Thesis for The Degree of Licentiate of Engineering

Long-Term Stable Communication in Centrally Scheduled Low-Power Wireless Networks

Oliver Harms

Division of Computer and Network Systems
Department of Computer Science & Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden, 2021

Long-Term Stable Communication in Centrally Scheduled Low-Power Wireless Networks

Oliver Harms

Department of Computer Science & Engineering
Division of Computer and Network Systems
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Digitaltryck,
Gothenburg, Sweden 2021.

Abstract

With the emergence of the Internet of Things (IoT), more devices are connected than ever before. Most of these communicate wirelessly, forming Wireless Sensor Networks. In recent years, there has been a shift from personal networks, like Smart Home, to industrial networks. Industrial networks monitor pipelines or handle the communication between robots in factories. These new applications form the Industrial Internet of Things (IIoT). Many industrial applications have high requirements for communication, higher than the requirements of common IoT networks. Communications must stick to hard deadlines to avoid harm, and they must be highly reliable as skipping information is not a viable option when communicating critical information. Moreover, communication has to remain reliable over longer periods of time. As many sensor locations do not offer a power source, the devices have to run on battery and thus have to be power efficient. Current systems offer solutions for some of these requirements. However, they especially lack long-term stable communication that can dynamically adapt to changes in the wireless medium.

In this thesis, we study the problem of stable and reliable communication in centrally scheduled low-power wireless networks. This communication ought to be stable when it can dynamically adapt to changes in the wireless medium while keeping latency at a minimum. We design and investigate approaches to solve the problem of low to high degrees of interference in the wireless medium. We propose three solutions to overcome interference: MASTER with *Sliding Windows* brings dynamic numbers of retransmissions to centrally scheduled low-power wireless networks, OVERTAKE allows to skip nodes affected by interference along the path, and AUTOBAHN combines opportunistic routing and synchronous transmissions with the Time-Slotted Channel Hopping (TSCH) MAC protocol to overcome local wide-band interference with the lowest possible latency. We evaluate our approaches in detail on testbed deployments and provide open-source implementations of the protocols to enable others to build their work upon them.

Keywords

Time-Slotted Channel Hopping (TSCH), Central Scheduling, Routing, Opportunistic Routing, Synchronous Transmissions, Wireless Sensor-Actuator Networks, WSN, (Industrial) Internet of Things, IoT, IEEE 802.15.4

Acknowledgment

A PhD is like a big project. In the beginning, you might have a plan. You might have every step laid out. However, as often in life, planning is great, but you can never prepare for all contingencies. In case you hit a low, you can try to pull yourself back on track, but it helps incredibly to have friends, colleagues, and a supporting supervisor around you. Thus, I would first of all like to thank my supervisor, Olaf Landsiedel. I would like to thank him for taking me on this journey towards a PhD and for all the guidance he has offered me. I want to thank him especially for believing in me even in times I didn't see myself continuing this journey. I hope we will continue having exciting discussions on the next leg of this journey.

I would also like to thank all the colleagues I met during my time at Chalmers and at Kiel University. Without you, this journey would have been much harder. I always enjoyed discussions with you on the hallways or during fika breaks, and I hope you liked them as well. Thank you (in no particular order) Valentin, Patrick, Dimitris, Christos, Beshr, Babis, Georgia, Bastian, Thomas, Karl, Hannah, Carlo, and all the other great people I have met so far during my PhD. Also, a special thanks to Brigitte, Steffi, Gerd, Rebecca, Eva, Agneta, and all administrative staff making daily work at the university much more effortless.

Moreover, I want to thank my best friend Marcel, who was always there for me and always had an open ear. I also want to thank him for his valuable outside perspective on my research as someone without knowledge in computer science.

Finally, I want to thank my parent and my sisters who helped me become the person I am and always were there for me, even if hundreds of kilometers apart. Thank you!

Oliver Harms
Kiel, November 2021

List of Publications

Appended publications

This thesis is based on the following publications:

[A] **O. Harms**, O. Landsiedel "MASTER: Long-Term Stable Routing and Scheduling in Low-Power Wireless Networks"
Proceedings of the 16th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2020.

[B] **O. Harms**, O. Landsiedel "(POSTER) OVERTAKE: Opportunistic Routing and Concurrent Transmissions for TSCH"
Proceedings of the 16th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2020.

[C] **O. Harms**, O. Landsiedel "Opportunistic Routing and Synchronous Transmissions Meet TSCH"
Proceedings of the 46th IEEE Conference on Local Computer Networks (LCN), 2021.

