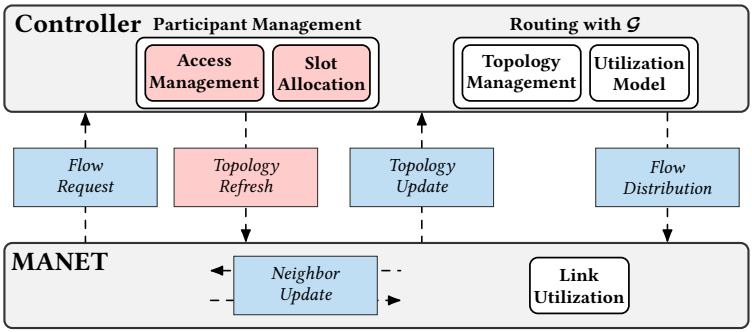


Figure 5.4: Message types and components of Re-CeTUP build in Cc-MANET architecture.

The controller starts the **Slot Allocation** each time the **Access Management** shows MANET member changes after a *Flow Request* arrives. Otherwise, a *Topology Refresh*, containing transmission slots (uplink and MANET channel) for all nodes is delivered to the MANET via the uplink channel. A *Topology Refresh* also instructs members to start sending *Neighbor Updates* according to their allocated slots. After sub-topologies are exchanged via *Neighbor Updates*, nodes deliver their neighborhood information to the controller via the uplink using their second allocated slot, which is also stored in the *Topology Refresh*. *Topology Updates* also comprise the current transmission utilization status of nodes. The controller computes the path for the requested flow from origin to destination, considering the link utilizations and also reroutes already deployed flows if new paths prove better after each *Topology Update* has been received. The *Flow Distribution* is then broadcasted to the MANET containing the list of paths including the newly requested one.

The number of slots for both the MANET and uplink channel are computed based on the node's radio configurations. This guarantees efficient and interference-free utilization of both channels as nodes start *Neighbor Updates* and *Topology Updates* after each other. Introducing the *Topology Refresh*, we have now designed the topology update process reactively, resulting in an accurate and up-to-date topology representation immediately before path computation begins.

In terms of the data types of the framework, the following parameter is added: The controller has a unique id, as shown in Table 5.2. The red-colored row depicts this newly introduced parameter. The unique id of

Table 5.2: Global parameters and variables used in multiple RQs.

Properties	Belonging	Description
n	$n \in \mathcal{N}$	Unique id of a MANET participant.
τ	$--$	Unique id of the controller.
o^f	$o \in \mathcal{N}, f \in \mathcal{F}$	Origin node of a flow.
t^f	$t \in \mathcal{N}, f \in \mathcal{F}$	Target node of a flow.
p^f	$p \subseteq \mathcal{N}, f \in \mathcal{F}$	Path as sequence of nodes of a flow.
d^f	$f \in \mathcal{F}$	Demanded data rate of a flow.
u_l	$l \in \mathcal{L}$	Current utilization of a link.
c_l	$l \in \mathcal{L}$	Capacity of a link.

the controller is stored in τ and associated to the network graph $\mathcal{G}^\tau = (\mathcal{N}^\tau, \mathcal{L}^\tau, \mathcal{F}^\tau)$. The controller's unique id is not belonging to a certain data structure so far. The id changes if, for instance, MANET members experience a controller outage, identifying another outsourced controller having a different id.

This application scenario, comprising the process of electing a new outsourced controller and potential constraints in the context of that is beyond the scope of this thesis. According to that definition, a controller can manage multiple MANETs consisting of multiple graphs in theory. Regarding Cc-MANET architecture and Re-CeTUP, we focus on a single controller managing one MANET.

The controller id is not part of the set of MANET participants and is unique, meaning $\mathcal{N} \cap \{\tau\} = \emptyset$. If the network graph $\mathcal{G}^\tau$ is described without the associated controller τ (just $\mathcal{G}$), we imply that the controller id is represented by τ. This definition is also set for all sets identifying a graph. The set of nodes $\mathcal{N}$, the set of links $\mathcal{L}$, and the set of flows $\mathcal{F}$ belong to the controller id τ if not otherwise specified. Nevertheless, the graph almost always belongs to the controller id τ since we focus on a single MANET in this RQ.

5.3.2 Access Management

This section discusses the node registration process in detail. In that context, participants also disconnect from the MANET. The process of how this is handled is also introduced in this section. Furthermore, we deep-dive into how slots of both channels (uplink and MANET channel) are allocated by

the controller, which also results in an adaptive MAC switching routine, introduced in this section as well.

Network Registration

As mentioned in Section 5.3.1, the registration process comprises two messages, named *Registration Request* and *Registration Confirmation*. The first one is transmitted from a node to the controller at runtime in order to access the network using the uplink. The latter is used by the controller to respond in case the registration was successful. A *Registration Request* does contain radio configurations for the uplink and the MANET channel of the node, indicating to the controller that the current node has not been registered yet. Both configurations are later used for slot size computations by the controller. The controller determines unique ids and returns this information to the node that previously sent the *Registration Request*.

Formally, the controller determines n, representing the id and inserts this id in $\mathcal{N}$. We define the parameter `rs` identifying whether the set of MANET participants ($\mathcal{N}$) has changed. The parameter is either set to 0, referring to no change, or to 1 informing the controller that the set of nodes has changed. The controller flags this parameter to 1 before transmitting a *Registration Confirmation* to a node, that sent the *Registration Request* to remember to recompute slots for all participants later. Finally, the controller responds with the *Registration Confirmation* containing the new id.

The registration process could also be skipped. In that case, the controller applies an additional control message after a *Flow Request* has been received, triggering all MANET participants to report their radio configurations. This must happen each time since a new MANET member could have decided to join the network in the meantime. The benefit of this approach is that no resources are wasted by assigning slots to nodes that have left the network. However, multiple messages are likely to collide as new nodes access the channel via CSMA/CA and interfere with *Topology Updates* of already registered nodes. Missing sub-topologies in the MANET snapshot would decrease the actuality tremendously.

Leaving the Network

It happens that MANET members leave the network at runtime. Defining a leave message from MANET node to the controller via the uplink channel is not sufficient as nodes could run out of battery or experience a hardware failure, thus not sending a notification to the controller. Leaving this issue

```
1 begin UpdateParticipants(𝒩′)
2     foreach n ∈ 𝒩 \ 𝒩′ do
3         mt_n = mt_n + 1
4         if dt = mt_n then
5             𝒩 = 𝒩 \ {n}
6             rs = 1
7         end
8     end
9 end
```

Algorithm 2: Verifying nodes' slot usage after Re-CeTUP finished. Repetitive unused slots are defined as obsolete and allocated to another node in the successive update process.

unresolved results in unused slots during update cycles. Consequently, the controller has to keep the list of registered nodes up-to-date, meaning nodes that left the network must be detected and their assigned slots must be allocated to other nodes.

Algorithm 2 shows how the controller detects and reallocates unused slots. The set $\mathcal{N}' \subseteq \mathcal{N}$, which is a subset of $\mathcal{N}$ stores all n, having delivered a *Topology Update* during the last Re-CeTUP. Next, from Lines 2 to 8, the controller removes node n from the set of $\mathcal{N}$ if n missed its update slot several times. The global threshold is defined with $\texttt{dt}$, where $\texttt{dt} \in \mathbb{N}$. This parameter $\texttt{mt}$ is set to 3 per default. In Line 3, the controller increments $\texttt{mt}_n$ corresponding to any node n, where $n \in \mathcal{N}$, as these nodes missed their slots during the last update period. Next, the frequency of missed *Topology Updates* in a row of each node n is compared against the threshold $\texttt{dt}$. The controller removes a MANET node n from the set of registered nodes $\mathcal{N}$ if $\texttt{mt}_n = \texttt{dt}$. The parameter $\texttt{rs}$ is set to 1 since the set of nodes is modified, forcing the controller to recompute all slots when the next *Flow Request* arrives.

The controller assigns the slots of nodes having exceeded the $\texttt{mt}$ to other participants in the subsequent *Topology Refresh*. Nodes might lose connectivity to the controller at runtime and expect their reserved slots when receiving a *Topology Refresh*. Nodes not considered during the current Re-CeTUP but willing to take part skip the current update cycle and send a *Registration Request* after Re-CeTUP finished.

```
1  begin DetermineSlots()
2      if rs = 1 then
3          T_m = {}
4          md, ud = 0
5          foreach n ∈ 𝒩 do
6              ns_n = md
7              T_m ∪ (n, ns_n)
8              md+ = ComputeSlotStart(md, mc_n)
9          end
10         ud = md
11         foreach n ∈ 𝒩 do
12             ts_n = ud
13             T_u ∪ (n, ts_n)
14             ud+ = ComputeSlotStart(ts_n, uc_n, |𝒩| − 1)
15         end
16         rs = 0
17     end
18 end
```

Algorithm 3: Computing and allocating *Neighbor Update* and *Topology Update* slots for registered nodes.

5.3.3 Slot Allocation

In general, the slot allocation process starts when the controller receives a *Flow Request* from an arbitrary node. The controller's job is to compute routes based on a complete topology representation. Therefore, the controller determines slot times of MANET participants to be used for an interference-free and up-to-date update process.

Algorithm 3 shows the computation and allocation of slots for both the uplink and the MANET channel. The function DetermineSlots fills T_m and T_u with slot start times. In particular, the controller computes the start and length of each slot based on the associated radio configuration of each node and stores these slots and their node ids as tuples $(n, \texttt{ns}_n)$ and $(n, \texttt{ts}_n)$, where $n \in \mathcal{N}$ in the sets T_m and T_u. The first tuple contains slot information for the MANET channel and the second for the uplink channel. For that, the controller verifies if a new member has registered within the last update phase (Line 2). If not, no further slot computations are necessary as slots for all members remain unchanged with respect to the MANET

and uplink channel. Otherwise, slots must be recomputed starting with the MANET channel.

In this case, the parameter T_m is initialized as an empty set, and continuously filled with slot start times and associated node ids. The set T_m is formally described as $T_m = \{(n_0, \texttt{ns}_{n_0}), \ldots, (n_{|\mathcal{N}|-1}, \texttt{ns}_{n_{|\mathcal{N}|-1}})\}$, where $n \in \mathcal{N}$ and `ns` represents the start time of the associated MANET slot. Parameters in line 4 store the start time of an upcoming slot of the MANET (`md`) and uplink (`ud`) channel and at the end of this function the timestamps when the last MANET member sent a *Neighbor Update* and *Topology Update* to the controller, respectively. The slot computation and allocation algorithm loops through all members to compute start times for all nodes with respect to the MANET channel. In particular, the associated start times are computed and stored in `md`. The duration of each slot is based on the radio configurations of each node. The value `md` is also the slot time for the upcoming node and represents the end of the entire *Neighbor Update* cycle. In Line 7, the controller inserts id and the associated start time (n and $\texttt{ns}_n$) as tuple in T_m. These slot times consider all header overhead and apply guard times of 8.25 bit, additional 6 bit for setup, and tear down of the transmitter to each slot in order to avoid interference of two neighboring slots [Kuk18].

The next step is to update the slots when each MANET participant accesses the uplink channel. First, the end time of the neighbor update cycle is set as the starting time stamp for the upcoming topology update duration on the uplink channel. Similar to the previous loop, each start time of the slot is stored in $\texttt{ts}_n$, associated to each node and stored in T_u combined with the corresponding node id n. Unlike the neighbor update cycle, determining the slot durations requires accounting for the number of MANET participants involved. In the worst case, each MANET node is connected to all other participants via the MANET channel, which is referred to a fully meshed topology. This increases the size of *Topology Updates* in B as each node stores every other participant as a direct neighbor and transmits this neighborhood information to the controller. Function computeSlotStart considers this and assumes the worst case to avoid time-driven slot overlaps.

At the end, `rs` is set to 0 as all new MANET nodes are now considered in the current slot computation process of Re-CeTUP. The controller delivers both T_m and T_u via the uplink to the MANET using the *Topology Refresh* message. Each node picks the start time associated to its id and starts transmitting its *Neighbor Update* and *Topology Update* in the given

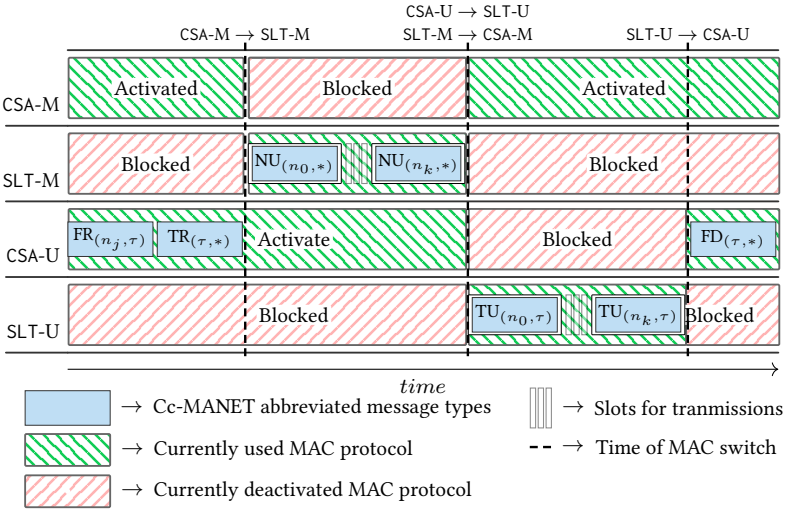


Figure 5.5: Dynamic MAC switching strategy to achieve TDMA provided update cycles and CSMA/CA for access management.

slot.

TDMA is a MAC technique which relies on preconfigurations to allocate the length of slots and to also optimize spatial reuse. The controller defines the start and end time of Re-CeTUP, meaning the time the first *Neighbor Update* starts and the time the last *Topology Update* is delivered via the uplink. It is mandatory to know when the last *Topology Update* arrives at the controller since path computation must take place immediately thereafter. The parameter ud determines the time the last *Topology Update* arrives at the controller which varies on the number of nodes. Packet processing times on nodes [Zil+17] and transmission durations of signals are considered during computations for all slots.

5.3.4 MAC Switching at Runtime

This section discusses the dynamic MAC switching between TDMA [Rod06, Section 14.7] and CSMA/CA [IEEE802.11] of MANET nodes and the controller during the reactive topology update process, respectively.

Figure 5.5 highlights MAC protocol switching times of all MANET participants and also shows time spans in which channels are free and blocked

regarding data delivery. Each swimlane represents TDMA and CSMA/CA protocol usage of either the MANET or the uplink channel of an arbitrary node $n \in \mathcal{N}$ and the controller τ. Protocol switches from one to another MAC technique only take place if MANET topology update occurs. Otherwise, the uplink and the MANET channel organize channel access via CSMA/CA.

The first two swimlanes from top to bottom represent the MANET channel (CSA-M and SLT-M). CSA-M stands for CSMA/CA which defines that each MANET participant uses CSMA/CA to send and receive frames within the MANET. The term SLT-M defines that each participant accesses the MANET channel via TDMA. The last two swimlanes form the uplink channel (CSA-Us and SLT-U). For instance, swimlane CSA-M represents the MAC protocol CSMA/CA used for the MANET channel. The other two uplink channel techniques are configured equally.

Cc-MANET used message types are abbreviated for this section. For instance, *Flow Request* is abbreviated with $\text{FR}_{(n_j,\tau)}$. Indices refer to the origin o and target where $o, t \in \mathcal{N}$of the message. With respect to the example $\text{FR}_{(n_j,\tau)}$ also appearing in Figure 5.5, node $n_j \in \mathcal{N}$, where $0 \leq j < |\mathcal{N}| - 1$ transmits a *Flow Request* to τ. The parameter τ represents the id of the controller. Also, the addressing scheme $(n_j, *)$ defines a message from node n_j to all nodes in $\mathcal{N}$ (broadcast message), whereas (n_j, n_k) is a unicast message from n_j to n_k, both $\in \mathcal{N}$ and $0 \leq j, k < |\mathcal{N}|$.

According to Figure 5.5, a node requests a route and transmits a $\text{FR}_{(n_j,\tau)}$ to the controller via the uplink. The controller replies with a $\text{TR}_{(\tau,*)}$, representing a *Topology Refresh*, which each node of the MANET receives via the CSA-U. At this point, uplink and MANET channels operate both with CSMA/CA as data transmission takes place at this time.

A *Flow Request* typically follows a *Topology Refresh*, initiated by the controller triggering nodes to gather neighborhood information and to report their gained knowledge to the controller. A *Topology Refresh* contains specific slot start times for each participant and general start and stop times of the neighbor update cycle and topology update cycle. Based on this information, nodes switch MAC protocols accordingly when receiving the broadcast $\text{TR}_{(\tau,*)}$. In particular, when receiving *Topology Refresh* via the uplink channel, nodes deactivate CSA-M and activate SLT-M. With respect to Figure 5.5, each $n \in \mathcal{N}$ switches to SLT-M when the *Topology Refresh* has successfully been received, as depicted by the first black dashed vertical line. Ongoing transmissions via the CSA-M protocol are queued during the neighbor update time period. The CSA-U can still be used for further

Flow Requests by MANET nodes until the topology update cycle starts as this channel is idle until then.

After this MAC switch took place, all participants transmit *Neighbor Updates*, and actualize their sub-topologies when their allocated slot starts. This process continues till the neighbor update cycle ends. In this figure, $\text{NU}_{(n_0,*)}$ to $\text{NU}_{(n_k,*)}$, which stands for a *Neighbor Update*, represents the transmission of *Neighbor Updates* of all registered nodes $\mathcal{N} = \{n_0, \ldots, n_k\}$, where $k = |\mathcal{N}| - 1$ at its allocated time slot.

Thereafter, nodes switch MANET MAC protocols back from SLT-M to CSA-M and continue delivering data when the general neighbor update stop time is reached, which is illustrated by the second black dashed vertical line. The end of the neighbor update cycle also defines the start of the topology update cycle. Each node activates the SLT-U and deactivates CSA-U to deliver their sub-topologies to the controller. In doing so, each node transmits the *Topology Update* via the uplink channel to the controller when the allocated slot starts. The controller receives the sub-topologies of each registered node during this cycle. Similar to the *Neighbor Update* cycle, $\text{TU}_{(n_0,\tau)}$ to $\text{TU}_{(n_k,\tau)}$, which stands for a *Topology Update*, represents the set $\mathcal{N} = \{n_0, \ldots, n_k\}$, where $k = |\mathcal{N}| - 1$.

The network graph is constructed by the controller followed by path computations when a topology update cycle runs out. Now, nodes again switch from SLT-U to CSA-U to be able to receive the subsequent *Flow Distribution* message, abbreviated with $\text{FD}_{(\tau,*)}$.

In rare occasions, nodes attempt to transmit *Flow Requests* when the topology update cycle is ongoing. This happens if these *Flow Requests* are stored in MAC queues at the time the switch from CSA-U to SLT-U happens. In such a case, the finite state machine of the DCF of CSMA/CA has already determined further actions of these frames, which is why these *Flow Requests* are exclusively deleted to not interfere with any *Topology Update* message.

Figure 5.5 discussed MAC protocol switches of MANET nodes. There is no need to implement and provide a further MAC protocol in addition to CSMA/CA on the controller. During data delivery, the controller is free for reception of *Flow Requests* via CSMA/CA. If the controller does not use TDMA during both update cycles, it must be ensured that the CSMA/CA protocol does not send ACKs specifically for successfully received *Topology Updates*. These messages could interfere with the topology update cycle.

5.3.5 Path Recomputation at Runtime

As a recap, nodes transmit *Flow Requests* if a new data stream must be delivered or a lost connection is detected. The Re-CeTUP process does not constrain MANET participants to stay idle until the corresponding *Topology Refresh* arrives. Several *Flow Requests* may follow after the initial one also while the neighbor update cycle is already ongoing. Interference between any *Neighbor Update* and a potential *Flow Request* is not possible as messages propagate on different channels. *Flow Requests* that arrive during a neighbor update cycle are queued and considered for the subsequent path computation, which starts after the topology update cycle ends.

It may happen that a new Re-CeTUP process starts immediately after the previously one finished. This is the case if a new *Flow Request* arrives at the controller after the current Re-CeTUP process delivered the *Flow Distribution*. The number of *Flow Requests* increases with the number of flows currently deployed on the MANET. Re-CeTUP stops data delivery for at least the neighbor update cycle time. This decreases the throughput and is more likely to drop data frames as MAC queues exceed their capacities.

To decrease the number of recognized lost connections, the controller recomputes and renews paths of other existing flows. Recomputation of already ongoing transmissions on the MANET may change the paths as nodes are moving, generating new topologies and routes towards requested destinations. This decreases the amount of transmitted *Flow Requests* because all routes are renewed before they might become obsolete.

5.4 RESULTS

The previous section discussed Re-CeTUP, which uses a TDMA technique to exchange topology information without interference. We also highlighted the integration of Re-CeTUP in the Cc-MANET architecture, switching form the proactive update mechanism to the reactive one.

This evaluation investigates at first the extent, the TDMA improves upon the previously applied protocol CSMA/CA focusing on runtime and completeness in terms of nodes and their connections. Secondly, Re-CeTUP is evaluated against Pro-CeTUP to determine the impact of an up-to-date topology, focusing on data delivery rates and routing load.

5.4.1 Comparing Channel Access Methods for Update Process

This section compares runtimes in relation to completeness of Re-CeTUP configured with CSMA/CA and with TDMA. As mentioned at the beginning of Section 5.3, plain Re-CeTUP is meant to be configured with TDMA. We compare whether connections delivered to the controller are actually present and also if connections in the actual MANET are missing in the representation of the controller. This evaluation investigates both update techniques in isolation and compares the results. Therefore, no mobility or data transmission is happening during the comparison.

With respect to the findings of Section 5.1, CSMA/CA experiences interference leading to lost topology information. Thereby, the number of successfully delivered frames decreases with increasing number of participants. The competing logic of CSMA/CA reacts to interfered frames and minimizes the probability of simultaneous channel access in increasing the CW when the previous attempt was unsuccessful. This decreases interference during consecutive attempts. With an increased CW, the probability increases that nodes pick higher backoffs resulting in longer waiting periods before attempting transmissions. We experimented with different backoff configurations aiming to achieve the most complete topology representation with the smallest possible CW. Test runs revealed an optimal CW of 511.

Based on that, we compared Re-CeTUP$_{31}$, Re-CeTUP$_{511}$, and plain Re-CeTUP with each other, focusing on execution times and the number of correctly reported connections. Re-CeTUP$_{31}$ stands for Re-CeTUP configured with CSMA/CA, operating with a CW of 31. Re-CeTUP$_{511}$ defines Re-CeTUP also running CSMA/CA having a CW of 511. Both Re-CeTUP$_{31}$ and Re-CeTUP$_{511}$ update the topology to the controller immediately before routing starts. However, all participants access the channel via CSMA/CA during the neighbor update and topology update cycles. A more detailed description is provided in Section 5.1.

We compared the three approaches starting from 4 to 60 nodes randomly placed on a $300\ m \times 300\ m$ playground. The MANET interface exhibits data rates of $2\ Mbps$, reaching communication ranges of $150\ m$. The communication range of the uplink channel ranges up to $35\ km$ and achieves data rates of up to $1\ Mbps$. For each initial configured number of nodes our simulation operates as follows: Nodes are randomly placed on the mentioned playground size. Runtime measurements start when the controller transmits the *Topology Refresh*. Measurements stop when the

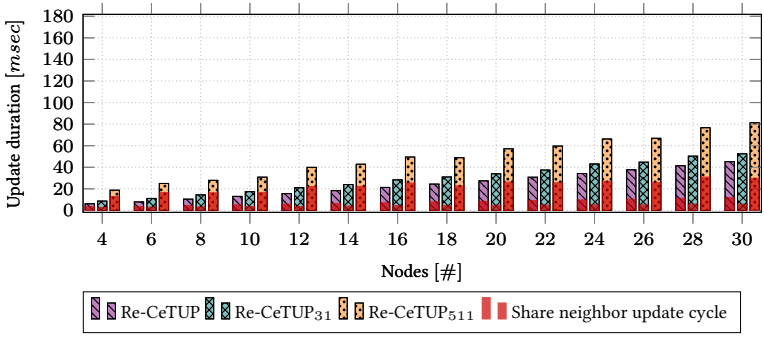


Figure 5.6: Cumulated runtimes of Re-CeTUP, Re-CeTUP$_{31}$, and Re-CeTUP$_{511}$ split in neighbor update and topology update cycles from 4 to 30 nodes.

controller receives the last *Topology Update*. All configurations ran 200 times to average out measurement errors.

Figures 5.6 and 5.7 show runtimes of Re-CeTUP, Re-CeTUP$_{31}$, and Re-CeTUP$_{511}$ partitioned in neighbor update cycles and topology update cycles. Bars show the sum of neighbor update and topology update cycles of each approach whereas the red bars depict the duration of neighbor update cycles exclusively. Re-CeTUP outperforms the two other approaches that have CSMA/CA equipped with respect to total runtimes.

CSMA/CA provides simultaneous transmissions of *Neighbor Updates* if two participants are out of each other's communication range which decreases the execution times of approaches. Unicast transmissions with CSMA/CA recognize interfered transmissions in a self-organized way and increase the CW to avoid simultaneous transmissions. However, broadcast fashion does not consider ACKs and CTSs, only competing for channel access leaving no clue whether each MANET node in communication range has received the *Neighbor Update*. The more nodes, the more channel utilization is sensed by participants attempting to start *Neighbor Updates*, resulting in restarted DIFSs and backoffs whereas configured CWs remain. Restarted contention processes increase runtimes of neighbor update and topology update cycles but only slightly because of the firmly defined CW.

Re-CeTUP$_{31}$ takes more time until the last participant delivers the *Topology Update* to the controller compared to Re-CeTUP. However, runtimes of the neighbor update cycle of Re-CeTUP$_{31}$ are lower compared to Re-

CeTUP, with respect to Figures 5.6 and 5.7 whereas Re-CeTUP$_{511}$ takes the most time to finish the neighbor update cycle. Smaller backoff periods obviously obtain a noteworthy impact on runtimes outperforming Re-CeTUP regarding the neighbor update cycle whereas in total Re-CeTUP updates the topology the fastest with respect to all number of nodes.

The share of topology runtimes increases with increasing number of nodes. The message size in *bit* increases with the number of nodes as more neighbors exist and are stored in *Topology Updates*. The limited transmission capacity of the uplink increases the transmission times compared to the MANET channel.

Next, we compare the number of reported neighbors that form the topology representation of the controller of all 3 approaches, as shown in Plot 5.8. Re-CeTUP reports the most neighbors to the controller followed by Re-CeTUP$_{511}$ and Re-CeTUP$_{31}$. TDMA guarantees interference-free data delivery as long as guard times provide sufficient space to prevent overlaps. This requires accurate computation of arising frame processing times on nodes which depends on hardware components and their utilization at runtime. Also the arising transmission times that depend on data rates, antenna specifications, and accurate clock synchronization must be included for slot time computation.

Applying TDMA is appropriate in this use case because the size in *bit* of *Topology Update* per node is known in advance which makes slotted channel access even more reasonable. Consequently, the controller is able to allo-

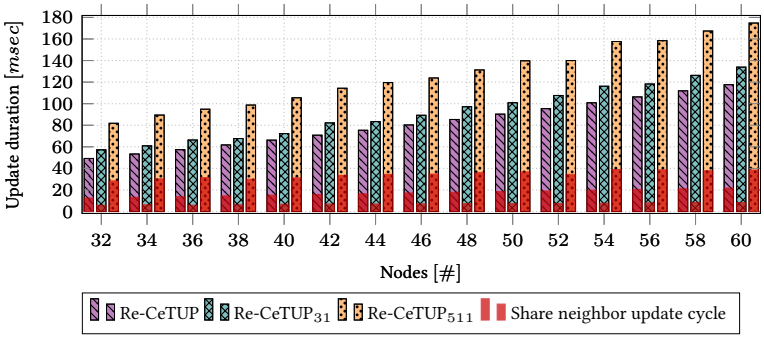


Figure 5.7: Cumulated runtimes of Re-CeTUP, Re-CeTUP$_{31}$, and Re-CeTUP$_{511}$ split in neighbor update and topology update cycle from 32 to 60 nodes.

cate channel capacity almost without waste of resources. These facts explain why Re-CeTUP outperforms Re-CeTUP$_{31}$ and Re-CeTUP$_{511}$ in terms of the number of reported neighbors. As previously explained, increasing the CW to 511 decreases the probability of simultaneous *Neighbor Updates* and *Topology Updates* resulting in more complete topology representation compared to Re-CeTUP$_{31}$.

To sum up, CSMA/CA as channel access method is inappropriate if aiming for an up-to-date topology snapshot on the controller in which each actual existing connection must be represented in the controller's topology. Re-CeTUP applied with both CSMA/CA configurations either delivers incomplete topology information when heading for runtimes that reach Re-CeTUP's execution times or requires significantly more time to finish the topology update while still losing topology information. Using CSMA/CA with CW set to 31 for topology update provides incomplete topology information when trying to achieve execution times similar to Re-CeTUP. By contrast, setting the CW to 511 takes significantly more time to complete the topology update while still losing topology information. Re-CeTUP, by contrast, exploits channel capacity most efficiently resulting in the lowest runtimes and complete topology knowledge.

5.4.2 Re-CeTUP and Pro-CeTUP in Action

Section 5.1 proves that the controller often computes paths that do not exist in the actual topology using the reactive topology update concept with CSMA/CA. Pro-CeTUP computes paths based on an obsolete MANET representation as the topology update interval is configured to 1 sec.

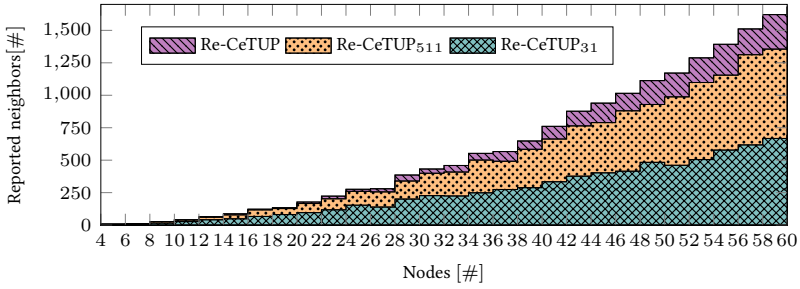


Figure 5.8: Reported neighbors to the controller of Re-CeTUP, Re-CeTUP$_{31}$, and Re-CeTUP$_{511}$ from 32 to 60 nodes.

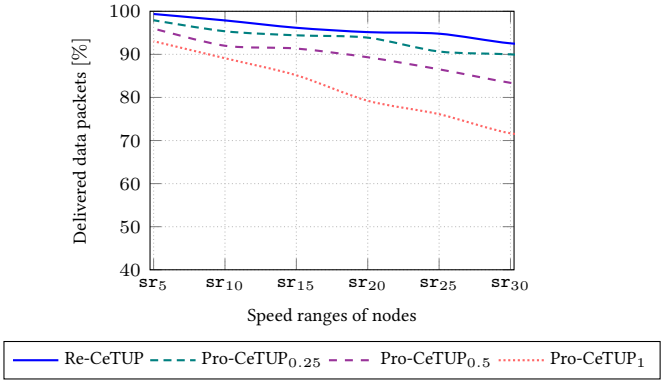


Figure 5.9: Delivered data in relation to sent data in percentage of Re-CeTUP, Pro-CeTUP$_{0.25}$, Pro-CeTUP$_{0.5}$, and Pro-CeTUP$_1$.

We investigate how the obsolete topology information influences data delivery and compare the results with Pro-CeTUP. To be more precise, this section discusses the packet delivery fraction and routing load of Re-CeTUP and Pro-CeTUP. In addition, we want to answer the question to which extent shorter update intervals of Pro-CeTUP increase data delivery with the expense of generated routing load. Therefore, we evaluate Pro-CeTUP with neighbor update and topology update intervals of 1 sec (Pro-CeTUP$_1$), 0.5 sec (Pro-CeTUP$_{0.5}$), and 0.25 sec (Pro-CeTUP$_{0.25}$) and also compare the results with Re-CeTUP equipped with TDMA. We apply equal velocities $V = \{\mathtt{sr}_5, \mathtt{sr}_{10}, \mathtt{sr}_{15}, \mathtt{sr}_{20}, \mathtt{sr}_{25}, \mathtt{sr}_{30}\}$ as defined in Section 5.1. The MANET playground is set to 550 $m \times 250$ m. 60 nodes are randomly placed on that playground and equipped with radios reaching 150 m, and data rates up to 2 $Mbps$. The uplink channel provides 0.7 $Mbps$ data rate. Three randomly selected nodes transmit data to arbitrarily elected destinations throughout each run. Each run lasts 17 min. We ran each of these configurations with each speed range 10 times to average out measurement errors.

Figure 5.9 depicts successfully delivered data in percentage in relation to the sent payload. Re-CeTUP outperforms all interval configurations of Pro-CeTUP with respect to all speed ranges. Pro-CeTUP$_{0.25}$ achieves data delivery rates almost comparable to Re-CeTUP even with simulation configurations where node speeds exceed 20 km/h. The most common reason for data loss is if path participants lose connectivity to the next hop. In this

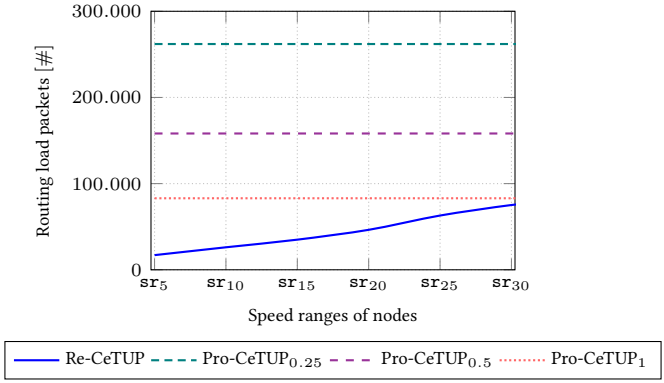


Figure 5.10: Routing load of Re-CeTUP,Pro-CeTUP$_{0.25}$, Pro-CeTUP$_{0.5}$, and Pro-CeTUP$_1$ with increasing speed ranges.

case, nodes continue to forward data based on the entries in their routing tables. Since the data stream is still transmitted but not delivered to the next hop, frames are dropped at nodes as long as no new route is reported by the controller. Active link breaks of path participants increase with increasing speed ranges. Consequently, the packet delivery fraction decreases when update intervals take too long to actualize the current topology, referring to results of Pro-CeTUP$_{0.5}$ and Pro-CeTUP$_1$.

So far, Re-CeTUP instantly corrects lost connections in the topology representation which is why Cc-MANET architecture equipped with this update technique experiences almost no data loss, closely followed by Pro-CeTUP$_{0.25}$.

The short update interval increases the routing load tremendously, as shown in Figure 5.10. We classified all Cc-MANET defined message types as routing load. This includes all topology update messages excluding *Flow Requests*, and *Flow Distributions*. Routing load of all Pro-CeTUP configurations are as assumed to be equally independent of the speed of nodes as interval cycles do not change during simulations. Re-CeTUP increases the routing load with increasing speed as more connections break, resulting in triggered topology updates. This increases the routing load. Pro-CeTUP configurations would generate even more routing load because nodes continue sending *Flow Requests* until the next neighbor update and topology update period starts since these messages are not included in the measurements. This would also increase the routing load of Re-CeTUP slightly.

5.5 SUMMARY

This chapter answered RQ 2, tackling an up-to-date topology representation on an outsourced controller. Proactive Cc-MANET architectures lack the timeliness of the topology representations due to interval-driven update periods. This results in computed routes not present in the actual MANET topology. Shortening the update intervals of the proactive topology update technique improves completeness of the topology representation at the expense of increased routing load.

We first outlined the drawbacks of the proactive topology update process Pro-CeTUP regarding protocol processes and topology reliability. In particular, we measured topology actuality and concluded that topology representations become outdated in the worst case resulting in 50% wrongly computed routes with increasing speed.

Our proposed reactive topology update process Re-CeTUP leverages TDMA during network updates forming an up-to-date topology snapshot. TDMA is convenient, guaranteeing interference-free and very efficient channel utilization, covering the objective to deliver topology information in minimum runtime. In that context, we proposed a Participant Management model maintaining registered MANET nodes to avoid unused slots and also to convey radio configurations for slot computations by the controller. Results emphasize improvements of Re-CeTUP compared to Pro-CeTUP regarding runtime and up-to-date topology knowledge.

So far, each lost transmitting connection and desired data transmission results in a new *Flow Request*. The more data transmissions are ongoing and with increasing node speeds, the more nodes start *Flow Requests* as transmitting connections are more likely to break. No rule limits the number of complete topology updates if *Flow Requests* arrive one after another at the controller. This increases the routing load unnecessarily and could be limited if the current topology representation is up-to-date enough. An actuality threshold based on certain topology characteristics would eliminate consecutive topology update processes.

Also, the general Cc-MANET, having the controller outsourced, is unstable and vulnerable as controller reachability relies on a single uplink connection. Link failures can be global or local if specific MANET nodes lose connectivity. Single participants could experience hardware problems or further technical outtakes. Sub and in the worst case global outtakes must be taken into account due to jamming attacks. Various jamming attacks exist and are applied especially in the military field [Vad+16]. It is

obvious that the uplink is a single point of failure and an all-time connectivity seems unrealistic.

The following chapter proposes a topology update process, having the controller deployed on an arbitrary MANET member, reachable via the MANET channel if the uplink channel does not provide connectivity to the outsourced controller.

CHAPTER 6

Self-organized Topology Management

This chapter proposes an approach how to report an up-to-date topology network representation to a controller if the uplink channel is not available. A MANET participant takes over the controller role, applying the Reactive Self-organized Topology Update Process (Re-SoTUP) to gather the entire topology information if needed.

At first, we determine the timeliness of the complete topology snapshot of OLSR as representation for several proactive routing protocols designed for MANETs. Next, we introduce the reactive topology update process, called Re-SoTUPs, which is evaluated later and compared with OLSR. Finally, we propose a controller self-management process, called Self-organized Controller Management (SoCM), to guarantee controller maintenance and reachability at runtime.

6.1 PROACTIVE UPDATE RELIABILITY

Section 3.3, which discusses related work in this field, summarizes protocols and controller-managed MANET approaches focusing on full-topology knowledge. Existing routing protocols and SDN-like techniques are introduced heading for a complete network representation stored on participants. All these approaches, aiming for complete topology knowledge gather topology information proactively, making them inappropriate for the global research objective. Multi-flow routing of data rate demanding services generating highly-utilized MANETs requires accurate and up-to-date connection characteristics to not exceed any link when computing paths for multiple flows.

Because of that, we verify the quality of topology representations of the well-known proactive routing protocol OLSR [Cla+14]. In particular, we compare the topologies generated via OLSR with the actual network constellations and identify the number of missing links of OLSR. With OLSR,

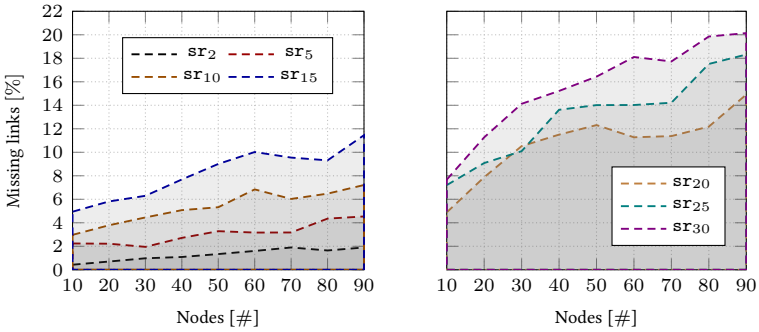
so-called MPRs periodically flood the MANET with topology control messages and also provide instant network information exchange if link breaks are detected. This makes OLSR more dynamic compared to FSR [PGC00] and GSR [CG98].

In order to determine the actuality of OLSR, we simulate the protocol in various MANET topology sizes combined with different participant speeds. We compare the topology information provided by OLSR with the current representation. In particular, we collect the link set table, the topology set table, as well as 1-hop, and 2-hop neighbor sets of an arbitrary participant, and construct the network graph. Any change in these mentioned sets triggers routing table re-computations. Intervals and holding time parameters are set to default values, as proposed by Gomez et al. [GGP05]. The network graph representations are calculated after $40\ sec$ simulation time to provide OLSR time to exchange control messages. During this period and beyond, participants exchange hello and topology control packets, populate their 1-hop and 2-hop neighborhood information based on them, and populate their topology tables.

In order to compare the topology representation of an arbitrary OLSR node with the actual topology, we collect coordinates of each participant also after $40\ sec$ simulation time and compute distances between the individual participants. A connection exists if nodes are within each others' coverage range which is set to $100\ m$, referring to the applied simulation configuration. Based on that, we construct the actual topology including nodes and their connections, which we compare with the constructed topology of an arbitrary node using OLSR.

Each MANET member is equipped with a IEEE 802.11 interface reaching $2\ Mbps$. Following the RWP [AWS06], nodes move straight ahead between $20\ sec$ and $40\ sec$ and thereafter change directions by randomly picking angles between $70°$ and $90°$. This moving pattern is repeated continuously by each node. We applied various speed ranges from $\mathtt{sr}_2$ to $\mathtt{sr}_{30}$ defined as follows: The speed range $\mathtt{sr}_2$ returns the value x, where $0\ km/h \leq x \leq 2\ km/h$ and $x \in \mathbb{N}$. Also, nodes pick a speed of x, where $0\ km/h \leq x \leq 30\ km/h$ and $x \in \mathbb{N}$ when speed range $\mathtt{sr}_{30}$ is defined. In particular, each node randomly picks a speed of the defined speed range and also randomly changes the direction after moving towards a direction for a randomly chosen timespan. Based on that, we define the velocity ranges $V = \{\mathtt{sr}_2, \mathtt{sr}_5, \mathtt{sr}_{10}, \mathtt{sr}_{15}, \mathtt{sr}_{20}, \mathtt{sr}_{25}, \mathtt{sr}_{30}\}$ for this evaluation.

A speed range is combined with a number of MANET nodes starting from 10 to 90 in steps of ten. The playground size increases with increasing



(a) Missing links of OLSR in percentage from 2 km/h to 15 km/h.

(b) Missing links of OLSR in percentage from 20 km/h to 30 km/h.

Figure 6.1: Comparing actual topology with OLSR representation regarding missing links with increasing number of MANET nodes and increasing speeds.

number of nodes from 250 $m \times 250$ m to 1050 $m \times 1050$ m. We run each combination of speed range and number of nodes 200 times to average out measurement errors.

Figures 6.1a and 6.1b show the topology representations of an arbitrary MANET node equipped with the routing protocol OLSR. In particular, both figures present missing links of OLSR in percentage, recorded with different speed ranges and an increasing number of MANET participants compared to the actual topology with all speed ranges and number of node configurations. With 2 km/h, OLSR misses almost no links in the representation while a minimal increase is observed with an increasing number of nodes. Increasing the speed ranges reduces the completeness of the topology and as a consequence increases the number of missing links.

Inconsistencies in the topology representation of OLSR also increase with increasing number of participants and equal speed range configurations. MPRs must forward topology control messages across the entire network to update routing tables of each node. The wider the network becomes the more hops topology control messages must be forwarded to reach all MANET members. That takes longer to spread new connections and generates inconsistencies in the topology representation on each node.

Overall, a OLSR node could assume that its predecessor of the route computes the same path if link costs and the routing topology representa-

tion are identical. However, based on the results, the topology representations stored on the participants are not the same. This means that MANET nodes route packets based on different routing information. United routing, aiming at the same objective could be accomplished if topology representations on each participant would be the same. Speed ranges from $\mathtt{sr}_{20}$ to $\mathtt{sr}_{30}$ reinforce inconsistencies of topology representations. In general, the number of missing links is remarkable and reduces routing options making multi-flow path computation in highly-utilized MANETs less successful if only a few path settings in the current topology representation exist that do not over-utilize the MANET.