Other publications

The following publications were published during my PhD studies, or are currently in submission/under revision. However, they are not appended to this thesis, due to contents overlapping that of appended publications or contents not related to the thesis.

[a] A. Varshney, **O. Harms**, C. Pérez-Penichet, C. Rohner, F. Hermans, T. Voigt "LoRea: A Backscatter Architecture That Achieves a Long Communication Range"
Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SenSys), 2017.

[b] **O. Harms** "C-TSCH: A Centralized Scheduler for TSCH"
Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN), 2019.
Poster Abstract

[c] **O. Harms**, O. Landsiedel "Competition: Centrally Scheduled Low-Power Wireless Networking for Dependable Data Collection"
Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN), 2019.
Poster Abstract

Contents

List of Figures

List of Tables

1

Introduction

Communication is an essential part of humanity. Without communication, modern civilizations are not imaginable. Through communication, we share information with each other. We use this information to generate meaningful conclusions influencing our further actions. While the information we share can be anything, humans are particularly interested in data. Data are observable units of information like, i.e., temperature, humidity, light (day or night), or answers to potentially existential questions or threats, like whether somewhere is a fire or not. Quantifying this data to be able to share it either needs a human observer with or without tools like a thermometer. While this method of quantifying data and communicating it to other humans is possible, it is somewhat limited in the amount of collectible data.

To overcome limitations, humans build tools helping them to fulfill their tasks. One of these tools is sensors. Sensors are electronic devices that do not need a human to read a value. Instead, a computer can collect the relevant data often with even higher precision and higher reliability. To communicate these sensor data, we can rely on human-to-human communication. However, with the growing amount of data and automation, computers should be able to communicate these data to other computers independent of humans. Fortunately, we have a globally connected network of computers at our hands – the Internet. The Internet allows all connected devices to communicate with each other. However, many sensors are connected to low-power devices (e.g., microcontrollers) not even capable of directly communicating over the Internet.

To nonetheless communicate the data, low-power devices often contain a low-power radio. This radio enables the devices to send the data either to a more powerful device connected to the Internet or another device connected to, e.g., an actuator. Actuator devices control processes, e.g., opening a valve or turning down the heater. Certain sensor data might only be relevant to these devices and thus does not need to be collected. These sensor and actuator devices are often personal devices, like smart light bulbs or thermostats found in the context of Smart Home or devices like smartwatches that communicate with a phone.

All these kinds of communicating low-power devices form networks, the so-called Internet of Things (IoT). IoT devices often operate on batteries.

Therefore, they must not waste energy. The biggest consumer of these devices is commonly neither the microcontroller nor the sensor, but the radio. A radio has a current of several milliamperes (e.g., 6.4 mA [1, p. 58], or 24 mA [2, p. 10]), emptying a coin cell battery (200 mA) in under 32 hours. Therefore, a long lifetime of IoT devices is strongly related to the device's communication, and thus, to the used communication protocols.

Industrial Internet of Things (IIoT)

In recent years, the Internet of Things has gained more and more traction outside of the private context, in industrial settings, or as a part of public infrastructure. There are smart meters that measure the electricity consumption of customers of an electricity provider and communicate this data back to the provider. Intelligent parking sensors notify the owner of the parking if a car has exceeded the maximum parking duration. Manufacturing facilities have humans and machines working next to each other. In such a setting, a robot might have to stop if a human enters a specific area. There, a sensor has to communicate this presence to the robot. Alternatively, in a fully automated factory, a machine has to notify another one that it can take over, or a machine should be able to report problems to an observer. Furthermore, sensors monitoring infrastructure like pipelines need to order maintenance if something unexpected, like a leakage, happens. All of these IoT scenarios have much higher requirements than personal devices, and we commonly refer to them as the Industrial Internet of Things (IIoT).

While in the context of Smart Homes, in the worst case, a sensor reading is not communicated, or turning on the light takes a bit longer than anticipated, this behavior is unacceptable in most industrial contexts. There, systems are often critical, and failed or delayed communication can lead to a robot harming a human or a pipeline's leakage, damaging the environment. Therefore the Industrial Internet of Things has specific requirements regarding reliability and latency towards communication systems. In addition to high reliability and low latency requirements, communication systems for the IIoT should offer long-term stability. If communication is not stable throughout time and requires frequent updates, it again negatively impacts the communication's overall reliability. Meeting these requirements is not a trivial task, especially for a wireless network. Such a network shares the available spectrum with other communication and thus has to handle all kinds of interference.