It must be noted that nodes detect lost connections to neighbors if entries of their neighbor sets exceed the validity time which is three times the refresh interval. The refresh interval is set to 2 sec per default. In addition, OLSR characterizes links with a quality. Thereby, the protocol follows the hysteresis strategy, determining this quality and the associated validity of a link. Links are not considered for routing if their qualities are below a defined quality threshold. Furthermore, the associated entry in the 1-hop neighbor set and all associated entries in the 2-hop neighbor set are deleted by a node that detects a lost connection to a previously known neighbor. The MPR selector set must also be adapted in case the node is a MPR. According to the RFC recommendation, nodes should start hello messages before the associated timer runs out if 1-hop and 2-hop sets change. [Cla+14]

In general, the results restrict OLSR to networks, comprising minimum speeds and number of participants as missing links make routing unreliable.

To sum up, the proactive routing technique of OLSR makes multi-flow path computation for data rate-demanding services unreliable due to the following reasons: Over-utilized path settings might be computed by OLSR as actual existing and required connections are not present in the topology representation. Multi-flow path computations will have to restart several times as selected links might not exist anymore in the actual MANET topology constellation.

Therefore, it is necessary to design a topology update technique to eliminate these drawbacks, which is introduced in the following sections.

6.2 REACTIVE SELF-ORGANIZED TOPOLOGY UPDATE

Section 6.1 demonstrated the drawbacks of the proactive routing protocol OLSR [Cla+14] in terms of topology actuality when mobility comes into play. Missing and outdated links create reliable routing difficult, motivating

us to switch to a reactive routing fashion as the required topology knowledge is then gathered immediately before routing takes place.

So far, the reactive routing technique is not designed to provide full topology knowledge, only providing partial network representations distributed on multiple nodes. We introduce Re-SoTUP [Str+20], gathering full topology knowledge reactively to a controller deployed on a MANET participant. This closes the research gap and provides similar routing capabilities if no uplink channel is available to provide outsourced topology knowledge.

Re-SoTUP aims for minimum runtime while gathering an up-to-date snapshot for the controller. Firstly, the general architecture and message types are introduced in Section 6.2.1. Secondly, the functionality and operation of Re-SoTUP are discussed briefly to better understand the integration of this algorithm in the Self-controlled MANET (Sc-MANET) architecture. Thirdly, we partition Re-SoTUP into two parts, called topology detection and topology reporting and deep dive into both parts in Sections 6.2.3 and 6.2.5.

6.2.1 Framework Components

The network architecture only comprises the MANET, operating with one wireless communication channel. This MANET channel is used for data transmissions and control overhead, respectively.

Figure 6.2 depicts the Sc-MANET architecture. This figure comprises the MANET and controller components. The controller is placed on an arbitrary participant, which means that the controller is placed inside the MANET All red-colored messages are newly defined for this algorithm. The network example shows a MANET comprising 5 nodes. Participant 0 owns the controller role. This role extends the controller node with **Topology Management** and **Utilization Model** capabilities. The first represents Re-SoTUP, aiming for up-to-date topology knowledge. The second provides accurate transmission capacity utilizations of links, described in Section 4.2.5. The blue area covers these extra controller capabilities, assigned exclusively to one MANET participant, owning the controller role.

All MANET participants, including the controller, record their channel utilization by applying the module **Link Utilization** which is defined and evaluated in Section 4.2.3. This is identified by the orange area which surrounds all MANET participants.

When applying Re-SoTUP in the Sc-MANET architecture, an arbitrary

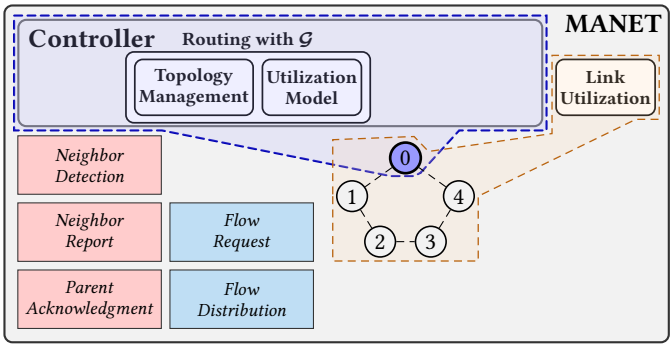


Figure 6.2: Sc-MANET architecture, deploying the controller on a participant, which applies Re-SoTUP to gather complete topology knowledge. Participants utilize a single wireless channel for data and protocol transmissions.

node starts a *Flow Request* addressed to the controller if the MANET member desires to receive a route for the requested destination. In this case, nodes use the MANET channel to deliver the message, which is forwarded via network participants. This requires a route to the controller for each MANET participant to request a route from the controller, which is described in Section 6.4.

When receiving the *Flow Request*, the Controller starts Re-SoTUP. This triggers successive *Neighbor Detections* and associated *Parent Acknowledgements*, forming a tree to the controller as root and also collecting directly reachable neighbors. This phase of Re-SoTUP is called topology detection. Thereafter, participants report their sub-topologies to the controller with *Neighbor Reports* along the branches starting with the leaves of the tree. This phase of Re-SoTUP is called topology reporting. The algorithm finishes when the last directly connected child of the controller reports its sub-branch. Now, the controller computes the paths of requested flows based on the gathered topology and informs all MANET members about new routes with a *Flow Distribution* via the commonly used MANET channel.

As already mentioned, the controller is included in the set of MANET participants, meaning $\tau \in \mathcal{N}$. Because of that, the controller node treads itself as a potential path participant when computing routes from origins to targets. In particular, the controller can be the origin or target of a flow,

Table 6.1: Global parameters and variables used in multiple RQs.

Properties	Belonging	Description
n	$n \in \mathcal{N}$	Unique id of a MANET participant.
τ	$\tau \in \mathcal{N}$	Unique id of the controller.
o^f	$o \in \mathcal{N}, f \in \mathcal{F}$	Origin node of a flow.
t^f	$t \in \mathcal{N}, f \in \mathcal{F}$	Target node of a flow.
p^f	$p \subseteq \mathcal{N}, f \in \mathcal{F}$	Path as sequence of nodes of a flow.
d^f	$f \in \mathcal{F}$	Demanded data rate of a flow.
u_l	$l \in \mathcal{L}$	Current utilization of a link.
c_l	$l \in \mathcal{L}$	Capacity of a link.

monitor the transmission utilization, and store the configured transmission capacity $c_{(\tau,m)}$, where $(\tau, m) \in \mathcal{L}$. This defines, that a graph, representing a MANET belongs to a controller identified by its id. This definition differs compared to the Cc-MANET architecture of RQ 2, where the controller is reachable via a dedicated channel and as a consequence is not included in the set of nodes. Addressing this definition, the MANET participant currently having the controller role is identified by τ if the current situation is about a single MANET. Furthermore, the graph $\mathcal{G}$ and all sets $\mathcal{N}$, $\mathcal{L}$, and $\mathcal{F}$ belong to the controller τ if not further specified.

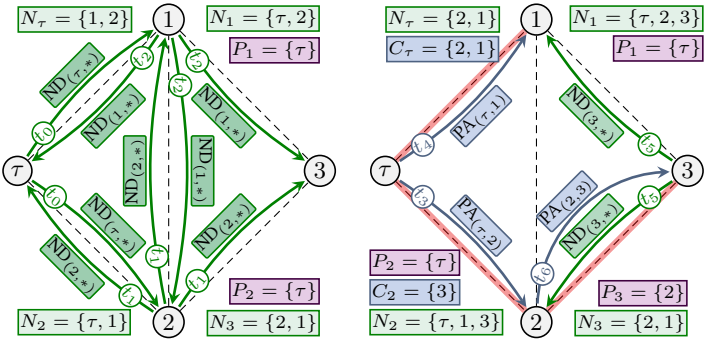
Table 6.1 shows all parameters including this definition. The red-colored row depicts the newly defined parameter.

The following sections introduce algorithms and data structures of Re-SoTUP required to construct a complete topology representation.

6.2.2 Operation and Structure

Re-SoTUP is designed for MANET members accessing the channel with CSMA/CA. Referring back to Re-CeTUP, discussed in Section 5.3, this technique, in-theory, does not miss any member and connection during the topology update process as clocks are synced and each participant utilizes the allocated slot respectively to deliver the topology information.

Slot allocation techniques in self-organized multi-hop networks, such as MANETs, are challenging regarding spatial slot reuse and control information exchange. MANETs must avoid allocating the same slot to multiple nodes within a 2-hop neighborhood to guarantee interference-free transmissions. The challenge is to define the least number of slots for each 2-hop



(a) Broadcasting *Neighbor Detections* from the root to the leave while forming the tree.

(b) Delivering *Parent Acknowledgements* along the tree back to the controller node.

Figure 6.3: Gathering one-hop neighbor topology per MANET participant and tree construction during topology detection.

region for the most possible data throughput that is NP-complete [Llo02]. Distributed slot allocation is error-prone, especially with increasing mobility, and requires computations and as a result decreases battery levels while CSMA/CA provides self-managed and interference-free channel access due to the provided carrier sense technique, which is why participants use this access technique for Re-SoTUP.

The controller node starts Re-SoTUP immediately before all routes are recomputed in order to determine paths on an up-to-date topology representation. Re-SoTUP is divided into two phases, named topology detection and topology reporting. The first phase applies a special flooding technique for reliable neighbor detection while constructing a tree with the controller as root. The second phase reports gathered sub-topologies along each branch to the controller.

Figure 6.3a depicts *Neighbor Detection* message flooding, abbreviated with $\mathrm{ND}_{(o,*)}$, where $o \in \mathcal{N}$ and $*$ stands for all nodes $\in \mathcal{N}$. The addressing scheme (o, t) represents a unicast transmission comprising the origin and the target of the message in round brackets, where $o, t \in \mathcal{N}$. In general, the abbreviated addressing scheme is the same as in Section 5.3.4.

The algorithm starts when the controller transmits a $\mathrm{ND}_{(\tau,*)}$ containing the unique id. This message reaches all directly connected nodes which populate their neighbor sets with the origin id of this *Neighbor Detection*

message. In particular, any node $n \in \mathcal{N}$ maintains the set of neighbors N_n, where N_n comprises all origin ids, node n already received a *Neighbor Detection* message. Next, each node immediately broadcasts a *Neighbor Detection* containing its own id when this node receives a *Neighbor Detection* for the first time.

A *Neighbor Detection* has two purposes: First of all, it informs nodes about the presence of the sender identified by the id. Nodes learn about their directly connected neighbors with this message. In particular, the target nodes of this message know that they can receive and decode the message sent by the origin. Second of all, we leverage this message to connect a child to its parent (origin of the *Neighbor Detection*) in order to construct the tree comprising the controller as root. Therefore, each participant $n \in \mathcal{N}$ maintains a set of parents P_n and also stores the next hop towards the controller in the parameter p_n. Nodes obtain the next hop towards the root from the first received *Neighbor Detection* and store the id of the origin in p_n.

With respect to Figure 6.3a, nodes 1 and 2 receive the $\text{ND}_{(\tau,*)}$ from the controller and insert τ in their set of neighbors N_1 and N_2, respectively. Also, both declare τ to their priority 1 parent, storing the id (τ) in p_1 and p_2 and in addition insert the controller id in their set of parents P_1 and P_2, respectively. The set of potential parents $P_n \subset \mathcal{N}$, where $n \in \mathcal{N}$ and $n \notin P_n$ comprises additional nodes which might be chosen as the next new parent if the current parent stored in p_n is not reachable. Further potential parent ids are received at runtime and inserted in P_n.

Broadcast messages, such as *Neighbor Detections*, are likely to get lost due to simultaneous transmissions of nodes in close vicinity. It is important to connect to an upstream parent to deliver the entire sub-topology of this branch to the controller. Because of that, nodes collect multiple parents and repeat sending *Neighbor Detections* if the desired parent does not acknowledge the message within a defined time window. The parent election process is introduced in Section 6.2.4.

At t_1 and t_2, nodes 1 and 2 also broadcast a $\text{ND}_{(1,*)}$ and $\text{ND}_{(2,*)}$ containing p_1 and p_2. Their messages reach node 3, also populating the set of neighbors N_3.

Nodes reply with *Parent Acknowledgements*, abbreviated as $\text{PA}_{(m,n)}$, where $m, n \in \mathcal{N}$ if they receive a *Neighbor Detection* and are elected as the parent node by n ($p_n = m$). This confirms the parent-child connection between nodes m and n. In contrast, each parent m maintains a set of children C_m, where $m \in \mathcal{N}$ to track, from which node they will receive a

Neighbor Report. This message contains a sub-graph of the complete network representation.

With respect to Figure 6.3b, the controller node receives *Neighbor Detections* of nodes 1 and 2 storing τ as parent ($p_1 = \tau$) and ($p_2 = \tau$), respectively. The parent answers with *Parent Acknowledgements* directly addressed to both nodes. Also, τ inserts nodes 1 and 2 in the set of children C_τ. Nodes 1 and 2 now show an edge (displayed as red transparent line) to τ.

This represents that both nodes are registered as children at their parent at t_3 and t_4. At t_5 node 3 broadcasts its initial *Neighbor Detection* containing node 2 as p_3 ($p_3 = 2$). Node 2 also replies with a *Parent Acknowledgement* confirming the parent-child connection.

The topology report phase follows the bottom-up principle. To begin, nodes identify themselves as leaves if they are not connected to children. Each MANET participant having no children registered starts a timer after receiving a *Parent Acknowledgement*. Nodes start the report process and transmit *Neighbor Reports* if they do not receive a *Neighbor Detection* containing their id before the timer runs out.

Any node $n \in \mathcal{N}$ constructs an adjacency set $L_n = \{(i,j) \in \mathcal{L} | j = n\}$, where node i is the originator of all previously received *Neighbor Detections*. Each sink node j of link (i,j), where $(i,j) \in L_j$ in addition obtains the current utilization $u_{(i,j)}$ and further parameters, such as the link quality, discussed in Section 7.3.4, which are later reported to the controller. Leaves include their adjacency sets in their *Neighbor Reports*.

Parents unite all neighbor adjacency lists received from their branch and include the sub-topology in their *Neighbor Report* before forwarding this message to their parent. Parents wait for all their children to deliver their *Neighbor Reports* until they forward their own *Neighbor Report* to their upstream parent. Each node merges the branches and forwards this information towards the root of the tree. The entire topology reporting process is described in Section 6.2.5.

According to Figure 6.4, node 1 transmits at t_7 a *Neighbor Report*, abbreviated as $\text{NR}_{(1,\tau)}$ to the controller as no further participant elected this node as its parent. At t_8, node 3 follows and also transmits $\text{NR}_{(3,2)}$. Node 2 unites both adjacency lists, resulting in $L_2 = \{(\tau,2),(1,2),(3,2),(2,3)\}$. The set of links is included in $\text{NR}_{(2,\tau)}$ and delivered to the controller node τ. The algorithm finishes when τ receives *Neighbor Reports* from nodes 1 and 2.

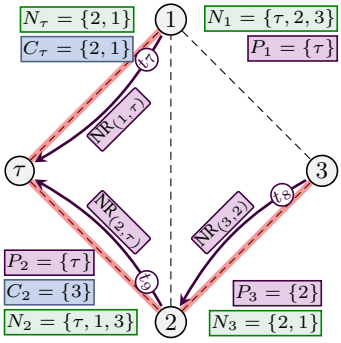


Figure 6.4: Re-SoTUP topology report phase delivering sub-topology knowledge to all parents towards the controller node.

```
1  begin NeighborSensing(ND_(n,*))
2      N_m = N_m ∪ {n}
3      if |N_m| = 1 and m ≠ τ then
4          p_m = n
5          send(ND_(m,*))                    // ND_(m,*) stores p_m = n.
6          startTimer(at_m)
7      else if p_n = m then
8          C_m = C_m ∪ {n}
9          send(PA_(m,n))
10         stopTimer(rt_m)
11     else if C_m = ∅ and hp_m = 1 then
12         startTimer(rt_m)
13     end
14 end
```

Algorithm 4: Node m receives $\mathrm{ND}_{(n,*)}$ from node n.

6.2.3 Topology Detection

This section explains the process nodes follow when receiving a *Neighbor Detection* from another node in the MANET. Algorithm 4 shows all actions when node m receives a $\mathrm{ND}_{(n,*)}$, where $m, n \in \mathcal{N}$.

To begin, node m inserts node n in the list of neighbors N_m. The reception of a *Neighbor Detection* creates exactly three situations: (i) Node m receives a *Neighbor Detection* for the first time and is not the controller

node (Line 3). (ii) The node m is elected as parent of the sender node n of the *Neighbor Detection* (Line 7). (iii) The node m is already connected to a parent node and has no registered child when receiving the *Neighbor Detection* from node n (Line 11).

In case the first condition is true, node m elects node n as its parent, as this is the first *Neighbor Detection* node m has received so far. In particular, node m sets n to be its desired parent $p_m = n$ and includes this parameter in its $\text{ND}_{(m,*)}$. Next, m broadcasts the $\text{ND}_{(m,*)}$ in order to inform neighbors about its presence and also to connect to the desired parent n, which will also receive this message. It happens that broadcast messages get lost, which is why node m starts the acknowledgement timer $\texttt{at}_m$ triggering the node to broadcast a *Neighbor Detection* again when the timer runs out and no response from the potential parent n has been received so far. In general, nodes start this timer each time they send a *Neighbor Detection*.

The acknowledgment timer represents the time of sending a *Neighbor Detection* including the subsequent reception of a *Parent Acknowledgement*. More precisely, $\texttt{at}_m = t(\text{ND}_{(m,*)}) + t(\text{PA}_{(n,m)})$, where $m, n \in \mathcal{N}$ adds up the durations when transmitting both messages. The function t returns these transmission durations, which implies runtimes of the complete CSMA/CA overhead. Transmission of both messages considers backoff, DIFS time, and theoretical propagation time based on the bit size of messages, which considers broadcast and unicast messages. The theoretical time of unicast messages also considers the overhead for RTS, CTS, and ACK frame transmissions as well as SIFS durations between consecutive frames.

The second condition is true if node m is selected as the parent of the sender n (Line 7). To begin, m inserts or updates n in its list of children C_m. Next, the parent-child connection request must be answered, which is why m responds with the unicast message $\text{PA}_{(m,n)}$. In this case, Re-SoTUP defines a report timer. This timer is used by nodes to identify themselves as a leaf node of the tree. Nodes start this timer if a *Parent Acknowledgement* has been received confirming their requested parent-child connection and no child has requested to connect to the node so far that now has an upstream connection to a parent. Nodes identify themselves as leaves if the report timer runs out and no *Neighbor Detection* has been received from any MANET participant, requesting to connect to as child. In Line 10, this timer is stopped since node m received the $\text{ND}_{(n,*)}$ containing m as requested parent of n ($p_n = m$). If the report timer $\texttt{rt}_m$ runs out, node m would transmit its *Neighbor Report*, which contains the list of direct neighbors. The exact composition of this timer is described in Section 6.2.4.

Nodes repeat sending *Neighbor Detections* if their requested parent-child connections are not confirmed with *Parent Acknowledgements* after each defined acknowledgment timer duration. In some cases, parent nodes must transmit *Parent Acknowledgements* one after another responding to multiple children. Because of that, it happens that parents do not meet the duration of the acknowledgment timer which has previously been started by the origin (potential child) of the *Neighbor Detection*. Each *Neighbor Detection* requires a *Parent Acknowledgement* to successfully construct the parent-child connection. Parents send *Parent Acknowledgements* only once to overcome retransmissions and also to decrease the entire algorithm time. *Parent Acknowledgements* are unicast frames relying on CSMA/CA, which implies an ACK frame for a successfully delivered data frame. Because of that, parents detect lost *Parent Acknowledgements* and as a consequence automatically send this message again. This ensures that parent-child connections are successfully constructed if a *Neighbor Detection* arrived at the parent node without interference.

The last condition is true when a node receives a *Neighbor Detection* (Line 11) in combination with a successfully constructed parent-child connection. In particular, this means that the parameter $\texttt{hp}_m$ (abbreviation for has parent) is flagged to 1, while no child has registered to node m so far. This of course indicates that node m previously received a *Parent Acknowledgement*. Because of that, this node starts its report timer $\texttt{rt}_m$ to start the topology report process if no child connects to this node before the timer runs out, as shown in Line 12. Node m stops and restarts the timer if $\texttt{rt}_m$ was currently running.

It is important that this timer does not run out until each node within communication range has successfully transmitted its *Neighbor Detection* to be included in the subsequent topology representation. Multiple *Neighbor Detections* are sent in dense topologies where several children try to connect to their parents at almost the same time. Interferences of transmissions are more likely to occur in these topology constellations forcing children to start additional connection attempts. Retransmissions of lost *Neighbor Detections* ensure that children connect to parents if the started report timer of the desired parent does not run out before. Because of that, the report timer is restarted if this timer is currently running. Additionally, the timer must be defined in relation to the retry interval of lost *Neighbor Detections*, which is discussed in more detail in Section 6.2.5.

The controller τ starts the report timer immediately after the $\text{ND}_{(\tau,*)}$ has been sent and also stops this timer when the first $\text{ND}_{(n,*)}$, where $n \in$

$\mathcal{N} \setminus \{\tau\}$, arrives at the controller, where $p_n = \tau$. Also, the initial *Neighbor Detection* of the controller has no parent node p_τ as no upstream node above the controller exists.

To sum up, a node also broadcasts its own *Neighbor Detection* if it receives a *Neighbor Detection* from a node of the MANET for the first time and also schedules the acknowledgment timer to also broadcast a *Neighbor Detection* if it did not receive a *Parent Acknowledgement* after the acknowledgment timer runs out. In addition, nodes stop the report timer and respond with *Parent Acknowledgements* to create a parent-child connection if they are elected as a parent by a node that sent the *Neighbor Detection*. Finally, parents not having children connected until then, start the report timer to trigger the topology report phase if no child connects to these nodes until the timer runs out.

6.2.4 Parent-Child Connection

Successful upstream parent connections require an efficient selection by the children in terms of decision and positive feedback (acknowledgment). In this case, children request to connect to specific parents, as discussed in this section. Thereafter, we introduce how parents confirm all child requests in order to build the tree structure.

Parent Election

Broadcast messages are more likely get lost or interfere with other messages compared to unicast transmissions and as a consequence cannot be decoded by the receivers. This happens more often if several nodes try to transmit *Neighbor Detections* at the same time. Sub-topologies are lost and not known in the controller's topology representation if children that are also parents miss connecting to their parents. Because of that, nodes will rebroadcast *Neighbor Detections* after their acknowledgment timers run out. During proof-of-concept runs, it turned out that *Neighbor Detections* never arrived at their potential parents although children broadcasted these messages multiple times. This happens if transmissions of nodes in close vicinity interfere with *Neighbor Detections* every single time. This results in an incomplete topology representation at the controller.

To overcome this situation, Re-SoTUP uses the constant MAXRETRIES, which specifies how often a MANET member tries to register as a child at the desired parent. A node changes its first-choice parent and elects the next in the row to connect to if it sent *Neighbor Detections* MAXRETRIES times in

a row and did not receive an answer. At this point, potential parents are chosen out of the set of neighbors N_n by the node $n \in \mathcal{N}$ which is filled at runtime.

In addition, Re-SoTUP defines the constant parameter maxParents that restricts the number of parents a node requests a potential parent-child connection from. Nodes most likely retransmit *Neighbor Detections* in dense topology constellations as parents are occupied in transmitting *Parent Acknowledgements* due to several connection requests. Further, *Parent Acknowledgements* arrive at the desired children with even more delay, if all these parents are in crowded areas. This increases the number of *Neighbor Detections* of each node and as a result maxRetries is reached more often. Consequently, children request parent-child connections from several parents even though the initial parent will respond with some delay. A child does not change the parent even if a connection currently exists and the initially requested parent answers with some delay. The value of maxParents was determined with several test runs and is set to 3. This parameter can be changed in the beginning before the algorithm starts.

Children only try to register to parents having lower or equal tree depths compared to the tree depth of the parent a child tried to register to initially. This prevents Re-SoTUP from generating loops in the tree during parent election. Each node $n \in \mathcal{N}$ stores its tree depth $\texttt{td}_n$, which refers to the position of the current tree. The tree depth of the controller is 0. Each node $n \in \mathcal{N}$, including the controller, includes this value in the $\text{ND}_{(n,*)}$. In addition, each node increments $\texttt{td}_n$ by 1 when receiving a *Neighbor Detection* for the first time. Also, there exists a set $R = \{0, 1, \ldots, \text{maxRetries}\}$ that stores all possible retry attempts. Function $r : P_n \rightarrow R$, where $n \in \mathcal{N}$, specifies how many times a node tried to register at parent $m \in P_n$, where $m \neq n$ and $m \in \mathcal{N}$.

Algorithm 5 shows how node $n \in \mathcal{N}$ selects a parent. First, node n verifies if the number of retries with respect to the current desired parent, stored in p_n exceeds the value of constant maxRetries. The node will choose a next potential parent to request a parent-child connection, if this condition is satisfied. If true, node n picks a parent from the list of neighbors N_n if the number of entries in P_n is below maxParents, see Line 5. In particular, node n selects an arbitrary parent from N_n whose tree depth is one level closer to the controller compared to its own tree depth. Node n stores this randomly chosen parent in the set of parents P_n (Line 5 - 7).

Otherwise, if P_n is equal to maxParents, node n chooses a parent already existing in the set of parents (Line 12). In that case, the node n selects

```
1  begin ParentElection()
2      if r(p_n) = maxRetries then
3          r(p_n) = 0
4          if |P_n| < maxParents then
5              P'_n = {m ∈ N_n | td_n > td_m}
6              p_n = a ∈_R P'_n  // a is a random element (∈_R) of P'_n and
                 stored in p_n.
7              P_n = P_n ∪ {p_n}
8          else
9              if P_n \ P'_n = ∅ then
10                 P'_n = {}
11             end
12             p_n = a ∈_R (P_n \ P'_n)  // a is a random element (∈_R) of
                 (P_n \ P'_n) and stored in p_n.
13             P'_n = P'_n ∪ {p_n}
14         end
15     end
16     send(ND_(n,*))            // ND_(n,*) stores current desired parent
17     r(p_n) = r(p_n) + 1
18     startTimer(at_n)
19 end
```

Algorithm 5: Parent election of node n.

the new parent from P_n that depends on the set P'_n, ensuring that a different parent of the set P_n is selected each time. In particular, node n randomly selects a parent p_n not stored in P'_n and inserts this p_n in P'_n afterwards. This ensures that the same parent is not chosen again in the subsequent iteration (Lines 12-13). Node n removes all elements in P'_n if P_n and P'_n are equal (Lines 9-11).

Node n broadcasts $\text{ND}_{(n,*)}$ with p_n and increments the number of transmission attempts by one to ensure that p_n is requested at most maxRetries, after selecting the current parent node p_n, see (Lines 16-17). In addition, node n restarts the $\texttt{at}_n$ to go through this algorithm again if no *Parent Acknowledgement* arrives before this timer runs out.

The controller node τ never uses this algorithm. The acknowledgment timer must be started by nodes to try to connect to an upstream parent. This timer will never be started by the controller, which is also not desired as the controller must not look after a parent connection.

```
1 begin ConnectionRequest(PA_(m,n))
2 |   if C_n = ∅ and hp_n = 0 then
3 |   |   startTimer(rt_n)
4 |   end
5 |   stopTimer(at_n)
6 |   hp_n = 1
7 end
```

Algorithm 6: Node n receives a *Parent Acknowledgement*.

Connection Confirmation

As mentioned earlier, the message *Parent Acknowledgement* confirms the child the previously requested connection to the parent. A node n replies with a *Parent Acknowledgement* if p_m stored in $\text{ND}_{(m,*)}$ matches n, where $n, m \in \mathcal{N}$. Algorithm 6 shows the process node n starts if it receives a *Parent Acknowledgement*.

The most important part of this process is to start the report timer if all conditions are satisfied. This is the case if no child ($C_n = \emptyset$) and no parent connection ($\texttt{hp}_n = 0$) of node n is known at this point. Node n does not start the $\texttt{rt}_n$ if it has registered children when receiving a *Parent Acknowledgement* as n waits for *Topology Refreshs* of all children to forward the sub-topology to its parent or controller.

However, an empty children set $C_n = \emptyset$ of n is likely to happen, as nodes do not wait for *Parent Acknowledgements* before broadcasting their own *Neighbor Detections*. This triggers downstream nodes to immediately broadcast their *Neighbor Detections* containing node n as their number-one parent of choice of those having received the previous *Neighbor Detection* of n for the first time.

That leads to a situation, where parents have connections to downstream children while not having set up their own upstream parent connection. This, for instance, happens if their desired upstream parent is placed in a crowded area with multiple parents also trying to respond to parent-child connection requests. In this case, a child receives several delayed *Parent Acknowledgements* from different parents at runtime, since nodes try to register at multiple parents, which is more likely to occur in dense topologies. In such a situation, the report timer should not run out more than once because a node would then send multiple *Neighbor Reports* to their upstream parents.

To avoid that, node n verifies with $\texttt{hp}_n$ if a received *Parent Acknowl-*

edgement is the first one. Because of that, node n only starts the report timer when receiving a *Parent Acknowledgement* if no children are known and no parent connection exists so far, as shown in Line 2.

The theoretical time that node n waits until it identifies itself as a leaf is computed as follows: $\texttt{rt}_n = t(\text{ND}_{(n,*)}) + \texttt{at}_n$. Similar to the computation of the acknowledgment timer, the function t returns the duration time to successfully transmit a *Neighbor Detection*. Re-SoTUP assumes the worst-case scenario, in the message $\text{ND}_{(j,*)}$, broadcasted by the potential child j, having node n elected as parent ($p_j = n$), has been interfered by another message. This is assumed to has happened immediately before this potential parent n has received its own *Parent Acknowledgement* $\text{PA}_{(m,n)}$, where $m, n, j \in \mathcal{N}$. In this case, child j would wait for a complete $\texttt{at}_j$ period before it starts an additional transmission attempt of $\text{ND}_{(j,*)}$ with its elected parent $p_j = n$. Function t considers the duration of the retransmission time $t(\text{ND}_{(j,*)})$.

In theory, the $\texttt{rt}_n$ of each node $n \in \mathcal{N}$ is designed to only start if n can ensure that other nodes in communication range have received their own *Neighbor Detections*. With respect to a real-world scenario, nodes cannot ensure that *Neighbor Detections* are delivered without any losses since broadcast messages do not consider acknowledgments. However, a received *Parent Acknowledgement* based on an earlier delivered *Neighbor Detections* increases the probability that nodes in communication range next to the currently elected parent node have also successfully received the previously sent *Neighbor Detection*. Consequently, the node n interprets a received *Parent Acknowledgement* in this context as proof that its previously broadcasted *Neighbor Detection* has also been received by nodes, considering to connect to n and then starts the $\texttt{rt}_n$ if n has no registered children at this time. Also, n stops the $\texttt{at}_n$ because n is now connected to a parent (Line 5).

6.2.5 Topology Reporting

Nodes select themselves as leaves and transmit *Neighbor Reports* to their connected upstream parent when their report timer runs out. These *Neighbor Report* messages contain all the neighbors, these nodes have received *Neighbor Detections* from. A node m sends a $\text{NR}_{(m,n)}$ to each parent node n, where $n \in P_m$, to exclude deadlocks during the topology report process. Each parent n waits for all *Neighbor Reports* of all children in the set of registered children C_n before they report their own *Neighbor Reports* to their parents.

```
1  begin ProcessingTopologyReporting(NR_(m,n))
2  |  L_n = L_n ∪ L_m
3  |  C_n = C_n \ {m}
4  |  if C_n = ∅ then
5  |  |  L_n = L_n ∪ {(j, n) ∈ L | j ∈ N_n}
6  |  |  foreach i ∈ P_n do
7  |  |  |  send(NR_(n,i))                    // NR_(n,i) stores L_n
8  |  |  end
9  |  end
10 end
```

Algorithm 7: Reporting *Neighbor Reports* to parent nodes.

Algorithm 7 describes the process when a parent node n receives a $\text{NR}_{(m,n)}$ from its child m. Nodes store a set of links that is continuously filled when a new *Neighbor Report* of a child arrives. Starting with Line 2, node n extends its own L_n by uniting the received L_m of its child m with L_n. Next, the node n removes m from the list of children C_n since no further message from this node is expected. Parent n waits until each child has delivered its sub-topology before its own reporting process starts. If the set of children of node n is empty ($C_n = \emptyset$), meaning all children have delivered their *Neighbor Reports* (Line 4), node n constructs its own set of neighboring links L_n, see (Line 5). Thereafter, node n transmits its *Neighbor Report* including L_n to each of its parents in P_n.

In general, a leaf node o constructs the set of neighbor links L_o and sends this set to all parents k, where $k \in P_o, o \in \mathcal{N}$, that previously requested an upstream connection. The controller terminates Re-SoTUP when all its directly connected children (C_τ) have reported their *Neighbor Reports*.

6.3 RESULTS

This evaluation aims to accomplish multiple goals. First, we investigate runtimes and completeness of Re-SoTUP with real hardware to prove the general functionality with a real-world scenario. Second, we simulate the algorithm with up to 90 MANET participants and different graph densities in order to investigate scalability. We sum up the evaluation with a comparison of Re-SoTUP with OLSR. The comparison focuses on missing links to identify, which technique provides more topology knowledge for routing.

In particular, we run Re-SoTUP with identical scenario conditions as OLSR in Section 6.1 and compare the results.

6.3.1 Hardware Performance Test

We implemented the proposed algorithm on low-performance hardware to take into account the share (processing time) of the network interface in relation to the overall runtime. The MAC library and physical layer have been implemented from scratch to have full flexibility on timers and modifications for Re-SoTUP. Each MANET node is equipped with a 32 bit `ARM-Cortex M3` microprocessor `STM32F103C8T6` attached to a 2.4 GHz radio transceiver module `nRF24L01`, capable of data rates up to 2 $Mbps$. In total, a frame comprises a maximum of 24 B: 6 B for either the transmitter or receiver MAC addresses, as well as 3 B for control and 15 B for payload overhead. The constructed test setup provides packet handling based on CSMA/CA. Antenna switch times from receiving to transmitting takes 130 μsec, which is why we adjusted SIFS to 150 μsec. The remaining 20 μsec are reserved for microcontroller processing times. The slot time for transmissions is set to 51.2 μsec. We set the CW to 16 slots. Re-SoTUP is also implemented and deployed on these microcontrollers and is placed on top of the data link layer.

We conducted experiments with a testbed of 3 to 5 nodes and evaluated its functionality in linear, fully meshed, and tree network topologies. A linear network is arranged like a chain, where intermediate nodes are connected to their predecessor and successor, except for the first and last nodes. Trees are homogeneous and symmetrical. The meshed topology represents an arbitrary placement of nodes.

Figure 6.5 illustrates the average runtimes for each network topology. It is obvious that the runtimes increase as the number of nodes increases. Additionally, the runtimes grow almost linearly, which was expected.

Regarding completeness of topology representations, in linear and tree network topologies, all links were successfully delivered to the controller. However, we experienced 1.67% loss of links with fully meshed topologies referring to 4 nodes. The number of lost links increases to 4.5% with 5 nodes. The higher the node density, the higher the probability of interference, leading to an increase in corrupted packets. These packets were dropped by the MAC layer due to the absence of Forward Error Correction (FEC), which could have been recovered by using the IEEE 802.11 standards [IEEE802.11].

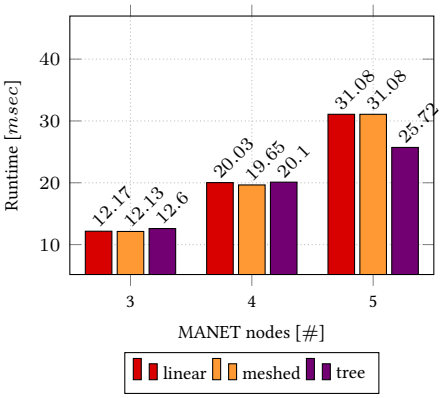
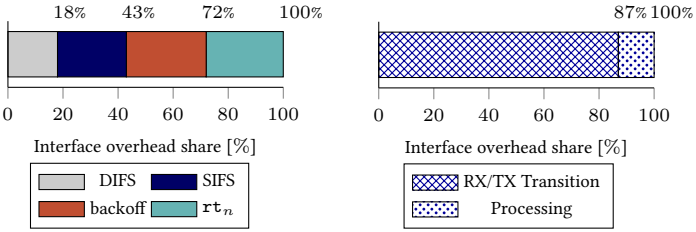


Figure 6.5: Runtimes comparison of all topologies.



(a) Share of CSMA/CA parameters and Re-SoTUP timer.

(b) Averaged SIFS composition.

Figure 6.6: CSMA/CA IFS and backoff overhead compared to $\texttt{rt}_n$, n is an node of the 3-node MANET of the linear topology.

The following results investigate the share of CSMA/CA with respect to Re-SoTUP timers to determine the efficiency of our algorithm. Figures 6.6 outlines the make-up of runtimes using a 3-node linear network as an illustration. We emphasize the share of Re-SoTUP in relation to the overall overhead of the interface and demonstrate how hardware limitations impact the utilization of the MAC layer.

The composition of the runtime for the 3-node linear network is depicted in Plot 6.6a. The duration consists of the DIFS, SIFS, backoff, and report timer $\texttt{rt}_n$ components, where n is a node of the 5-node MANET. It turns out that the report timer accounts for 28% of the total measured time.

Table 6.2: Simulation parameters

Parameter	Value
Playground sizes	$G = \{\texttt{pg}_1, \ldots, \texttt{pg}_5\}$
Number of nodes	$\{10, 20, \ldots, 90\}$
maxRetries	$\{1, 2, 3, \infty\}$
maxParents	3
Communication range	100 m
Configured bitrate of nodes	2 $Mbps$

It is important to note that the SIFS time also requires a 25% share of the total share of the wireless interface. As previously mentioned, this timer is adapted to hardware specifications.

Figure 6.6b provides a more detailed look at the SIFS overhead. The configured 150 $msec$ SIFS duration is composed of 87 $msec$ receive and transmit transitions, which in turn accounts for 21.92% of the total duration. The remaining 13% of SIFS overhead is attributed to hardware processing times.

In total, the duration for transmitting and receiving packets only makes up 76% of the runtime, while the remaining 73.32% is related to waiting periods. This shows that switching the radio transceiver of the radio module from transmitter to receiver mode takes a significantly longer time compared to other processes. Runtimes of Re-SoTUP can be reduced significantly by replacing the radio module with a more powerful one. Wireless interfaces nowadays provide higher data rates resulting in faster runtimes. Therefore, the CSMA/CA share could be reduced using up-to-date wireless network interfaces. Roughly 85% of the total runtime can be saved when using a common standard, such as IEEE 802.11ac reaching data rate peaks of up to 1 $Gbps$ [Nee11].

6.3.2 Large Scale Simulations

This section simulates Re-SoTUP on different playground sizes combined with different numbers of MANET participants to analyze the algorithm in various network densities and also to figure out how the increasing number of nodes affects the completeness of topology representations and runtimes. Simulation parameters are shown in Table 6.2.

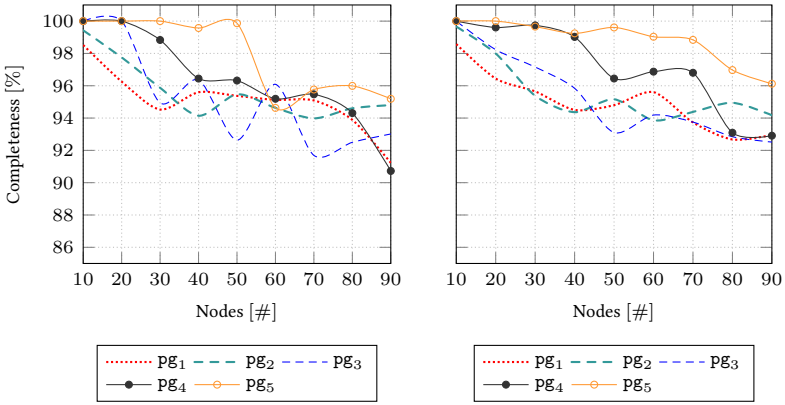
All nodes are equipped with IEEE 802.11 interfaces operating in ad-hoc

mode. We configured a transmission rate of $2\ Mbit$. In particular, the simulation comprises several scenarios, which differ in playground sizes (G) and the number of nodes. A playground size is defined by the length and width of the evaluation area. Playground $\mathtt{pg}_1$ comprises an area of $250\ m \times 250\ m$ and the length and width of each further playground increases by $200\ m$ each. Based on that, the area of playground $\mathtt{pg}_2$ reaches $450\ m \times 450\ m$. Playgrounds $\mathtt{pg}_3$, $\mathtt{pg}_4$, and $\mathtt{pg}_5$ increase the sizes as described, reaching a total area of $1050\ m \times 1050\ m$. The smallest playground generates a dense topology situation, making it challenging to successfully access the channel without interference. Both timers (acknowledgment and report timer) will have to restart more often due to missing *Parent Acknowledgements* and received *Neighbor Detections*. This constellation is error-prone regarding missing branches when Re-SoTUP finishes.

We varied the number of retries (maxRetries) during all simulations in combination with the number of nodes and playground sizes. The number of retries refers to the number of attempts a child tries to connect to the same parent until it chooses another parent to connect to. We averaged all runs with 1, 2, and 3 maxRetries and compared all results with maxRetries $= \infty$. All in all, the combination of playgrounds, number of nodes, and maxRetries simulate plenty of topology densities and outlines if different numbers of parents have a significant impact on the completeness of topologies and runtimes of Re-SoTUP.

Results focus on the number of links reported with Re-SoTUP compared to the actual number of links. We also measured execution times of Re-SoTUP from the first *Neighbor Detection* of the controller until the last *Neighbor Report* reaches the controller with up to 90 nodes. We ran each combination of number of participants, playground size, and maxRetries 500 times to average out measurement errors.

Figures 6.7a and 6.7b show the number of successfully delivered links to the controller. Figure 6.7a depicts Re-SoTUP where children only try to register to the same parents (maxRetries $= \infty$) whereas Figure 6.7b describes the average of parent retries 1, 2, and 3. The edge density increases with increasing number of nodes from left to right as each line represents a playground size. In general, both approaches show minor differences regarding reported topologies. Re-SoTUP configured with averaged maxRetries (Figure 6.7b) delivers slightly more links to the controller. This becomes more evident in dense topologies when the number of MANET participants increases. It is obvious that densely populated areas in terms of nodes challenge the algorithm as results of $\mathtt{pg}_5$ outperform all other configurations,



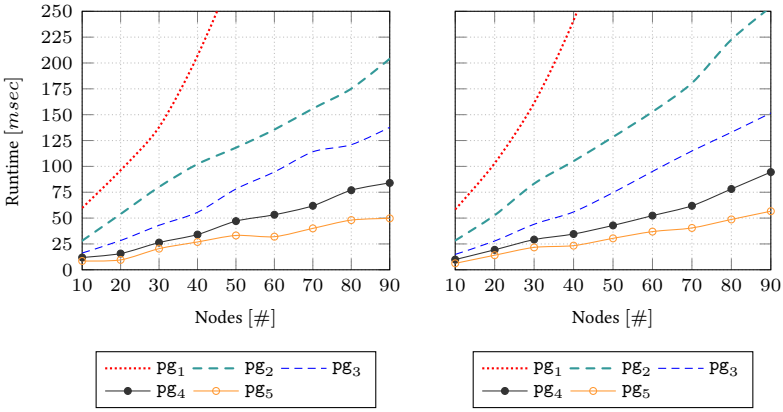
(a) Re-SoTUP with single parent election. (b) Re-SoTUP with averaged parent election.

Figure 6.7: Completeness of reported topologies to the controller with $\mathtt{pg}_{\{1,\ldots,5\}}$.

especially when nodes are allowed to connect to multiple parents.

Speaking of maxRetries, the left figure exhibits uneven graphs appearing as small dips. These small dips during topology representations result from children not having successfully connected to their desired parents before the topology reporting phase starts. The extent of incomplete topology knowledge varies depending on the connected children and their children if they exist. Sub-branches can get lost only if a single child misses connectivity to its desired upstream parent. We recorded the number of *Neighbor Detections* each node broadcasts during runs since nodes are searching for a parent connection by sending a *Neighbor Detection* after another when the acknowledgement timer runs out until connectivity to a requested parent exists. It turns out that Re-SoTUP with single parent registration propagates 8% more *Neighbor Detections* per node compared to multiple parent configuration when analyzing the number of transmitted *Neighbor Detections* of all runs of $\mathtt{pg}_1$.

Closing the gap to the aforementioned goal, both approaches deliver an almost complete topology representation to the controller. This applies to all scenario configuration combinations. However, requesting multiple parents if the initial one is busy improves reliability compared to the single parent configuration (maxRetries $= \infty$).



(a) Re-SoTUP with single parent election. (b) Re-SoTUP with averaged parent election.

Figure 6.8: Runtimes of Re-SoTUP with $\mathtt{pg}_{\{1,\ldots,5\}}$.

In the next part of the evaluation, the focus is on execution times of Re-SoTUP since the complete topology should be delivered to the controller as fast as possible. We considered 15 μsec processing time per packet to design realistic simulation conditions [Zil+17]. Figures 6.8a and 6.8b illustrate runtimes from the start of the first *Neighbor Detection* until the controller receives the *Neighbor Report* from its last child investigating all combinations of number of nodes and playground sizes. Like in the previous evaluation, the results consider various number of node configurations combined with different playground sizes. Figure 6.8a depicts Re-SoTUP configured with single parent connectivity (maxRetries set to ∞) whereas the second figure (6.8b) focuses on runtimes having averaged maxRetries configured (the average of all runs having maxRetries set to 1, 2, and 3).

Runtimes are minimal and increase with increasing number of nodes except for the results of $\mathtt{pg}_1$. Nodes experience more transmissions during channel sensing as more nodes are in each communication range with decreasing playground sizes. CSMA/CA is inefficient in high edge density since the slot access competition takes time especially if more nodes try to access the channel in close vicinity simultaneously. Results also show that the number of repeated *Neighbor Detections* per node increases in $\mathtt{pg}_1$ since parents are occupied with reporting *Parent Acknowledgements*. Also, nodes

are 2 hops away from each other at most meaning the network is densely meshed. Route computation is not necessary as transmissions reach destinations as no intermediate route participants exist in many cases. Because of that, no complex routing engine is required and one must call into question if the proposed Re-SoTUP is efficient for those topologies.

We learned that Re-SoTUP takes longer when applying this algorithm in smaller playgrounds. This is the case in situations in which nodes try to register multiple times at the desired parent, considering the configured maxRetries (simulation configuration is shown in Table 6.2) before they change the parent.

Each downstream node n must transmit its *Neighbor Report* to each parent in the set P_n it tried to register with a *Neighbor Detection* before, where $n \in \mathcal{N}$ to not run into algorithm deadlocks. Each parent p stores each received potential child n of the received $\text{ND}_{(n,*)}$ in the list of children C_p and expects a $\text{NR}_{(n,p)}$ during topology reporting phase, where $p_n = p$ and $p \in P_n$. The parent p waits for each *Neighbor Report* of each $n \in C_p$ until p sends its $\text{NR}_{(p,j)}$ where $p_p = j$ to its upstream parent and to other upstream nodes in P_p. Because of that, each child transmits a *Neighbor Report* to each parent it has tried to connect to previously.

Increased retries of *Neighbor Detections* and multiple *Neighbor Reports* extend the runtime of Re-SoTUP configured with multiple parents when playground sizes decrease. We also noticed that in rare cases parents drop *Parent Acknowledgements* due to overfilled MAC queues. This also leads to retransmissions of *Neighbor Detections*. Limiting the number of children per parent could solve this drawback.

In general, the results prove that the runtimes are minimal as topology update takes less than $250\ msec$ with all scenario configurations except for pg_1. In the worst case, Re-SoTUP reports more than 95% of all links to the controller. The algorithm is more reliable if configured with multiple parents. However, topology reporting takes longer as reporting *Neighbor Reports* to multiple parents produces more competition and transmission overhead.

6.3.3 OLSR versus Re-SoTUP

Re-SoTUP is designed for MANETs where mobility and topology changes happen continuously at runtime. Because of that, Re-SoTUP is evaluated during different mobility speeds. Full topology knowledge in MANETs is already provided, applying specific proactive routing protocols, such as

OLSR [Cla+14], as researched and summarized in Section 3.3. The conceptual protocol design enables complete knowledge for all or at least for a specific MANET participant. Previous research focuses on proactive topology reporting techniques lacking accuracy in terms of the number of missing links. To be more concrete, evaluations in Section 6.1 found out that participants miss connections of neighboring nodes within their coverage ranges that exist in the real MANET structure which increases with increasing speeds of participants. Inconsistencies in routing tables increase even more with increasing number of MANET participants.

We evaluated Re-SoTUP during different mobility configurations and also compared the algorithm with OLSR to better classify topology management with a state-of-the-art routing protocol. Therefore, the portion of missing links of Re-SoTUP are carried out and compared to OLSR. This evaluation adopts almost the entire scenario configuration described in Section 6.1. The comparison only skips speed range $\texttt{sr}_2$, which is why speed ranges start at $\texttt{sr}_5$ and end at $\texttt{sr}_{30}$. OLSR and Re-SoTUP run on equal playgrounds where nodes are moving randomly based on the same seeds applied to both approaches. Mobility simulations stop at equal times (40 sec) in each run followed by a topology snapshot on an arbitrary node, with respect to OLSR. The topology representation of the controller of Re-SoTUP and the topology snapshot are compared with the actual topology, respectively.

Figure 6.9 depicts the portion of missing links of Re-SoTUP and OLSR compared to actual topologies with all configured speed ranges. Both approaches are evaluated with scenarios from 10 to 90 MANET nodes, deployed on proportionally increasing playgrounds, combined with all defined speed ranges. Each scenario runs 500 times with different seeds.

Minimum speed configuration ($\texttt{sr}_5$) has almost no impact on topology representations, referring to both approaches. Both approaches miss almost the same amount of links (up to 5% at most). Increasing speed configuration increases the gap of missing links between Re-SoTUP and OLSR.

Re-SoTUP takes hundreds of milliseconds making almost no differences in terms of missing links. Topology completeness decreases slightly with increasing number of node configurations in combination with increasing speed ranges. These lost links are due to mobility changes at runtime as the number of missing links also increases with 10 to 40 nodes deployed, which differs from the static evaluations in Section 6.3.2. The actuality of topology representations decreases with 50 nodes deployed and more (Figure 6.7b).

The topology representation of Re-SoTUP generates the network graph

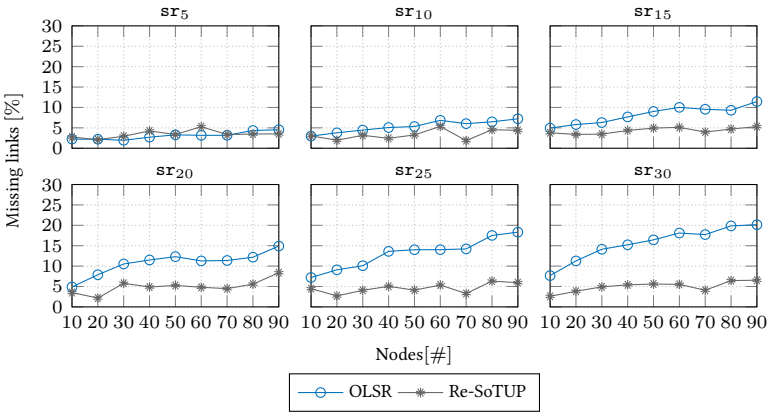


Figure 6.9: Missing links in topology representation of Re-SoTUP and OLSR with 10 to 90 nodes and speed ranges $\texttt{sr}_5$ to $\texttt{sr}_{30}$.

from scratch each time the algorithm starts. Links of previous runs are dropped before the controller starts the *Neighbor Detection*. We experienced a minor share of outdated links (below 1%). Connections existed during topology detection phase in which both source and sink of these connections were almost out of each other's range at this time. During topology reporting phase, these connections were not present anymore.

Summary In this chapter until this point, we introduced Re-SoTUP [Str+20] a reactive and self-organized topology update algorithm focusing on minimum runtime and on a complete snapshot in terms of MANET participants and their connections. We first highlighted weaknesses with respect to the timeliness of network topologies of current approaches dealing with proactive topology reporting techniques. Next, we introduced the new architecture Sc-MANET and added parameters and variables of Re-SoTUP to it. Furthermore, we deep-dived into the functionality of Re-SoTUP and introduced the topology detection and reporting process. We also showed real hardware results and evaluated large-scale simulations to prove the functionality and scalability of our approach. At last, we compared Re-SoTUP with the state-of-the-art routing protocol OLSR to determine the topology quality with related research.

The following section proposes a concept to manage the controller at runtime in MANETs. This ensures continuous controller reachability for each participant. We also discuss controller outages and the process in

case several MANETs, which are equipped with a controller reach each other.

6.4 AUTONOMOUS CONTROLLER MANAGEMENT

A full topology representation cannot be leveraged if the knowledge is not accessible by MANET participants. Up-to-date topology knowledge can be gathered by the controller using Re-SoTUP, referring to Section 6.2. Per design, each MANET participant potentially desires a data transfer between itself and another participant, which requires a route to this destination. The controller constructs routes to accomplish utilization-efficient and robust paths, in terms of long-living connections, for all requested routes. In order to compute these requested paths, MANET participants require a route to the controller which must be maintained continuously. Furthermore, controllers might experience outages, making it necessary to determine a new controller.

Keeping that in mind, the following requirements must be fulfilled to provide utilization-efficient routing of multiple flows at runtime: (i) Exactly one controller, deployed on a MANET participant, must be present at runtime. (ii) Continuously controller reachability for each node, focusing on minimum control message overhead as the wireless channel provides restricted transmission resources. (iii) Self-managed controller election in case of controller outages or if, for instance, multiple MANETs reach each other.

Considering these requirements, we present SoCM an algorithm providing self-organized controller election and all-time connectivity to a controller for each MANET node. Firstly, we define additional framework components and parameters defined for SoCM and integrate these components into the Sc-MANET architecture. Secondly, controller characteristics and requirements are discussed, distinguishing controllers from MANET participants. Thirdly, we deep dive into algorithm structures to accomplish all-time controller connectivity per node. Lastly, SoCM processes are introduced in case controller outages and MANET merge situations appear.

6.4.1 Further Framework Components

SoCM provides any-time controller reachability for all MANET participants at runtime. Furthermore, SoCM manages controller election. The latter ensures that only one participant has the controller role. Each node must

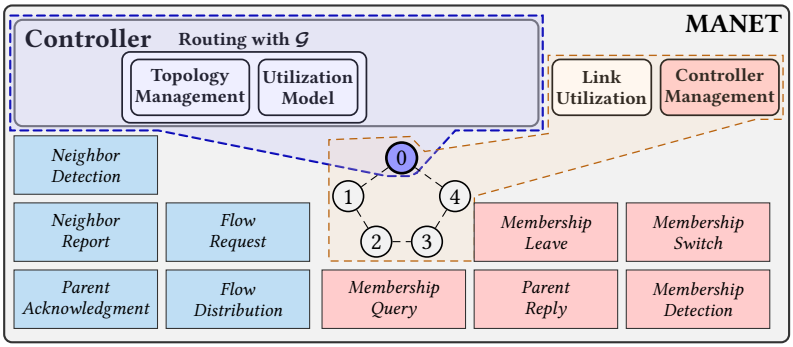


Figure 6.10: Sc-MANET architecture, applying Re-SoTUP with SoCM to accomplish self-organized controller management and reachability at runtime.

maintain a route to the controller in order to request routes to the desired destinations. Controller reachability and management require additional interactions and functionalities for Sc-MANET.

Figure 6.10 depicts the Sc-MANET architecture including all additional defined message types and components. Red-colored components and messages represent newly defined parts of the Sc-MANET. The blue-colored node identifies the controller comprising the capabilities **Topology Management** and **Utilization Model** (inside the blue surrounding). The first component is assigned to Re-SoTUP and is responsible for up-to-date topology knowledge. The second considers residual transmission capacities during route computations. Both, the **Utilization Model** and the **Link Utilization** are discussed in detail in Section 4.2.3.

The orange-colored surrounding contains components used by all MANET participants. Controller selection, solid reachability for each node, applied steps in case of controller outages, and MANET merge processes are defined in the module **Controller Management**. This component uses the tree of Re-SoTUP to provide controller reachability for each node. Tree optimization in terms of minimum control message overhead is one of the key features of SoCM. Participants store their upstream node towards the root node (controller) and switch this parent in case a new upstream node might offer less message overhead. Their sub-tree quality is indicator for potential tree changes, speaking of a connected child. Tree optimization and single controller management, such as MANET merge scenarios and

controller outages, are key components of the **Controller Management** module.

SoCM is partitioned into two phases, named tree maintenance and tree merge. We define five messages for these two phases covering tree optimization and single-controller policy management. In the tree maintenance phase, the controller and parents broadcast *Membership Queries* to trigger connection acknowledgments of their children. Thereby, we apply a resource-efficient response technique to in the best case reply with a single *Parent Reply*. Nodes broadcast *Membership Queries* interval-wise to maintain parent-child connections. These connections provide a route for each tree member to the controller. *Membership Queries* store the sub-tree qualities, which children use to switch parents to achieve better tree quality. The composition of sub-trees is defined in Section 6.4.5. *Membership Leaves* are sent to inform parents about a leaving child and *Membership Switches* inform the new parent about the new parent-child connection. Children are aware of the periodic keep-alive (receiving *Membership Queries*) from their parents. The entire tree maintenance phase is discussed in Section 6.4.3. Missing *Membership Queries* are classified as connection breaks. Each node that experiences a connection break elects itself as controller.

SoCM applies the tree merging process if more than one controller is present in a MANET. This happens if children lose connectivity to their parents or if at least one node of another MANET, each controlled by a controller, receives *Membership Queries* from a node of the other MANET. Isolated nodes elect themselves as controller and start broadcasting *Membership Queries* containing their tree quality. Receivers of these *Membership Queries* either connect to the new controller or advise this node that previously sent the *Membership Query* to connect to itself if its tree quality is better. Receivers respond with a *Membership Detection* to establish the parent-child connection, discussed in detail in Section 6.4.4.

6.4.2 Initialization and Definitions

SoCM is built on top of Re-SoTUP and uses the tree information after topology reporting finished. The data structure of SoCM is derived from the graph $\mathcal{G}^\tau = (\mathcal{N}^\tau, \mathcal{L}^\tau, \mathcal{F}^\tau)$ belonging to controller τ (first introduced in Section 5.3.1), comprising a tree $T^\tau := (M^\tau, P^\tau, U^\tau)$ to organize all members, where τ of the tree and the graph are identical. Tree members $M^\tau := (\mathcal{N}^\tau \setminus \{\tau\})$ comprise all nodes of the MANET except the controller. Like with Re-SoTUP the controller id τ is the root of the tree. The set of upstream

connections $U^\tau \subset \mathcal{L}^\tau$ functions as the next hop for each member towards the controller root. A link $(j,k) \in U^\tau$ is directed, connecting a member j to its upstream tree member k, which has either the parent or controller role. SoCM maintains a parent set $P^\tau \subset M^\tau$ containing all nodes that own the parent role. Also, each tree member n stores the parameter tree depth $\mathtt{td}_n$, where $n \in M^\tau$, representing the placement of n in the tree. The tree depth of the controller is 0. Directly connected children have tree dept 1. Roles and usages including the associated connections are discussed in detail within the next paragraphs of this section.

SoCM provides controller switches leading to new controller ids in the graph and tree data structures. In particular, if the controller role switches from, for instance, id a to b, tree T^a changes to T^b and graph $\mathcal{G}^a$ switches to $\mathcal{G}^b$. This includes all data structures of the graph and the tree. The default controller id is τ if the current context is about a single tree.

Roles and Usages

Tree participants have at least one of the following roles: **Controller**, **parent**, or **child**. The **controller** is defined as the root node having no upstream parent and zero or more downstream nodes (children). Children of the controller also own the role parent if at least one child is connected to this node. A controller τ computes the property tree quality $\mathtt{tq}_\tau$, which is derived from the current tree structure, as discussed in Section 6.4.5. This value represents the tree arrangement also considering mobility characteristics of tree participants. The tree quality is indicator of which tree connects to another tree in case two MANETs reach each other. The tree with worse quality drops the controller role and connects to the other one. A controller τ computes $\mathtt{tq}_\tau$ summing up all specific quality parameters of downstream parents.

Furthermore, a controller stores a set of parent nodes in $P^\tau = \{n \in M^\tau | \mathtt{bq}_n > 0\}$. A node $p \in P^\tau$ is parent if the sub-tree quality (branch quality) $\mathtt{bq}_p$ is greater than 0. Based on that, the controller node maintains P^τ that stores all downstream nodes having assigned the parent role. Parents forward their branch qualities up to the controller node. A controller τ also maintains the set of potential children $C_\tau = \{n \in M^\tau | (n,\tau) \in U^\tau\}$ assuming a connection to the controller node. Setting up a connection between a child and a controller or a parent is discussed in Section 6.4.3. The set of children C_τ is defined as a total ordered set. The sequence of this set is based on the parameter child quality $\mathtt{cq}_c$, where $c \in C_\tau$, which stores the quality of a child. The quality of a child is based on their speed and

movement directions. The child with the worst child quality is positioned at the beginning of the ordered set C_τ. The child with the best quality is positioned at the end of the set of children. The computation of all qualities (tree, branch, and child qualities) is discussed in Section 6.4.5. A controller periodically broadcasts *Membership Queries* to gather information about directly connected children. A child confirms its connection with a *Parent Reply* to maintain its route to the controller.

Nodes have the role **parent** if they have at least one connected child. Parents also own the child role, since they are also connected to their upstream parent or controller. Similar to controllers, parents also periodically broadcast *Membership Queries* to maintain the connection to their children if they have any. Each parent $n \in P^\tau$ has the property tree depth $\texttt{td}_n$, which represents the position of n in the tree. A parent n also stores the id of its controller in the parameter r_n. Similar to controllers, parents also store the property $\texttt{tq}_n$. The tree quality is initially computed by the controller of this tree. Parents compare their own tree quality with the tree quality of an approaching MANET in case another MANET reaches the connection range of any node. Depending on the result, the parent either informs the other tree to connect as a child to itself or connect itself to the node of the other tree. Each $p \in P^\tau$ forwards $\texttt{tq}_p$ to their children included in each *Membership Query* to ensure that all tree participants store the current value. Because of that, the parameter tree quality is forwarded from the root towards the leaves reaching each tree participant. In addition, each parent p computes the branch quality $\texttt{bq}_p$. Parents also maintain the totally ordered set of children $C_p := \{c \in M^\tau | (c, p) \in U^\tau\}$. Also, parents store the id of the controller in r_p.

The role **child** is defined for nodes that are connected to a parent. Each $c \in C_p$ of a parent p, where $p \in P^\tau$, computes its child quality $\texttt{cq}_c$, which comprises the speed and the associated movement direction. Equal to parents, each child c stores the properties $\texttt{tq}_c$, and $\texttt{bq}_c$, which have been computed by the controller and its parent p. In addition, the child c stores the controller id in the parameter r_c. Children process received *Membership Queries* to verify the origin of this message as their connected parent. Therefore, a child c maintains p_c storing the id of the connected parent, which they compare with the origin of the received *Membership Query*. A child c also maintains the totally ordered set of siblings S_c included in the received *Membership Queries* storing the update sequence that children send their *Parent Replies* to their parent. The set of siblings of a child compared to the set of children of a parent node are the same in terms of the

stored nodes and the sequence of them in case this child is actually a child of this parent.

Siblings follow a defined order when to transmit their *Parent Reply*, as discussed in Section 6.4.3. Children do not have downstream nodes connected unless they are also parents.

Control Messages

Abbreviated control messages of SoCM are notated with indices, similar to Re-CeTUP and Re-SoTUP. For instance, a $\text{MQ}_{(p,*)}$ represents a *Membership Query*, originated by node $p \in P^\tau \cup \{\tau\}$. This message is broadcasted by p to all nodes within communication range, identified by $*$ as target. Unicast control messages replace the $*$ with the desired destination node $n \in \mathcal{N}$. SoCM uses the following message types for communication:

Membership Queries are periodically sent by parents p and controllers τ to verify if at minimum one child is connected to them, where $p \in P^\tau \cup \{\tau\}$. Nodes stop sending *Membership Queries* in case no child responds with a *Parent Reply* after a defined time period. A *Membership Query* message is abbreviated as $\text{MQ}_{(p,*)}$ and stores the id of the parent p and C_p. The latter represents the report order children follow with *Parent Replies*. Also, a $\text{MQ}_{(p,*)}$ contains the following parameters: $\texttt{td}_p$, $\texttt{bq}_p$, and $\texttt{tq}_p$. These parameters are decision-makers for children whether to stay connected to their current parent or to connect to another one. Children broadcast *Parent Replies* when receiving a *Membership Query* to maintain and verify their route to the controller. This message also instructs to notify their parent to continue broadcasting *Membership Queries*.

A *Parent Reply*, abbreviated as $\text{PR}_{(c,*)}$, where $c \in C_p$ and $p \in P^\tau$, contains the id of the parent, stored by each child in the parameter p_c, their own id c, and the child quality $\texttt{cq}_c$. Tree participants broadcast *Membership Detections* if they receive *Membership Queries* of different trees having worse tree quality. Receivers of *Membership Detection* will connect to the node that sent this message.

A *Membership Detection*, abbreviated as $\text{MD}_{(c,*)}$, where $c \in M^a$ of tree T^a, is sent as a response to a $\text{MQ}_{(p,*)}$, where $p \in M^b$, belonging to T^b if the tree quality of T^a is better compared to T^b.

Tree members send *Membership Switches* to reply to a previously received *Membership Detection* with the intention to connect to the origin. Node c broadcasts a *Membership Switch*, abbreviated as $\text{MS}_{(c,*)}$ storing $p_c = n$ on the protocol layer as destination, where $c \in M^b$ belonging to tree T^b

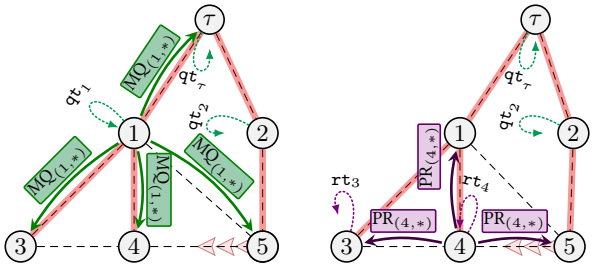
and $n \in P^a$, to connect to parent n as child to the other tree. In addition, *Membership Switches* comprise the id of its new parent, the id of the node that sends this message, and also the child quality of the origin of this message.

Membership Leaves, abbreviated as $\text{ML}_{(c,p)}$, sent from node c to p, informs the parent p that node c disconnected and is now connected to another parent. The previous parent thereafter removes c from its list of children. In this situation, nodes c and p belong to the same tree.

6.4.3 Tree Maintenance

Parts of the tree maintenance phase are derived from the protocol processes of Internet Group Management Protocol (IGMP), version 2 [Fen97]. This multicast protocol, operating mainly in edge networks or Local Area Networks (LANs), routes multicast transmissions to clients subscribing to a multicast traffic. Multicast supportive routers periodically ask their directly connected clients with membership queries for subscriptions of a specific multicast address. Clients respond with membership reports if they subscribe to this traffic. Subscribers start a randomly chosen timer that triggers their membership report, to keep the report messages to a minimum. Subscribers discard their timer when they receive a report message for the same multicast address sent by another client. This ensures that only one client answers, which reduces the control message overhead to a minimum. SoCM follows a similar subscription process to achieve minimal message overhead.

Membership Queries and *Parent Replies* of SoCM manage the IGMP-like parent-child connectivity. This procedure constitutes the major part of the tree maintenance phase. Thereby, parents periodically poll information from multiple children with *Membership Queries* to verify connectivity of their children. All parents including the controller node $p \in P^\tau \cup \{\tau\}$ periodically broadcast *Membership Queries*, containing their origin id and all properties mentioned in Section 6.4.2. *Membership Queries* aim to trigger children to confirm their subscriptions with *Parent Replies*. This verifies from the children's point of view that their current parent connection is still active. In addition, the role of parents relies on subsequent situations. The parent continues broadcasting *Membership Queries* if at least one *Parent Reply* reaches this parent. This defines that one or more children chose this node as the next hop to reach the controller. Otherwise, the controller stops sending *Membership Queries* and switches from the role parent to the



(a) Parent 1 broadcasts *Membership Query* to one-hop neighbors after query timer runs out.

(b) Child 4 replies with a *Parent Reply* based on the report order.

Figure 6.11: Exemplary tree maintenance process. Node 5 decides whether to stay connected to parent 2 or to connect to parent 1. The Controller is deployed on node τ.

role child as no downstream node replied with a *Parent Reply*.

Reducing the number of parents of the tree also reduces *Membership Queries* and *Parent Replies* which minimizes the overall message overhead significantly. To guarantee correct functionality and also to reduce the control message overhead, SoCM applies a special *Parent Reply* process in which only one arbitrary child responds to a previously sent *Membership Query*. Also, children switch parents if the new tree structure increases the probability of sending fewer control messages.

Figure 6.11 introduces an exemplary tree maintenance situation of SoCM. The tree with controller node τ, parent nodes 1, 2, and the downstream nodes (children) make up the tree structure, illustrated with the red lines. Gray dashed lines represent connections between all participants in the MANET. Node 5 moves towards node 4 and already reached the transmission range of 4 and 1. According to SoCM, all parents and the controller periodically broadcast *Membership Queries*. These query timers $\mathtt{qt}_m$ of all tree members $m \in P^\tau \cup \{\tau\}$ are illustrated with round arrows around parents and the controller. In general, each participant has configured the same interval duration. However, initial start times are randomly chosen.

According to Figure 6.11a, node 1 starts broadcasting a *Membership Query* since the query timer ran out. The message reaches all nodes that are within transmission range. Downstream nodes 3 and 4 of their parent node

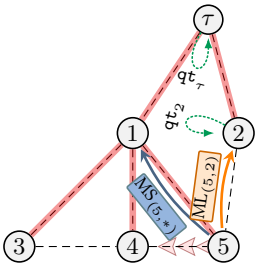


Figure 6.12: Node 5 switches parents and sends a $\text{MS}_{(5,*)}$ and a $\text{ML}_{(5,\tau)}$ message.

1 follow a report order generated by their parent to respond most efficiently in terms of control message overhead. The order is determined by an arranged sequence in which each time slot is associated to a connected child. This sequence is computed by the parent node. The sequence is related to child qualities, as discussed in Section 6.4.5. Children sense the channel for *Parent Replies* of their siblings until their defined time slots start when receiving a *Membership Query*. A downstream node broadcasts its *Parent Reply* if it has not received a *Parent Reply* from any sibling when its own slot starts. This ensures that only one child responds to a previously sent *Membership Query*.

As depicted in Figure 6.11b, node 4 is set to report position 1 and as a consequence replies with a *Parent Reply* which notifies node 3 to abort its report timer. The controller node τ remains idle after receiving the *Membership Query* from 1 since τ is the parent of 1. The *Parent Reply* of 4 triggers node 1 to continue sending *Membership Queries* since 1 is informed about a connected child. Also, all children (3 and 4) verified that their connection to their parent node 1 and as a consequence their connection to the controller is still present. Node 5 compares the sub-tree qualities of its currently elected parent 2 and of node 1 when receiving the *Membership Query* and decides to switch parents since the sub-tree quality of 1 is higher compared to the sub-tree quality of its current parent. The detailed computation of sub-tree qualities is introduced in Section 6.4.5.

In order to connect to 1, node 5 first transmits a *Membership Switch* to register to the desired parent 1. This message contains the child quality of 5 which is stored by 1 to consider this information when computing subsequent report orders, as discussed in Section 6.4.5. Also, the new parent inserts 5 into its list of children.

Furthermore, node 5 informs the previous parent with a *Membership Leave* that this parent-child connection no longer exists (Figure 6.12). The old parent (node 2) removes 5 from its list of children and will broadcast a *Membership Query* only once more. No child is connected which is why node 2 will drop the parent role.

Processing Membership Queries

Tree participants have to make various decisions when they receive a *Membership Query* despite responding with a *Parent Reply*. These decisions include whether switching to another parent node, setting the reply slot, as well as verifying the controller ids. The latter happens if several MANETs merge.

Algorithm 8 introduces all protocol steps of node $c \in M^\tau$ if this node receives a $\text{MQ}_{(p,*)}$ from node $p \in P^\tau$ of tree T^τ. To begin, c verifies if the currently selected parent sent this $\text{MQ}_{(p,*)}$. If p is the currently connected upstream node, c updates the tree properties (Line 3) and schedules the upcoming $\text{PR}_{(c,*)}$ invoking Function startTimer. Node c will broadcast a *Parent Reply* when this timer runs out. The duration is based on the received position of node $c \in S_c$. The report order process and all quality parameters, such as tree, sub-tree, and the child quality are discussed in detail in Section 6.4.5. All these quality parameters must be kept up-to-date as their values are decision-makers for parent switches and determine which node drops the controller role during a tree merge process.

Node c also updates the controller parameter r_c (Line 6) to always keep track of the current controller as this role is passed to another node with better tree quality during a tree merge process in case the tree quality of the other tree is better.

There are situations, in which the child's parent id differs, which is when downstream nodes receive *Membership Queries* from other parents within communication range. In such a situation, receivers of this message decide which parent to connect to in order to improve the tree quality. Parent switches aim to decrease protocol message overhead. Lines 7 to 28 elaborate on this process. The decision variables, stored in the ordered sequence Q_{cp}, determine if node c decides whether to connect to a new parent ($p_c \neq p$) or to another tree ($r_c \neq r_p$), see Lines 8 to 12. First of all, the tree dept, second of all the branch quality, and finally the ids are used to decide whether to stay or to switch the parent, in case the nodes c and p have stored the same controller (Line 9). Otherwise, if the controller ids are not identical, firstly both tree qualities, and secondly the controller ids are

```
1  begin ProcessingMembershipQuery(MQ_(p,*))
2      if p = p_c then
3          updateTreeProperties(td_p, tq_p, bq_p)
4          S_c = C_p
5          startTimer(rt_c, S_c) // Start timer based on position in S_c
6          r_c = r_p
7      else
8          if r_c = r_p then
9              Q_cp = ⟨(td_c, td_p), (bq_c, bq_p), (p_c, p)⟩
10         else
11             Q_cp = ⟨(tq_c, tq_p), (r_c, r_p)⟩
12         end
13
14         if evalTreeProperties(Q_cp) = 1 then
15             updateTreeProperties(td_p, tq_p, bq_p)
16             send(MS_(c,*))              // MD_(c,*) stores p, c and cq_c.
17             send(ML_(c,p_c))
18             p_c = p
19             S_c = C_p
20         else
21             if r_c ≠ r_p then
22                 startTimer(qt_c)
23                 return
24             else
25                 delete(MQ_(p,*))
26                 return
27             end
28         end
29     end
30 end
```

Algorithm 8: Node c receives $\mathrm{MQ}_{(p,*)}$ from p.

compared with each other (Line 11). Function evalTreeProperties loops over each entry in Q_{cp} comparing the values of each tuple. If the value of c is worse compared to the value of p with respect to each tuple in Q_{cp}, the function returns 1, otherwise 0. evalTreeProperties continuously loops over the tuple entries if both values of c and p are equal.

If this function returns 1, c will connect to the new parent, which is now p, as shown in Lines 14 to 19. In this case, c updates all quality properties $\texttt{td}_p$, $\texttt{tq}_p$, and $\texttt{bq}_p$. Next c sends an $\text{MS}_{(c,*)}$ and a $\text{ML}_{(c,p_c)}$ respectively to register at the new parent and to disconnect from the old parent. Also, the parent id p_c of c is set to p. Furthermore, the list of siblings is set to the new siblings.

If evalTreeProperties returns 0, either the own quality parameters to compare different two trees (Line 11) outperform or the parameters comparing both parents (Line 9) of the same tree are higher rated. In both cases, c checks if the own stored controller equals the controller of the new potential parent (Line 21). Node c starts the tree detection timer $\texttt{dt}_c$ if the controller ids differ. This timer schedules a *Membership Detection* to p in order to advise this node to connect to this tree.

We observed multiple *Membership Detections*, which have been sent by nodes in near vicinity during test runs. The previously sent *Membership Query* of a node of the other tree probably reaches several nodes which leads to multiple sent *Membership Detection* increasing the control message overhead. To avoid multiple broadcast messages, each recipient of this *Membership Query* starts a random timer. Meanwhile, each node $c \in C_p$, where $p \in P^\tau$ scans the channel for any *Membership Detection* containing the id p before their timer runs out. Each node c discards its timer in case a node s of set of siblings S_c broadcasts a *Membership Detection* before the timers of other nodes $S_c \setminus \{s\}$ runs out. Only one *Membership Detection* is sent in the best case which reduces the message overhead. In Line 22, node c behaves as mentioned and broadcasts its *Membership Detection* if its timer runs out. Otherwise, the timer will be stopped. In case nodes p and c have the same controller, c would discard the received *Membership Query* as a parent switch would not improve the control message overhead of this tree.

Children Maintenance

Children broadcast *Parent Replies* to inform their parents that they have at minimum one child connected. Parents are instructed to continue sending *Membership Queries* when receiving a *Parent Reply* storing its id as the upstream node. Also, children maintain their route to the controller as they maintain their upstream connection towards the root node.

Algorithm 9 walks through the process of node p, when receiving a $\text{PR}_{(c,*)}$ from c, where $c, p \in M^a$ of tree T^a. Starting with Line 2, node p checks if one of its children originated the $\text{PR}_{(c,*)}$ and also if the parent id

of c equals its own id. It is not ruled out that this $\text{PR}_{(c,*)}$ is sent by a node of another branch of the same tree as transmissions propagate omnidirectionally. Also, if $p \notin P^a$, meaning that p does not have the parent role, the set C_p would be empty.

The parent updates the ordered set of children, see Line 3, if the origin of the received *Parent Reply* is one of the parent's children, which means that both conditions of Line 2 are true. Updating means that all children having worse child quality with respect to c are removed. This affects all children located before $c \in C_p$. Parent p assumes that all nodes positioned before c are no longer connected as they also received the previously broadcasted $\text{MQ}_{(p,*)}$ and would have replied before c as their reply slot in the totally ordered set is before node c.

The sequence of the C_p is based on the child qualities $\texttt{cq}_c$, where $c \in C_p$. The child quality parameter $\texttt{cq}_c$ is described by a value between $[0, 2] \cap \mathbb{R}$, with higher values indicating better quality. The lower the value of the child quality of an arbitrary child, the more likely it is that a child will be out of the parent's communication range when the parent starts the next *Membership Query*. If it's probable that a child will be beyond the communication range of the parent after the next query timers, then the child quality of this child decreases. As a consequence, a child c with lower $\texttt{cq}_c$ is positioned towards the beginning of the C_p, where $c \in C_p$. As already mentioned, the arrangement of C_p determines the reply order each child follows when receiving a new $\text{MQ}_{(p,*)}$. If a child does not receive a *Parent Reply* from a sibling when its timeslot starts, it broadcasts its *Parent Reply* to the parent since nodes positioned beforehand in the set of siblings have obviously moved out of the communication range of their parent. All siblings positioned thereafter receive this *Parent Reply* and will stay idle for

```
1 begin MaintainingChildren(PR_(c,*))
2     if c ∈ C_p and p = p_c then
3         C_p = C_p \ {n ∈ C_p | cq_n ≤ C_c}
4         ∀n ∈ C_p ∪ {c} : compute(cq_n)
5         startTimer(qt_p, C_p)
6     else if c ∈ S_p and p_c = p_p then
7         stopTimer(rt_p)
8     end
9 end
```

Algorithm 9: Node p receives $\text{PR}_{(c,*)}$ from c.

this period. After the parent received a *Parent Reply* of a child $c \in C_p$, node p recomputes all child qualities since they vary with each query timer, as shown in Line 4. Detailed information, on how the child quality is computed is given in Section 6.4.5. To follow protocol, parent p schedules an additional *Membership Query* (Line 5) after recomputing all child qualities as a *Parent Reply* of a connected child has been received.

However, if the sender of this $\text{PR}_{(c,*)}$ is a sibling (Line 6), meaning that $c \in S_p$, node c is obviously placed ahead in the set of siblings with respect to p and transmitted its *Parent Reply* to the same parent. In this situation, the sibling p stops its timer, see Line 7. In general, each $c \in S_p$ discards its reply timer $\texttt{rt}_c$ to generate minimum control message overhead.

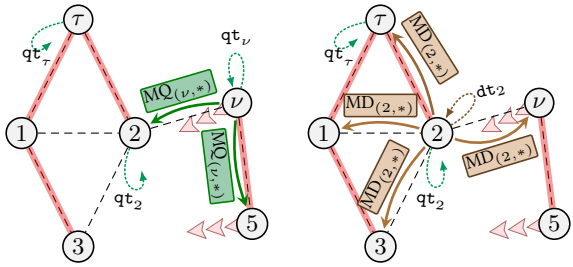
The order of the sets of children aims to reduce the number of *Parent Replies* to a minimum, in the best case to a single message, but also tries to keep the set of children up-to-date. Nodes that have not replied to a parent although positioned before the node that sent the *Parent Reply* are flagged as no longer connected. Because of that, a parent positions a node with the highest probability of being out of coverage range at the beginning of the ordered sequence of children.

However, it cannot be ruled out that nodes behind the one that sent the *Parent Reply* are still connected to the parent. The connections of these children are assumed to remain longer after recomputing their child qualities. Test runs showed that in some cases more than one *Parent Reply* is sent because nodes connected to the same parent are out of each other's communication range.

6.4.4 Tree Merging Process

This section describes protocol-specific processes when two or more trees reach each other. On the one hand, this involves receiving *Membership Detections* that instruct foreign participants to connect to the tree that transmitted this message. On the other hand, this section includes subsequent actions, necessary to achieve a successful tree merging process.

As mentioned earlier, it is most important that there is never more than one controller present in a MANET after a merge situation finished. Controller role negotiations must be completed successfully when two or more trees reach each others' transmission ranges. The parameter $\texttt{tq}_\tau$ is the key decision-maker when determining which tree must drop the controller role. The tree with worse tree quality connects to the tree with better tree quality. Detailed information on tree quality computation is discussed in Sec-



(a) *Membership Query* of Controller ν reaches node 2 of tree T^{τ}, starting tree merge process.

(b) Node 2 broadcasts *Membership Detection* considering tree detection timer $\mathtt{dt}_2$, after comparing both tree qualities.

Figure 6.13: The controller node ν reaches tree T^{τ} decides whether to drop the controller role or not.

tion 6.4.5. Controller negotiation also takes place if the current controller experiences an outage. Directly connected children detect the lost connection because regular *Membership Queries* of their controller are missing. These nodes elect themselves as controller and start transmitting *Membership Queries*, followed by SoCM merge process.

Actions and steps referring to the tree merging process are introduced using the example in Figure 6.13. Two trees exist at the beginning having nodes τ and ν as controllers. Both follow the SoCM process by broadcasting *Membership Queries* when their query timers of controllers and parents run out, as shown in Figure 6.13a with dashed green arrows, pointing to controllers and parents. Also, tree T^{ν} containing nodes ν and 5 moves towards node 2 of tree T^{τ}. Mobility is illustrated with red arrows, pointing in the moving direction. Node 2 is in communication range of ν, receiving its *Membership Query*. Node 2 recognizes that this *Membership Query* belongs to the tree T^{ν} and compares both tree qualities.

The tree quality of its own tree is better resulting in a *Membership Detection* broadcasted by node 2, depicted in Figure 6.13b. Node 2 transmits this message to inform participants of a different tree to connect to. *Membership Detections* are bound to the tree detection timer, which is randomly generated by the sender to avoid multiple *Membership Detections*. A sent *Membership Query* probably reached multiple members of a foreign tree resulting in several triggered *Membership Detection*. Because of that, node 2

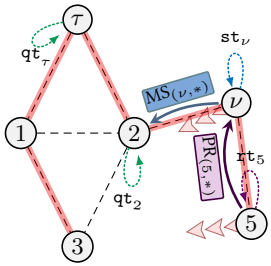


Figure 6.14: Tree merge process confirmed via $\text{MS}_{(2,*)}$ connecting node 2 to new parent 2 of tree T^{τ}.

counted down the randomly generated timer $\texttt{dt}_2$ before it broadcasted the *Membership Detection*. Nodes in close vicinity listen to broadcasted *Membership Detections* of their tree members and discard their timers thereafter. Nodes τ, 1, and 3 discard the *Membership Detection* since they receive the *Membership Detection* of 2 and are part of the same tree. However, receivers of *Membership Detections* belonging to different trees answer with *Membership Switches* messages to inform members of the other tree about a new child and their participants if available.

With respect to Figure 6.14, node ν drops the controller role and connects as a child to its new parent node 2, replying with a *Membership Switch*.

Foreign Tree Detection

Membership Detections are used to advise members of foreign trees to connect to the node that sent the message. These connection requests are only sent if the own tree quality is better after comparing them with the counterpart.

Algorithm 10 describes the process when node m processes a $\text{MD}_{(n,*)}$ received from node n. First, the receiver verifies that both tree members are connected to different controllers, see Line 2. If this identity check returns false, m discards the *Membership Detection* and remains idle since the received message was transmitted by a participant of the same tree (Line 12). If tree ids differ, node m collects all *Membership Detections* potentially broadcasted by multiple members of the foreign tree and selects the parent with the best sub-tree quality (Lines 3 to 10). We define two parameters $\texttt{qs}_m$ and $\texttt{qs}'_m$, initially set to 0, storing the best branch quality of the foreign tree when comparing all qualities received with several *Membership*

```
1  begin ConnectionRequest(MD_(n,*))
2      if r_n ≠ r_m then
3          if qs_m = 0 then
4              startTimer(st_m)
5              qs_m = extractProperties(MD_(n,*))
6          else
7              qs'_m = extractProperties(MD_(n,*))
8              if evalTreeProperties(qs_n, qs'_n) = 0 then
9                  qs_m = qs'_m
10             end
11         end
12     delete(MD_(n,*))
13 end
```

Algorithm 10: Node m receives $\text{MD}_{(n,*)}$ from n.

Detections. Node m uses $\texttt{qs}'_m$ to compare the value with currently received *Membership Detections*.

SoCM is designed to broadcast the least *Membership Detections* as possible. Participants discard desired *Membership Detections* when they receive a *Membership Detection* from neighboring nodes of the same tree addressed to the same destination, as defined in Line 12. In theory, node m should receive only 1 *Membership Detection*. However, disturbed transmissions result in multiple sent *Membership Detections*. Because of that, node m picks the parent having the best sub-tree quality when receiving multiple *Membership Detections*. In doing so, node m stores the branch quality in $\texttt{qs}'_m$ (Line 5) if no *Membership Detection* has been received earlier ($\texttt{qs}'_m = 0$). Also, m starts the tree switch timer $\texttt{st}_m$ in Line 4 indicating the time until the best sub-tree quality at this time is final. This behavior causes m to compare all received sub-tree qualities and select the best one (Lines 6 to 10). Line 8 practically compares both parameters and returns 0 if the newly received sub-tree quality is better.

Foreign Tree Connection

Parent-child connections are confirmed and established by children, delivering *Membership Switches* to desired parents. We now elaborate on node m that receives a $\text{MS}_{(n,*)}$. At first, node m ensures that the desired parent id p_n stored in the decoded *Membership Switch* matches the id of m. Next,

node $m \in M^{\tau}$ of tree T^{τ} computes the child quality of node n and inserts n at the appropriate position of the ordered set C_m. The position of each child $n \in C_m$ is based on the corresponding child quality. Section 6.4.5 introduces the order of this set more precisely. This process considers node n in the subsequent *Membership Queries* by m. Also, m assigns itself the parent role in case no child was connected beforehand.

A $\text{ML}_{(n,m)}$ causes m to remove n from its list of connected children C_m. Node n will no longer be considered in successive *Membership Queries*. Also, node m changes its role from parent to child if $C_m = \emptyset$

Controller Role Self-election

Defining the characteristics and requirements for the controller role depends on the use case. One could define the battery level as the highest priority for controller election. In this case, a node with the most remaining battery lifetime would receive the controller role. The MANET could also strongly rely on a military operation where a predefined sequence of controller ids determines, which controller is elected if the predecessor is not available anymore. In such cases, the tree quality $\texttt{tq}_{\tau}$ represents the battery level or the node ids. Branch and child qualities remain. However, both parameters could also represent other strategies apart from selecting nodes based on their sub-tree structure and their mobility behavior.

Controller redefinitions in MANETs take place if either a node loses direct connectivity to the current controller or a node loses connectivity to its upstream parent. Both can happen to multiple tree members at the same time.

To begin, each node n updates the controller parameter $r_n = n$, electing itself temporally as controller. Next, the corresponding tree quality $\texttt{tq}_n$ is set based on the use case. In case the node that experiences an upstream connection loss has defined the parent role, the branch quality of this node is inserted in $\texttt{tq}_n$. Otherwise, if the node n is a child, the tree quality stores the value of $\texttt{cq}_n$. The branch quality is always preferred over the child quality if two nodes reach each other in this situation. Now these nodes start broadcasting *Membership Queries* containing these modified parameters. Section 6.4.3 elaborates on how nodes process *Membership Queries* if controller ids differ. This is discussed in depth in Algorithm 8 starting at Line 11.

6.4.5 Tree Decision Parameters

We start in this section with how child qualities are composed and their intention referring to further characteristics, such as sub-tree and tree qualities. Next, we elaborate on the set of children and its order based on the child qualities. Lastly, the composition of sub-tree and tree qualities are discussed.

Child Quality

Tree reorganization aims to achieve the least control message overhead heading for minimum parents in trees as these roles permanently broadcast *Membership Queries*, which produce a significant share of the overall control message overhead. Gathering as many as possible children to the least required number of parents reaches this objective more likely. This could be realized self-organized as parents increase their branch qualities with increasing number of children. In contrast, branch qualities decrease when children leave a parent. However, frequent parent switches generate subsequent message exchanges, which lead to fluctuating sub-tree qualities. This also leads to increased parent switches. Sub-tree (branch) qualities are based on the connected child qualities focusing on long connection lifetimes between parent and downstream nodes. The probability that connection lifetime increases if parent and child are in close vicinity and if both mobility characteristics are similar.

Wireless connections between terminals rely on multiple factors, such as reception sensitivity, transmission power, environmental circumstances, among others, as described in Section 4.2.2. These indicators are useful to determine connection qualities and potential connection lifetimes. Using these parameters enriches connection characteristics and makes it possible to derive quality assumptions about them. However, accomplishing best child qualities based on Layer 1 (OSI model) parameters is not the focus of this research. We rather focus on protocol-conform functionality and also aim for minimum control message overhead at runtime. Because of that, we simplify the radio model and make assumptions discussed in the subsequent paragraphs to obtain the child quality for each node.

SoCM is implemented and evaluated with simulation environment OMNet++ [Var10] leveraging abstracted radio medium models. For simplicity's sake, each node has the same communication range, guaranteeing an active connection if origin and target communication distances overlap. Frames do not reach the desired destination only if two or more transmis-

sions within communication range at the same time are present. Moreover, nodes move based on the linear mobility model heading in the same direction for a randomly defined time span. OMNet++ ships in-time simulation parameters of nodes and the environment letting us obtain position data and motion vectors.

Based on mobility characteristics and position information, we can determine whether a child will still be within transmission range of the connected parent after a query timer interval runs out. Thereby, we assume that each participant's direction does not deviate during that time. Extracting the future position of each child $c \in C_p$, where $p \in P^\tau$, after the next query timer $\mathtt{qt}_p$ runs out lets us compute the upcoming assumed reachability $\mathtt{ar}_c = 1 - \frac{(\mathtt{cr}_p - \mathtt{bd}_c)}{\mathtt{cr}_p}$ to the communication border of parent p. The constant $\mathtt{cr}_p$ represents the communication range of p whereas $\mathtt{bd}_c$ stands for the distance of c to the communication border of p after the next query timer $\mathtt{qt}_p$ runs out. The variable $\mathtt{ar}_c$ provides an indication whether the c is within transmission range of the next $\mathrm{MQ}_{(p,*)}$ or not.

However, we also take into account if the direction of c changes within the next query interval. The parameter $\mathtt{wr}_c = 1 - \frac{(|\mathtt{md}_c^\circ - 180^\circ|)}{180^\circ}$ indicates the worst-case direction taking the shortest way out of the communication range of the parent node. Movement delta $\mathtt{md}_c$ represents the deviation of the current direction compared to the direction heading towards the shortest path out of the parent's communication range. Both $\mathtt{ar}_c$ and $\mathtt{wr}_c$ would be 0 if node c is out of the communication range of p at the time the next query interval ends. We assume that children are more likely to turn into the worst-case direction if their current angle is close to this worst-case direction. Combining both, the child quality is composed of $\mathtt{cq}_c = \mathtt{ar}_c + \mathtt{wr}_c$. Child qualities $\mathtt{cq}_c$ are classified as worse the closer the value gets to 0.

Ordering Connected Children

Each parent $p \in P^\tau$ updates the child qualities $\forall c \in C_p : \mathtt{cq}_c = \mathtt{ar}_c + \mathtt{wr}_c$ each time after receiving a $\mathrm{PR}_{(c,*)}$. Next, p arranges each child $c \in C_p$ ascending based on $\mathtt{cq}_c$. Suppose children c_1, c_2, c_3 are connected to parent p where $p \in P^\tau$, having the following child qualities: 0.65, 0.15, and 0.92. It follows that the set C_p is ordered as follows $\langle c2, c1, c3 \rangle$. Nodes at the beginning of $p \in P^\tau$ are more likely to be out of communication range of their parent p within subsequent query timers. The order of a set of children also defines the report order children follow, meaning that a node at the first position of this set starts broadcasting the *Parent Reply*, which is also

received by other children having the same parent. In this case, children of higher order drop their desired *Parent Reply*.

Sub-tree Quality

With the branch quality, it is possible to infer how many children are connected, as well as to draw conclusions about their qualities. Formally, the branch quality $\texttt{bq}_p$, where $p \in P^\tau$, sums up child qualities of all connected children $c \in C_p$ as follows: $\texttt{bq}_p = \sum_{c \in C_p} \texttt{cq}_c$. These branch qualities are decision-makers for children to switch to another parent in case they receive a *Membership Query* from another parent. Based on the protocol of SoCM, parents update their list of children each time they receive a *Parent Reply*.

Tree Quality

Only controllers compute the tree quality of the current tree. The computation is similar compared to the branch quality. Therefore, the controller sums up all known branch qualities as follows: $\texttt{tq}_\tau = \sum_{p \in P^\tau} \texttt{bq}_p$. All parents forward their branch qualities to their parent using *Parent Reply* until all qualities reach the controller. Controllers update the tree quality each time they receive a *Parent Reply*. In addition, the controller includes the tree quality in each *Membership Query*. This quality is forwarded with *Membership Queries* by each parent until leaves of this tree receive this information.

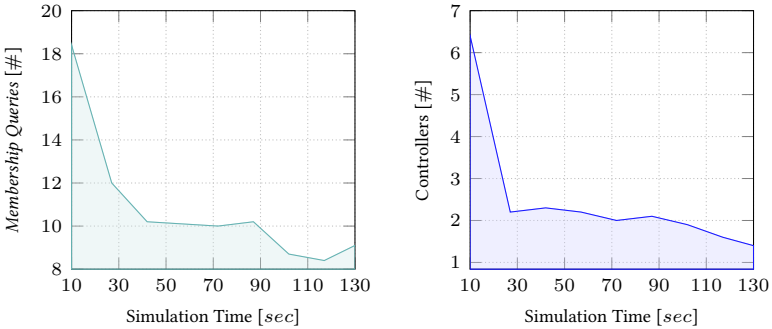
6.4.6 Results

In this section, we perform a tree merging process to examine if the SoCM process, including the proposed controller election, performs as expected. The objective is to verify the correct functionality of SoCM. Thereby, we also take into account produced message overhead. All simulation parameters are listed in Table 6.3.

We placed two initially isolated MANETs on the playground, meaning no participant of both trees reaches any node of the other one. MANET A comprises 19 nodes each connected to at least two participants. The second (MANET B) comprises 5 participants. MANET A is already initialized, meaning the role of the controller and a tree structure already exists. Each participant of MANET B starts from the north-east moving towards MANET A. At approximately simulation time $80\ sec$, both MANETs merge and

Table 6.3: Simulation parameters

Parameter	Value
Number of nodes	25
Communication range	100 m
Bitrate of nodes	2 $Mbps$
Initialization phase	20 sec
Membership Query interval	3 sec



(a) *Membership Queries* send during simulation.

(b) Controllers during simulation.

Figure 6.15: Message overhead and number of elected controller during simulation time.

negotiate the controller role. One controller should drop its role during the merge process resulting in a single MANET. The tree of MANET B will initialize at the beginning to verify if single controller election within the MANET works as expected. The main objective is to examine the behavior and the message overhead of SoCM at the time both MANET trees reach each other. We investigate the number of sent *Membership Queries* and the number of controllers during simulation time. The scenario ran 10 times applying different seeds to average out measurement errors.

Figure 6.15 depicts the number of sent *Membership Queries* and the number of controllers during the entire simulation time. *Membership Query* message overhead at the beginning stands out significantly, referring to the period until simulation time 30 sec in Plot 6.15a. We observed sev-

eral simultaneously broadcasted *Membership Queries* during this time span. MANET B experiences a higher message volume because participants negotiate the controller role, as assumed. Each node periodically broadcasts *Membership Queries* comparing tree qualities and controller ids. The control message overhead decreases thereafter and shrinks further after the described MANET merge situation finished. It can be concluded that fewer parent nodes exist over time.

Plot 6.15b depicts the number of controllers along the complete simulation time. The number of elected controllers is around 6 at the beginning as expected. It takes some time, until approximately simulation time $15\ sec$, until the initialization phase of MANET B finishes. We can observe that the number of controllers is slightly above 2, which is caused by the interference of several *Membership Queries*. Children classify the missing control message as a lost parent connection and assign themselves the controller role, which is dropped after the next *Membership Query* from their previous parent. We experience an averaged slight decrease of controller roles after the merge process at the time both MANETs finish (after simulation time $80\ sec$) reaching almost 1 controller. The temporary assumed controller role of participants is still present after the merging process. Here too, the simultaneous transmissions of *Membership Queries* that result in separations of parent and child connections cause the children to elect themselves as controllers.

This proof-of-concept evaluation verifies the correct functionality of SoCM. We also experience a slight control message decrease at runtime, which can be attributed to a meaningful reordering of the tree at runtime.

6.5 SUMMARY

Relying on an uplink channel in order to provide connectivity to routing services is risky as this single point of failure is error-prone. This uplink channel is used to feed the routing instance with current topology information. Having the complete network topology is essential because this knowledge is required to achieve the desired routing goal to the best possible extent. The objective of this thesis is to distribute all flows in a way to achieve the least network utilization while computing paths of flows that seem to be robust in terms of connection lifetimes.

We proposed a controller-equipped MANET architecture deploying the controller on any arbitrary participant of this network, called Sc-MANET, to overcome the single point of failure issue. This self-organized controller-

managed MANET provides up-to-date topology knowledge on the controller node ready to serve as the routing engine for route requests of MANET participants. To begin, we highlighted drawbacks in terms of incomplete and inconsistent topology knowledge of proactive full topology gathering strategies based on the routing protocol OLSR [Cla+14]. Built on that, we proposed Re-SoTUP [Str+20] providing a reactive topology update process focusing on an up-to-date and complete topology snapshot in terms of nodes and their connections. The algorithm, triggered by the controller, constructs a tree having the controller as root while generating neighborhood information of each node. Next, sub-topologies are forwarded by each node across each branch towards the controller, forming the complete network representation for routing on the controller node.

Re-SoTUP is tested on microcontrollers forming a MANET, and with simulation scenarios, on the one hand, to ensure correct functionality in real-life environments and, on the other hand, to investigate scalability of the algorithm. Results focused on execution times and completeness of responded topologies. The evaluations prove conceptual functionality on microcontrollers. Up to 90 nodes on various playground sizes provide extensive insights regarding completeness, runtimes, and scalability. Topology representations are almost complete but decrease with the number of nodes to approximately 95% if participants select multiple parents to report their sub-topologies to. Closing the gap, Re-SoTUP has been evaluated against OLSR with increasing node mobility configurations. Minimum runtimes of Re-SoTUP prove advantageous when mobility of the participants is configured. The topology representation remains complete even with mobility patterns up to $30\ km/h$. No noticeable inconsistencies were found in outdated or missing topology information. On the other hand, OLSR had to deal with significant inconsistencies, when comparing Re-CeTUP and OLSR regarding missing connections.

Full topology knowledge is only valuable if one has access to it. Route computation must be accessible for each MANET participant at runtime with minimal control message overhead as wireless channels, especially in the military sector, are restricted [MKL15]. Also, controller functionality must be provided in case of outages and MANET merge situations. We, therefore, introduced SoCM [Hag+22] focusing on all-time controller reachability and controller management. The first leverages the tree generated by Re-SoTUP applying node and branch rearrangements to minimize message overhead. Thereby each tree member applies a resource-efficient update mechanism to keep track of its upstream node. The former part of

SoCM provides that only one node is assigned the controller role, considering controller outages and tree merging scenarios of several MANETs. Preliminary results prove correct functionality of SoCM. In addition, we observed a reduced control message overhead.

CHAPTER 7

Time-sensitive Multi-flow Routing

The previous Chapters 4, 5, and 6 provide essential research defining and verifying preconditions, required for achieving minimum network utilization and the most robust paths. Within this RQ, we self-design, extend, and apply heuristics and exact pathfinding techniques to accomplish this routing objective.

RQ 1 investigates transmission overhead in highly-utilized MANET scenarios. More precisely, the chapter provides accurate utilization recordings of single connections and also of routes, using omni-directional antennas. Thereby, RQ 1 also determines an upper utilization recording threshold when measurements become inaccurate. This extension enriches MANET routing in avoiding links not providing sufficient transmission capacities. Even more, one may compute the future utilization of the MANET based on the potential route and data rate of an upcoming flow. The Multi-flow Routing Model (MfDM), proposed in this chapter, leverages utilization recordings in aiming for minimum network overhead.

Furthermore, RQ 2 and RQ 3 propose algorithms and processes to report up-to-date and complete MANET topology snapshots representing all connections and participants to the controller as a graph. This controller is either physically deployed on an outsourced instance, reachable via an uplink channel, or deployed on an arbitrary MANET participant. The latter approach uses one wireless channel for data and control message overhead. Both topology-gathering techniques provide fast runtimes and almost complete knowledge about current connections and MANET participants.

Unifying these MANET-specific extensions of RQ 1 to RQ 3 (Chapters 4 to 6) with the MfDM, generates a complete routing framework for MANETs. Based on that, the objective of this chapter is to investigate if and to which extent routing can benefit from global topology knowledge and knowledge about current utilization of transmitting and idle connections in MANETs. Therefore, we propose several self-developed, extended,

and adapted pathfinding techniques and compare their results with each other. In particular, all approaches are evaluated regarding runtimes, found capacity-conform path settings, and their robustness. The latter represents the lifetimes of found paths. In doing so, we present a link quality metric distinguishing robust from fragile connections.

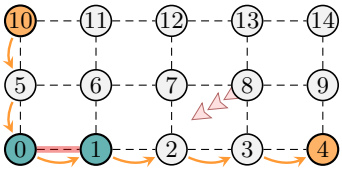
We also introduce the MfDM workflow comprising the MANET Generator and sub-components, the Scenario Generator, MANET Routing Engine, and the Mobility Simulator. The first constructs MANET-like topologies and includes properties of connections and nodes, such as the utilization model, among others, to mimic real-world conditions. The Scenario Generator artificially generates over-utilized network situations that must be resolved by pathfinding techniques of the MANET Routing Engine. The Mobility Simulator simulates mobility to determine how long the transmitting links are connected.

This chapter is organized as follows: To begin, we highlight specify MANET demands regarding multi-flow routing and point out the resulting requirements followed by the objective of this chapter. Next, we extend framework components and architecture parts related to this RQ. We thereafter introduce the MfDM, which represents a evaluation application considering and applying all findings of previous RQs. Thereby, we also introduce briefly how all parts logically fit together. Afterwards, we deep dive into all pathfinding techniques followed by an extensive evaluation. We conclude this chapter with a classification of all pathfinding techniques to recommend, which solution to use in different application scenarios.

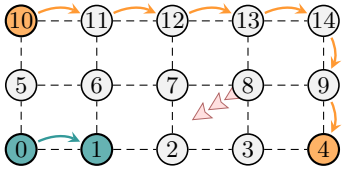
7.1 CHALLENGES AND ROUTING OBJECTIVE

In this chapter, we introduce challenges of multi-flow routing in MANETs with an exemplary scenario to address the requirements and the objective. Assume two flows ($f_{(10,4)}$ and $f_{(0,1)}$) are supposed to be delivered in the grid MANET, as shown in Figure 7.1. Flow $f_{(10,4)}$ (orange colored) is already deployed utilizing all transmitting connections. The link $(0, 1)$ is almost completely utilized, as shown with a red connection. An additional transmission between 0 and 1 (teal colored) is planned, which would exceed the capacity of link $(0, 1)$.

Re-distribution of the already deployed flow $f_{(10,4)}$ resolves the upcoming stressed segment and releases required transmission capacities for flow $f_{(0,1)}$. This guarantees capacity-conform path computations in MANETs. Furthermore, we also aim for robust connections of path participants to ex-



(a) Path of $f_{(10,4)}$ is already deployed utilizing especially edge $(0, 1)$ almost completely. Both flows together would over-utilized this red colored link.



(b) Re-distribution of the already deployed flow $f_{(10,4)}$ provides sufficient transmission capacities for flow $f_{(0,1)}$.

Figure 7.1: Exemplary routing example to avoid an upcoming over-utilized network situation. Colored arrows depict already deployed paths of a flow demanding capacity of the corresponding connection. Red colored links show over-utilized connections. Red arrows represent mobility of MANET participants in a specific direction.

tend the lifetimes of routes. Occurring link breaks of active connections require path re-distributions which produces computation overhead on the controller where routes are computed. In addition, transmissions of broken paths must be paused until new paths are available, otherwise frames will get lost. Speaking of time-sensitive transmissions, such as video calls, these interrupted data deliveries could cause QoS violations.

Link and node characteristics, such as reception powers and speed of nodes classify the robustness of connections. To extend connection lifetimes, path computation techniques must prefer connections, having low speed and high signal connection strength. Consequently, flow $f_{(10,4)}$ is detoured via nodes 13, 14, and 9 instead of node 13, 8, and 9 because node 8 moves away from 13 and 9, which is shown in Figure 7.1b with red arrows.

Being aware of the introduced challenges, multi-flow routing should not generate an over-utilized MANET situation. In particular, no link should be chosen for data delivery that exceeds its configured capacity boundary. On top, pathfinding techniques should focus on robust paths in terms of signal strength and mobility of nodes. Participants in MANETs are moving, resulting in an ever-changing topology. If pathfinding in addition focuses on these characteristics (connection strength and speed of nodes), the chosen links should provide long connection lifetimes.

Furthermore, the runtime of each path computation technique should be kept as small as possible to avoid path deployments on outdated topology

constellations, as participants are moving continuously.

All these mentioned requirements form the overall objective of the MANET Routing Engine which is summarized in the following sentence: Path computations aim to compute a long-living path for each flow in minimum runtime while not over-utilizing any active connection.

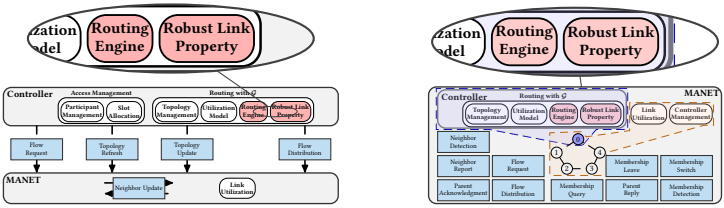
7.2 FRAMEWORK COMPONENTS

The thesis proposes a framework comprising two network architectures, named Cc-MANET and Sc-MANET. We present several pathfinding techniques for MANETs in this chapter, which are included in the MfDM. Each individual pathfinding approach is either deployed on the Sc-MANET, having the controller deployed on an arbitrary node of the MANET (Chapter 6), or included in the Cc-MANET architecture depending on the characteristic of each technique. The controller of the latter architecture is outsourced and reachable via an uplink channel, as introduced in Chapter 5.

Figure 7.2 shows both MANET architectures. Figures 7.2a and 7.2b introduce two new components, named the **Routing Engine** and **Robust Link Property**. We now discuss the functionality of both components and highlight interactions with already existing components.

The controller of both architectures is equipped with the **Routing Engine**, consisting of specific pathfinding techniques all heading for multi-flow distribution, considering the objectives, introduced in Section 7.1. Input parameters are a set of flows, each having origin-target pairs and the required data rate. The topology representation is the data basis for routing. Mobility patterns of participants and links, having characteristics, such as connection strengths, data rates, and current utilizations, are used to describe a MANET topology. Active and passive utilization of links increases due to transmissions as well as logical and physical occupation, based on CSMA/CA since this MAC technique is assumed to be used by all participants.

The **Routing Engine** starts to search for a path for each flow after the topology update process of either Cc-MANET or Sc-MANET is finished. Chapter 5 and Chapter 6 elaborate on how to provide an up-to-date topology knowledge. The former architecture proposes an **Access Management** technique for each participant, in case an uplink channel for each MANET participant is available to reach the controller. This topology update process, named Re-CeTUP, remains the same, meaning the topology snapshot is triggered when the controller receives a *Flow Request*, forcing all



(a) Cc-MANET with Routing Engine and Robust Link Property extension.

(b) Sc-MANET with Routing Engine and Robust Link Property extension.

Figure 7.2: Outsourced and self-organized controller-equipped MANET architectures providing multi-flow routing. Magnifiers highlight new controller entities having the **Routing Engine** and **Robust Link Property** deployed (red-colored). Section 5.3.1 shows Figure 7.2a in original size. This right figure is shown in original size in Figure 6.10. New components not included.

registered nodes to immediately report their adjacency lists via the uplink to the controller in their allocated slot. In this context, deployed and not yet deployed flows are included in the upcoming pathfinding process heading for the mentioned objective. Thereby, the **Robust Link Property** component supports selecting potentially long-living links during path computation. The newly computed path setting is transmitted via the uplink to the MANET. Nodes update their forwarding tables and deliver the flows accordingly.

In case an uplink is unavailable, nodes apply Re-SoTUP and SoCM, included in the framework architecture Sc-MANET. SoCM constantly provides a route to the controller for each participant so that *Flow Requests* reach the corresponding controller. After processing a *Flow Request* the controller starts Re-SoTUP instructing all participants to update their directly connected nodes and report a list, consisting all neighbors back to the controller. Both techniques, SoCM and Re-SoTUP, are introduced in Sections 6.2 and 6.4. After the topology has been reported, the controller takes over and starts the **Routing Engine** using the updated topology. Next, the controller leverages the tree constructed by Re-SoTUP to inform path participants to update their forwarding tables according to the routing information.

The rest of this section introduces a new framework parameter. We must quantify connection robustness to distinguish between seemingly robust and weak connections. We therefore extend the global parameters and

Table 7.1: Global parameters and variables used in multiple RQs.

Properties	Belonging	Description
n	$n \in \mathcal{N}$	Unique id of a MANET participant.
τ	$\tau \in \mathcal{N}$	Unique id of the controller.
o^f	$o \in \mathcal{N}, f \in \mathcal{F}$	Origin node of a flow.
t^f	$t \in \mathcal{N}, f \in \mathcal{F}$	Target node of a flow.
p^f	$p \subseteq \mathcal{N}, f \in \mathcal{F}$	Path as sequence of nodes of a flow.
d^f	$f \in \mathcal{F}$	Demanded data rate of a flow.
u_l	$l \in \mathcal{L}$	Current utilization of a link.
c_l	$l \in \mathcal{L}$	Capacity of a link.
q_l	$l \in \mathcal{L}$	Quality of a link.

variables of this thesis (Table 7.1) with the link quality q_l, where $l \in \mathcal{L}$ represents the degree of robustness. The red-colored row shows this newly defined link quality parameter. The value of the link quality q_l can be any floating number of $\mathbb{R} \cap [0, 1]$. The lower q_l, the better the assumed connection lifetime of l. Path computation techniques use this parameter during route computations to aim for the most robust links.

Subsection 7.3.4, which is part of the following section defines and evaluates the composition of the robust link quality.

7.3 THE MULTI-FLOW ROUTING MODEL

Investigating if and to which extent routing can benefit from accurate connection utilization and up-to-date MANET topology knowledge requires a complete evaluation workflow. This workflow must implement each pathfinding approach and must provide equal MANET scenarios. Equal scenario conditions must include the MANET topology structure, connection characteristics, origin and target pairs including the data rate of each flow, and mobility patterns of participants. To summarize, each run must provide equal conditions for each pathfinding technique to compare all approaches appropriately.

We therefore implemented a simulation environment in Java[1] comprising multiple modules. This evaluation workflow is called MfDM. MfDM provides various MANET topologies and mobility models aiming at real-

[1] https://www.java.com/en/

istic conditions, such as radio signal propagations and mobility patterns, among others. Pathfinding techniques are applied to these generated MANETs. The simulation environment and all dependencies are available on GitHub [2][3][4].

In this section, we first introduce the MfDM and deep-dive into each component. Thereby, we formalize MANET characteristics, such as wireless connection specifications, robust link qualities, and properties of nodes, among others. In general, this worflow unites all components in this evaluation workflow to illustrate how realistic MANETs based on these components are constructed.

7.3.1 Evaluation Workflow

Figure 7.3 depicts the process and all components of the proposed MfDM. The Topology Constructor is capable of generating various topologies, such as grid, random, cluster, among others topologies. Next, the workflow assigns characteristics to vertices and edges of these network graphs to construct realistic MANET topologies. The Topology Constructor is introduced in Section 7.3.2.

Further, the MANET Generator extends the MfDM by adding the Utilization Setup and the Link Property Integration to a generated MANET. These two components comprise the utilization model and the robust link quality. We introduce the former in Section 7.3.3 and the latter in Section 7.3.4.

Thereafter, the Scenario Generator constructs an over-utilized network situation for the previously constructed MANET. In particular, the Scenario Generator searches for a path setting of a predefined number of flows using Dijkstra [Dij59], applying the robust link quality. Thereby this component successively increases the data rates of all requested flows and restarts Dijkstra until a pre-configured degree of over-utilization is reached. The source-target pairs of all flows and their associated data rate demands are taken by each pathfinding approach in order to resolve these over-utilized segments by finding new paths in the same MANET. No transmitting link chosen as part of a path must exceed its transmission capacity. We elaborate on generating over-utilized network situations in Section 7.3.5.

The MANET Routing Engine comprises all pathfinding techniques introduced in Section 7.4. In a real scenario, the controller would receive

[2]https://github.com/F1-C0D3/JGraphLib
[3]https://github.com/F1-C0D3/MANETModel
[4]https://github.com/F1-C0D3/MANETRoutingInstance

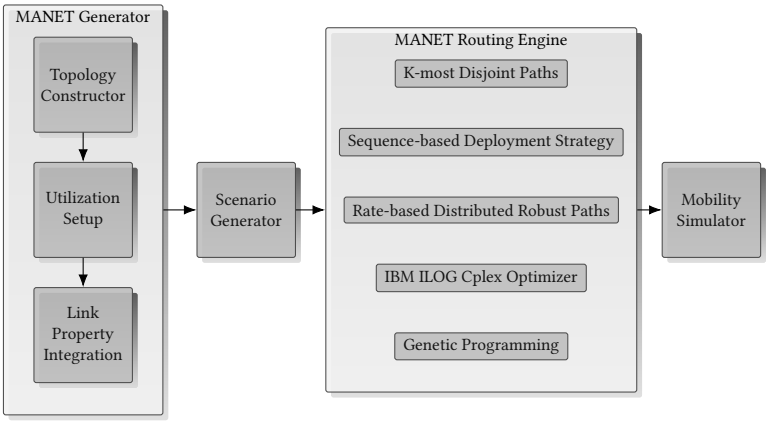


Figure 7.3: Evaluation workflow of the MfDM.

a *Flow Request*. Next, the controller would either start Re-CeTUP or Re-SoTUP, depending on the network architecture. The current topology representation is then used by the controller to find possible path combinations that would not overload the actual MANET in theory. After a successful calculation, the resulting routes are transmitted to the network with a *Flow Distribution*. The MANET Routing Engine simulates finding these paths, applying several pathfinding techniques to identify the advantages and disadvantages of each technique to discuss which approach to use in which situation.

Lastly, the Mobility Simulator evaluates the robustness of each capacity-conform path setting, computed by each approach, by measuring the time each path remains connected. Furthermore, the Mobility Simulator allows for the configuration of different mobility models, comprising speed and update times. This component is introduced in Section 7.3.6.

The following section introduces each component of the MfDM in detail.

7.3.2 Topology Constructor

The component Topology Constructor provides various typical MANET network topologies ranging from a grid, to a cluster, to random layouts. At the beginning, vertices are placed on the network connected via edges based on the topology type. These entities represent nodes and their links

to other participants when assigning common MANET characteristics.

In general, connections between nodes are generated depending on the distances between them. To begin, the MfDM allows to specify the maximum distance between two nodes. A connection between two participants exists if the distance is below the maximum. One can also specify the minimum distance between nodes to define the topology structure more precisely. Additional link properties, that characterize connections are discussed in Section 7.3.2.

The grid topology is mainly used for testing and debugging of routing, and shortest paths algorithms, as well as to verify the correct functionality of the utilization model, which represents the effect of using the wireless channel with the MAC protocol CSMA/CA [IEEE802.11]. The grid graph is configurable regarding the network size. The number of participants is adapted based on the configured height and width.

Cluster topologies are mainly developed for military transmission simulations. A cluster in this topology represents an individual unit in which nodes communicate with each other within and beyond their own cluster. Therefore, cluster graphs construct corridors, containing a crowd of nodes (clusters). Only a fraction of participants have a connection to a neighboring cluster node, constructing a potential route to each node of the entire network. Cluster topologies can be configured based on the number of nodes, the distances between them, and specific cluster properties. These specific properties range from size in terms of height and width to cluster distances.

Random topologies produce networks in which nodes are arbitrarily placed around already existing nodes following predefined constraints to guarantee a certain graph density. Thereby, node distances are also considered to stretch the graph in its size. One can increase the number of nodes around existing ones, thus increasing the number of neighbors of each participant as well as the graph density. This makes this topology scalable when increasing the number of topology participants. Also, when running the Topology Constructor with the random graph topology constructor, one can configure the playground size in length and width in combination with the number of participants. Combining these parameters with a minimum and maximum edge distance of nodes achieves a flexible and balanced number of connections. The minimum distance defines the distance, two randomly placed vertices must have from each other. The maximum distance determines if a link exists when generating link characteristics. The link characteristic of connections is described in this section.

Practically, the random topology generator starts with an initial vertex randomly placed on the defined playground size. Each further vertex is only placed on the playground if the randomly defined position is below the minimum configured distance to any surrounding node and if this node is within the maximum defined distance of at least one node. This generates topologies in which the minimum distances between nodes are provided.

This topology structure is well suited to outline the advantages and disadvantages of all pathfinding techniques as the amount of possible path settings increases based on the node degree and the distances between them. Because of that, this topology type is mainly applied throughout upcoming evaluations.

After generating a suitable topology, each vertex receives capabilities to act as MANET participant. These characteristics are described in the upcoming section. Furthermore, we extend all edges with additional characteristics to formalize robust link qualities, also discussed in this section.

Node Characteristics

Vertices become MANET nodes when the Topology Constructor extends the vertices with simple and complex properties. MANET participants are expected to move which is why nodes have assigned mobility patterns generating various velocities heading in different directions. The MfDM records speed and directions of all nodes at predefined time stamps, called ticks.

Starting with the simple properties, each node is assigned a transmission range that is identical to the maximum distance used in the MANET Generator to place participants on the area. With respect to complex configuration parameters, nodes are configured with a carrier frequency. We used the $2.4\ GHz$ frequency for our evaluations. Also, each node $n \in \mathcal{N}$ has the property transmission power $\mathtt{tp}_n$ as well as the reception threshold $\mathtt{rt}_n$ (both in watts).

The node mobility is also partitioned into simple and complex properties. The simple property returns the last recorded tick of each node which can be used to form the link quality. The complex property computes the weighted moving average of i previous ticks, where i can be configured. Ticks closer to the current time are given more weight. We use the complex speed recording technique for our evaluations.

Link Characteristics

Edges become links in extending them with additional properties. Therefore, each edge of a graph is also extended with either simple or complex properties.

Each simple link has the properties capacity c_l and utilization u_l, defined in Table 7.1, where $l \in \mathcal{L}$. When using the simple radio model, the first property (c_l) is set before the simulation starts. When applying the complex model, each link $l \in \mathcal{L}$ also has the property reception power $\mathtt{rp}_l$, which is based on the free space path loss model [Cer19] and defined as follows:

$$\mathtt{rp}_{(n,m)} = \mathtt{tp}_n \cdot \left(\frac{w}{4 \cdot \pi \cdot d_{(n,m)}}\right)^2 \tag{7.1}$$

The reception power $\mathtt{rp}_{(n,m)}$ of the link (n, m) with receiver m depends on the transmission power $\mathtt{tp}_n$ of the sender n which is multiplied by the free space path loss model. This model describes the quadratic signal power loss of radio waves depending on the distance between sender and receiver and the wavelength. The parameter w represents the wavelength. The function $d_{(n,m)}$ returns the distance between nodes n and m, where $(n, m) \in \mathcal{L}$. In general, frames can only be decoded if $\mathtt{rp}_{(n,m)} \geq \mathtt{rt}_m$, where $\mathtt{rt}_m$ represents the reception threshold of node m.

The link capacity c_l, where $l \in \mathcal{L}$, is computed with the Shannon-Hartley theorem [Des09] when using the complex model (Equation 7.2).

$$c_l = \mathtt{bw} \cdot \log_2\left(1 + \left(20 \cdot \log_{10}\left(\frac{\mathtt{rp}_l}{\mathtt{bn}}\right)\right)\right) \tag{7.2}$$

The voltage of radio waves arriving at the receiver's antenna depends on the distance between two nodes. The transmission rates (Mbps) of links decrease with increasing distances between two nodes. This equation uses the received reception power $\mathtt{rp}_l$ of link $l \in \mathcal{L}$ to compute the theoretical transmission capacity c_l. Thereby, $\mathtt{bw}$ and $\mathtt{bn}$ represent the channel bandwidth and the background noise, respectively.

7.3.3 Utilization Setup

At this point, nodes and their links are classified MANET-like, comprising connection characteristics, such as maximum transmission capacities, reception powers, and robust link qualities, among others. Nodes have transmission powers and apply mobility patterns.

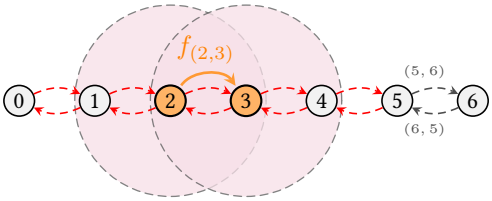


Figure 7.4: Utilized links (red-colored) based on a deployed flow $f_{(2,3)}$ from node 2 to 3 applying the MAC protocol CSMA/CA. Dashed circles represent transmission ranges of 2 and 3.

Next, we propose the formal description of link utilization and implement this model in the MfDM. This model is used to theoretically identify all links being utilized because of a transmission of a node in the MANET. All affected links reduce their residual transmission capacities according to the required data rate of a flow. Applying this model lets us analyze which links exceed their transmission capacities in decreasing their capacity according to the data rates when computing a path setting of multiple flows.

Section 4.2.2 elaborates on signal propagation of wireless transmissions especially in multi-hop networks, such as MANETs. The formal definition of link utilization is based on this section and determines the links being utilized based on the path and data rate demand of a specific flow when applying the MAC protocol CSMA/CA.

The utilization concept is introduced with the example in Figure 7.4, showing a MANET as a directed graph, forming a chain where node 2 transmits flow $f_{(2,3)}$. Red-colored links show which connections between nodes are occupied and consequently utilized during frame sequences of this flow. A frame sequence describes the delivery of a data frame based on CSMA/CA comprising its mandatory control frames RTS, CTS, and ACK. A detailed description is introduced in Section 2.4.1. Dashed circles represent the transmission ranges of nodes 2 and 3.

At first, node 2 broadcasts a RTS to 3 trying to reserve the channel if 3 is free for reception. The transmitted RTS also reaches node 1 since the transmission range of 2 is omni-directional. This forces node 1 to stay idle according to the NAV for the upcoming frame sequence between 2 and 3 when decoding the RTS of 2. Suppose, node 0 starts a transmission request to 1 during the frame sequence between 2 and 3. Node 1 would not respond in this situation. Next, node 3 responds to 2 with a CTS also reaching node 4, which, like node 1, will stay idle during the upcoming data

exchange. Generally, all surrounding nodes receiving RTS or CTS are not able to transmit or receive data. Because of that, transmission capacities of all red-colored links which are connected to nodes receiving either a RTS or CTS are reduced by the data rate of flow $f_{(2,3)}$.

More formally, the set of utilized links $\mathcal{U}(n,m)$ based on the transmission of a link (n,m), where $(n,m) \in \mathcal{L}$ is defined as follows:

$$\mathcal{U}(n,m) := \{(i,j) \in \mathcal{L} | i \in N(n,m) \vee j \in N(n,m)\} \tag{7.3}$$

The set $N(n) := \{m \in \mathcal{N} | (m,n) \vee (n,m) \in \mathcal{L}\}$ represents the set of nodes which are directly connected to n. $N(n,m) := N(n) \cup N(m)$, where $(n,m) \in \mathcal{L}$ unites all neighboring nodes of n and m. Based on that, each link $l' \in \mathcal{U}(l)$ increases its utilization $u_{l'}$ based on the configured data rate d^f of flow f. Equation 7.3 is applicable to all active links currently transmitting data to determine all links in the network being utilized at this point. This lets one compute to which degree each link is utilized and also how much capacity is left for an upcoming flow and its data rate demand.

In general, we distinguish between actively and passively utilized links. An active link is defined as a connection currently transmitting or receiving a flow and consequently utilizing the link. A passively utilized link is also a connection of the set of utilized links that is also utilized but is not transmitting or receiving data actively of any flow. In particular, suppose multiple flows are deployed resulting in multiple paths. A link a might be included in $\mathcal{U}(b)$ and $\mathcal{U}(c)$, where connections b and c forward different paths and a is not actively transmitting. Link a would be utilized twice by the data rates of both flows and would decrease its transmission capacity c_a accordingly. However, a link could also be elected by multiple flows to actively transmit or receive data.

The controller uses the network graph $\mathcal{G}$ (links and nodes) including all properties, such as transmission capacities c_l, mobility patterns, and speed s_n, among other, where $l \in \mathcal{L}$ and $n \in \mathcal{N}$, and applies the pathfinding techniques to the network representation. In doing so, the controller deploys and undeploys paths of flows on its current topology representation, not on the actual MANET. Deploying a path on the topology representation increases the utilization of links whereas undeploying does the opposite.

7.3.4 Link Property Integration

This section first verifies to which extent speed and connection distances of nodes affect the lifetimes of links in MANETs. We carried out a proof-of-concept simulation in order to determine the efficiency of node speeds and

connection distances as robust link quality representation. Therefore, we adapted the costs of links to determine the robust link quality when computing paths with Dijkstra [Dij59]. Furthermore, we formalize the robust link quality which is applied in all evaluations of this chapter.

Assessing Speed and Connection Characteristics

The mobility of MANET participants forces path computations to respond quickly with an adequate path setting for all current flows as active links are prone to disconnect and new links may emerge. The faster new paths are deployed on a current topology, the longer the new path setting will last as participants are moving even during path computations. However, route lifetimes can further be influenced when including additional characteristics of connections and mobility patterns of nodes in path computations. For instance, two nodes in near vicinity both moving slowly will probably be connected longer compared to two nodes moving fast in opposite directions.

Routing protocols, such as AODV [DPB03] aim for the least number of hops when generating the shortest paths from origins to targets, assuming no further influences, like overloaded network segments. Destinations respond to the first RREQ message with a RREP and ignore further RREQs. The first request most likely uses the least number of intermediate hops as packet processing on the nodes takes the most time during route construction. However, such paths do not last long as the next hops of established routes are more likely almost out of coverage of their predecessor nodes, which makes path robustness of AODV fragile. Disconnections arise if path participants move out of each others' communication ranges.

We have observed the same behavior when applying standard Dijkstra [Dij59] with edge weight 1 as this weight aims for the same objective. This configuration also returns paths for a particular node or all participants of a network having minimum intermediate nodes which also results in short connection lifetimes [SSG20].

Based on that knowledge, we propose a proof-of-concept scenario to investigate if and to which extent speeds and distances of participants influence the connection lifetimes of active links compared to the lifetimes when using standard Dijkstra (link weight 1). The scenarios aim to test the self-defined connection qualities, which is why less emphasis was placed on a realistic simulation environment. For instance, the distances between participants were used as an indicator to characterize connections in terms of robustness, as general improvements of our defined link quality are in

focus at first.

Keeping that in mind, the following evaluation is carried out with the Cc-MANET architecture using Pro-CeTUP as topology update technique. Only for these runs, the controller has a modified link weight metric equipped to generate robust paths. We therefore defined and implemented a link weight comprising mobility and connection distances. For these scenarios, MANET participants include positions (x-coordinates and y-coordinates) and their speed patterns in *Topology Update* messages, enabling the controller to include distances between nodes and their mobility patterns in the path computations.

We simulated 30 mobile MANET nodes, equipped with 2 $Mbit$ data rate and 110 m communication range on a 430 m $\times$ 190 m playground. Nodes are moving based on the RWP mobility model [AWS06]. Each participant randomly chooses a speed between 0 km/h and 60 km/h for 8 sec to 10 sec and changes direction (0° to 180°) randomly. 5 flows, each demanding 50 $Kbps$, are transmitted during the simulation time which is set to 150 sec. Origin nodes request a route from the controller using the uplink channel and transmit data till the simulation ends. The controller detects disconnections in already existing routes when constructing the network graph with *Topology Updates* and recomputes paths if link breaks are detected. We evaluated this setting with the *Topology Update* time spans $S = \{\mathtt{ui}_1, \mathtt{ui}_2, \mathtt{ui}_3, \mathtt{ui}_4, \mathtt{ui}_5, \mathtt{ui}_6, \}$ to also investigate if route lifetimes decrease with decreasing accuracy as topology representation becomes inaccurate with increasing neighbor update and topology update intervals. For instance, update interval $\mathtt{ui}_1$, triggers each MANET participant to transmit *Neighbor Updates* and *Topology Updates* every second.

The link weight is partitioned into two parts. The first comprises the speed of the next hop node of the potential route towards the destination. The second represents the distance between two nodes n and m, where $(n, m) \in \mathcal{L}$. Together, these two characteristics represent the link quality, stored in each edge weight. With respect to the speed classification, we define 4 speed classes, each ranging 15 km/h. Class 1 for instance comprises 0 km/h to 15 km/h. The link weight increases linearly with each speed class. Furthermore, the closer the participants are to each other, the lower (better) the link quality is defined.

Distances of connections below 15 m and above 85 m are not considered during path construction to first of all avoid to many route participants lined up one after another and second of all to eliminate connection interruptions arising immediately after route construction. The speeds of the

Table 7.2: Comparing robust link quality (w_r) based on speeds and distances of nodes with standard link weight 1 (w_s) using Dijkstra.

S	**Througput** $[Mbps]$		**Route Replies**		**Route Length**	
	w_r	w_s	w_r	w_s	w_r	w_s
$\texttt{ui}_1$	14.28	15.97	20365.0	51333.9	6.06	5.27
$\texttt{ui}_2$	15.35	14.63	13122.4	56795.0	5.68	3.47
$\texttt{ui}_3$	15.18	13.03	7505.6	24776.2	5.25	4.11
$\texttt{ui}_4$	15.59	12.27	6912.7	27240.1	5.95	4.09
$\texttt{ui}_5$	12.08	11.31	13393.1	26111.9	4.45	2.90
$\texttt{ui}_6$	12.92	12.39	8273.1	32622.0	4.13	3.26

residual connections are similarly defined compared to the distance classification between nodes. Compared to the distance boundaries, nodes are also considered if their speeds approach $0\ km/h$. Overall, velocities are normalized in the range $[0, 1] \in \mathbb{R}$ and also partitioned into 4 classes of equal size.

Each speed and distance class is assigned a weight. The weights of speed and distances are the same, meaning the speed of class 2 has the same weight as a distance of class 2. All weights of links, whose distances are below $15\ m$ and above $85\ m$ are set to $|\mathcal{L}| \cdot |\mathcal{N}|$. The controller adds up both weights of each link when computing the paths for the requested flows.

The focus of this evaluation is to investigate if and to which extent speeds and distances play a role in extending the lifetimes of computed paths. We also measure the throughput to find out if the route length has an influence on the lifetimes of paths. Table 7.2 shows all recorded results which have been run 10 times to average out measurement errors. The link weight w_r represents the robust link characteristic whereas Dijkstra uses 1 as link weight when w_s is configured.

Starting with throughput measurements, the link weight w_r delivers more data compared to w_s. It can be concluded that the routes of w_r remain connected for longer periods of time than paths configured with w_s. More data is lost with w_s as connections break down more often resulting in dropped packets at least until the next *Topology Update* period ends. In addition, the controller transmits fewer *Flow Distributions* to the MANET via the uplink channel if routing with w_r. New routes are only deployed if an active connection fails, otherwise, the controller stays idle. If nodes

lose connectivity frequently, they will notify this issue to the controller and then they will receive new routes via *Flow Distributions*. It turns out that routes computed with w_r last longer, referring to the number of sent *Flow Distributions*. However, w_r excludes connections above $85\ m$ radius, leading to more path participants, which is also shown in the last two columns of this table. Integrating more nodes in routing increases the utilization of the entire MANET. This leads to less transmission capacity being available for further transmissions but on the other hand, leads to more link breaks of active connections.

These proof-of-concept results show that applying a compromise between connection distances and speed of nodes increases the lifetimes of routes. Furthermore, aiming for the least distances between intermediate route participants increases the utilization of the MANET as more nodes forward data to the destinations.

Formalizing the Robust Link Quality

Having a MANET comprising node and link capabilities as well as a wireless utilization model, we extend the MANET with the robust link quality to classify each connection in the network accordingly. The previous Section 7.3.4 provided insights through the proof-of-concept assessment, which connection properties are well suited to describe robust links. Based on that knowledge, we formalize a link quality, focusing on speed of nodes, connection strength, and the relative directions of nodes.

We apply the weighted moving average to describe the speed over a time period starting in the past. This technique includes multiple speed ticks starting at a specific point in the past to the latest speed tick. Current speed recordings are ranked higher than ones obtained in the past. Based on that, function $s(n)$ returns the normalized speed of each node $n \in \mathcal{N}$. Unlike the link characteristic of the proof-of-concept evaluation, we use the reception power to evaluate the connection quality and reliability. The maximum and minimum defined distances between participants, applied to the proof-of-concept evaluation, have proven to be promising and are therefore also used. In doing so, we define the confidence range comprising a lower bound reception power $\texttt{lr}_{(n,m)}$ and an upper bound reception power $\texttt{ur}_{(n,m)}$. Similar to Equation 7.1, we compute $\texttt{lr}_{(n,m)}$ and $\texttt{ur}_{(n,m)}$ by adjusting the distance $d_{(n,m)}$ between nodes n and m to obtain the upper and lower bounds. The function $z(n, m)$ returns the normalized value representing the connection strength.

The robust link quality also depends on the movement directions of n

and m, where $(n, m) \in \mathcal{L}$. Therefore, we compare the direction deviations of the last two mobility ticks of nodes n and m. The closer the deviation of both directions the better the normalized result of function $a(n, m)$.

Equation 7.4 applies all functions to compute the robust link quality of link $(n, m) \in \mathcal{L}$

$$q_{(n,m)} = z\big(n, m\big) \cdot w_z + s\big(m\big) \cdot w_s + a\big(n, m\big) \cdot w_a \tag{7.4}$$

Weights w_z, w_s, and w_a, associated to each function increase and decrease the value of each property. We obtained the best weight setting during test runs and used it for all evaluations.

Having determined the link quality of each connection in a MANET, one can derive the quality of an arbitrary path p^f of flow $f \in \mathcal{F}$. In particular, the $\mathtt{qp}^f$ is defined as the sum of all link qualities chosen for a specific path of flow f. Equation 7.5 defines the path quality $\mathtt{qp}^f$ where $f \in \mathcal{F}$.

$$\mathtt{qp}^f = \sum_{i=0}^{|p^f|-1} q_{(n_i, n_{i+1})} \tag{7.5}$$

A path p^f, defines an indexed sequence of nodes $n_i \in p^f$, where $0 \leq i < |p^f|$ and each $(n_i, n_{i+1}) \in \mathcal{L}$. Further characteristics are described in Section 4.2.1. According to Equation 7.5, we sum up each link quality of such a path p^f of flow $f \in \mathcal{F}$, starting with origin source o^f and ending at the link with sink node t^f.

The quality of a path $\mathtt{qp}^f$ always depends on the path of flow $f \in \mathcal{F}$ as long as each flow only stores a single path. However, specific pathfinding techniques, introduced in the Section 7.4 will store multiple paths of each flow, which requires to compute a path quality for each path of each flow. The individual data structures, describing multiple paths of a flow are introduced in these corresponding sections.

7.3.5 Scenario Generator

The MANET Generator ends up providing a complete and realistic MANET topology representation. To resolve an over-utilizing path setting, one must generate routes of multiple flows, exceeding the transmission capacities of links. The Scenario Generator focuses on generating over-utilzed MANET scenarios in defining data rates of flows that exceed the transmission capacities of links.

With the Scenario Generator, one can configure the number of flows, the degree of over-utilization in percentage, and a factor specifying how the data rate of each flow increases during computation. The latter specifies the accuracy of the MANET's over-utilization. Furthermore, one can configure unique source-target pairs or random ones. The unique configuration forces the model to not select an already taken node as origin or target for any further flow. This configuration distributes all flows even more across the MANET, also generating a more efficient transmission utilization distribution. Also, it is unlikely that participants use two real-time applications simultaneously. As a recap, the field of applications assumes real-time services, such as video calls or VoIP calls, as described in Section 1.4.

At runtime, the Scenario Generator applies Dijkstra multiple times for each flow and increases the transmission rates during this process until the predefined over-utilization degree is reached. All data rate demands are set to 0 at begin. During each iteration, the Scenario Generator applies Dijkstra [Dij59] in combination with the robust link quality and computes and deploys each path of each flow successively. Thereby, each $l' \in \mathcal{U}(l)$ increases $u_{l'}$ by the data rate d^f of flow $f \in \mathcal{F}$ if l is selected as a link actively transmitting data, where $l \in \mathcal{L}$. Thereafter, if over-utilization in the MANET is present, meaning that at least one active link exceeds its transmission capacity, the utilization of all active links is set in relation to their capacities and the Scenario Generator verifies if the over-utilization exceeds the predefined degree of over-utilization. If not, the data rate of each flow is increased by a value, randomly picked out of 0 and the initially defined increase factor. Next, the Scenario Generator undeploys each path in order to set the MANET links to the initial conditions, meaning the complete capacity of each connection is available again.

This process is repeated until the degree of over-utilization is reached. The Scenario Generator undeploys and deletes all paths and returns all flows, comprising origin-target pairs and their data rate demands which generated the over-utilized path setting, when the configured degree is reached. Both, origin-target pairs and the data rate demands are input parameters for the MANET Routing Engine to find paths in the same MANET which do not over-utilize any active link of the network.

The degree of over-utilization mostly exceeds the predefined configuration as the algorithm stops when the current setting is reached. The increase factor must be kept small to provide minimal deviation, which is important as runs should be comparable.

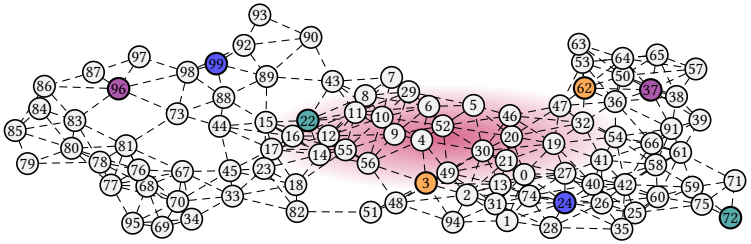


Figure 7.5: Exemplary MANET applying the Scenario Generator comprising 4 colored origin target pairs after artificially generating an over-utilized situation. The red ellipse represents a possible network region having active links that must carry data beyond the configured capacity. Circles represent MANET nodes and dashed lines the connections between them.

Figure 7.5 depicts an exemplary MANET after running through all steps from Section 7.3.2 to Section 7.3.5. At this stage of the MfDM, paths and data rate demands of flows have been determined, over-utilizing a single or multiple network segments to the configured degree which are depicted with red ellipses in the background. Having data rates and origin-target pairs at hand, the next step is to apply pathfinding techniques to resolve the over-utilization and also to construct long-living paths.

7.3.6 Mobility Simulator

The mobility model and required parameters are defined and applied with the MANET Generator, as a couple of mobility ticks are simulated before pathfinding starts. This is due to the fact that the robust link qualities, described in Section 7.3.4, require information about the movement patterns of participants to make a statement about their lifetimes. Based on that, the Topology Constructor computes 20 ticks of each participant before all link qualities are determined. Nevertheless, the Mobility Simulator is intended to verify how long active links stay connected when nodes move based on the configured mobility model. Therefore, the next movement patterns are appended to the already calculated ones or continued according to the configured mobility model.

The MfDM provides the RWP and the GM mobility models [AWS06]. The RWP model introduces a straightforward mathematical description to define the movement pattern of nodes with only a minimum of deterministic rules. With the RWP, each node randomly chooses speed and direc-

tion and moves accordingly. The time each node maintains a certain speed and direction depends on the configuration of the RWP model. Usually, this timespan can also be chosen randomly. When this timespan is exceeded, nodes choose directions and speeds again randomly and behave as described. Speeds and timespans, the RWP randomly chooses from are configured as ranges. Both models also include waiting times before nodes choose new directions and velocities.

In contrast to the RWP, GM generates a more intuitive and natural movement pattern, so that direction and velocity changes are determined based on the previous ones. In particular, one can configure the degree of speed and direction deviations. This means that it can be defined, how far the current direction can differ from the previous one and how much velocity of the current tick can be changed compared to the previous. Deviations are also modeled as ranges, randomly chosen by the GM mobility model. At the beginning, speeds and directions are defined completely randomly.

Our evaluation in this chapter is based on the GM mobility model as it imitates the behavior of pedestrians which is well suited for our purpose.

Coming back to the workflow, each active link is monitored if the connection after each elapsed tick is still present. In doing so, the MfDM compares the reception power with the reception threshold to verify if transmissions are still decodeable which is true if $\mathtt{rp}_{(n,m)} \geq \mathtt{rt}_m$, where $n, m \in \mathcal{N}$.

In addition, MfDM records the time when a capacity-conform path setting has been deployed until the connection of any active link breaks. The elapsed time is compared against all other pathfinding techniques which ran on exactly the same scenario (equal MANET, over-utilizing flow setting, and mobility model).

7.4 THE MANET ROUTING ENGINE

We introduce five pathfinding techniques integrated into the MfDM aiming for capacity-conform and robust path settings in minimum runtimes. To provide diversity, the techniques comprise exact and heuristic approaches, having been self-developed, extended, or applied.

The first self-developed approach is called K-most Disjoint Paths (KmDP). This technique generates paths having minimum similarity in terms of commonly utilized links per flow while searching for combinations aiming for a robust and capacity-conform path setting. The Sequence-based Deployment Strategy (SbDS) is also self-developed, applying the Greedy algorithm when comparing various path combinations considering all flows. Rate-

based Distributed Robust Paths (RbDRP) extends previous work of Akin et al. [AK19] with the wireless utilization model and the robust link characteristic while focusing on minimum runtime. The CPLEX, provided by IBM [5] is an approximation technique, applying methods like Branch & Cut and Branch & Bound [Mit09] to compute the global optimum. We propose and inject a formalized model describing the objective and associated constraints for this solver to first of all claim if a solution exists for each run and second of all to compare the various approaches. For the latter, we restrict the execution runtime of CPLEX accordingly. Also, we apply GP, which self-generates and improves solutions in interpreting the result of the fitness function. GP is modified to accomplish the desired objectives while keeping runtimes as small as possible.

The following sections introduce each pathfinding technique in detail.

7.4.1 K-most Disjoint Paths

The purpose of KmDP is to generate plenty of paths for each flow and pick a combination consisting of exactly one path of each flow, having good link quality and no over-utilization. KmDP is inspired by the Baseline Solution of Chondrogiannis et al. [Cho+15]. Their research focuses on multiple paths for a flow with minimum similarity while also trying to reduce the execution time of their approach.

Based on that, a well-defined number of paths for each flow is computed at first having a defined set of characteristics, distinguishing them from each other. Secondly, a rating is established, determining the best combination (a potential path setting) having no over-utilization and long-living connections. This requires the following predefinitions: Adequate disjoint-path characteristics must be defined regarding the similarity of paths of a single flow. Also, it must be considered which path of each flow to combine with a path setting in order to elect the setting fulfilling the objective the best.

We believe that searching for multiple paths consisting of a most disjoint characteristic, such as the most different links is a promising approach as generating a diverse distribution of connections increases the probability to find a capacity-conform path setting for multiple flows. Having a list of most disjoint paths per flow requires an appropriate selection strategy as the number of possible combinations increases with the number of flows.

[5] https://www.ibm.com/de-de/products/ilog-cplex-optimization-studio

```
1  begin evolveKmostDisjointPaths(F, k)
2      S, D = {}
3      foreach f ∈ F do
4          P^f = computeKDisjointPaths(f, k)
5          D = D ∪ P^f;
6      end
7
8      while k > 0 do
9          while |s| < |F| do
10             P^f = randNotIn(s, D)
11             dr^best = ∞
12             foreach p_n^f ∈ P^f do
13                 dr_n^f = sim(p_n^f, s)
14                 if dr_n^f < dr^best then
15                     dr^best = dr_n^f
16                     p^best = p_n^f
17                 end
18             end
19             s = s ∪ p^best
20             P^f = P^f \ {p_n^f}
21         end
22         S = S ∪ s
23         s = ⟨⟩
24         k = k − 1
25     end
26     return bestSetting(S)
27 end
```

Algorithm 11: Path setting computation using KmDP.

Because of that, we compared the so-called random with the top-down selection strategy and compared, which approach finds more capacity-conform path settings. Both techniques returned a similar number of found path settings whereas the random selection strategy found a few more capacity-conform results, which is why we decided to include this approach in KmDP.

Enlightened by the k-shortest path algorithm [Yen71] we propose an adaption, focusing on maximum disjoint paths while computing k number

of paths per flow where k is configured at startup of the MfDM when using KmDP.

The Algorithm 11 introduces each step of KmDP in detail. Function COMMPUTEKDISJOINTPATHS returns a set of paths with an equal origin and target pair for each flow. In particular, mutation paths of the initial or $k-1$ paths are generated, removing nodes and edges to provoke uniqueness. All unique paths of the k run are stored in a set comparing each of them with already elected paths of previous shortest path computations of this run. Therefore, we set all links of both paths in relation to common links to determine minimum similarity when comparing two unique paths of the same flow. The most disjoint k paths of a flow f are stored in the set P^f, where $f \in \mathcal{F}$, as shown in Line 4. This function returns the set $P^f := \{p^f_0, \ldots, p^f_{k-1}\}$ of length k, where each path from index 0 to $k-1$ is unique and depends to the flow $f \in \mathcal{F}$. The set of k paths of all flows is stored in $D := \{P^{f_0}, \ldots, P^{f_{|\mathcal{F}|-1}}\}$, when altering through this for loop, as shown in Line 5.

The next part of the algorithm generates combinations, so-called potential path settings, each stored in a sequence s. All these elected settings are stored in the set S, where $S := \{s_0, \ldots, s_{k-1}\}$. Each setting $s = \langle p^{f_0}_{if_0}, \ldots, p^{f_z}_{jf_z} \rangle$ stores exactly one path of each requested flow, where $z = |\mathcal{F}| - 1$. In other words, there must exist exactly one path of each flow in this list. As a consequence, each $s \in S$ stores exactly $|\mathcal{F}|$ elements. The paths $p^{f_0}_{if_0}$ and $p^{f_z}_{jf_z}$ of flows f_0 to f_z of a path setting s are picked based on a specific logic, introduced in the subsequent lines of this algorithm. The indices i and j identify each path of the corresponding flow where both i and j of flow f_0 and f_z and of course of each other path of the flows in s represent a value between 0 and $k-1$. We exclusively define that indices i and j depend on a specific flow, in this case on flows f_0 and f_z, to show that each index is related to a different flow.

Lines 8 to 25 deep dive into how each potential path setting is generated. Function RANDNOTIN randomly selects the P^f where no path of the corresponding flow is yet stored in the current computed path setting s. Lines 12 to 18 return a path of this P^f having minimum similarity in terms of common links compared to all previously found paths, applying the Jaccard Coefficient [Jac12]. In particular, function SIM computes the Jaccard Coefficient, dividing the intersection of active and passive utilized links of all paths in s with all active and passive links of the potential path p^f_n, where path $p^f_n \in P^f$ of flow $f \notin s$. This result is stored in the disjoint rating $\mathtt{dr}^f_n$. The lower $\mathtt{dr}^f_n$ the less similarity is present. The best score is stored in

$\mathtt{dr}^{best}$ after all paths in P^f have been compared (Line 15). The associated path with the best score is stored in p^{best} (Line 16). KmDP then inserts p^{best} in the sequence s and also removes the path p^f_n from P^f in order to not consider this path more than once during the remaining k result setting generations, as shown in Line 20. KmDP continues and evaluates all paths of further flows not stored in the current path setting s until a path of each flow is stored in s. All complete path settings are stored in the set S, as shown in Line 22. The algorithm continues until S stores k path settings.

When KmDP finishes and $k = 0$, each P^f where $f \in \mathcal{F}$ is an empty set. Function BESTSETTING returns a single path setting s having the following criteria: Firstly, KmDP drops all settings in S where any active link exceeds its transmission capacity. Secondly, all residual settings are compared with each other regarding their path qualities. In particular, KmDP sums up all path qualities $\mathtt{qp}^f$ of each p^f in s and selects the path setting of S having the best result, where $f \in \mathcal{F}$.

In general, KmDP compares for each path setting s the similarities to avoid that routes are computed having the same transmitting links. This increases the probability of computing paths of flows not over-utilizing the MANET. However, function RANDNOTIN randomly selects the set of paths which's paths are compared with already elected paths of other flows in s. No specific selection strategy is applied in order to increase the chance for further flows not to over-utilize the network.

Suppose KmDP computes path settings for two flows. Which paths to insert at first in the sequence s when generating a new path setting might be important. The function SIM always returns the same path out of the set of paths when given an empty sequence s. Selecting the set of paths of the second flow at first could eliminate the only path setting in which no active link utilizes the MANET. However, selecting the first flow with the function RANDNOTIN at the beginning might result in over-utilized path settings only. Specific characteristics describing each set individually regarding utilization and path qualities could improve the selection process and the outcome of KmDP.

Furthermore, KmDP exclusively searches for k paths of each flow $f \in \mathcal{F}$ at the beginning. Thereafter, KmDP generates path settings having the least similarity and lastly returns the setting with ideally comprising no over-utilization and the best link qualities. It must be noted that KmDP provides no guarantee of finding a robust path setting in case such a setting exists. The probability of finding such a setting depends on the number of paths (k) for each flow, which in turn strongly influences the runtime.

```
1  begin EnvolveSbDS(F)
2      s = ⟨⟩
3      while |s| < |F| do
4          q^best = ∞
5          foreach f ∈ {f ∈ F | p^f ∉ s} do
6              qp^f, p^f = scoreAndPathOf(f)
7              if q^f ≤ q^best then
8                  q^best = qp^f
9                  p^best = p^f
10             end
11         end
12         deployPath(p^best)
13         s = p^best
14     end
15     return s
16 end
```

Algorithm 12: Path setting computation using SbDS.

7.4.2 Sequence-based Deployment Strategy

The SbDS pathfinding strategy focuses on the local best path quality when heading for a potential path setting for all requested flows. For good path quality results, SbDS also applies Dijstkra [Dij59] with our robust link quality q_l as the metric, where $l \in \mathcal{L}$, to successively generate a single path for each flow. Paths are compared regarding their quality and either dropped or deployed. Flows whose computed paths are already deployed are no longer considered during computations to reduce the execution times of SbDS.

Algorithm 12 introduces each step of SbDS in detail. We first define the sequence s representing the resulting list (path setting) of paths if a capacity-conform path setting has been found by SbDS. This sequence is filled at runtime, which is why SbDS stops if the size of the result s equals the size of all flows, as shown in Line 3.

From Lines 5 to 11, the algorithm successively determines the local best path p^f for each flow f. In particular, the SbDS computes the path p^f and its quality qp^f for each flow using function scoreAndPathOf(f). This function applies Dijkstra to compute those paths and uses q_l as the metric. A computed path is stored in p^f and the associated path quality is stored in qp^f. From Lines 7 to 10, the current path quality is compared with the best

path quality of the current sequence and stored in the associated parameters (p^{best} and q^{best}). Each determined path p^f of each flow $f \in \mathcal{F}$ where a path p^f of the flow f is not in the path setting s (Line 5) is compared with the current best path p^{best} regarding their robust path qualities. The current p^f is set to the best path p^{best} if $\mathsf{qp}^f \leq q^{best}$. Function scoreAndPathOf returns $\mathsf{qp}^f = \infty$ if p^f over-utilizes the MANET. This ensures that p^f does not take over the current p^{best} if qp^f has better robust link quality compared to the quality of p^{best} but over-utilizes the MANET.

When p^{best} has been identified after comparing the paths of all flows, the path with the best robust path quality (q^{best}) is deployed on the topology representation, which affects the further pathfinding process of SbDS since links increase their utilization and might not provide sufficient transmission capacities when computing paths of additional flows. Next, p^{best} is stored in the s. This sequence is returned when a path for each requested flow has been found. Storing the currently found path (p^{best}) in s also has the effect that any path of this flow f is not considered for further Greedy pathfinding operations, as the loop, starting at Line 5 only considers flows not in s. After each deployed path, SbDS verifies if a path for each requested flow has been found and stops the algorithm if this condition is true.

To sum up, SbDS firstly computes a path for each flow and compares the path qualities. Next, the algorithm selects the path with best quality and deploys this path on the MANET representation. After one path has been deployed, SbDS determines the best path for all flows based on the current network utilization situation, excluding the already deployed paths. Dijkstra path computation detours paths of not yet deployed flows around stressed links if their residual transmission rates do not provide sufficient capacities for the current path. However, this approach does not identify conflicting links if no capacity-conform path setting could be found. More precisely, each path of each flow is computed individually not considering paths of following flows that have not been deployed yet. Furthermore, the probability increases that more conflicting links arise that exceed the connection capabilities when increasing the initially configured over-utilization of the Utilization Setup component.

7.4.3 Rate-based Distributed Robust Paths

RbDRP also focuses on finding a capacity-conform path setting in minimum runtime while avoiding path deployments comprising stressed network segments. In contrast to SbDS and KmDP, RbDRP introduces, be-

```
1  begin evolveRbDRP(F)
2      dijkstraDeployAll(F)
3      foreach l ∈ L do
4          foreach f ∈ F do
5              if l ∈ ⋃_{i=0}^{|p^f|−1} U(n_i, n_{i+1}) then
6                  cs_l^f = capacityShare(l, d^f)
7              end
8          end
9          cu_l = 1
10     end
11     undeployAllFlows(F)
12     foreach f ∈ F do
13         p^f = dijkstraDeploy(f)
14         foreach v_j ∈ p^f do
15             foreach (m, n) ∈ U(v_j, v_{j+1}) do
16                 cu_(m,n) = cu_(m,n) + 1
17             end
18         end
19     end
20 end
```

Algorithm 13: Path setting computation using RbDRP.

sides the robust link quality, an additional link weight that adapts the value according to the current amount of active and passive utilization. Dijkstra [Dij59] combines the additional weight with our defined robust link quality to compute a path for each flow. Because of that, RbDRP has the effect of utilizing a link only with a single flow resulting in less stressed MANET segments while also aiming for capacity-conform and robust paths.

In doing so, RbDRP identifies links, which will most likely be chosen by at least more than one flow. These links are penalized. This approach extends the research published by Akin et al. [AK19] with wireless capabilities and with a newly defined link weight computation to first of all provide similar functionality in MANETs and second of all to also consider the required path characteristics. In doing so, RbDRP at first identifies links more likely to be chosen for multiple flows. Secondly, path distribution considers these links when searching for robust paths for each flow.

Algorithm 13 deep-dives into the functionality of RbDRP. The initial

step of the algorithm is to compute one path of each flow with Function DIJKSTRADEPLOYALL using the robust link quality. Once all the paths have been computed, this function proceeds to deploy all the discovered paths. The utilization caused by deploying these paths spreads beyond the active connections, formally described in Section 7.3.3. As a consequence, this decreases the transmission capacities of surrounding links not actively utilized, based on the demand d^f of flow $f \in \mathcal{F}$. RbDRP computes a path of each flow in isolation, meaning not considering the occurring utilization of each other. Next, these isolated paths are deployed one after another increasing active and passive utilization of links in the MANET topology representation. This deployment strategy highlights stressed network segments as path computations are based on topology representations, having initially no active transmission deployed.

After this first step, RbDRP determines adequate link weights, aiming for robust paths whose active links are in the best case utilized by only a single path of a flow. In doing so, the capacity share of each flow with respect to each link $l \in \mathcal{L}$ is evaluated by RbDRP. Lines 3 to 10 describe this procedure in detail. Thereby, RbDRP precisely determines the utilization volume of each connection, which is based on the data rate of each flow, in case the link l is actively or passively utilized. Therefore, RbDRP unites all actively and passively utilized links of the currently investigated path and verifies if the current connection is part of this set, as defined in Line 5. Each node n in Line 5 is of the current path p^f. In the next line, Function CAPACITYSHARE determines the exact utilization of each connection of the currently utilized link (passively and actively), which depends on the data rate demand of the flow, according to the previously defined path of the current flow. The result is stored in the capacity share weight $\mathtt{cs}_l^f$. In particular, the capacity share $\mathtt{cs}_l^f$ defines the portion of the demanded data rate of each flow in relation to the utilization of each link, defined as: $\mathtt{cs}_l^f = \frac{d^f}{u_l}$, where $f \in \mathcal{F}$ and $l \in \mathcal{L}$. The parameter $\mathtt{cs}_l^f$ is set to 1 if link l is neither actively nor passively utilized by any path.

Having determined the capacity share of each flow with respect to each link, RbDRP undeploys the computed path of each flow in the MANET topology representation, as shown in Line 11. This resets the utilization of each link back to 0.

Next, the actual path computation and deployment take place considering the previously gained utilization knowledge, as shown with Lines 12 to 19. Therefore, Function DIJKSTRADEPLOY returns the path p^f of each flow

$f \in \mathcal{F}$, applying the individual link weight $\mathtt{cs}_l^f$ for each connection, where $l \in \mathcal{L}$. Including the new weight $\mathtt{cs}_l^f$ forces Dijkstra to consider the paths of other flows during computations. In particular, Dijkstra investigates the costs to traverse to each next sink node n of a link $(m, n) \in \mathcal{L}$ when being at source m. At his point, RbDRP applies Equation 7.6 to compute the customized link weight considering the following aspects:

$$\mathtt{lw}_{(m,n)}^f = \frac{\mathtt{cu}_{(m,n)}}{(q_{(m,n)} + \mathtt{cs}_{(m,n)}^f) \cdot (c_{(m,n)} - u_{(m,n)})} \tag{7.6}$$

Path computation must avoid links potentially being chosen by routes of multiple flows. Therefore, the cost must decrease and increase depending on the demand of each flow. Furthermore, RbDRP must prefer robust connections over fragile ones. In general, Dijkstra prefers connections that have minimum costs. The link usage parameter $\mathtt{cu}_{(m,n)}$ indicates the number of flows a connection carries which is set to 1 at the start (Line 9). Lines 14 to 18 increment $\mathtt{cu}_{(m,n)}$ if link (m, n) is passively or actively utilized according to path p^f, where $f \in \mathcal{F}$, which increases $\mathtt{lw}_{(m,n)}^f$. The costs of these links increase if $\mathtt{cu}_{(m,n)}$ increases which further influences path computations of additional flows. The parameter $\mathtt{cs}_{(m,n)}^f$, which is returned by function CAPACITYSHARE, influences the decision of Dijkstra which connection to select at node m investigating all directly connected neighbors. Thereby, lower values in the parameter $\mathtt{cs}_{(m,n)}^f$ with respect to a flow f also reduce the custom cost $\mathtt{lw}_{(m,n)}^f$, where $f \in \mathcal{F}$ and $(m, n) \in \mathcal{L}$. To also consider the long-living connections, the robust link quality parameter $q_{(m,n)}$ pushes to select n as the next hop if the link quality seems long-living.

RbDRP computes paths using the $\mathtt{lw}_{(m,n)}^f$ as cost to traverse from m to n, which potentially changes after the path of flow f has been deployed. In particular $\mathtt{lw}_{(m,n)}^f$ changes according to the $\mathtt{cs}_{(m,n)}^f$ and also depends on $\mathtt{cu}_{(m,n)}$, which increases if the link (m, n) has been chosen by an already deployed flow.

To sum up, RbDRP firstly identifies stressed MANET segments in computing and deploying all paths in a naive way, not considering mutual transmission interference of all flows during path computation. These resulting stressed segments are considered and penalized individually per flow during actual path construction to eliminate previously found stressed network areas. On top, RbDRP also considers robustness and includes the robust link weight in the link cost of Dijkstra when computing a path for each flow.

7.4.4 IBM ILOG Cplex Optimizer

CPLEX is a solver receiving an objective function and associated constraints as input, applying mathematical solving strategies. An optimization function and corresponding constraints represent an abstract model of the optimization problem comprising decision variables. The best values of the decision variables must be found considering the defined constraints. This so-called optimization model is the basis for implemented solving strategies such as Branch & Bound and Branch & Cut [Mit09].

CPLEX solves linear and integer optimization problems and can be multi-processed, decreasing the runtimes [SFK03]. The CPLEX solver has continuously been improved regarding underlining hardware support and solving strategies, producing faster results over the past years. Optimization problems that utilized a computer for a whole day in 1998 take less then $5\ sec$ almost 25 years later [Nic22]. Leveraging the computation power makes it possible to investigate the entire search space in reasonable time to compute the optimal solution for a given problem, especially if runtime is not a matter of fact.

We apply CPLEX for route computations heading for robust and capacity-conform paths, given any MANET topology representation, the flows, and their data rate demands. Thereby, we firstly adopt CPLEX as a baseline solution for each evaluation scenario and secondly also limit the runtime of CPLEX to design the solver comparable to all other pathfinding techniques. The baseline solution can prove if a potential path setting exists in the current topology representation including link and node characteristics, which comprises only capacity-conform paths. The second configuration restricts runtimes of each optimization challenge to the average runtime of SbDS when comparing CPLEX regarding found solutions and their quality with other approaches. This might reduce the robust link qualities, as the number of found solutions also decreases. The objective of this RQ is to compute robust and capacity-conform paths for a set of flows with their currently defined paths in minimum runtime. These paths over-utilize the MANET to a certain degree. According to that goal, we formalize a model comprising an optimization function and associated constraints.

In doing so, Optimization Function 7.7 minimizes the link qualities q_l of all links $\mathcal{L}$ which are chosen for data delivery according to each flow. As mentioned, a link becomes more robust if the link quality q_l is minimal.

$$\min_{(x)} \sum_{f \in \mathcal{F}} \sum_{l \in \mathcal{L}} x_l^f \, q_l \tag{7.7}$$

x_l^f determines those links $l \in \mathcal{L}$ whose qualities q_l are minimal regarding all flows $f \in \mathcal{F}$. This binary decision variable is set to 1 if link l forwards data of the flow f, otherwise 0. Constraint 7.8 defines this formally.

$$x_l^f \in \{0,1\}, \forall f \in \mathcal{F}, \forall l \in \mathcal{L} \tag{7.8}$$

Expression 7.9 describes the binary variable y_l, which is set to 0 if any flow f traverses the link l. If not, y_l is set to 1. Also, y_l is initially set to 1.

$$y_l \in \{0,1\}, \forall l \in \mathcal{L} \tag{7.9}$$

Next, we must ensure that only active links, actually defined for data transmissions must not be over-utilized. Therefore, constraint 7.10 verifies if link l is elected for transmission of any flow f and sets the binary y_l to 0 if it is planned that link l actively transmits data. This is important since a passively utilized link $l' \in \mathcal{U}(l)$ also increases its utilization based on the data rate d^f of flow f if $l \in \mathcal{L}$ is elected to transmit data, as described in Section 7.3.3. In addition, passive utilization must be considered if l' is also elected to transmit data.

$$0 < (\sum_{f \in \mathcal{F}} x_l^f) \Rightarrow y_l = 0, \forall l \in \mathcal{L} \tag{7.10}$$

Constraint 7.11 defines not to over-utilize any transmitting $l \in \mathcal{L}$ with the data rate demand d^f, if x_l^f of any flow is set to 1. This also considers passive connection utilization.

$$\sum_{f \in \mathcal{F}} \sum_{l' \in \mathcal{U}(l)} d^f (x_{l'}^f - y_l) \leq c_l, \forall l \in \mathcal{L} \tag{7.11}$$

If the decision variable $y_l = 1$, the capacity c_l will never be reached, so that CPLEX ensures that only active connections need to comply with the transmission speeds. Constraint 7.12 defines the range of q_l, meaning that the robust link quality is normalized between 0 and 1, where values towards 0 represent more robust connections.

$$0 \leq q_l \leq 1, \forall l \in \mathcal{L} \ , q \in R \tag{7.12}$$

The set $N^+(m) := \{n \in \mathcal{N} | (n,m) \in \mathcal{L}\}$ returns all incoming neighbors of node m. Speaking of a link (n,m), this set returns all source nodes of links where m is the sink of these links. The set $N^-(m) := \{n \in \mathcal{N} | (m,n) \in \mathcal{L}\}$

returns all outgoing nodes of m where a link is stored in $\mathcal{G}$.

$$\sum_{m \in N^+(m)} x^f_{(n,m)} d^f - \sum_{n \in N^-(m)} x^f_{(m,n)} d^f = \begin{cases} 0, m \in \mathcal{L} \setminus \{o^f, t^f\} \\ -d^f, m = o^f \\ d^f, m = t^f \end{cases}$$
$$, \forall f \in \mathcal{F} \quad (7.13)$$

Constraint 7.13 verifies that the data rate d^f of each flow $f \in \mathcal{F}$ that reaches node m must be equal to the amount of data rate that leaves node m. This excludes the origin o^f and the target t^f nodes. If m is the origin node of a link, the amount of data that leaves m must be equal to the demand of the flow d^f. In case m is the target, the amount of data that reaches node m must be equal to d^f. The constraint also ensures that CPLEX constructs loopless paths from o^f to t^f.

$$\sum_{n \in N^+(m)} x^f_{(n,m)} + \sum_{n \in N^-(m)} x^f_{(m,n)} = \begin{cases} 2, m \in \mathcal{L} \setminus \{o^f, t^f\} \\ 1, m = o^f \\ 1, m = t^f \end{cases}$$
$$, \forall f \in \mathcal{F} \quad (7.14)$$

Constraint 7.14 ensures that only a single path per flow is constructed from o^f to t^f regarding each $f \in \mathcal{F}$. In particular, if $m = t^f$, only 1 link (n, m) exists per flow where $x^f_l = 1$. Also, only a single link $(m, n) \in \mathcal{L}$ exists, where $m = o^f$ and where $x^f_{(m,n)} = 1$.

CPLEX uses this model to solve the Optimization Function 7.7. Therefore, MfDM provides graph $\mathcal{G}$, comprising all nodes, connections, and their characteristics as the basis of the search space, used by the solver to approximate towards the objective.

To sum up, CPLEX operates as a problem solver given a formalized mathematical model comprising an optimization function and associated constraints. Due to the improved performance in recent years and the ability for multiprocessing, CPLEX is comparable to all other approaches since runtime can be restricted and configured. CPLEX can and will be used as a benchmark in all scenarios, as it returns a result if a path setting exists if runtimes are not restricted. Furthermore, we restrict the runtime of CPLEX to the average runtime of SbDS and compare the path setting regarding robustness with all other approaches.

7.4.5 Genetic Programming

Unlike all other approaches mentioned from Sections 7.4.1 to 7.4.4, GP is a meta-heuristic inspired by nature. GP is characterized by its ability to learn from the impact of previous solutions of the environment and based on that independently generates new solutions. In addition, GP operates round-based, applying modifications and manipulations during these periods. These rounds are called generations. If we apply these capabilities to our objective, GP generates new potential path settings during generations based on previous ones, with the aim of achieving less network utilization and better path robustness.

Additionally, this learning-enriched approach allows for a compromise between diversity and minimum runtime. Diversity is provided since path construction is based on randomness and the computed quality of previous results. GP uses techniques to select path settings which are investigated next and also modifies and manipulates them to increase diversity. Also, GP provides termination conditions to restrict runtimes of approaches, which also makes this technique comparable to all other approaches. The following sections introduce the modifications to obtain path settings each comprising a path of each requested flow which represent possible solutions.

Initial Population

Initial populations are often created randomly, meaning that the composition of the first possible results are generated without any specific pattern. Each of these individual compositions represent a possible solution. This set of individuals is generated at the beginning of GP. At runtime, entries of the initial population are modified in order to produce additional potential solutions having better qualities.

With respect to our objective, the initial population $I = \langle i_0, \ldots, i_v \rangle$ represents a set of possible solutions, comprising of a path of each flow. For our approach, we define a sequence of paths $P^f = \langle p_0^f, \ldots, p_v^f \rangle$ for each flow, where $f \in \mathcal{F}$. The indices of a path, in this representation 0 and v, always point to a specific path of their flow f. The index v represents $|I|-1$ which defines that the sequence of individuals and the sequence of paths of each flow have the same length. Also, the path p_v^f and the individual i_v are located at the same position of their corresponding sequences. This also defines that each set of paths P^f of each flow, where $f \in \mathcal{F}$, has equal length, which is $|I|$, whereas the length of each individual path in this sequence differs. Furthermore, each path of P^f is unique meaning that

each path differs by at least a node compared to all other paths in the list. This goes for each P^f, where $f \in \mathcal{F}$.

A single individual $i_z \in I$, where the index $0 \leq z < |I| - 1$ is defined as follows: $i_z = \langle p_z^{f_0}, \ldots, p_z^{f_j} \rangle$, where $j = |\mathcal{F}| - 1$. Each $i_z \in I$ represents a sequence of ordered paths, which contains exactly the path with index z of the corresponding flow f. This path p^f of the flow f is of the set of paths P^f. Also, exactly one path of each flow is stored in each individual which defines that each $i \in I$ is of length $|\mathcal{F}|$.

We propose two techniques to generate the initial population, which can be combined. MfDM provides random path and k-shortest path [Yen71] generation to fill the initial population. We, therefore, combine both techniques and propose the instruction factor adjusting the share of each method of generated paths for the set of individuals. If this factor is set to 0.5, the set of individuals I will have almost the same number of paths of each approach (random and k-shortest path generation) whereas 1 causes the set I to be filled with k-shortest paths only.

For random path generation, GP starts at the origin o^f of the current flow $f \in \mathcal{F}$ and selects a random neighboring node n where $(o^f, n) \in \mathcal{L}$. Thereafter, the node n is marked as visited to exclude this node from the further random path generation for this particular path. Next, the algorithm repeats this step and randomly selects a neighbor m of n, where $(n, m) \in \mathcal{L}$ and m is not in the set of visited nodes. The algorithm repeats this step until either it reaches the destination or no neighbor m of the current node n is selectable. If the latter is true, the random pathfinding algorithm starts from scratch until a random path has been generated. If the first situation is present, the algorithm returns the generated path.

K-shortest path [Yen71] applies Dijkstra [Dij59] to generate k unique paths. In doing so k-shortest path removes one link (n, m) of the previously generated path from the graph starting at the beginning and thereafter recomputes Dijkstra, starting from node n to generate a new path with o^f as origin and t^f as target, where $f \in \mathcal{F}$ and $o, t \in \mathcal{N}$. K-shortest path concatenates the partial path of the previously generated path of the same flow from o^f to n with the second part of the part of the new partial path from n to t^f. In general, the nodes of two paths differ at least at one position when comparing them position-wise and if the length of both paths is equal. Otherwise, these 2 paths differ anyway.

Either random path or k-shortest path generation is invoked when generating the initial population. This depends on the instruction factor, previously described. The initial population generation stops when the set of

individuals reaches the configured size which can be configured before the MfDM is started.

Test runs have proven that the instruction factor, which defines the share of both methods, must be adapted in case runtime matters in order to produce path settings satisfying the objective, which is to not over-utilize the MANET and to find long-living paths. However, the instruction factor also influences the robustness of paths. Test runs showed that the robust link quality score increases when increasing the number of random paths in the initial population. In addition, the more entries in the initial population the higher the probability of generating high-quality results. However, the number of entries in I increases the runtime of GP. We elaborate on these runs and their configurations in Section 7.5 to highlight which configuration fits best for the proposed scenarios.

Evaluate Fitness

The fitness function evaluates potential solutions and guides the direction in which GP optimizes. In doing so, the initial population as well as at runtime generated solutions are evaluated with this fitness function in order to distinguish between promising and inadequate solutions. Referring to the objective of this thesis, the fitness function rates robust and capacity-conform results over fragile and over-utilizing path settings.

In order to optimize towards this goal, our Fitness Function 7.15 computes a score, which rates the quality of each $i \in I$ in the current population investigating each path and their links.

$$\lambda(i) = \rho(i) + o(i) + \upsilon(i) \tag{7.15}$$

First of all, solutions must compute capacity-conform path settings followed by path and link quality score improvements. Hence, $\lambda(i)$ evaluates i based on the current utilization which is determined with $\upsilon(i)$. Equation 7.16 returns the share of the occurring passive and active utilization of i and all links in the topology representation that would be utilized by each flow. Thereby, $\upsilon(i)$ adds up the demands of each path of each flow and divides the amount by the utilization that would occur if each flow would traverse via each link of the given MANET, where $f \in \mathcal{F}$ and $i \in I$.

$$\upsilon(i) = \frac{\sum_{p^f \in i} \sum_{l \in p^f} |\mathcal{U}(l)| \cdot d^f}{\sum_{f \in \mathcal{F}} \sum_{l \in \mathcal{L}} |\mathcal{U}(l)| \cdot d^f} \tag{7.16}$$

The Function $\rho(i)$ returns the robust quality score of each individual, where $0 \leq j \leq |\mathcal{F}| - 1$ and $0 \leq k \leq |I| - 1$. Furthermore, the definition

$p_k^{f_j} \in P^{f_j}$ implies that the path at index k belongs to flow $f_j \in \mathcal{F}$. Because of that, $\rho(i)$ adds up all path qualities in the current individual, where each quality belongs to exactly one flow, and divides this quality by the number of links of all paths in the current individual. This averages the robust link quality of the current individual, described in Equation 7.17.

$$\rho(i) = \sum_{j=0}^{|i|-1} \frac{\mathfrak{qp}_k^{f_j}}{|p_k^{f_j}|} \tag{7.17}$$

The Function $o : i \rightarrow \{0, 1\}$ returns 1 if setting $i \in I$ creates an over-utilized MANET situation and 0 otherwise. This ensures that promising results computed by $\lambda(i)$ exclusively compute capacity-conform path settings. Over-utilizing settings are rated worse to exclude them from the set of individuals and from further computations.

Crossover

During each generation of GP, crossover operations are applied on specific $i \in I$. Therefore, two individuals are elected for this action where each chromosome randomly selects a gene where the crossover operation takes place. Individuals are elected by the configured tournament selection which prefers better solutions over worse ones. The fitness function ranks these individuals. Both chromosomes are cut in two pieces at the defined crossover point and the second part of the second chromosome is concatenated with the first part of the first chromosome. The other two parts behave in the same way. This technique is called one-point crossover.

We extended this technique as each path, included in an individual must follow certain constraints. The one-point crossover technique treats an individual as a sequence of genes having no further composition except for the current order. Newly constructed path-based chromosomes must represent exclusively complete paths from origin to target when crossover operations took place. Origin and target pairs must remain and the consecutive participants of each path must always have a link to each other. The definition of a complete path is introduced in Section 4.2.1. When crossover takes place on a selected individual, all paths of this individual will be considered and manipulated for this operation.

Figure 7.6 exemplifies the customized crossover technique considering the mentioned constraints. The associated MANET is shown in Figure 7.7.

Individuals i_a and i_b, both $\in I$ are selected for the operation. For simplicity, we only introduce the crossover operation for the two paths $p_a^{f_h}$ and

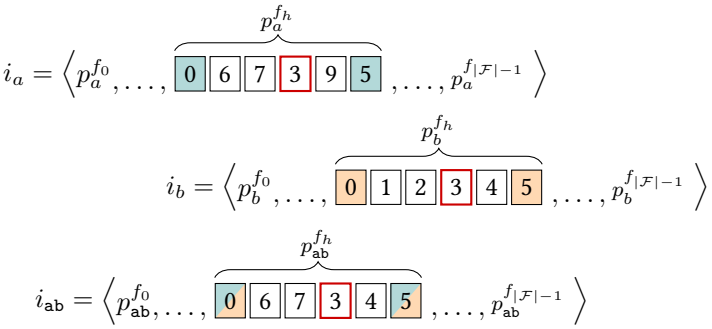


Figure 7.6: Exemplary custom crossover operation modifying paths $p_a^{f_h}$ and $p_b^{f_h}$ of individual i_a and i_b, both $\in I$ and both paths $\in P^{f_h}$. The MANET in Figure 7.7 shows the paths of the flow $f_{(0,5)}$. GP looks for common nodes in both paths for crossover operation. Crossover technique selects node 3 which results in the new path $p_{ab}^{f_h}$ consisting of the left part of $p_a^{f_h}$ and the right part of $p_b^{f_h}$.

$p_b^{f_h}$ of individuals i_a and i_b of flow $f_h \in \mathcal{F}$. All others are treated the same way. Both paths are in the same set of paths P^{f_h}, which defines that $p_a^{f_h}$ and $p_b^{f_h}$ are of flow f_h. In general, a path p_n^f always depends on the flow $f \in \mathcal{F}$ since p_n^f is stored in the set P^f.

According to the figure, GP selects a node that is part of both paths $p_a^{f_h}$ and $p_b^{f_h}$ excluding origin and target nodes. This process is formally defined as follows: One common node of set $C_{ab}^{f_h} = (p_a^{f_h} \cap p_b^{f_h}) \setminus \{o^{f_h}, t^{f_h}\}$ is randomly chosen as the crossover point. This maintains coincidence and diversity, which make up the main features of GP. Suppose node $c \in C_{ab}^{f_h}$ is chosen. Referring to Figure 7.6, the chosen node is 3. Now, a complete path comprising equal o^{f_h} at position 0 and t^{f_h} at the last position with respect to paths $p_a^{f_h}$ and $p_b^{f_h}$ is constructed. Therefore, GP contacts the left part of $p_a^{f_h}$, including the node c, with the right part, excluding c, which generates the new path $p_{ab}^{f_h}$. This introduced crossover operation is applied to each path in the individuals $i_a, i_b \in I$. However, no action is performed during crossover if no common node is found when investigating two paths of both individuals, which means that $C_{ab}^{f_h} = \emptyset$, where $f^h \in \mathcal{F}$.

Mutation

GP applies mutation to maintain coincidence during result generation. Selection of the individual is completely random and configurable before GP starts. This means that the probability that more individuals are chosen during each generation can be enforced. If mutation takes place at an individual, GP modifies exactly one node of every path in this individual ensuring that all constraints are met.

Naive mutation randomly selects a gene of the current chromosome and exchanges it with any gene of the given alphabet. As mentioned in the previous section, specific requirements of path manipulation exist also restricting the mutation process to guarantee the complete path constraint. In doing so, any node of a path of an individual is selected to be exchanged with a current participant, which must also connect the path entirely. Also, origin and target nodes must remain the same after mutation took place.

Figure 7.8 depicts the custom single node mutation process. There, mutation is performed on the individual i_b. For simplicity, we focus in this example exclusively on $p_b^{f_h}$. All further paths in the individual i_b are treated the same way. Paths are constructed based on the exemplary MANET in Figure 7.7.

To begin, GP selects a random node n_r of path $p_b^{f_h}$ which is not the origin or target node. The path $p_b^{f_h}$ is stored in the set of all paths P^{f_h} of flow f_h, implying that this path and in general each path in P^{f_h} depends on the flow f_h.

According to the figure, all neighboring preceding n_{r-1} and successive n_{r+1} nodes of the randomly selected node n_r are chosen, ensuring that this chosen selection is not part of $p_b^{f_h}$, in order to meet the complete path

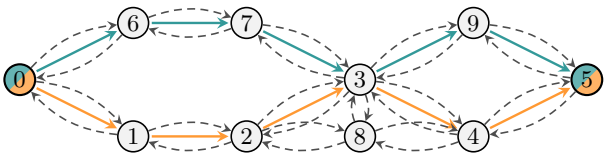


Figure 7.7: Exemplary MANET graph to introduce custom Crossover and Mutation techniques. Orange and teal arcs illustrate possible paths $p_a^{f_h}$ and $p_b^{f_h}$ both in P^{f_h} from nodes 0 to 5. Directed dashed arcs represent edges (links) between nodes.

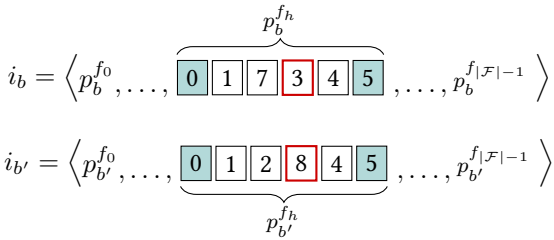


Figure 7.8: Exemplary mutation process with MANET topology, as shown in Figure 7.7. Node 3 is selected and exchanged with 8 since this node fulfills the complete path criteria as 2 and 4 are also connected to node 8.

requirement, where $1 \leq r \leq |p_b^{f_h}| - 2$. The definition of r also ensures that neither the origin nor the target node of a path is chosen. All potential nodes are stored in the set of mutable nodes $M_b^{f_h}$, as defined with Equation 7.18.

$$M_b^{f_h} = (N^-(n_{r-1}) \cap N^+(n_{r+1})) \setminus p_b^{f_h} \tag{7.18}$$

Each element in $M_b^{f_h}$ can be used to exchange this element with n_r at the same position in $p_b^{f_h}$ as it connects two participants of the current path, where $f^h \in \mathcal{F}$. The sets N^- and N^+ are defined in Sections 7.4.4.

Referring to Figure 7.8, node 3 is selected for mutation and the mutation process generates the set $M_b^{f_h} = \{8\}$, as shown in Figure 7.7. Node 8 is the only MANET participant that is connected to 3. In addition, node 8 is not part of the current path.

This mutation process is applied to all paths of each flow $f \in \mathcal{F}$ in the individual i_b which creates $i_{b'}$. Also, if $|p_b^{f_h}| < 3$ or if $M_b^{f_h}$ is empty, this custom mutation process generates a new random path for this flow.

Termination Condition

We propose a customized termination condition that best fits our purpose. Test runs showed that the size and the instruction factor have a major influence on whether and how GP converges toward a solution. Several runs and scenarios generated populations whose fitness score remained the same through multiple generations. In this situation, no termination condition exists to stop GP until the configured number of generations is reached. We defined and applied a customized termination condition to overcome

the mentioned problems. GP terminates the execution based on the following criteria: (i) Predefined runtime or number of generations exceeds its configuration. (ii) Any individual meets the defined fitness value. (iii) The same best fitness value is computed successively after multiple generations.

7.5 RESULTS

This section compares the proposed pathfinding techniques by focusing on found capacity-conform path settings, link quality scores, and their runtimes. We also highlight how configurations of individual approaches influence the results regarding link quality and found solutions.

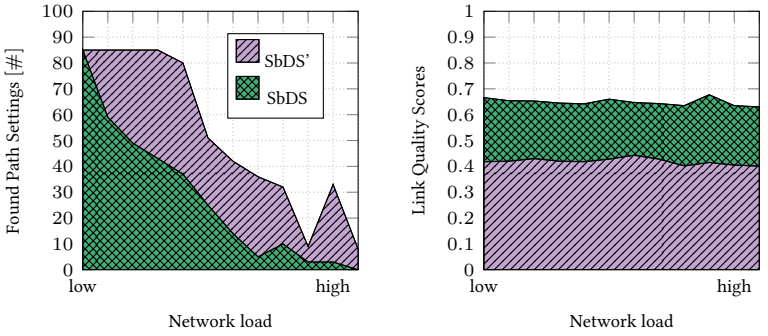
Firstly, we propose results regarding found path settings and their execution times if no time restriction is present. Secondly, we restrict runtimes of all approaches and compare found solutions and their link quality scores. Further, we investigate how link quality scores correlate with the measured time that these found path settings stay connected.

7.5.1 Scenario Configurations

We compare all pathfinding techniques of this chapter focusing on found capacity-conform path settings, runtimes, and link qualities. Each approach searches for paths having equal conditions in terms of network topologies, requested flows, as well as mobility. In particular, each technique is applied to the same run having equal random seeds. Because of that, configurations regarding network size, topology constellations, number of flows, the associated computed data rates, and mobility patterns of nodes are equal for each pathfinding approach with respect to each run.

Each run comprises 100 nodes generating topologies similar to Figure 7.5. To be more concrete, the average node degree is roughly 5.5 which corresponds to a graph density of 0.0726. Physical distances between neighbors are set between $[35.0\ m, 100.0\ m]$. This configuration offers the best conditions to highlight the advantages and disadvantages of all approaches as plenty of alternative paths are available to redistribute an over-utilized network situation.

The radio model resembles the $2.4\ GHz$ band supplying approximately $12\ Mbps$. The exact data rate per link depends on the distance between connections, as described in detail in Section 7.3.2. Gauss-Markov mobility model [AWS06] is used to generate movement patterns of participants mimicking pedestrians. The speed of nodes varies between $[2\ km/h, 7\ km/h]$.



(a) Number of capacity conform path settings.

(b) Link quality scores of found capacity conform path settings.

Figure 7.9: Found capacity-conform path settings and link quality scores during increasing initial over-utilization of SbDS and SbDS'.

Initial results of the SbDS have indicated that the use of the link quality score q_l results in a lower number of found capacity-conform paths compared to a link quality score $q'_l = 1$, where $l \in \mathcal{L}$. The score $q'_l = 1$ essentially calculates the path for each flow with the least number of transmitting links not considering robust connection characteristics. The specific results of these findings are depicted in Figure 7.9, which compares the found path settings and associated link quality scores of SbDS and SbDS' configured with $q'_l = 1$. It clearly shows that, on the one hand, SbDS' more often computes capacity conform path settings but, on the other hand, selects links having worse link quality compared to SbDS. The results of robust link quality scores in this figure are inverted for better readability, which means higher results represent better link qualities.

Both SbDS and SbDS' are discussed in the following sections since the different link quality scores of both approaches can be used to determine whether better link qualities increase the lifetime of paths.

7.5.2 Capacity-conform Path Settings

This section measures the execution times of all approaches and also focuses on the number of found corresponding capacity conform path settings in order to highlight how techniques perform if the execution time is not an issue.

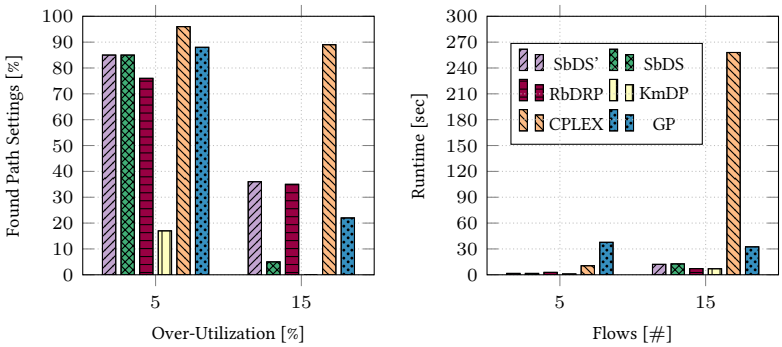


Figure 7.10: Number of found capacity-conform path settings and runtimes of all approaches.

On the left side, Figure 7.10 shows the number of found capacity-conform path settings with 5% initial over-utilization combined with 5 predefined flows and their origin-target pairs. Almost all approaches reach similar results, finding around 85% capacity-conform path settings. GP and CPLEX stand out somewhat. RbDRP is slightly below SbDS and SbDS', while CPLEX almost always finds a solution for each constructed scenario. KmDP finds by far the fewest capacity-conform path settings. Only 17% of all overloaded network constellations can be resolved.

KmDP focuses on combining maximum disjoint paths electing at first k paths with minimum similarity of each flow. Next, path settings, containing a path of each flow are selected also focusing on minimum common active and passive links, as defined in Section 7.4.1. By using as many different and distributed paths as possible, the transmission load is assumed to be distributed across the network. We believed that this would result in individual active connections bearing the burden of multiple transmissions less frequently. However, KmDP only focuses on using the maximum disjoint path ignoring the resulting transmission utilization when selecting appropriate path settings. Only after that, a path setting is chosen from a set of possible settings, which is first of all capacity-conform and second of all appears to be the most robust. We found out that almost all path settings of this set are over-utilized.

The left barplot of Figure 7.10 also shows all found path settings with 15 flows combined with 15% initial over-utilization. This configuration reduces valid solutions in the search space, challenging each path setting

approach even more to find possible solutions. KmDP does not find any capacity-conform path setting. SbDS, SbDS', GP, and RbDRP find notably fewer path settings compared to the previous scenario (5 flows and 5% initial over-utilization). SbDS and SbDS' do not change their already deployed paths at runtime even if the additional paths that also need to be deployed will over-utilize the current scenario. We often observed, that GP is stuck in a local optimum, in which link quality is high while the capacity-conform network criteria is not met. As expected, CPLEX computes the most capacity-conform paths compared to all other optimization techniques. The runtime of CPLEX is restricted to 8 minutes. This duration is adequate for identifying path configurations that meet the capacity constraints.

The right barplot of Figure 7.10 depicts runtimes with 5 flows combined with 5% initial over-utilization and with 15 flows and 15% over-utilization. Speaking of both 5 and 15 flows, SbDS, SbDS', RbDRP, KmDP as well as GP do not exhibit notable deviations regarding the measured execution times. One notices a slight increase as the increasing number of paths results in additional combinations and computation times. GP employs the termination condition described in Section 7.4.5, which can lead to longer and uncontrolled runtimes. CPLEX almost exhausts the pre-configured 8 minute runtime, when calculating paths for 15 flows combined with 15% over-utilization. The fewer valid solutions are present in the search space, the longer it takes for CPLEX to return one of those path settings.

As a reminder, computation times of paths must be kept minimal, as our focus is on MANETs where mobility continuously changes the topology. In order to account for this, we applied the averaged runtimes of SbDS as a baseline and restricted the computation times of CPLEX and GP accordingly. The following section discusses found path settings of all approaches with restricted execution times.

7.5.3 Valid Path Settings under Runtime Restrictions

Figure 7.11 depicts the number of valid path settings that are found by all approaches having configured averaged runtimes of SbDS. We do not consider KmDP as almost no valid path setting is found since findings of Figure 7.10 prove that starting with 5% initial over-utilization combined with 5 flows, this technique already finds the worst valid path settings.

The gray bars depict found capacity conform path settings of CPLEX running approximately 24 hours. The thick red lines represent the averaged

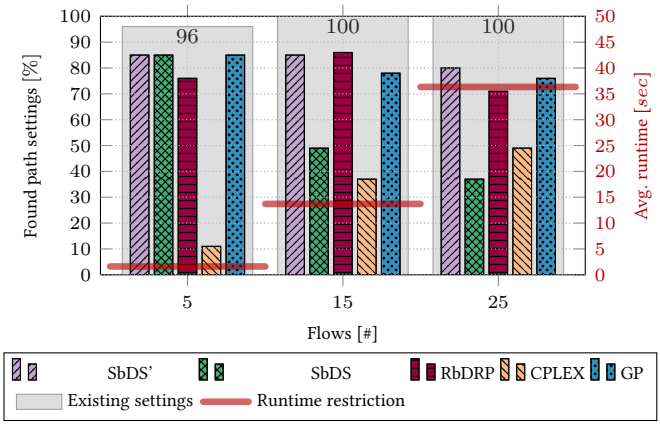


Figure 7.11: Comparing found capacity-conform path settings with 5% initial over-utilization.

runtimes of SbDS configured as the execution time threshold of all other approaches.

Our findings reveal that CPLEX requires more time to attain better results compared to all other approaches except for KmDP, as this pathfinding technique is not included in all evaluations. However, CPLEX reaches better results with increasing number of flows proportionally to other approaches as the averaged runtimes of SbDS also increase with the increasing number of initial flows. GP discovers a similar number of path configurations as SbDS'. The initial population is mostly generated using k-shortest paths [Yen71], to speed up the process of GP, finding capacity-conform solutions. Improving potential path settings from a group of randomly generated paths takes GP longer to produce valid path settings. RbDRP generates promising results considering valid path settings and the associated execution times. In general, the pathfinding approaches SbDS', RbDRP, and GP exhibit acceptable results according to Figure 7.11.

Figure 7.12 depicts found path settings with 10% initial over-utilization. It is clearly visible that the number of found path settings significantly decreases with increasing utilization if one compares Figure 7.11 with Figure 7.12 with respect to the number of initial configured flows. As assumed, the scenario configuration comprising 10% initial over-utilization and 5 flows shows the least loss compared to all other scenarios. Restricted execution times (horizontal red lines) show almost the same results compared

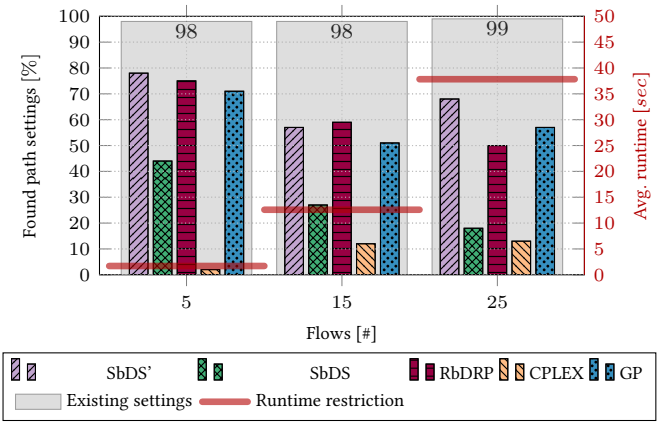


Figure 7.12: Comparing found capacity-conform path settings with 10% initial over-utilization.

to Figure 7.11 since increasing initial over-utilization of flows does not affect path computation times of SbDS.

However, there exist fewer valid results in search spaces of each scenario with 10% initial over-utilization compared to scenarios configured with 5% initial over-utilization. Because of that, CPLEX returns almost no valid path setting. This technique requires even more time to compute potential path settings when initial over-utilization increases. Only minor improvements can be observed with increasing number of flows. GP must tackle similar challenges compared to CPLEX when exploring the search space since newly generated potential results are built on top on previous ones. Compared to CPLEX, this technique copes better but also decreases the number of valid path settings with an increasing number of flows.

The results of SbDS and RbDRP are as expected according to the scenario configuration with 10% over-utilization, meaning the number of valid path settings decreases in general and also in relation to the number of flows. Surprisingly, SbDS finds more valid paths for 25 flows than for 15. All in all, the number of valid results decreases with respect to each pathfinding approach while the number of actual existing path settings (measured with CPLEX having no time constraint configured) does not decrease.

Figure 7.13 depicts the results of all pathfinding techniques with 15% initial over-utilization. All approaches find fewer valid path settings compared to the previous scenario configuration, except for CPLEX. SbDS strugg-

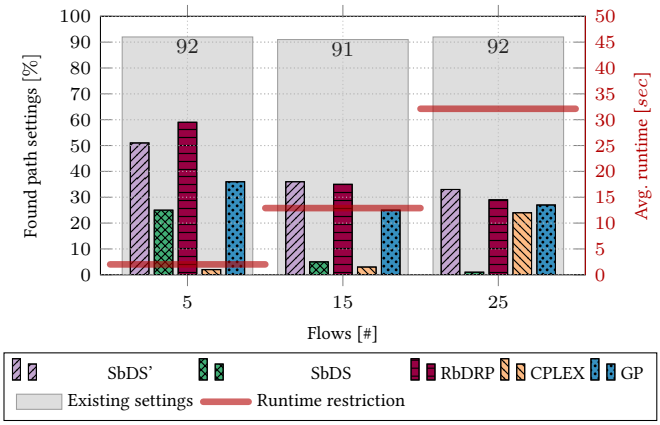


Figure 7.13: Comparing found capacity-conform path settings with 15% initial over-utilization.

les to find up to 5% valid path settings with 25 flows and 15% initial over-utilization. RbDRP, SbDS', as well as GP outperform CPLEX and SbDS. However, each approach returns less then 40% of all available valid results when increasing the number of flows to 10 and 15.

Results deteriorate even further when the utilization increases to 25% initial over-utilization. With 25% initial over-utilization, SbDS computes 10% capacity-conform path settings with 5 flows, returning 0 with 15 and 25 flows. CPLEX computes even fewer valid results and finds only two valid path settings for 5 to 25 flows. GP returns similar results. The number of found path settings decreases to 3 with 15 and 25 flows. It is worth mentioning that in 78% of the generated scenarios with 25 paths and an initial over-utilization of 25%, at least one solution, having no link over-utilized was found by CPLEX, having no runtime constraint configured. This result confirms that the proposed pathfinding techniques have their limits in finding valid results that do not over-utilize the MANET when the initial over-utilization is increased to 15% or more, even though there is at least one capacity-conform path setting per run in the search space.

To sum up, RbDRP, SbDS, and SbDS' take from approximately 1.5 sec to 35 sec execution time to compute 5 to 25 flows. Runtimes of these three pathfinding techniques are expected to remain similar with increasing initial over-utilization as approaches always go through the same deterministic process when searching for valid path settings, independent of the initial

over-utilization. GP surprisingly generates as many valid path settings as RbDRP and SbDS' with 15 flows and more when restricting the runtime to SbDS.

This technique mostly applies k-shortest path [Yen71] to generate the initial population, speeding up the execution significantly. CPLEX needs more time to generate meaningful results which we have proven in Section 7.5.2. So far, RbDRP and SbDS' should be chosen over all other approaches when paths of around 5 flows must be found and runtime is an issue. With 15 and more flows, one can choose GP as well. The robust link quality and the lifetimes of computed paths also play an important role when deciding which pathfinding technique to use. These results are discussed in the next section.

7.5.4 Correlating Link Qualities with Lifetimes

In this section, we first compare the link quality scores obtained by all approaches. Next, we discuss whether and how these scores affect the lifetimes of the discovered paths for all flows. The objective is to investigate if high link quality scores affect the connection lifetimes of the computed paths. Therefore, we start several runs of all approaches and compare link qualities and the corresponding connection lifetimes.

Comparing Link Quality

Referring to Figure 7.9b of Section 7.5.1, connection qualities appear to be poor when applying shortest path algorithm with $q_l = 1$, where $l \in \mathcal{L}$. Because of that, SbDS' is a good standard of comparison to determine whether our robust link quality has an impact on the lifetimes of active connections.

To ensure comparability and to generate reliable results with the objective of identifying, which technique optimizes better, we conducted multiple runs of all pathfinding techniques to obtain at least 25 scenarios where all approaches found capacity-conform path settings. It is important to compare all techniques with the same runs, including equal topologies, source-target pairs, and initial over-utilization to, first of all, determine the results of each pathfinding technique and also to detect correlations between link qualities and connection lifetimes.

The following paragraphs elaborate on link quality scores and path lifetimes of all pathfinding techniques. Again, CPLEX and GP are evaluated with averaged SbDS runtimes as well as with maximum 8 minutes runtime, respectively.

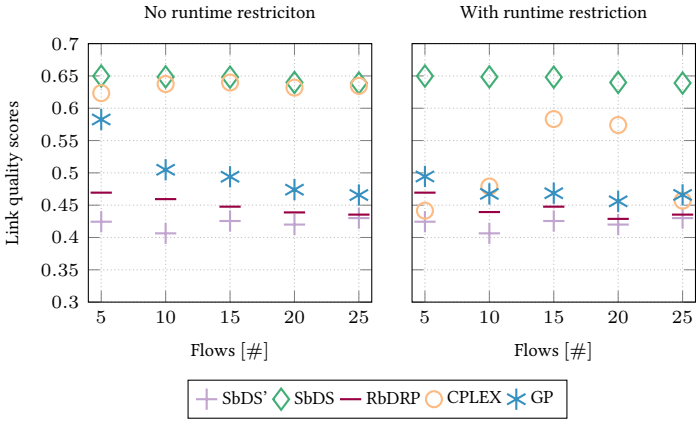


Figure 7.14: Comparing robust link quality scores with and without runtime restriction. The initial over-utilization is set to 5%.

Figure 7.14 depict robust link quality scores of all pathfinding techniques on the left-hand side without execution time restriction and on the right-hand side with runtime restriction. Initial over-utilization is set to 5%. The link quality scores of this figure are inverted to present them in a more readable form, meaning the higher the score the better the link quality.

Results show that with no rumtime restriction, CPLEX and SbDS produce the best scores, followed by GP. However, as the number of flows increases, the performance of GP deteriorates. The fewer potential solutions exist in the search space, the lower the probability to either generate these valid path settings in the initial population or to construct them with crossover or mutation. The fitness function is defined to rank capacity-conform path settings high in the first place followed by the robust link quality. The fewer potential solutions that can be found by GP the lower the probability is that these solutions comprise high link quality scores. SbDS' computes the worst scores since our link quality metric is not applied.

Evaluations, having runtime restrictions, only show different results of GP and CPLEX as RbDRP and SbDS require almost the same time, which is negligible. With limited runtimes, CPLEX is forced to terminate the computation and to return the current best solution, if available. This leads to worse link quality scores, especially with 5, 10, and 25 configured flows. GP once again generates the initial population with almost only k-shortest paths [Yen71], which, on the one hand, speeds up the execution times but,

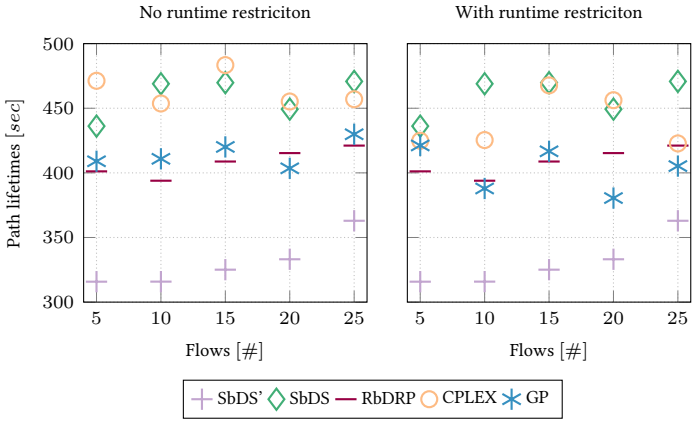


Figure 7.15: Comparing path connection lifetimes with and without runtime restriction. The initial over-utilization is set to 5%.

on the other hand, results in less diversity. This limitation, along with the time constraint, slightly reduces the link qualities. The differences become apparent with 5 flows.

Correlating Path Lifetimes

This section investigates if and to which extent robust link quality scores affect the path lifetimes. We simulated the mobility as explained in Section 7.3.6 and recorded the time how long each active link remains connected until transmitting or receiving nodes of deployed paths move out of each others coverage ranges. All MANET participants are moving continuously, closing and creating connections with nodes resulting in an ever-changing network topology. In order to generate meaningful results, we averaged the connection lifetimes of all active links of all paths of all common runs.

The left plot of Figure 7.15 displays the lifetimes of paths without runtime restriction, while the right plot of the same figure shows the lifetimes of paths with runtime restriction. Both figures display the connection lifetimes for 5, 10, 15, 20, and 25 flows, each with an initial over-utilization of 5%.

In most scenarios, results prove that longer path connectivity is accomplished when choosing links, which have better link qualities. This compar-

ison can best be seen in the results without time restrictions in the left plots of Figures 7.14 and 7.15. There, SbDS' computes paths with the worst robust link qualities and the corresponding paths of SbDS' lose connectivity first compared to all other approaches. Furthermore, GP and RbDRP have path connectivity lifetimes in the lower range. The paths of SbDS and CPLEX are connected for the longest period of time compared to other approaches.

Comparing path lifetimes and robust link qualities with restricted runtimes reinforces this relation. Results of RbDRP, SbDS, and SbDS' remain the same since the runtime restriction is aligned with their execution times. CPLEX returns worse path connection lifetimes in relation to obtained link quality scores with restricted runtimes. This becomes even more obvious with 5, 10, and 25 flows. Path lifetimes increase with 15 and 20 flows since link quality scores are high as well. The path lifetimes of GP are as expected when correlating them with link quality scores in which no runtime restriction is configured. When configuring pathfinding techniques with runtime restrictions, GP unexpectedly produces worse path lifetimes, yet they are still acceptable as the resulting link quality scores are worse than RbDRP. Also, RbDRP achieves surprisingly long connection lifetimes as well as surprisingly good link quality scores. The recorded lifetimes are in the same range as GP and SbDS'.

Summing up, we introduced the results of all pathfinding techniques of Section 7.4 focusing on found capacity-conform path settings, their link qualities, and connection lifetimes. There, we investigated the impact of robust link qualities in relation to their measured connection lifetimes. Furthermore, we obtained link quality scores and path lifetimes with and without runtime restriction and discussed the outcome. We proved that all of our approaches resolve most of the initial over-utilized MANET situations, speaking of 5% initial over-utilization. However, the number of found valid path settings decreases when initial over-utilization increases to 10% and 15%. Evaluations also revealed that the better the robust link quality score, the longer the corresponding connection lifetime is. The following section discusses when to apply each technique in order to generate the most promising results.

7.6 PATHFINDING CLASSIFICATION

This section defines application scenarios and classifies each pathfinding technique according to its advantages and disadvantages. We evaluated these approaches with and without runtime restriction as MANET par-

ticipants must receive routing updates as quickly as possible because the MANET topology changes due to the mobility of nodes. During the route-finding process, the topology also changes, and time-consuming calculations may result in short data transmissions because nodes of selected connections might have moved in opposite directions by the time pathfinding finished. As a consequence, paths must be requested more often. In the worst case, the connections may no longer exist when the pathfinding technique has finished its calculations. The shorter pathfinding takes, the higher the probability that paths stay longer connected.

We introduced the uplink channel that connects each MANET participant with the controller to convey network topology information to the controller and also to request paths from the controller. This architecture is named Cc-MANET.

We also discussed Sc-MANET which installs the controller on an arbitrary MANET participant in case no uplink channel is available. There, the MANET channel is utilized for routing load and data transmissions, respectively. Computation power is limited on mobile devices, seeking for minimum path computation runtimes to respond quickly to *Flow Requests*.

We assume that Cc-MANET provides high computation powers and pathfinding can, for instance, be paralyzed on cloud services or deployed on high-performance clusters, making computation runtimes of all approaches negligible. Because of that, one of the classification criterion for all pathfinding techniques is the available Computation Power.

Path computation takes place during various network transmission utilization situations. This thesis especially focuses on high network load situations, which is why pathfinding techniques deal with initially over-utilized MANETs. Because of that, application scenarios address, apart from the Computation Power low to high network over-utilization situations (Network Congestion). So far, a possible scenario would, for instance, comprise high Network Congestion combined with limited Computation Power. Figure 7.16 visualizes all application scenarios delimited by gray lines.

Furthermore, the classification focuses on the performance results Reliability and Robustness of each pathfinding technique. The former refers to how often each pathfinding technique returns a capacity-conform path setting. The latter represents the path lifetime of each result. Each approach is classified regarding these performance results with respect to each scenario, as depicted with inner coordinate systems in Figure 7.16. To be more precise, all pathfinding techniques are classified based on their found paths (Reliability) and their lifetimes (Robustness) taking into account each applica-

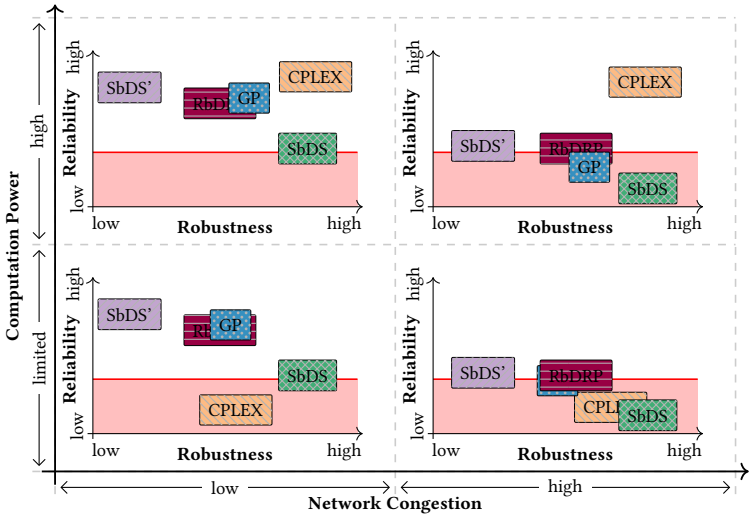


Figure 7.16: Application scenarios containing all pathfinding techniques classified in the associated criteria (Reliability and Robustness). The outer coordinate system depicts scenario combinations. The Inner coordinate systems classify pathfinding approaches in the current scenario regarding both criteria.

tion scenario (each combination of Computation Power and Network Congestion). For instance, pathfinding techniques, returning capacity-conform and long-living path settings in scenarios, comprising high Network Congestion and high Computation Power are declared as highly reliable and highly robust because their quality of the associated path lifetimes are long.

As mentioned earlier, a technique must return a capacity-conform path setting in the first place to take into account the associated path lifetime. Because of that, Reliability is more important than Robustness which is why each inner coordinate system comprises a horizontal red border declaring if the Reliability of a particular approach has reached a critical minimum threshold. An approach appears critical if a pathfinding technique touches or appears below this threshold.

The classifications in this figure are based on the results of Section 7.5.2. However, as the number of valid path settings per approach decreases with increasing initial over-utilization, the probability of comparing the path lifetimes of found results decreases because fewer results of each approach

exist. Results of high network overload situations make the comparison in this case less significant because equal runs, which returned a valid result of each approach must be used to compare robust link qualities with each other and also to rank their path lifetimes. Because of that, classifications of path lifetimes regarding each application scenario and path setting are based on an initial over-utilization of 5% as the number of common runs being successfully completed by each approach is sufficient. With the knowledge of path lifetimes during low Network Congestion, we assume that the path lifetimes (Robustness) of each pathfinding approach do not deviate significantly with increasing Network Congestion. Because of that, we map these results to high Network Congestion scenarios.

High Computation Power and Low Network Congestion In this scenario, the computation time to compute capacity-conform path settings is not an issue. CPLEX returns a capacity-conform path setting in this scenario constellation. Hence, we declare CPLEX as the most reliable pathfinding technique compared to all other approaches. This technique searches for the global optimum in this scenario. Consequently, CPLEX returns the longest path lifetimes.

We found out that SbDS' computes paths consisting of only short connection lifetimes. However, the number of found capacity-conform paths is very high, which is why SbDS' in turn offers high Reliability. In contrast, SbDS computes paths consisting of high connection qualities, resulting in long connection lifetimes. Found path settings became worse with either 15 flows and 5% over-utilization or 5 flows and 15% over-utilization, leading to the worst performance regarding Reliability.

GP and RbDRP obtain similar results regarding Reliability and Robustness. Computed paths of GP are slightly more robust compared to RbDRP, leading to longer connection lifetimes.

High Computation Power and High Network Congestion High network congestion, such as 15% initial over-utilization combined with 15 flows, decreases the results of CPLEX only marginally, which is why Reliability and Robustness are classified as high compared to all other approaches.

SbDS performs the worst regarding found capacity-conform paths resulting in the worst Reliability with respect to the performance classification. However, these few found capacity-conform paths have long path lifetimes. SbDS' still generates paths with short lifetimes and also computes significantly fewer paths under increased network load making this approach also insufficient regarding reliability at this point.

RbDRP computes as few valid paths as SbDS' and is also on the verge of the critical zone with respect to the classification Reliability. A similar classification is defined for GP regarding Reliability and Robustness.

Limited Computation Power and Low Network Congestion If only limited hardware resources are available to compute paths, long calculation times are to be expected, that are not tolerable in MANETs. Previous results show that only a fraction of possible paths can be found with restricted runtimes, which is why CPLEX is classified as insufficient regarding Reliability in this application scenario. In addition, lifetimes of found paths of CPLEX also decreased.

We experienced a slight path lifetime decrease of GP when limited computation time was present. The number of found paths is comparable to RbDRP resulting in similar reliability.

The classification of RbDRP, SbDS, and SbDS' is equal to the upper coordinate system depicting equal Network Congestion but high Computation Power since approaches already respond with valid path settings quickly.

Limited Computation Power and High Network Congestion Path lifetimes surprisingly increased with CPLEX while the number of capacity-conform path settings remains low which is why Reliability is classified as insufficient.

GP computed slightly less capacity-conform path settings and also reduced the lifetimes compared to the scenario configured with high Computation Power and high Network Congestion. Other approaches show similar results as limited runtimes do not change the result of found capacity-conform paths and their lifetimes.

7.7 SUMMARY

This chapter discussed RQ 4, tackling multi-flow path computation under runtime restrictions to guarantee capacity-conform routing in highly-utilized MANET situations. We focused on stressed network situations in terms of overloaded wireless connections and tried to calculate a routing result, which distributes all requested flows without any active connection being overloaded. Therefore, we generated artificially over-utilized network constellations and investigated if and to which degree pathfinding techniques resolve the initially generated stressed network segments. Thereby, pathfinding approaches must respond as quickly as possible with a valid path setting as nodes in MANETs are moving continuously generating new topology constellations. Lastly, it is important that the computed

routes remain connected for a long time, as the path calculations are time-consuming.

Following these objectives, resolving over-utilization and computing long-living paths while considering minimum runtime, led to additional tasks. We began to investigate connections to distinguish high-quality connections in terms of their lifetimes from lower-quality connections (Section 7.3.4). Combining node speeds, reception powers, and moving directions has proven to be promising and applied to describe wireless connections in MANETs as robust.

Next, over-utilized network situations are required to determine if and to which extent these stressed segments can be resolved in redistributing the currently deployed paths, which over-utilize transmitting connections. An evaluation environment, which we call MfDM, was necessary to create these over-utilized situations and to evaluate the desired result, as discussed in Section 7.3.

To begin, this workflow generates randomized MANETs similar to Figure 7.5. This topology constellation provides plenty of alternative paths and as a consequence highlights advantages and disadvantages of each pathfinding approach. Next, we artificially generated over-utilized network situations. In particular, we computed paths while continuously increasing the data rates of associated flows until we reached a predefined degree of initial over-utilization.

Next, 5 optimization techniques have been developed, extended, or configured to find a path for each flow in order to resolve the over-utilized network situation in recomputing the paths, considering the data rate demands of the flows. SbDS follows the Greedy approach in deploying the current best path of a flow with respect to the robust path quality and processes the same with all remaining flows until all paths are deployed on the topology representation (Section 7.4.2). This technique guarantees fast response times but sticks to the process even if the current best path over-utilizes the topology.

KmDP, introduced in Section 7.4.1, at first generates a pre-defined number of paths of each flow with the least similarity. Secondly, a process, enriched with randomness constructs possible path settings, containing a path of each flow. Thirdly, the best setting is chosen and deployed. Runtimes of this approach depend on the number of defined paths per flow. Preliminary results were disappointing, which is why this technique was not included in the results.

RbDRP identifies potentially over-utilized connections and penalizes

them when searching after the final path of each flow (Section 7.4.3). This technique is inspired by the similar work of Akin et al. [AK19]. RbDRP, first applies Dijkstra [Dij59] when searching for links that will potentially be selected by multiple flows. Secondly, we penalized all actively and passively utilized links in these areas and adjusted the costs to reinforce routing to compute paths around these stressed segments.

We also applied GP that self-improves potential solutions according to the fitness function, introduced in Section 7.4.5. The initial population comprises random and Yen's k-shortest paths [Yen71]. The fraction is adjustable based on the desired result. We designed special crossover and mutation techniques fitting for the complete path requirement to generate capacity-conform path settings containing robust connections.

Lastly, we applied CPLEX, a solver, taking a formalized model to use Branch & Bound and Branch & Cut techniques [Mit09] in order to converge to the global optimum (Section 7.4.4). This technique, on the one hand, is used to figure out if a capacity-conform path setting exists and, on the other hand, is applied to return a potential setting with a time constraint configured.

MANET participants move according to the GM [AWS06] mobility model to evaluate how long the links of the returned path settings of each approach stay connected. Therefore, we measured the time transmitting links stay connected.

The results focus on the number of found path settings, not over-utilizing the MANET with and without runtime restriction and on their connection lifetimes. Having no time limitation configured, CPLEX outperforms all other approaches with 5% and 15% over-utilization configured, followed by GP configured with 5% over-utilization. With 15% initial over-utilization, SbDS' (SbDS with link quality set to 1) and RbDRP outperform GP. The number of found path settings of CPLEX decreases to a minimum when runtimes are restricted to the execution time of SbDS. CPLEX finds by far the fewest paths in all configurations with the runtime constraint set. There, we evaluated MANETs configured with 5, 10, and 15 flows combined with 5%, 10%, and 15% initial over-utilization. All approaches except of CPLEX and SbDS find a large portion of valid path settings, see Plot 7.11. Results decrease with increasing initial over-utilization, which is true for each approach. The longest connection lifetimes are carried out by CPLEX if no time constraint is configured. Considering runtime restrictions, paths having the longest lifetimes are found by SbDS.

Revisiting the initial objective, we now can claim that over-utilization in

MANETs can be resolved depending on the current level of over-utilization and the number of requested flows. The higher the initial over-utilization and the more flows are requested, the more the probability decreases that one of the proposed pathfinding techniques returns a path combination considering the runtime restriction where no active link exceeds its configured capacity. However, until 10% initial over-utilization each technique returns plenty of valid path settings.

We also proved that our defined robust link quality has a positive impact on the connection lifetimes of found solutions. Further, we presented a classification that recommends which pathfinding technique to apply in which application scenario in order to generate the best results in terms of found capacity-conform path settings and connection lifetimes.

CHAPTER 8

Conclusions and Outlook

In this final chapter, we summarize all RQs highlighting each problem statement and achievements. Based on that, we create a logical connection of all RQs in Section 8.3, called Connecting the Dots. Moreover, we take a closer look at the answers of our RQs and discuss the outcome. In the end, we elaborate on future research opportunities to extend and improve our proposed framework.

8.1 SUMMARY

Routing in MANETs has always been and continues to be very demanding. MANETs are characterized by a single-layer architecture not depending on further dedicated routing services or hardware for routing [Gio02]. The self-organized fashion requires offloading the route-finding process on participants, acting as clients and routers, respectively [Toh96]. A commonly used wireless channel, providing only restricted transmission capacity, connects each participant whereby connections change constantly as nodes are moving.

It is obvious that one aims to exploit available transmission capacities most efficiently in terms of determining paths utilizing the MANET the least to provide the most capacity for additional transmissions. In return, one tries to exchange as few routing management messages as possible to leave the most transmission capacities for data. Considering the mobility of nodes and the changing network topology, control messages should also be sent at the most reasonable time providing up-to-date and complete routing information when needed. Further, distributed routing complicates utilization-efficient routing as participants forward data based on their individual entries in their routing tables. This makes multi-flow routing complicated as these local routing decisions commonly do not pay attention to whether the new paths disturb existing transmissions.

We addressed these drawbacks with the aim to make these networks more resilient and reliable. Our achievements are especially important to the military and civil protection organizations as these organizations must not rely on a central point of failure [Jah+18].

In order to address the restricted data rates, we introduced a transmission overhead recording model providing accurate information about connection utilization to determine the residual transmission capability of the MANET and further if there are sufficient capacities for additional flows. In the course of this thesis, we developed a framework comprising two network architectures that eliminate isolated routing decisions of MANET participants by centralizing the routing competence in a controller. We extended both architectures with network topology update processes reporting connections to the central routing instance to construct a network graph as a representation for routing purposes. Now, routing can rely on accurate network utilization monitoring and complete topology information, reaching a level of timeliness despite the mobility of nodes. We summarized all findings at that point in time and developed and compared several pathfinding processes to find paths for multiple flows in highly-utilized MANET situations. The objective was to determine to which extent these approaches are useful of resolving situations in which multiple flows over-utilize the MANET to a certain degree. This sheds light on the question of whether central routing in MANET more efficiently distributes multiple flows in terms of the generated transmission utilization. Furthermore, we developed a robust connection property detecting potentially long-living connections with the objective of increasing the lifetimes of computed paths since nodes move and connections break down in the future.

8.2 REVISITING THE RESEARCH QUESTIONS

In this section, we address each RQ, introduced in Section 1.2 and provide a comparison of the initially identified problem with the outcome, concluding with the added value of our research.

RQ 1 - *How to record and compute accurate connection utilization during capacity-demanding transmissions?*

The first RQ focused on recording connection utilization and computation methods in MANETs. We presented two approaches. The experiences and obtained results of the first significantly improved the recording and com-

putation technique of the second approach, and as a consequence the outcome in terms of results. We began to elaborate on radio wave propagation in wireless multi-hop networks, such as MANETs, to take co-channel interference of route participants into account. This provided the foundation for generating classifications of each MAC layer state, which determine the type and duration of the current situation, and whether this refers to occupation or availability of a connection. One of the takeaways of the first link utilization recording approach was to apply a single controller storing the MANET topology to more accurately compute the potential overhead of an upcoming flow for chosen route participants as well as the arising passive utilization of nearby nodes. We investigated several scenarios, focusing on highly-utilizing transmissions in MANETs to determine the extent to which recorded utilization overhead equals the computations of the controller. The results show accurate utilization computations with increasing data rates until path participants reach their transmission boundaries.

Overall, this RQ initiated the integration of the Cc-MANET architecture where an outsourced controller, reachable via an uplink channel, exploits MANET topology knowledge for utilization-aware routing. This architecture in combination with the connection utilization recording and computation concept allowed us to conduct further research and brought us one step closer to answer the question to which extent the MANET capacity can be exploited to compute routes that are robust and do not over-utilize the network.

RQ 2 - *Which controller-equipped MANET modifications are required to provide a complete topology representation for an outsourced routing instance?*

With connection utilization measurements we noticed slow reports of changes in the actual MANET topology to the controller, particularly when mobility came into play. The initial Cc-MANET architecture was equipped with a proactive topology update process (Pro-CeTUP). Exploiting the available transmission capacity of the MANET at the best required an up-to-date topology snapshot provided on the controller. After highlighting and arguing the lack of actuality, reinforced by evaluations, we introduced Re-CeTUP, a reactive MANET topology reporting concept focusing on actuality in terms of participants and their connections. In particular, a variety of paths not present in the actual network topology has been computed by the controller using the proactive topology update technique. Applying a reactive update technique, triggering the MANET participants to report their

directly connected neighbors to the controller, eliminates this drawback.

In the course of that, we introduced tailored MAC modifications on MANET participants, managed by the controller, that aim for exclusive channel access per node during topology reporting. Integrating Re-CeTUP in Cc-MANET splits the network operation in topology reporting and data delivery. In that context, we introduced a MAC switching routine to enable the most suitable access technique at the respective point in time. The isolated evaluation of Re-CeTUP regarding completeness in terms of connections and execution time yielded satisfying results. Re-CeTUP outperformed Pro-CeTUP when comparing topology completeness. Re-CeTUP showed only negligible packet loss due to the MAC switching routine making the update process a reliable component for up-to-date topology reporting to an outsourced controller.

To sum up, the second RQ exchanged Pro-CeTUP with Re-CeTUP in the Cc-MANET architecture. This, among others, required a MAC switching routine to use both data delivery and topology reporting in the best ways. Re-CeTUP, embedded in the Cc-MANET architecture achieved convincing results, pointing to evaluations on complete topology reports, short topology update process times, and a high packet delivery fraction during phases of increasing mobility.

RQ 3 - *How to provide and access complete topology knowledge when the controller is deployed on a MANET member?*

Assuming continuous connectivity to the outsourced controller via a dedicated uplink is naive as unforeseen circumstances, such as a power outage could cut the connection of each MANET participant to the controller. The third RQ aims to provide similar functionality, meaning centralized and up-to-date topology knowledge, in case the uplink is not available.

We first evaluated the proactive routing technique, also providing full topology knowledge on MANET nodes, by focusing on missing and obsolete links. The outcome helped us to develop and test a reactive topology reporting process. In doing so, we proposed the algorithm Re-SoTUP using exclusively the MANET channel to reactively report the entire topology in minimum time to a controller, deployed on an arbitrary MANET node. This partitions the controller and the MANET only logically, meaning the controller is also a participant taking part in data transmissions. We named this architecture Sc-MANET, having a controller as a routing engine deployed on an arbitrary node.

In that context, we introduced a tree, dynamically self-constructing out

of the MANET, which in the first place spreads neighborhood information starting at the controller (root node) through the tree until reaching all leaves. This ensures upstream connectivity to the parents in order to thereafter report sub-topologies to them which are carried back to the controller. Complete and up-to-date topology representations on the controller could reliably be reported in plenty of scenario constellations, referring to evaluations experiencing no noteworthy deviations regarding the up-to-date topology representation with increasing mobility. The controller triggers this algorithm if participants request new routes via *Flow Requests*.

Furthermore, we defined a controller self-management concept. We contributed further functionalities helping to apply Re-SoTUP at runtime. Firstly, nodes continuously maintain their route to the currently elected controller for potential *Flow Requests*. Secondly, MANETs elect a single controller per network in a self-organized way in case of outages or if multiple controllers are identified, for instance, in MANET merge situations. We focused on minimum control message overhead, keeping in mind the restricted channel capacities. These functionalities are bundled in one algorithm, called SoCM, leveraging the tree of Re-SoTUP to achieve all requirements. Evaluations prove correct functionality while reducing the control message overhead.

RQ 4 - *To which extent can time-sensitive routing compute robust paths of multiple flows while not over-utilizing the network?*

This RQ leverages the capabilities of RQ 1 to RQ 3 introducing a routing engine during highly-utilized MANET situations. The objective was to determine a capacity-conform path for each flow currently over-utilizing the network while focusing on robust connections. Robust in this context refers to long connection lifetimes of transmitting links.

We therefore introduced the MfDM guiding us throughout the entire process from generating various random MANET topologies to measuring the lifetimes of computed paths. MfDM generates an over-utilized MANET situation on these randomly generated topologies having a configured number of flows deployed. The workflow applies wireless channel characteristics to each connection and also implements mobility models defining mobility patterns, such as waypoints and velocities of nodes, mimicking realistic MANETs. Based on that, we defined the link quality, distinguishing between fragile and robust connections to identify potential paths as long-living. We selected several parameters, such as the velocity of nodes and re-

ception powers among others, and evaluated combinations of those searching for suitable results to determine potential long-living connections. We also formalized the utilization model to determine in theory, which links are utilized, and in addition to which degree these links have to decrease their transmission capacities. This allows us to conclude whether over-utilization of the MANET would be present due to the paths of all requested flows. The MfDM applies this model to, first of all, generate over-utilized situations in increasing data rates of flows to reach a predefined level and, second of all, to determine the extent of the configured over-utilization before heading for the objective to resolve these bottlenecks.

Most importantly, we proposed five pathfinding techniques with the objective of computing robust paths not over-utilizing the MANET. Mobility of MANETs forces approaches to respond quickly, which is why our pathfinding techniques KmDP, RbDRP, SbDS, and GP focused on fast execution times or can be configured to behave so. We also used the solver CPLEX to compare results with other approaches when configured with restricted execution time and also to prove whether a capacity-conform path setting exists in a given scenario. The MfDM simulates the mobility of MANET participants before and after pathfinding takes place. We recorded the time these transmitting connections are connected in order to evaluate the robustness.

We presented evaluations concerning results with and without runtime restrictions. Thereby, all approaches are evaluated with scenarios where initial over-utilization increases step by step. All in all, until 15% initial over-utilization, there exists a pathfinding technique providing adequate results in terms of capacity-conform paths. Further, we compared the link qualities of the found paths of all approaches with each other to figure out which pathfinding technique computes paths having the longest connection livetimes. In addition, we proved that paths, which have promising link qualities obtain longer connectivity and should be preferred.

In that context, we defined application scenarios and discussed which pathfinding technique should be preferred in each situation in order to detect the best option.

8.3 CONNECTING THE DOTS

This section logically connects all individual pieces of all RQs together to a cohesive whole. The thesis comprises multiple invented algorithms, measurement methods, and designed models among others with the objective

of addressing each RQ. It is important to understand where all components, developed algorithms, and designed and evaluated optimization techniques fit into the big picture.

We first recap the challenges and constraints of MANETs, introduced in Chapter 1, to understand how all parts of the RQs fit together. Secondly, we introduce the big picture and discuss, which component addresses which challenges and constraints.

MANETs characteristically comprise single-layer architecture, organizing routing and data delivery in a self-organized way. The wireless channel provides only restricted transmission rates, especially in the military sector [MKL15; LAM12]. Real-time traffic, such as video calls and streaming services [Ham20] among others, have capacity-demanding data rate requirements [Adobe], pushing MANET participants to their limits as multi-hop routes demand the commonly used channel even more [FKQ19; RGM18]. Routing in MANETs commonly acts in a distributed fashion where each node makes forwarding decisions in isolation based on its current routing knowledge, referring to common routing protocols [Cla+14; Neu+08; DPB03; HMJ07]. Path constructions of multiple flows potentially disturb each other and are placed sub-optimally wasting transmission capacities for additional flows. Also, the key difference in contrast to traditional networks is that participants are moving continuously, resulting in an ever-changing topology. Routing information on each participant comprises inconsistencies resulting in incomplete and outdated topology knowledge stored on each participant.

The identified problems are summarized as follows: (i) Restricted transmission capacities, (ii) incomplete and outdated topology knowledge, and (iii) isolated routing behavior of participants. The constraints and challenges form the desired outcome and objective which define a framework comprising a centrally managed routing engine taking over path computation based on the up-to-date topology information, that stores the current complete topology (nodes and connections). Furthermore, all characteristics, such as flows, data rate demands, connection capacities, their characteristics, and mobility information of MANET participants are required to achieve the desired goal.

Figure 8.1 depicts both architectures and corresponding components developed and evaluated throughout this thesis. This figure represents the relation of components, algorithms, among others to each architecture. Starting with architectures, we designed Cc-MANET and Sc-MANET equipped with a controller, responsible for routing, respectively. The first architec-

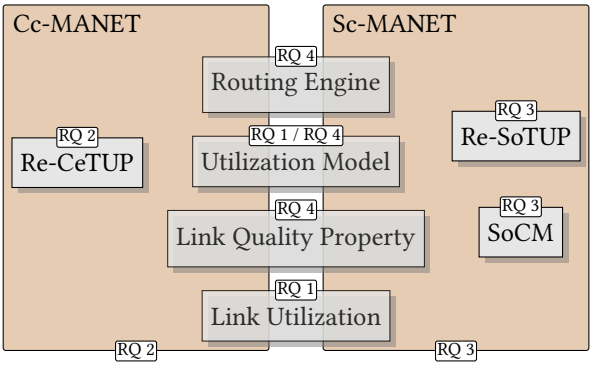


Figure 8.1: The framework, comprising architectures and components created throughout this thesis. Brown rectangles represent architectures and gray rectangles represent components associated to the corresponding architecture. Components used in two architectures overlap in both.

ture outsources the controller and instantiates the uplink channel for each MANET participant to connect to the controller. In contrast to Cc-MANET, Sc-MANET deploys the controller on an arbitrary node participating in the MANET reachable via the commonly used MANET channel, which is the first choice for the MANET in case the uplink is not available.

Next to Cc-MANET designed in RQ 1, we investigated how accurately link utilization can be measured when connections reach their capacity limits. Computing paths for multiple flows in highly-utilized MANET situations requires determining whether the defined links that are selected to carry the flow exhibit sufficient capacities. This depends, on the one hand, on how accurately the utilization is recorded and, on the other hand, how newly deployed flows demand the MANET. The components Link Utilization and parts of the Utilization Model comprise mentioned functionalities where the former is completely and the latter is partially covered by RQ 1. Both components are designed to be used in Cc-MANET and Sc-MANET.

After identifying how utilization propagation in wireless multi-hop networks can be exploited for routing, we further require complete and reliable topology knowledge at the controller in terms of existing connections and their participants. The topology must be as complete and up-to-date as possible since routing must be provided with the best possible information. More specifically, in some cases there are only a minority of potential paths in the current MANET that allow no connection to be over-utilized. Keep-

ing this requirement in mind, we defined Re-CeTUP gathering the MANET topology to the controller immediately before routing takes place via the uplink. This algorithm and the MAC switching routine are both defined and evaluated in RQ 2. The latter, which is not depicted in the figure, ensures that MANET participants and the controller switch MAC protocols at runtime ensuring interference-free transmissions during topology update cycles.

In order to also provide a complete and up-to-date topology representation for the controller in the Sc-MANET architecture, we designed and evaluated Re-SoTUP also providing the controller, deployed on a MANET participant, with the topology representation. Questions arise about how MANETs manage the controller role in situations such as controller outages and in case several MANETs merge that are all equipped with a single controller. Furthermore, one must ensure controller reachability for each MANET participant since topology changes and *Flow Requests* often traverse multiple nodes before reaching the controller. These challenges are solved with the component SoCM taking care of controller management and reachability. Thereby, SoCM uses functionality of Re-SoTUP.

With all the mentioned components so far, we aimed for robust multi-flow route computation in highly-utilized MANET situations. Before that, we defined a connection classification applied to each link to determine robust connections potentially providing long-living paths. We also generalized the Utilization Model in the RQ 4 to be able to identify the impact of utilization if all flows are deployed. Now, the controller uses utilization information, link characteristics, and topology knowledge gathered with Re-CeTUP or Re-SoTUP and forms the MANET representation. The controller starts the MANET Routing Engine and applies a pathfinding technique best fitting for the current application scenario as each technique has its advantages and disadvantages. The component Routing Engine points to Cc-MANET and Sc-MANET as multiple pathfinding techniques are provided fitting both architectures. The same goes for the Robust Link Quality metric and the Utilization Model as each architecture uses this information for routing.

To sum up, Figure 8.1 comprises all components and architectures to provide capacity-conform robust multi-flow routing in MANET. We therefore bring together MANET transmission utilization information, link characteristics, and up-to-date topology information to construct the routing with the Routing Engine leveraging all this information.

8.4 FUTURE RESEARCH CHALLENGES

This section introduces future research not explored in this thesis due to time constraints but maybe relevant in the field of MANETs. In this context, we highlight potential and promising extensions.

Applying Re-SoTUP only in Sub-trees of SoCM. SoCM comprises a tree having the controller as root, aiming to send as few control messages as possible. A further goal is that parents roughly keep track of sub-tree nodes. Children send *Membership Leaves* to their parents when disconnecting and also send *Parent Replies* containing sub-tree information if triggered by their parents. This information provides indications of whether a particular child is connected to a sub-tree of a parent.

Re-SoTUP reactively collects topology information of the entire MANET starting at the root, also known as the controller, each time a *Flow Request* arrives. This generates control message overhead to report each existing connection and participant to the controller. Each node along the route toward the controller could check if the destination of the *Flow Request* is stored in its sub-tree information. If true, this parent starts Re-SoTUP only in this sub-tree, to gather up-to-date topology information of this partial tree. This parent forwards the route request including the up-to-date topology representation of the sub-tree to its parent or controller triggering it to also look for the destination in the sub-tree and behave equally if this node is listed.

This would reduce the message overhead of Re-SoTUP as this algorithm would start at a specific parent node only gathering sub-tree topologies. In case the desired destination is not included in the sub-tree after running Re-SoTUP, a parent or controller runs Re-SoTUP again. Sub-trees already reported the sub-topology to their parent stay idle as this parent stops the algorithm at this point to reduce the control message overhead.

The following possible research questions are derived when comparing this extension with the traditional Re-SoTUP: To which extent can the control message overhead be reduced? Furthermore, routing multiple flows requires knowledge of the entire topology. How to use the sub-topology knowledge when further topology information is necessary to determine a path between origin and target?

Centrally managed and self-organized MAC techniques for MfDM. The Utilization Model in this thesis assumes CSMA/CA [IEEE802.11] when delivering data in the MANET. MfDM represents an evaluation workflow, testing to which extent pathfinding techniques can resolve over-utilized

MANET segments due to transmissions. Paths of flows are computed by various pathfinding techniques and deployed in the forwarding tables of nodes if they are part of a path.

CSMA/CA is a probabilistic MAC technique including fairness during channel access. Competing for channel access is time, not used for transmitting or receiving data frames, which affects each participant in the MANET. This is one of the disadvantages of CSMA/CA. The advantage in this situation is that no additional channel allocation computation for each MANET participant is necessary, which would be the part of the controller.

However, the controller computes the paths and nodes use CSMA/CA to access the channel when delivering packets of the flow. The controller could also apply TDMA to compute and allocate slots to each transmitting node since data rates of flows and radio configurations of MANET participants are available. Competition times when using CSMA/CA would be eliminated, thus generating additional transmission capacities in the MANET. One of the research questions would be whether the slot computation overhead is proportionate compared to additional transmission capacities.

Combining Re-CeTUP with Re-SoTUP. Situations must be taken into account where the uplink of Cc-MANET does not cover the entire MANET. Re-CeTUP reports all connections of nodes having connectivity via the uplink to the controller. The *Topology Refresh* message triggers each node to start *Neighbor Updates* and *Topology Updates* thereafter. A MANET participant could start Re-SoTUP, mimicking the controller, to gather the topology of nodes not connected to the actual controller, when receiving a *Topology Refresh*. Participants would only take part if they receive a message of Re-SoTUP and in combination have no connection via the uplink to the controller. The root node of the tree would include the tree, gathered via Re-SoTUP in the *Topology Update* when its slot starts.

In this case, one must consider the following extensions for Re-CeTUP: *Neighbor Updates* and messages belonging to Re-SoTUP must not interfere with other transmissions as both potentially happen at the same time. Slots for *Neighbor Updates* are sized to fit a fully meshed MANET, not considering further information gathered via Re-SoTUP. Lastly, the controller determines the start and end time of Re-CeTUP. Additional time must be considered by the controller to be able to include further topology information by MANET participants.

Real-World Testbed for Cc-MANET and Sc-MANET. Several parts of Re-CeTUP and Sc-MANET comprising their components and algorithms

Re-CeTUP, Re-SoTUP, and SoCM are simulated with OMNet++ [1] [Var10] and INETMANET [2]. Thereby, functionality and performance tests were the main focus of evaluations. Although Re-SoTUP and parts of Re-CeTUP have been implemented on microcontrollers, such as nodes of the IoT-Lab[3], both complete frameworks comprising a meaningful number of MANET participants have not yet been deployed on a real-world testbed. The number of microcontrollers mimicking MANET nodes that have been used for performance tests in RQ 3 (Section 6.3.1) is limited to 5. Extending the network to 30 nodes increases the complexity and also the probability of missing topology knowledge during update process. It is important to verify if the completeness of the topology representation on the controller decreases with an increasing number of microcontrollers, functioning as MANET participants.

A similar potential exists for SoCM. There, it is also interesting to figure out if at first, the message overhead remains the same with a small real-world testbed and also if the control message overhead increases in relation to the number of nodes when increasing the testbed participants. As a recap, SoCM maintains a route to the controller for each MANET participant aiming for the least control message overhead.

Furthermore, the reactive neighbor update technique of Re-CeTUP has been evaluated regarding number of delivered *Neighbor Updates* with up to 30 microcontrollers. However, the complete concept Re-CeTUP, comprising MAC switching routine and ongoing data delivery has not been tested with real hardware.

Cc-MANET entails similarities to SDN architectures whereas the wireless communication of Cc-MANET is one of the biggest differences. Nevertheless, research in the field of SDN applied in MANETs has been conducted. So far, previous research approaches introduced architectures and identified potential as well as challenges [YQR17; Pou+19b]. It seems promising to combine current real-world experiments with Cc-MANET and evaluate a scaled-up scenario in terms of MANET participants in a real environment.

Considering Battery lifetimes of MANET participants. MANETs are active and provide connectivity to each other until nodes run out of battery. It is obvious that different battery levels are present with respect to each participant. The more MANET members exist, the more routing opportu-

[1] https://inet.omnetpp.org
[2] https://github.com/aarizaq/inetmanet-3.x
[3] https://www.iot-lab.info

nities are available for flow distribution. Future research could investigate how transmissions and receptions affect battery usage. Furthermore, this information should be available when the MANET Routing Engine starts computing paths for all requested flows. Participants running out of battery could be avoided during path computation.

Glossary

ACK Acknowledgment.

ACO Ant Colony Optimization.

AODV Ad-Hoc on-demand Distance Vector.

AODV-CBR Capacity-based Routing.

ARP Address Resolution Protocol.

AS Autonomous System.

BFS Breadth First Search.

CAKE Central Authorized Key Extension.

Cc-MANET Central-controlled MANET.

CP Control Plane.

CPLEX IBM ILOG Cplex Optimizer.

CRT Chinese Remainder Theorem.

CSMA/CA Carrier Sense Multiple Access / Collision Avoidance.

CTS Clear-to-Send.

CW Contention Window.

D2D Device-to-Device.

DARPA Defense Advanced Research Projects Agency.

DCF Distributed Coordination Function.

DFS Depth First Search.

DIFS Distributed Interframe Space.

DoS Daniel of Service.

DP Data Plane.

DSCP Differential Services Code Point.

DSR Dynamic Source Routing.

ECMP Equal Cost Multipath.

ECN Explicit Congestion Notification.

ETX Expected Transmission Count.

FANET Flying Ad-Hoc Network.

FDMA Frequency-Division Multiple Access.

FEC Forward Error Correction.

FSLS Fuzzy Sighted Link State.

FSR Fisheye State Routing.

GA Genetic Algorithm.

GM Gauss-Markov.

GP Genetic Programming.

GPS Global Positioning System.

GSM Global System for Mobile Communications.

GSR Global State Routing Protocol.

IEEE Institute of Electrical and Electronics Engineers.

IFS Interframe Space.

IGMP Internet Group Management Protocol.

IoT Internet of Things.

IP Internet Protocol.

KmDP K-most Disjoint Paths.

LAN Local Area Network.

LLDP Link Layer Discovery Protocol.

LP Linear Programming.

LTE Long Term Evolution.

MANET Mobile Ad-Hoc Network.

MAC Medium Access Control.

MCNFP Multi-commodity Network Flow Problem.

MfDM Multi-flow Routing Model.

MhCD2D Multi-hop Cellular Device-to-Device.

MPR Multi-Point Relay.

MTN Mobile Tactical Network.

MTU Maximum Transmission Unit.

NATO North Atlantic Treaty Organization.

NAV Network Allocation Vector.

NBWF Narrowband Waveform.

OFDMA Orthogonal Frequency Division Multiple Access.

OFDP OpenFlow Discovery Protocol.

OLSR Optimized Link State Routing.

ONOS Open Network Operating System.

OSI Open Systems Interconnection.

OSPF Open Shortest Path First.

Pro-CeTUP Proactive Topology Update Process.

PRS Packet Radio System.

QoS Quality of Service.

RbDRP Rate-based Distributed Robust Paths.

Re-CeTUP Reactive Topology Update Process.

Re-SoTUP Reactive Self-organized Topology Update Process.

RERR Route Error.

RFC Request for Comment.

RQ Research Question.

RREP Route Reply.

RREQ Route Request.

RSU Road Side Unit.

RTS Request-to-Send.

RWP Random Waypoint.

SATCOM Satellite Communication.

SbDS Sequence-based Deployment Strategy.

Sc-MANET Self-controlled MANET.

SDMANET Software-defined Mobile Ad-Hoc Network.

SDN Software-defined Networking.

SIFS Short Interframe Space.

SNR Signal-to-Noise Ratio.

SoCM Self-organized Controller Management.

SRW Soldier Radio Waveform.

STAR Source-tree Adaptive Routing.

TCP Transmission Control Protocol.

TCP/IP Transmission Control Protocol/Internet Protocol.

TDMA Time-division Multiple Access.

TTL Time to Live.

UAV Unmanned Aerial Vehicle.

UDP User Datagram Protocol.

UHF Ultra High Frequency.

US United States.

VANET Vehicular Ad-Hoc Network.

VHF Very High Frequency.

VoIP Voice over IP.

WAN Wide Area Network.

WMN Wireless Mesh Network.

WSN Wireless Sensor Network.

List of Figures

List of Tables

List of Algorithms

Bibliography

[Abd+17] Nasim Abdolmaleki, Mahmood Ahmadi, Hadi Tabatabaee Malazi and Sebastiano Milardo. "Fuzzy Topology Discovery Protocol for SDN-based Wireless Sensor Networks". In: *Simulation Modelling Practice and Theory* 79 (2017), pp. 54–68.

[Abo+15] Mehran Abolhasan, Justin Lipman, Wei Ni and Brett Hagelstein. "Software-Defined Wireless Networking: Centralized, Distributed, or Hybrid?" In: *IEEE Network* 29.4 (2015), pp. 32–38.

[Abo+18] Mehran Abolhasan, Mahrokh Abdollahi, Wei Ni, Abbas Jamalipour, Negin Shariati and Justin Lipman. "A Routing Framework for Offloading Traffic From Cellular Networks to SDN-Based Multi-Hop Device-to-Device Networks". In: *IEEE Transactions on Network and Service Management* 15.4 (2018), pp. 1516–1531.

[AK19] Erdal Akin and Turgay Korkmaz. "Rate-Based Dynamic Shortest Path Algorithm for Efficiently Routing Multiple Flows in SDN". In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 2019, pp. 1–7.

[AKS17] Singh Ajeet, Ramesh Kumar and Pant Sonu. "Advanced Routing on AODV Using Link Prediction in Mobile Ad-Hoc Network". In: *2017 3rd International Conference on Advances in Computing, Communication Automation (ICACCA) (Fall)*. Sept. 2017, pp. 1–7.

[Alm92] Philip Almquist. *Type of Service in the Internet Protocol Suite*. RFC 1349. RFC Editor, July 1992, pp. 1–28. URL: https://www.rfc-editor.org/info/rfc1349.

[AM18] Muhammad Yeasir Arafat and Sangman Moh. "Location-Aided Delay Tolerant Routing Protocol in UAV Networks for Post-Disaster Operation". In: *IEEE Access* 6 (Oct. 2018), pp. 59891–59906.

[AMM01] R. Ananthapadmanabha, B.S. Manoj and C.Siva Ram Murthy. "Multi-hop Cellular Networks: The Architecture and Routing Protocols". In: *12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. PIMRC 2001. Proceedings (Cat. No.01TH8598)*. Vol. 2. 2001, pp. 78–82.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. USA: Prentice-Hall, Inc., 1993.

[And+14] Sergey Andreev, Alexander Pyattaev, Kerstin Johnsson, Olga Galinina and Yevgeni Koucheryavy. "Cellular Traffic Offloading onto Network-Assisted Device-to-Device Connections". In: *IEEE Communications Magazine* 52.4 (2014), pp. 20–31.

[APF15] Leonardo Ochoa Aday, Cristina Cervelló Pastor and Adriana Fernández Fernández. "Current Trends of Topology Discovery in OpenFlow-based Software Defined Networks". 2015.

[Ass78] Arjang A. Assad. "Multicommodity network flows—A survey". In: *Networks - An International Journal* 8.1 (1978), pp. 37–91.

[AWS06] Jinthana Ariyakhajorn, Pattana Wannawilai and Chanboon Sathitwiriyawong. "A Comparative Study of Random Waypoint and Gauss-Markov Mobility Models in the Performance Evaluation of MANET". In: *2006 International Symposium on Communications and Information Technologies*. 2006, pp. 894–899.

[Ban+98] Wolfgang Banzhaf, Peter Nordin, Robert Keller and Frank Francone. *Genetic Programming: An Introduction on the Automatic Evolution of computer programs and its Applications*. Morgan Kaufmann, Jan. 1998.

[Ban01] Wolfgang Banzhaf. "Artificial Intelligence: Genetic Programming". In: *International Encyclopedia of the Social & Behavioral Sciences*. Ed. by Neil J. Smelser and Paul B. Baltes. Oxford: Pergamon, 2001, pp. 789–792.

[Bar+16] Christoph Barz, Christoph Fuchs, Jonathan Kirchhoff, Julia Niewiejska and Henning Rogge. "Extending OLSRv2 for Tactical Applications". In: *2016 International Conference on Military Communications and Information Systems (ICMCIS)*. 2016, pp. 1–8.

[Bar13] Dave Barker. "Bringing Mobile Ad Hoc Networks to the battlefield using COTS open standards". In: *Extreme Engineering Solutions* (2013), pp. 22–26.

[BDG18] Paolo Bellavista, Alessandro Dolci and Carlo Giannelli. "MANET-oriented SDN: Motivations, Challenges, and a Solution Prototype". In: *2018 IEEE 19th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. June 2018, pp. 14–22.

[Bel58] Richard Bellman. "ON A ROUTING PROBLEM". In: *Quarterly of Applied Mathematics* 16.1 (1958), pp. 87–90.

[Bem+19] Amaury Van Bemten, Jochen W. Guck, Carmen Mas Machuca and Wolfgang Kellerer. "Bounded Dijkstra (BD): Search Space Reduction for Expediting Shortest Path Subroutines". In: *CoRR* abs/1903.00436 (2019). arXiv: 1903.00436. URL: http://arxiv.org/abs/1903.00436.

[Bha+94] Vaduvur Bharghavan, Alan Demers, Scott Shenker and Lixia Zhang. "MACAW: A Media Access Protocol for Wireless LAN's". In: *Proceedings of the Conference on Communications Architectures, Protocols and Applications*. SIGCOMM '94. Association for Computing Machinery, 1994, pp. 212–225.

[Bia00] Giuseppe Bianchi. "Performance Analysis of the IEEE 802.11 Distributed Coordination Function". In: *IEEE Journal on Selected Areas in Communications* 18.6584745 (2000), pp. 535–547.

[BIO17] Daniel Baeza, Christian F. Ihle and Julián M. Ortiz. "A comparison between ACO and Dijkstra algorithms for optimal ore concentrate pipeline routing". In: *Journal of Cleaner Production* 144 (2017), pp. 149–160.

[BJ08] Osama Bazan and Muhammad Jaseemuddin. "Multicommodity Flow Problem for Multi-hop Wireless Networks with Realistic Smart Antenna Model". In: *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next*

Generation Internet. Ed. by Amitabha Das, Hung Keng Pung, Francis Bu Sung Lee and Lawrence Wai Choong Wong. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 922–929.

[BK16] Deepti Badal and Rajendra Singh Kushwah. "Energy Efficient Approach of Route Selection for Dynamic Source Routing Protocol in MANET". In: *Proceedings of the International Conference on Advances in Information Communication Technology & Computing*. AICTC '16. New York, NY, USA: ACM, 2016, pp. 1–4.

[Bou+11] Azzedine Boukerche, Begumhan Turgut, Nevin Aydin, Mohammad Z. Ahmad, Ladislau Bölöni and Damla Turgut. "Routing protocols in ad hoc networks: A survey". In: *Computer Networks* 55.13 (2011), pp. 3032–3080.

[BS96] Arjen W. Brander and Mark C. Sinclair. "A Comparative Study of k-Shortest Path Algorithms". In: *Performance Engineering of Computer and Telecommunications Systems: Proceedings of UKPEW'95, Liverpool John Moores University, UK. 5–6 September 1995*. Ed. by Madjid Merabti, Michael Carew and Frank Ball. London: Springer London, 1996, pp. 370–379.

[BST13] İlker Bekmezci, Ozgur Koray Sahingoz and Şamil Temel. "Flying Ad-Hoc Networks (FANETs): A survey". In: *Ad Hoc Networks* 11.3 (2013), pp. 1254–1270.

[CC12] Fei-xue Chu and Shi-yi Chen. "Optimal Design of Pipeline Based on the Shortest Path". In: *Physics Procedia* 33 (2012). 2012 International Conference on Medical Physics and Biomedical Engineering (ICMPBE2012), pp. 216–220.

[CED19] Farah Chahlaoui, Mohammed Raiss El-Fenni and Hamza Dahmouni. "Performance Analysis of Load Balancing Mechanisms in SDN Networks". In: *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*. NISS19. Association for Computing Machinery, 2019.

[Cer19] Steve Cerwin. *Radio Propagation and Antennas: A Non-Mathematical Treatment of Radio and Antennas*. AuthorHouse, 2019, pp. 31–35.

[CFZ18] Yunfei Chen, Wei Feng and Gan Zheng. "Optimum Placement of UAV as Relays". In: *IEEE Communications Letters* 22.2 (2018), pp. 248–251.

[CG98] Tsu-Wei Chen and Mario Gerla. "Global State Routing: A New Routing Scheme for Ad-Hoc Wireless Networks". In: *ICC '98. 1998 IEEE International Conference on Communications. Conference Record. Affiliated with SUPERCOMM'98 (Cat. No.98CH36220)*. Vol. 1. 1998, pp. 171–175.

[CH05] Lei Chen and Wendi B. Heinzelman. "QoS-Aware Routing Based on Bandwidth Estimation for Mobile Ad Hoc Networks". In: *IEEE Journal on Selected Areas in Communications* 23.3 (Mar. 2005), pp. 561–572.

[Cho+15] Theodoros Chondrogiannis, Panagiotis Bouros, Johann Gamper and Ulf Leser. "Alternative Routing: K-Shortest Paths with Limited Overlap". In: *23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '15. Association for Computing Machinery, 2015, pp. 1–4.

[Cla+14] Thomas H. Clausen, Christopher Dearlove, Philippe Jacquet and Ulrich Herberg. *The Optimized Link State Routing Protocol Version 2*. RFC 7181. RFC Editor, Apr. 2014, pp. 1–115. URL: https://www.rfc-editor.org/info/rfc7181.

[CM07] Antonio Capone and Fabio Martignon. "A Multi-Commodity Flow Model for Optimal Routing in Wireless MESH Networks". In: *J. Networks* 2 (2007), pp. 1–5.

[CML99] Derya H. Cansever, Arnold M. Michelson and Allen H. Levesque. "Quality of Service Support in Mobile ad-hoc IP Networks". In: *MILCOM 1999. IEEE Military Communications. Conference Proceedings (Cat. No.99CH36341)*. Vol. 1. 1999, pp. 30–34.

[CMP07] Pietro Ciciriello, Luca Mottola and Gian Pietro Picco. "Efficient Routing from Multiple Sources to Multiple Sinks in Wireless Sensor Networks". In: *Wireless Sensor Networks*. Ed. by Koen Langendoen and Thiemo Voigt. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 34–50.

[COC10] Miguel Camelo, Carlos Omaña and Harold Castro. "QoS Routing Algorithm Based on Multi-Objective Optimization for Wireless Mesh Networks". In: *2010 IEEE Latin-American Conference on Communications*. 2010, pp. 1–6.

[Cor+09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009, pp. 658–662.

[Cro+97] Brian P. Crow, Indra Widjaja, Jeong G. Kim and Prescott T. Sakai. "IEEE 802.11 Wireless Local Area Networks". In: *IEEE Communications Magazine* 35.9 (1997), pp. 116–126.

[CS21] Juliusz Chroboczek and David Schinazi. *The Babel Routing Protocol*. RFC 8966. RFC Editor, Jan. 2021, pp. 1–54. URL: https://www.rfc-editor.org/info/rfc8966.

[DAM14] Alejandro De Gante, Mohamed Aslan and Ashraf Matrawy. "Smart Wireless Sensor Network Management Based on Software-Defined Networking". In: *2014 27th Biennial Symposium on Communications (QBSC)*. June 2014, pp. 71–75.

[DB21] Anne Delaporte and Kalvin Bahia. *The State of Mobile Internet Connectivity 2021*. Tech. rep. Global System for Mobile Communications, 2021.

[Des09] Emmanuel Desurvire. "Gaussian channel and Shannon–Hartley theorem". In: *Classical and Quantum Information Theory: An Introduction for the Telecom Scientist*. Cambridge University Press, 2009, pp. 264–282.

[Dij59] Edsger W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numer. Math.* 1.1 (Dec. 1959), pp. 269–271.

[DKB11] Peter Dely, Andreas Kassler and Nico Bayer. "OpenFlow for Wireless Mesh Networks". In: *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*. 2011, pp. 1–6.

[DKS16] Mayank Dixit, Rajesh Kumar and Anil Kumar Sagar. "VANET: Architectures, Research Issues, Routing Protocols, and its Applications". In: *2016 International Conference on Computing, Communication and Automation (ICCCA)*. 2016, pp. 555–561.

[DLV04] Jing Deng, Ben Liang and P.K. Varshney. "Tuning the Carrier Sensing Range of IEEE 802.11 MAC". In: *IEEE Global Telecommunications Conference, 2004. GLOBECOM '04.* Vol. 5. 2004, pp. 2987–2991.

[DMC96] Marco Dorigo, Vittorio Maniezzo and Alberto Colorni. "Ant System: Optimization by a Colony of Cooperating Agents". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996), pp. 29–41.

[DMS18] Ayush Dusia, Vinod K. Mishra and Adarshpal S. Sethi. "Control Communication in SDN-based Dynamic Multi-hop Wireless Infrastructure-less Networks". In: *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS).* 2018, pp. 1–6.

[DPB03] Samir R. Das, Charles E. Perkins and Elizabeth M. Belding-Royer. *Ad hoc On-Demand Distance Vector (AODV) Routing.* RFC 3561. RFC Editor, July 2003, pp. 1–37. URL: https://www.rfc-editor.org/info/rfc3561.

[DRS19] Ayush Dusia, Ram Ramanathan and Adarshpal S. Sethi. "CORR: Centralized Opportunistic Reactive Routing for Mobile Multi-Hop Wireless Networks". In: *2019 28th International Conference on Computer Communication and Networks (ICCCN).* 2019, pp. 1–6.

[DWJ15] Jerzy Domżał, Robert Wójcik and Andrzej Jajszczyk. *Guide to Flow-Aware Networking: Quality-of-Service Architectures and Techniques for Traffic Management.* Switzerland: Springer International Publishing, 2015.

[DZ10] Agrawal Dharma and Qing-An Zeng. *Introduction to Wireless and Mobile Systems.* Cengage Learning, 2010.

[Epp98] David Eppstein. "Finding the k Shortest Paths". In: *SIAM J. Comput.* 28.2 (Jan. 1998), pp. 652–673.

[ESM07] El-Sayed M. El-Alfy, Shokri Selim and Syed Mujahid. "Solving the Minimum-Cost Constrained Multipath Routing with Load Balancing in MPLS Networks Using an Evolutionary Method". In: *2007 IEEE Congress on Evolutionary Computation.* 2007, pp. 4433–4438.

[FAG95] David D. Falconer, Fumiyuki Adachi and Björn Gudmundson. "Time Division Multiple Access Methods for Wireless Personal Communications". In: *IEEE Communications Magazine* 33.1 (1995), pp. 50–57.

[FBB18] Noel Farrugia, Johann A. Briffa and Victor Buttigieg. "An Evolutionary Multipath Routing Algorithm using SDN". In: *2018 9th International Conference on the Network of the Future (NOF)*. 2018, pp. 1–8.

[Fen97] Bill Fenner. *Internet Group Management Protocol, Version 2*. RFC 2236. RFC Editor, Nov. 1997, pp. 1–24. URL: https://www.rfc-editor.org/info/rfc2236.

[FF04] Lester R. Ford and Delbert R. Fulkerson. "A Suggested Computation for Maximal Multi-Commodity Network Flows". In: *Management Science* 50.12_supplement (2004), pp. 97–101.

[FG75] Stanley C. Fralick and James C. Garrett. "Technological Considerations for Packet Radio Networks". In: *Proceedings of the May 19-22, 1975, National Computer Conference and Exposition*. AFIPS '75. Association for Computing Machinery, 1975, pp. 233–243.

[FKQ19] Martin Fleury, Dimitris Kanellopoulos and Nadia N. Qadri. "Video Streaming over MANETs: An Overview of Techniques". In: *Multimedia Tools Appl.* 78.16 (Aug. 2019), pp. 23749–23782.

[Fod+12] Gábor Fodor, Erik Dahlman, Gunnar Mildh, Stefan Parkvall, Norbert Reider, György Miklós and Zoltán Turányi. "Design Aspects of Network Assisted Device-to-Device Communications". In: *IEEE Communications Magazine* 50.3 (2012), pp. 170–177.

[Fot+16] Hossein Fotouhi, Maryam Vahabi, Apala Ray and Mats Björkman. "SDN-TAP: An SDN-based Traffic Aware Protocol for Wireless Sensor Networks". In: *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*. 2016, pp. 1–6.

[FSR06] Liping Fu, Dihua Sun and L.R. Rilett. "Heuristic shortest path algorithms for transportation applications: State of the art". In: *Computers & Operations Research* 33.11 (2006). Part Special Issue: Operations Research and Data Mining, pp. 3324–3343.

[Gal+15] Laura Galluccio, Sebastiano Milardo, Giacomo Morabito and Sergio Palazzo. "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks". In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. 2015, pp. 513–521.

[GGP05] Carles Gomez, David Garcia and J. Paradells. "Improving Performance of a Real Ad-hoc Network by Tuning OLSR Parameters". In: *10th IEEE Symposium on Computers and Communications (ISCC'05)*. 2005, pp. 16–21.

[GHX10] Lifang Guo, Khaled Harfoush and Huimin Xu. "Distribution of the Node Degree in MANETs". In: *2010 Fourth International Conference on Next Generation Mobile Applications, Services and Technologies*. 2010, pp. 162–167.

[Gio02] Silvia Giordano. "Mobile Ad Hoc Networks". In: *Handbook of Wireless Networks and Mobile Computing*. Second Edition. Wiley Series on Parallel and Distributed Computing, 2002, pp. 325–343.

[GJV16] Lav Gupta, Raj Jain and Gabor Vaszkun. "Survey of Important Issues in UAV Communication Networks". In: *Commun. Surveys Tuts.* 18.2 (Apr. 2016), pp. 1123–1152.

[GM07] Zhihao Guo and Behnam Malakooti. "Predictive Delay Metric for OLSR Using Neural Networks". In: *Proceedings of the 3rd International Conference on Wireless Internet*. WICON '07. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, pp. 1–9.

[GM78] Israel Gitman and Daniel Minoli. "ON CONNECTIVITY IN MOBILE PACKET RADIO NETWORKS". In: *28th IEEE Vehicular Technology Conference*. Vol. 28. 1978, pp. 105–109.

[Goo91] David J. Goodman. "Trends in Cellular and Cordless Communications". In: *IEEE Communications Magazine* 29 (1991), pp. 31–40.

[GS99] J.J. Garcia-Luna-Aceves and Marcelo Spohn. "Source-Tree Routing in Wireless Networks". In: *Proceedings. Seventh International Conference on Network Protocols*. 1999, pp. 273–282.

[Gug+18] Tobias Guggemos, Klement Streit, Markus Knüpfer and Nils gentschen Felde. "No Cookies, just CAKE: CRT based Key Hierarchy for Efficient Key Management in Dynamic Groups". In: *13th International Conference for Internet Technology and Secured Transactions (ICITST)*. 2018, pp. 1–8.

[Gui+16] Antonio Guillen-Perez, Ramon Sanchez-Iborra, Maria-Dolores Cano, Juan Carlos Sanchez-Aarnoutse and Joan Garcia-Haro. "WIFI NETWORKS ON DRONES". In: *2016 ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT)*. 2016, pp. 1–8.

[Gut04] Ron Gutman. "Reach-based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks". In: *in Proceedings 6th Workshop on Algorithm Engineering and Experiments (ALENEX)*. Vol. 20. 2004, pp. 100–111.

[Hag+22] Klement Hagenhoff, Maximilian Tränkler, Corinna Schmitt and Gabi Dreo Rodosek. "RTC: Route to Controller Algorithm Providing SDN Capabilities in MANETs". In: *MILCOM 2022 Track 2 - Networking Protocols and Performance (MILCOM 2022 Track 2)*. Rockville, USA, Nov. 2022.

[Hal+15] Evangelos Haleplidis, Kostas Pentikousis, Spyros Denazis, Jamal Hadi Salim, David Meyer and Odysseas Koufopavlou. *Software-Defined Networking: Layers and Architecture Terminology*. RFC 7426. RFC Editor, Jan. 2015, pp. 1–35. URL: https://www.rfc-editor.org/info/rfc7426.

[Hal60] Paul Halmos. *Naive Set Theory*. Reprinted by Springer-Verlag, Undergraduate Texts in Mathematics, 1974. Van Nostrand, 1960.

[Hil+06] Moritz Hilger, Ekkehard Köhler, Rolf H. Möhring and Heiko Schilling. "Fast Point-to-Point Shortest Path Computations with Arc-Flags". In: *The Shortest Path Problem*. 2006.

[HL02] Yu-Ching Hsu and Ying-Dar Lin. "Base-Centric Routing Protocol for Multihop Cellular Networks". In: *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE.* Vol. 1. 2002, pp. 158–162.

[HMJ07] Yih-Chun Hu, Dave A. Maltz and David B. Johnson. *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4.* RFC 4728. RFC Editor, Feb. 2007, pp. 1–107. URL: https://www.rfc-editor.org/info/rfc4728.

[HNR68] Peter E. Hart, Nils J. Nilsson and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[Hol92] John H. Holland. *Adaptation in Natural and Artificial Systems.* Second Edition. Ann Arbor, MI: Complex Adaptive Systems, 1992.

[Hop00] Christian Hopps. *Analysis of an Equal-Cost Multi-Path Algorithm.* RFC 2992. RFC Editor, Nov. 2000, pp. 1–8. URL: https://www.rfc-editor.org/info/rfc2992.

[HP59] Walter Hoffman and Richard Pavley. "A Method for the Solution of the Nth Best Path Problem". In: *J. ACM* 6.4 (Oct. 1959), pp. 506–514.

[HSS13] Shabana Habib, Somaila Saleem and Khawaja Muhammad Saqib. "Review on MANET Routing Protocols and Challenges". In: *2013 IEEE Student Conference on Research and Developement.* 2013, pp. 529–533.

[HST16] Jörg Hähner, Klement Streit and Sven Tomforde. "Cellular Traffic Offloading Through Network-Assisted Ad-Hoc Routing in Cellular Networks". In: *2016 IEEE Symposium on Computers and Communication (ISCC).* 2016, pp. 469–476.

[HT00] Christian Hopps and Dave Thaler. *Multipath Issues in Unicast and Multicast Next-Hop Selection.* RFC 2991. RFC Editor, Nov. 2000, pp. 1–9. URL: https://www.rfc-editor.org/info/rfc2991.

[HT07] Lajos Hanzo and Rahim Tafazolli. "A SURVEY OF QOS ROUTING SOLUTIONS FOR MOBILE AD HOC NETWORKS". In: *IEEE Communications Surveys & Tutorials* 9.2 (2007), pp. 50–70.

[HT18] Sabastien Henri and Patrick Thiran. “Optimal Number of Paths with Multipath Routing in Hybrid Networks”. In: *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*. 2018, pp. 1–7.

[HVD22] Klement Hagenhoff, Eike Viehmann and Gabi Dreo Rodosek. “Time-sensitive Multi-Flow Routing in Highly Utilized MANETs”. In: *18th International Conference on Network and Service Management (CNSM)*. Thessaloniki, Greece, Oct. 2022.

[HVD24] Klement Hagenhoff, Eike Viehmann and Gabi Dreo Rodosek. “Eliminating Bottlenecks in MANETs”. In: *NOMS 2024-2024 IEEE/IFIP Network Operations and Management Symposium*. 2024, pp. 1–5.

[HXG02] Xiaoyan Hong, Kaixinm Xu and Mario Gerla. “Scalable Routing Protocols for Mobile Ad Hoc Networks”. In: *IEEE Network* 16.4 (2002), pp. 11–21.

[HYM16] Samira Hayat, Evşen Yanmaz and Raheeb Muzaffar. “Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint”. In: *IEEE Communications Surveys & Tutorials* 18.4 (2016), pp. 2624–2661.

[IEEE802.11] *IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. Tech. rep. 2021, pp. 1–4379.

[IEEE802.15] *IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs): Amendment 1: Add Alternate PHYs*. Tech. rep. 2007, pp. 1–210.

[IH10] Teerawat Issariyakul and Ekram Hossain. *Introduction to Network Simulator NS2*. 1st. Springer Publishing Company, Incorporated, 2010.

[Jac12] Paul Jaccard. “The Distribution of the Flora in the Alpine Zone”. In: *The New Phytologist* 11.2 (1912), pp. 37–50.

[Jah+18] Yasmin Jahir, Mohammed Atiquzzaman, Hazem Refai, Anirudh Paranjothi and Peter Lopresti. "Routing Protocols and Architecture for Disaster Area Network: A Survey". In: *Ad Hoc Networks* 82 (Aug. 2018).

[Jar+14] Michael Jarschel, Thomas Zinner, Tobias Hossfeld, Phuoc Tran-Gia and Wolfgang Kellerer. "Interfaces, Attributes, and Use Cases: A Compass for SDN". In: *IEEE Communications Magazine* 52.6 (2014), pp. 210–217.

[Jia+14] Jehn-Ruey Jiang, Hsin-Wen Huang, Ji-Hau Liao and Szu-Yuan Chen. "Extending Dijkstra's Shortest Path Algorithm for Software Defined Networking". In: *The 16th Asia-Pacific Network Operations and Management Symposium.* 2014, pp. 1–4.

[JIN14] Waheb A. Jabbar, Mohamod Ismail and Rosdiadee Nordin. "MBA-OLSR: A Multipath Battery Aware Routing Protocol for MANETs". In: *2014 5th International Conference on Intelligent Systems, Modelling and Simulation.* 2014, pp. 630–635.

[KA06] Vinay Kolar and Nael B. Abu-Ghazaleh. "A Multi-Commodity Flow Approach for Globally Aware Routing in Multi-Hop Wireless Networks". In: *Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM06).* 2006, pp. 10–317.

[Kaf+22] Dimitrios Kafetzis, Spyridon Vassilaras, Georgios Vardoulias and Iordanis Koutsopoulos. "Software-Defined Networking Meets Software-Defined Radio in Mobile ad hoc Networks: State of the Art and Future Directions". In: *IEEE Access* 10 (2022), pp. 9989–10014.

[Kah+78] Robert E. Kahn, Steven A. Gronemeyer, Jerry Burchfiel and Roland C. Kunzelman. "Advances in Packet Radio Technology". In: *Proceedings of the IEEE* 66.11 (1978), pp. 1468–1496.

[Kat19] Naoto Katayama. "A combined fast greedy heuristic for the capacitated multicommodity network design problem". In: *Journal of the Operational Research Society* 70.11 (2019), pp. 1983–1996.

[Ken78] Jeff L. Kennington. "A Survey of Linear Cost Multicommodity Network Flows". In: *Operations Research* 26.2 (1978), pp. 209–236.

[KG10] Suchi Meena Kumari and N Geethanjali. "A Survey on Shortest Path Routing Algorithms for Public Transport Travel". In: *Global Journal of Computer Science and Technology* 9.5 (2010), pp. 73–76.

[KGL01] Manthos Kazantzidis, Mario Gerla and Sung-Ju Lee. "Permissible Throughput Network Feedback for Adaptive Multimedia in AODV MANETs". In: *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*. Vol. 5. 2001, pp. 1352–13565.

[Khe+22] Mehdi Kherbache, Otabek Sobirov, Moufida Maimour, Eric Rondeau and Abderrezak Benyahia. "Reinforcement Learning TDMA-Based MAC Scheduling in the Industrial Internet of Things: A Survey". In: *IFAC-PapersOnLine* 55.8 (2022). 6th IFAC Symposium on Telematics Applications TA 2022, pp. 83–88.

[KHS18] Ahmed Kadhim, Seyed Amin Hosseini Seno and R.A. Shihab. "ROUTING PROTOCOL FOR SDN-CLUSTER BASED MANET". In: *Journal of Theoretical and Applied Information Technology* 96 (Aug. 2018), pp. 5398–5412.

[Kim+05] Byung-Seo Kim, Yuguang Fang, T.F. Wong and Younggoo Kwon. "Throughput Enhancement Through Dynamic Fragmentation in Wireless LANs". In: *IEEE Transactions on Vehicular Technology* 54.4 (2005), pp. 1415–1425.

[KIM82] Naoki Katoh, Toshihide Ibaraki and H. Mine. "An efficient algorithm for K shortest simple paths". In: *Networks* 12.4 (1982), pp. 411–427. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230120406`.

[KK00] 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, NY, USA: Association for Computing Machinery, 2000, pp. 243–254.

[KK18] Vadivelu Kalpana and S. Karthik. "Bandwidth Constrained Priority Based Routing Algorithm for Improving the Quality of Service in Mobile Ad hoc Networks". In: *2018 International Conference on Soft-computing and Network Security (ICSNS)*. 2018, pp. 1–8.

[KLG14] Ian Ku, You Lu and Mario Gerla. "Software-Defined Mobile Cloud: Architecture, Services and Use Cases". In: *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*. 2014, pp. 1–6.

[KM16] Sumit Kumar and Shabana Mehfuz. "Intelligent probabilistic broadcasting in mobile ad hoc network: a PSO approach". In: *Journal of Reliable Intelligent Environments* 2.2 (July 2016), pp. 107–115.

[Koz92] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, 1992.

[KQK19] Muhammad Asghar Khan, Ijaz Mansoor Qureshi and Fahimullah Khanzada. "A Hybrid Communication Scheme for Efficient and Low-Cost Deployment of Future Flying Ad-Hoc Network (FANET)". In: *Drones* 3.1 (2019).

[Kre+15] Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Veríssimo, Christian Esteve Rothenberg, Siamak Azodolmolky and Steve Uhlig. "Software-Defined Networking: A Comprehensive Survey". In: *Proceedings of the IEEE* 103.1 (2015), pp. 14–76.

[Kuk18] Alexander Kukushkin. "Global System Mobile, GSM, 2G". In: *Introduction to Mobile Network Engineering: GSM, 3G-WCDMA, LTE and the Road to 5G*. 2018, pp. 59–102.

[KWT16] Sahrish Khan, Abdul Wahid and Sadaf Tanvir. "Comparative Study of Routing Strategies in Software Defined Networking". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. SAC '16. Association for Computing Machinery, 2016, pp. 696–702.

[KY17] Padmavathi Kora and Priyanka Yadlapalli. "CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW". In: *International Journal of Computer Applications* 162.10 (2017).

[LAM12] Jan Leduc, Markus Antweiler and Torleiv Maseng. "Spectrum Issues of NATO Narrowband Waveform: On the Spectral Efficiency of CPM-Modulation with small Modulation Indices". In: *2012 Military Communications and Information Systems Conference (MCC)*. 2012, pp. 1–5.

[Lan+08] William B. Langdon, Riccardo Poli, Nicholas F. McPhee and John R. Koza. "Genetic Programming: An Introduction and Tutorial, with a Survey of Techniques and Applications". In: *Computational Intelligence: A Compendium*. Ed. by John Fulcher and L. C. Jain. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 927–1028.

[Law72] Eugene L. Lawler. "A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem". In: *Management Science* 18.7 (1972), pp. 401–405.

[LBF16] Mohamed Labraoui, Michael M. Boc and Anne Fladenmuller. "Software Defined Networking-Assisted Routing in Wireless Mesh Networks". In: *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*. Sept. 2016, pp. 377–382.

[LDP06] Stephane Lohier, Yacine Ghamri Doudane and Guy Pujolle. "Link Available Bandwidth Monitoring for QoS Routing with AODV in Ad Hoc Networks". In: *IFIP/IEEE International Conference on Management of Multimedia Networks and Services*. Springer. 2006, pp. 37–48.

[Leb+17] Andrei Lebedev, JooYoung Lee, Victor Rivera and Manuel Mazzara. "Link Prediction Using Top-k Shortest Distances". In: *Data Analytics*. Cham: Springer International Publishing, 2017, pp. 101–105.

[Lei+12] Lei Lei, Zhangdui Zhong, Chuang Lin and Xuemin Shen. "Operator controlled device-to-device communications in LTE-advanced networks". In: *IEEE Wireless Communications* 19.3 (2012), pp. 96–104.

[LH00] Ying-Dar Lin and Yu-Ching Hsu. "Multihop cellular: A New Architecture for Wireless Communications". In: *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*. Vol. 3. 2000, pp. 1273–1282.

[Li+02] H. Li, Matthias Lott, Martin Weckerle, Wolfgang Zirwas and E. Schulz. "MULTIHOP COMMUNICATIONS in FUTURE MOBILE RADIO NETWORKS". In: *The 13th IEEE Interna-*

tional Symposium on Personal, Indoor and Mobile Radio Communications. Vol. 1. 2002, pp. 54–58.

[Liu+12] Ling Liu, Lei Zhu, Long Lin and Qihui Wu. "Improvement of AODV Routing Protocol with QoS Support in Wireless Mesh Networks". In: *Physics Procedia* 25 (2012), pp. 1133–1140.

[LK05] Changling Liu and Jörg Kaiser. "A Survey of Mobile Ad Hoc Network Routing Protocols." In: *Open Access Repositorium der Universität Ulm und Technischen Hochschule Ulm.* (2005), p. 37.

[Llo02] Errol L. Lloyd. "Broadcast Scheduling for TDMA in Wireless Multihop Networks". In: *Handbook of Wireless Networks and Mobile Computing*. USA: John Wiley & Sons, Inc., 2002, pp. 347–370.

[LMO16] Jesús Loo, Jonathan Mauri and Jaime Ortiz. *Mobile Ad Hoc Networks*. Ed. by Jonathan Loo, Jaime Lloret Mauri and Jesus Hamilton Ortiz. Boca Raton, FL: CRC Press, Apr. 2016, pp. 22–35.

[LP02] William Langdon and Riccardo Poli. *Foundations of Genetic Programming*. Jan. 2002.

[LSC08] Xue Jun Li, Boon-Chong Seet and Peter Han Joo Chong. "Multihop cellular networks: Technology and economics". In: *Computer Networks* 52.9 (2008), pp. 1825–1837.

[Luo+03] Haiyun Luo, Ramachandran Ramjee, Prasun Sinha, Li Li and Songwu Lu. "UCAN: A Unified Cellular and Ad-Hoc Network Architecture". In: *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*. MobiCom '03. Association for Computing Machinery, 2003, pp. 353–367.

[LYC03] Hui Li, Dan Yu and Hui Chen. "New Approach to Multihop - Cellular Based Multihop Network". In: *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003.* Vol. 2. 2003, pp. 1629–1633.

[Mag+15] Naércio Magaia, Nuno Horta, Rui Neves, Paulo Rogério Pereira and Miguel Correia. "A multi-objective routing algorithm for Wireless Multimedia Sensor Networks". In: *Applied Soft Computing* 30 (2015), pp. 104–112.

[Mar+17] Kelvin Marcus, Christoph Barz, Jonathan Kirchhoff, Henning Rogge, Jan Nilsson, Ronald in 't Velt, Niranjan Suri, Anders Hansson, Ulf Sterner, Mariann Hauge, King Lee, Arjen Holtzer, Boyd Buchin, Markus Peuhkuri and Levent Misirlioglu. "Evaluation of the Scalability of OLSRv2 in an Emulated Realistic Military Scenario". In: *2017 International Conference on Military Communications and Information Systems (ICMCIS)*. 2017, pp. 1–8.

[McK+08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker and Jonathan Turner. "OpenFlow: Enabling Innovation in Campus Networks". In: *SIGCOMM Comput. Commun. Rev.* 38.2 (Mar. 2008), pp. 69–74.

[MD05] Mahesh K. Marina and Samir R. Das. "Routing in Mobile Ad Hoc Networks". In: *Ad Hoc Networks: Technologies and Protocols*. Ed. by Prasant Mohapatra and Srikanth V. Krishnamurthy. Boston, MA: Springer US, 2005, pp. 63–90.

[MDS18] Vinod K. Mishra, Ayush. Dusia and Adarsh. Sethi. *Routing in Software-Defined Mobile Ad hoc Networks (SD-MANET)*. Tech. rep. US Army Research Laboratory Aberdeen Proving Ground United States, June 2018, pp. 1–34.

[Mic+18] Matias Micheletto, Vinicius Petrucci, Rodrigo Santos, Javier Orozco, Daniel Mosse, Sergio F. Ochoa and Roc Meseguer. "Flying Real-Time Network to Coordinate Disaster Relief Activities in Urban Areas". In: *Sensors* 18.5 (2018).

[Mit09] John E. Mitchell. "Integer programming: branch and cut algorithmsInteger Programming: Branch and Cut Algorithms". In: *Encyclopedia of Optimization*. Boston, MA: Springer US, 2009, pp. 1643–1650.

[MK17] Oliver Michel and Eric Keller. "SDN in Wide-Area Networks: A Survey". In: *2017 Fourth International Conference on Software Defined Systems (SDS)*. 2017, pp. 37–42.

[MKL15] Miriam S. Marwick, Corinne Ms Kramer and Evan J. Laprade. "Analysis of Soldier Radio Waveform Performance in Operational Test". In: *Defense Technical Information Archive*. 2015, pp. 1–92.

[Mon+11] Álvaro Monares, Sergio F. Ochoa, José A. Pino, Valeria Herskovic, Juan Rodriguez-Covili and Andrés Neyem. "Mobile computing in urban emergency situations: Improving the support to firefighters in the field". In: *Expert Systems with Applications* 38.2 (2011). Intelligent Collaboration and Design, pp. 1255–1267.

[Moy98] John Moy. *OSPF Version 2*. RFC 2328. RFC Editor, Apr. 1998, pp. 1–244. URL: https://www.rfc-editor.org/info/rfc2328.

[MPG15] Naércio Magaia, Paulo Pereira and Antonio Grilo. "High Throughput Low Coupling Multipath Routing for Wireless Multimedia Sensor Networks". In: *Ad Hoc & Sensor Wireless Networks* 25 (Jan. 2015), pp. 165–198.

[MR19] Abdul Manazir and Khalid Raza. "Recent Developments in Cartesian Genetic Programming and Its Variants". In: *ACM Comput. Surv.* 51.6 (Jan. 2019).

[MW15] Yasushi Matsumoto and Kia Wiklundh. "Evaluation of Impact on Digital Radio Systems by Measuring Amplitude Probability Distribution of Interfering Noise". In: *IEICE Transactions on Communications* E98.B.7 (2015), pp. 1143–1155.

[MWH01] Martin Mauve, Jörg Widmer and Hannes Hartenstein. "A Survey on Position-Based Routing in Mobile Ad Hoc Networks". In: *IEEE Network* 15.6 (2001), pp. 30–39.

[NAP05] Lars Relund Nielsen, Kim Allan Andersen and Daniele Pretolani. "Finding the K shortest hyperpaths". In: *Computers & Operations Research* 32.6 (2005), pp. 1477–1497.

[Nee11] Richard Van Nee. "Breaking the Gigabit-per-second barrier with 802.11AC". In: *IEEE Wireless Communications* 18.2 (2011), pp. 4–4.

[Neu+08] Axel Neumann, Corinna Aichele, Marek Lindner and Simon Wunderlich. "Better Approach to Mobile Ad-hoc Networking (BATMAN)". In: (2008), pp. 1–24. URL: https://datatracker.ietf.org/doc/draft-openmesh-b-a-t-m-a-n/00/.

[Nic22] Stefan. Nickel. *Decision Optimization with IBM ILOG CPLEX Optimization Studio A Hands-On Introduction to Modeling with the Optimization Programming Language (OPL)*. eng. 1st ed. 2022. Graduate Texts in Operations Research. Berlin, Heidelberg: Springer Berlin Heidelberg, 2022.

[NL07] Ping Chung Ng and Soung Chang Liew. "Throughput Analysis of IEEE803.11 Multi-Hop Ad Hoc Networks". In: *IEEE/ACM Transactions on Networking (TON)* 15.2 (Apr. 2007), pp. 309–322.

[Nob+16] Jéferson Nobre, Denis Rosario, Cristiano Both, Eduardo Cerqueira and Mario Gerla. "Toward Software-Defined Battlefield Networking". In: *IEEE Communications Magazine* 54.10 (2016), pp. 152–157.

[NY18] James Nguyen and Wei Yu. "An SDN-Based Approach to Support Dynamic Operations of Multi-Domain Heterogeneous MANETs". In: *19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. 2018, pp. 21–26.

[Ole16] Jorge Olenewa. *Guide to Wireless Communications*. 4th. Boston, MA, USA: Course Technology Press, 2016, pp. 77–79.

[PAR05] Ramesh Chembil Palat, A. Annamalau and Jeffrey R. Reed. "COOPERATIVE RELAYING FOR AD-HOC GROUND NETWORKS USING SWARM UAVS". In: *MILCOM 2005 - 2005 IEEE Military Communications Conference*. Vol. 3. 2005, pp. 1588–1594.

[Pat+21] Vaios Patras, Ioannis Fudos, Kyriakos Koritsoglou and Georgios Tsoumanis. "Revisiting shortest path algorithms for navigation systems". In: *2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*. 2021, pp. 1–5.

[PB01] Charles E. Perkins and Elizabeth M. Belding-Royer. *Quality of Service for Ad hoc On-Demand Distance Vector Routing*. RFC draft-perkins-manet-aodvqos-01. Work in Progress. RFC Editor, Dec. 2001, pp. 1–11. URL: https://datatracker.ietf.org/doc/draft-perkins-manet-aodvqos/01/.

[PGC00] Guangyu Pei, M. Gerla and Tsu-Wei Chen. "Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks". In: *2000 IEEE International Conference on Communications. ICC 2000. Global Convergence Through Communications. Conference Record*. Vol. 1. 2000, pp. 70–74.

[Phe+16] Kévin Phemius, Jawad Seddar, Mathieu Bouet, Hicham Khalifé and Vania Conan. "Bringing SDN to the Edge of Tactical Networks". In: *MILCOM 2016 - 2016 IEEE Military Communications Conference*. 2016, pp. 1047–1052.

[PIT18] Konstantinos Poularakis, George Iosifidis and Leandros Tassiulas. "SDN-Enabled Tactical Ad Hoc Networks: Extending Programmable Control to the Edge". In: *IEEE Communications Magazine* 56.7 (2018), pp. 132–138.

[PLM08] Riccardo Poli, William Langdon and Nicholas Mcphee. *A Field Guide to Genetic Programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, Jan. 2008.

[Pos81] Jon Postel. *Internet Protocol*. RFC 791. RFC Editor, Sept. 1981, pp. 1–51. URL: `https://www.rfc-editor.org/info/rfc791`.

[Pou+17] Konstantinos Poularakis, Qiaofeng Qin, Erich Nahum, Miguel Rio and Leandros Tassiulas. "Bringing SDN to the Mobile Edge". In: *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. 2017, pp. 1–6.

[Pou+19a] Konstantinos Poularakis, Qiaofeng Qin, Kelvin M. Marcus, Kevin S. Chan, Kin K. Leung and Leandros Tassiulas. "Hybrid SDN Control in Mobile Ad Hoc Networks". In: *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2019, pp. 110–114.

[Pou+19b] Konstantinos Poularakis, Qiaofeng Qin, Erich M. Nahum, Miguel Rio and Leandros Tassiulas. "Flexible SDN Control in Tactical Ad Hoc Networks". In: *Ad Hoc Networks* 85 (2019), pp. 71–80.

[Pra13] Laishram Prabhakar. "A Comparative Study of Three TDMA Digital Cellular Mobile Systems (GSM, IS-136 NA-TDMA and PDC) Based On Radio Aspect". In: *International Journal of Advanced Computer Science and Applications* 4 (2013), pp. 139–143.

[PS06] Alan Piszcz and Terence Soule. "A Survey of Mutation Techniques in Genetic Programming". In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. GECCO '06. Seattle, Washington, USA: Association for Computing Machinery, 2006, pp. 951–952.

[Raj20] Alagan Ramasamy Rajeswari. "A Mobile Ad Hoc Network Routing Protocols: A Comparative Study". In: *Recent Trends in Communication Networks*. Ed. by Pinaki Mitra. IntechOpen, 2020. Chap. 13.

[RGM18] Shahbaz Rezaei, Mohammed Gharib and Ali Movaghar. "Throughput Analysis of IEEE 802.11 Multi-Hop Wireless Networks With Routing Consideration: A General Framework". In: *IEEE Transactions on Communications* 66.11 (2018), pp. 5430–5443.

[Rod+18] Nils Rodday, Klement Streit, Gabi Dreo Rodosek and Eiko Pras. "Empirical Study of DSCPand ECNUsage by Application in Internet Traffic Traces". Poster @ CoNEXT. 2018.

[Rod+19a] Nils Rodday, Raphael Labaca Castro, Klement Streit and Gabi Dreo Rodosek. "Evaluating TCP Connection Healthiness". In: *2019 29th International Telecommunication Networks and Applications Conference (ITNAC)*. 2019, pp. 1–4.

[Rod+19b] Nils Rodday, Klement Streit, Gabi Dreo Rodosek and Aiko Pras. "On the Usage of DSCP and ECN Codepoints in Internet Backbone Traffic Traces for IPv4 and IPv6". In: *2019 International Symposium on Networks, Computers and Communications (ISNCC)*. 2019, pp. 1–6.

[Rod06] Dennis Roddy. *Satellite Communications, Fourth Edition*. Professional Engineering. Mcgraw-hill, 2006.

[RT99] Elizabeth M. Royer and Chai-Keong Toh. "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks". In: *IEEE Personal Communications* 6.2 (1999), pp. 46–55.

[Sam20] Ghassan Samara. "Wireless Sensor Network MAC Energy - efficiency Protocols: A Survey". In: *2020 21st International Arab Conference on Information Technology (ACIT)*. 2020, pp. 1–5.

[SB20] Khodakaram Salimifard and Sara Bigharaz. "The multicommodity network flow problem: state of the art classification, applications, and solution methods". In: *Operational Research* (Apr. 2020), pp. 1–47.

[Sch04] Mischa Schwartz. *Mobile Wireless Communications*. USA: Cambridge University Press, 2004, pp. 396–406.

[SD20] Klement Streit and Gabi Dreo Rodosek. "Cetup: Controller-equipped Topology Update Process for Tactical Ad-hoc Networks". In: *Proceedings of the 17th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*. 2020, pp. 57–66.

[Sey05] John S. Seybold. "Communication Systems and the Link Budget". In: *Introduction to RF Propagation*. John Wiley & Sons, Ltd, 2005. Chap. 4, pp. 66–86. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471743690.ch4`.

[SFK03] Yuji Shinano, Tetsuya Fujie and Yuusuke Kounoike. "Effectiveness of Parallelizing the ILOG-CPLEX Mixed Integer Optimizer in the PUBB2 Framework". In: *Euro-Par 2003 Parallel Processing*. Ed. by Harald Kosch, László Böszörményi and Hermann Hellwagner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 451–460.

[SH04] Tim Szigeti and Christina Hattingh. *End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs*. Cisco Press, 2004.

[Shi+06] Kuei-Ping Shih, Chih-Yung Chang, Yen-Da Chen and Tsung-Han Chuang. "Dynamic bandwidth allocation for QoS routing on TDMA-based mobile ad hoc networks". In: *Computer Communications* 29.9 (2006). ICON 2004, pp. 1316–1329.

[Shi+13] Seugwon Shin, Phillip Porras, Vinod Yegneswaran, Martin Fong, Guofei Gu and Mabry Tyson. "FRESCO: Modular Composable Security Services for Software-Defined Networks". In: *Internet Society NDSS*. Feb. 2013, pp. 1–16.

[Shu+20] Demeke Shumeye Lakew, Umar Sa'ad, Nhu-Ngoc Dao, Woongsoo Na and Sungrae Cho. "Routing in Flying Ad Hoc Networks: A Comprehensive Survey". In: *IEEE Communications Surveys & Tutorials* 22.2 (2020), pp. 1071–1120.

[SI21] Rabia Saleh and Lilia Georgieva Idris Skloul. "SDN-based MANET Using Existing OpenFlow Protocol". In: *11. International Conference on Mobile Services, Resources, and Users* 11 (2021), pp. 15–21.

[SMM20] Eva Shayo, Prosper Mafole and Alfred Mwambela. "A survey on time division multiple access scheduling algorithms for industrial networks". In: *SN Applied Sciences* 2 (Dec. 2020), p. 10.

[Spe+99] Lee Spector, William B. Langdon, Una-May OŔeilly and Peter J. Angeline. *Advances in Genetic Programming: Volume 3*. Cambridge, MA, USA: MIT Press, 1999.

[SRR18] Klement Streit, Nils Rodday and Gabi Dreo Rodosek. "AODV-CBR: Capacity-based Path Discovery Algorithm for MANETs with High Utilization". In: *2018 Advances in Wireless and Optical Communications (RTUWO)*. 2018, pp. 234–239.

[SRS01] César A. Santiváñez, Ram Ramanathan and Ioannis Stavrakakis. "Making Link-State Routing Scale for Ad Hoc Networks". In: *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing*. MobiHoc '01. Association for Computing Machinery, 2001, pp. 22–32.

[SSG20] Klement Streit, Corinna Schmitt and Carlo Giannelli. "SDN-Based Regulated Flow Routing in MANETs". In: *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2020, pp. 73–80.

[SSJ07] Szu-Lin Su, Yi-Wen Su and Jing-Yen Jung. "A Novel QoS Admission Control for Ad Hoc Networks". In: *2007 IEEE Wireless Communications and Networking Conference*. 2007, pp. 4193–4197.

[Str+19a] Klement Streit, Raphael Labaca Castro, Nils Rodday and Gabi Dreo Rodosek. "Topology Update Algorithm for Wireless Networks". In: *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*. 2019, pp. 184–185.

[Str+19b] Klement Streit, Nils Rodday, Florian Steuber, Corinna Schmitt and Gabi Dreo Rodosek. "Wireless SDN for Highly Utilized MANETs". In: *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. 2019, pp. 226–234.

[Str+20] Klement Streit, Eike Viehmann, Florian Steuber and Gabi Dreo Rodosek. "Improving Routing with Up-to-date and Full Topology Knowledge in MANETs". In: *2020 Military Communications and Information Systems Conference (MilCIS)*. 2020, pp. 1–8.

[Sur+16] Niranjan Suri, Anders Hansson, Jan Nilsson, Piotr Lubkowski, Kelvin Marcus, Mariann Hauge, King Lee, Boyd Buchin, Levent Mısırhoglu and Markus Peuhkuri. "A Realistic Military Scenario and Emulation Environment for Experimenting with Tactical Communications and Heterogeneous Networks". In: *2016 International Conference on Military Communications and Information Systems (ICMCIS)*. 2016, pp. 1–8.

[SW17] Farrukh S. Shaikh and Roland Wismüller. "Centralized Adaptive Routing in Multihop Cellular D2D Communications". In: *2017 2nd International Conference on Computer and Communication Systems (ICCCS)*. July 2017, pp. 158–162.

[Swe+10] Ahmed M. Sweedy, Abdellatif I. M. Semeia, Sanaa Y. Sayed and A. H. Konber. "The Effect of Frame Length, Fragmentation and RTS/CTS mechanism on IEEE 802.11 MAC performance". In: *2010 10th International Conference on Intelligent Systems Design and Applications*. 2010, pp. 1338–1344.

[Szy12] Ted H. Szymanski. “Achieving Minimum-Routing-Cost Maximum-Flows in Infrastructure Wireless Mesh Networks”. In: *2012 IEEE Wireless Communications and Networking Conference (WCNC)*. 2012, pp. 2031–2036.

[TMB01] Mineo Takai, Jay Martin and Rajive Bagrodia. “Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks”. In: *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing*. MobiHoc ’01. Long Beach, CA, USA: Association for Computing Machinery, 2001, pp. 87–94.

[Toh96] Chai-Keong Toh. *Wireless ATM and Ad-Hoc Networks: Protocols and Architectures*. USA: Kluwer Academic Publishers, 1996.

[TP21] Thananop Thongthavorn and Sumet Prabhavat. “Integration of MANET with In-Band SDN Controller”. In: *Recent Advances in Information and Communication Technology 2021*. Ed. by Phayung Meesad, Dr. Sunantha Sodsee, Watchareewan Jitsakul and Sakchai Tangwannawit. Cham: Springer International Publishing, 2021, pp. 322–331.

[TSR15] Shobba Tyagi, Ajay V. Singh and Q. P. Rana. “A QoS Aware Variant of AODV with Probabilistic Broadcast of Control Packets in MANETs”. In: *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*. Sept. 2015, pp. 1–6.

[TW11] Andrew S. Tanenbaum and David Wetherall. *Computer Networks*. 5th ed. Boston: Prentice Hall, 2011.

[Vad+16] Satish Vadlamani, Burak Eksioglu, Hugh Medal and Apurba Nandi. “Jamming attacks on wireless networks: A taxonomic survey”. In: *International Journal of Production Economics* 172 (2016), pp. 76–94.

[Var10] Andras Varga. “OMNeT++”. In: *Modeling and Tools for Network Simulation*. Ed. by Klaus Wehrle, Mesut Güneş and James Gross. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 35–59.

[VDC16] Kumar Mishra Vinod, Kumar Verma Dinesh and Williams Christopher. *Leveraging SDN for Cyber Situational Awareness in Coalition Tactical Networks*. Tech. rep. Cyber Defence Situation Awareness, Oct. 2016, pp. 1–10.

[Wal06] Krzysztof Walkowiak. "New Algorithms for the Unsplittable Flow Problem". In: *Proceedings of the 2006 International Conference on Computational Science and Its Applications - Volume Part II*. ICCSA'06. Springer-Verlag, 2006, pp. 1101–1110.

[WJ22] Jingjing Wang and Chunxiao Jiang. "Introduction of Flying Ad Hoc Networks". In: *Flying Ad Hoc Networks: Cooperative Networking and Resource Allocation*. Singapore: Springer Singapore, 2022, pp. 2–6.

[WS10] Pattana Wannawilai and Chanboon Sathitwiriyawong. "AODV with Sufficient Intermediary Bandwidth Aware". In: *ECTI-CON2010: The 2010 ECTI International Confernce on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*. May 2010, pp. 61–65.

[XGB02] Kaixin Xu, Mario Gerla and Sang Bae. "How Effective is the IEEE 802.11 RTS/CTS Handshake in Ad Hoc Networks". In: *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*. Vol. 1. 2002, pp. 72–76.

[XZ15] Cao Xuelin and Song Zuxun. "An Overview of Slot Assignment (SA) for TDMA". In: *2015 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*. 2015, pp. 1–5.

[Y+15] Adarbah Haitham Y., Shakeel Ahmad, Bassel Arafeh and Alistair Duffy. "Efficient Broadcasting for Route Discovery in Mobile Ad-hoc Networks". In: *2015 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*. July 2015, pp. 1–7.

[YA06] Hujun Yin and Siavash Alamouti. "OFDMA: A Broadband Wireless Access Technology". In: *2006 IEEE Sarnoff Symposium*. 2006, pp. 1–4.

[Ye+05] Fan Ye, Gary Zhong, Songwu Lu and Lixia Zhang. "GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks". In: *Wireless Networks* 11 (May 2005), pp. 285–298.

[Yen71] Jin Y. Yen. “Finding the K Shortest Loopless Paths in a Network”. In: *Management Science* 17.11 (1971), pp. 712–716.

[YGW14] Hu Yuan, Weisi Guo and Siyi Wang. “Emergency Route Selection for D2D Cellular Communications During an Urban Terrorist Attack”. In: *2014 IEEE International Conference on Communications Workshops (ICC)*. 2014, pp. 237–242.

[Yin+14] Wang Yinghe, Chang Lin, Yu Kai, Tan Chong and Bu Zhiyong. “A DYNAMIC SHORTEST PATH WEIGHTED ROUTING MECHANISM FOR MANET BASED ON COMPLEX NETWORK THEORY”. In: *2014 4th IEEE International Conference on Network Infrastructure and Digital Content*. 2014, pp. 17–21.

[YK18] Yun-Shuai Yu and Chih-Heng Ke. “Genetic algorithm-based routing method for enhanced video delivery over software defined networks”. In: *International Journal of Communication Systems* 31 (2018).

[Yoo+15] Changhoon Yoon, Taejune Park, Seungsoo Lee, Heedo Kang, Seungwon Shin and Zonghua Zhang. “Enabling Security Functions with SDN”. In: *Comput. Netw.* 85.C (July 2015), pp. 19–35.

[YQR17] Hans C. Yu, Giorgio Quer and Ramesh Rao. “Wireless SDN Mobile Ad Hoc Network: From Theory to Practice”. In: *2017 IEEE International Conference on Communications (ICC)*. May 2017, pp. 1–7.

[Zab06] Arwa Zabian. “Topology Discovering Mechanism for Power Saving in Ad-hoc Wireless Networks”. In: *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*. Vol. 3. 2006, pp. 910–915.

[Zab07] Arwa Zabian. “Topology Discovering and Power saving Mechanism for Routing in a Tree of Ad-hoc Wireless Networks”. In: *Journal of Computer Science* 3 (Aug. 2007), p. 26.

[ZC09] Weng Zeng and R. L. Church. “Finding shortest paths on real road networks: the case for A*”. In: *International Journal of Geographical Information Science* 23.4 (2009), pp. 531–543.

[ZG05] Yihai Zhang and T.A. Gulliver. "Quality of Service for Ad hoc On-demand Distance Vector Routing". In: *(WiMob'2005), IEEE International Conference on Wireless And Mobile Computing, Networking And Communications, 2005.* Vol. 3. 2005, pp. 192–196.

[Zil+17] Noa Zilberman, Matthew Grosvenor, Diana Andreea Popescu, Neelakandan Manihatty-Bojan, Gianni Antichi, Marcin Wójcik and Andrew W. Moore. "Where Has My Time Gone?" In: *Passive and Active Measurement.* Ed. by Mohamed Ali Kaafar, Steve Uhlig and Johanna Amann. Cham: Springer International Publishing, 2017, pp. 201–214.

[ZW13] Xu Zhen and Yang Wenzhong. "Bandwidth-aware Routing for TDMA-based Mobile Ad Hoc Networks". In: *The International Conference on Information Networking 2013 (ICOIN).* Jan. 2013, pp. 637–642.

Webography

[Adobe] *Recommended bit rates for live streaming | Adobe Developer Connection*. Accessed on: May, 15th 2019. 2019. URL: https://www.adobe.com/devnet/adobe-media-server/articles/dynstream_live/popup.html.

[BWI20] BWI-Redaktion. *Programmplan „TEN“: Weg zur digitalen Integration deutscher und niederländischer Streitkräfte*. Online; accessed 11 August 2022. 2020. URL: https://www.bwi.de/news-blog/news/artikel/programmplan-ten-weg-zur-digitalen-integration-deutscher-und-niederlaendischer-streitkraefte.

[Cisco] *Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper*. Accessed on: May, 15th 2019. 2019. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[Dor19] Frank Dorothee. *Neue VHF/UHF-Funkgeräte für die VJTF*. Online; accessed 11 August 2022. 2019. URL: https://esut.de/2019/07/meldungen/ruestung2/14052/neue-vhf-uhf-funkgeraete-fuer-die-vjtf-l-2023.

[Ericcson] *Mobile data traffic outlook*. Accessed on: December, 28th 2022. 2022. URL: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast.

[Ham20] David Hambling. *Battlefield Augmented Reality Getting Real For U.S. Army*. Online; accessed 11 August 2022. 2020. URL: https://www.forbes.com/sites/davidhambling/2020/05/22/battlefield-augmented-reality-gets-real/#6098e7737d8e.

[Onos] *Onos - open network operating system*. Accessed on: May, 08th 2019. 2019. URL: https://onosproject.org/.

[Wak19] Jane Wakefield. *Hong Kong protesters using Bluetooth Bridgefy app*. Online; accessed 12 August 2022. 2019. URL: https://www.bbc.com/news/technology-49565587.

PUBLICATIONS BY THE AUTHOR

This is a list of publications by the author, sorted in reverse chronological order.

- **Klement Hagenhoff**, Eike Viehmann and Gabi Dreo Rodosek. “Eliminating Bottlenecks in MANETs”. In: *NOMS 2024-2024 IEEE/IFIP Network Operations and Management Symposium*. 2024, pp. 1–5
- **Klement Hagenhoff**, Eike Viehmann and Gabi Dreo Rodosek. “Time-sensitive Multi-Flow Routing in Highly Utilized MANETs”. In: *18th International Conference on Network and Service Management (CNSM)*. Thessaloniki, Greece, Oct. 2022
- **Klement Hagenhoff**, Maximilian Tränkler, Corinna Schmitt and Gabi Dreo Rodosek. “RTC: Route to Controller Algorithm Providing SDN Capabilities in MANETs”. In: *MILCOM 2022 Track 2 - Networking Protocols and Performance (MILCOM 2022 Track 2)*. Rockville, USA, Nov. 2022
- **Klement Streit**, Corinna Schmitt and Carlo Giannelli. “SDN-Based Regulated Flow Routing in MANETs”. In: *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2020, pp. 73–80
- **Klement Streit**, Eike Viehmann, Florian Steuber and Gabi Dreo Rodosek. “Improving Routing with Up-to-date and Full Topology Knowledge in MANETs”. In: *2020 Military Communications and Information Systems Conference (MilCIS)*. 2020, pp. 1–8
- **Klement Streit** and Gabi Dreo Rodosek. “Cetup: Controller-equipped Topology Update Process for Tactical Ad-hoc Networks”. In: *Proceedings of the 17th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*. 2020, pp. 57–66
- **Klement Streit**, Raphael Labaca Castro, Nils Rodday and Gabi Dreo Rodosek. “Topology Update Algorithm for Wireless Networks”. In: *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*. 2019, pp. 184–185
- Nils Rodday, Raphael Labaca Castro, **Klement Streit** and Gabi Dreo Rodosek. “Evaluating TCP Connection Healthiness”. In: *2019 29th International Telecommunication Networks and Applications Conference (ITNAC)*. 2019, pp. 1–4
- **Klement Streit**, Nils Rodday, Florian Steuber, Corinna Schmitt and Gabi Dreo Rodosek. “Wireless SDN for Highly Utilized MANETs”. In: *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. 2019, pp. 226–234
- Nils Rodday, **Klement Streit**, Gabi Dreo Rodosek and Aiko Pras. “On the Usage of DSCP and ECN Codepoints in Internet Backbone Traffic Traces for IPv4 and IPv6”. In: *2019 International Symposium on Networks, Computers and Communications (ISNCC)*. 2019, pp. 1–6

- Nils Rodday, **Klement Streit**, Gabi Dreo Rodosek and Eiko Pras. "Empirical Study of DSCPand ECNUsage by Application in Internet Traffic Traces". Poster @ CoNEXT. 2018
- **Klement Streit**, Nils Rodday and Gabi Dreo Rodosek. "AODV-CBR: Capacity-based Path Discovery Algorithm for MANETs with High Utilization". In: *2018 Advances in Wireless and Optical Communications (RTUWO)*. 2018, pp. 234–239
- Tobias Guggemos, **Klement Streit**, Markus Knüpfer and Nils gentschen Felde. "No Cookies, just CAKE: CRT based Key Hierarchy for Efficient Key Management in Dynamic Groups". In: *13th International Conference for Internet Technology and Secured Transactions (ICITST)*. 2018, pp. 1–8
- Jörg Hähner, **Klement Streit** and Sven Tomforde. "Cellular Traffic Offloading Through Network-Assisted Ad-Hoc Routing in Cellular Networks". In: *2016 IEEE Symposium on Computers and Communication (ISCC)*. 2016, pp. 469–476