Building upon these communication requirements for industrial applications, we envision smart, connected industrial systems. These systems shall be software-based and thus flexible in making intelligent decisions to communicate data from sensors to actuators successfully.

In this thesis, we design, implement and evaluate protocols for the Industrial Internet of Things achieving long-term stable, reliable communication with low latency and overcoming interference. We build these protocols for low-power wireless networks with limited energy budgets.

1.1 Challenges

Many IIoT applications have strict latency and reliability requirements for data communication. Moreover, several concurrent communications need to reach their destinations at similar points in time. Unreliable or delayed communication could delay processes or could hinder the detection of faults, leading to devastating results in the worst case. Moreover, communication between these devices has to coexist with other wireless communication, like WiFi or Bluetooth, and withstand unavoidable wireless interference.

To meet the strict requirements of industrial applications, efficient scheduling algorithms are necessary. Current systems and algorithms address some of these requirements but solve the challenges only in part. For example, Orchestra [3] offers highly reliable communication while sacrificing latency. Palattella et al. [4], Saifullah et al. [5], and Ferrari et al. [6] achieve acceptable reliability and low latency. However, they either require a wireless spectrum (almost) free of interference [4, 5], or allow only a single communication at a time [6].

In addition to the challenges above, energy efficiency is another crucial component for efficient wireless protocols. While communication devices attached to robots can share a robot's energy source, other sensors will be battery-powered. For example, sensors monitoring a pipeline, detecting fires, or operating in remote locations have to be powered by batteries or harvest their energy through, e.g., solar. As many devices will be powered this way, a protocol has to by design ensure power efficiency. Moreover, communication in these systems should be stable over long periods to minimize the amount of maintenance, like installing a new schedule and keep highly reliable at all times.

While there has been much work on low-power wireless networks in the past, existing protocols that allow concurrent communications, achieve high reliability, and low latency, lack the flexibility to act upon local changes of the wireless medium during communication or between individual transmissions. Most current systems cannot immediately adapt to changes in interference. However, as one can never plan the exact amount of interference or its pattern, a stable solution should offer flexibility to adapt to it.

We derive the following goals from these challenges and observations: We want to provide a scheduling system and algorithms that ensure long-term stable, reliable communication in industrial wireless sensor networks while keeping latency as low as possible. We will design, implement and evaluate algorithms and set these into perspective to state-of-the-art solutions.

1.2 Low-Power Wireless Networks

Low-power wireless networks are networks that usually allow the communication of low amounts of data over various distances. There are, on the one hand, wireless personal area networks (WPAN) building upon the physical layers of, e.g., Bluetooth Low Energy (BLE), IEEE 802.15.4, or Ultra-Wideband (UWB). These networks cover short distances of a couple of meters with data rates of hundreds of kbit/s up to several Mbit/s. On the other hand, low-power wide-area networks (LPWAN) build on physical layers like LoRa or Sigfox. These

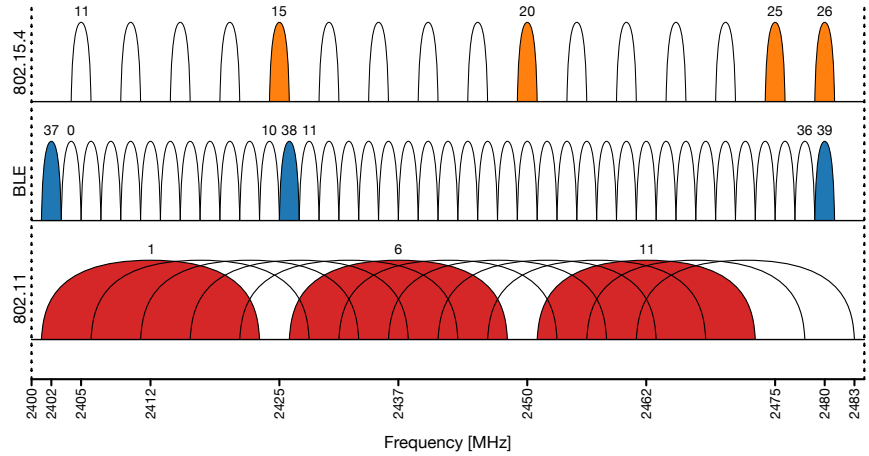


Figure 1.1: Channel mapping of BLE, IEEE 802.15.4, and IEEE 802.11 (WiFi) sharing the 2.4 GHz frequency band. BLE and IEEE 802.15.4 channels are 2 MHz wide with 2 MHz and 5 MHz spacing between center frequencies of adjacent channels, respectively. WiFi channels are 22 MHz wide. Blue: primary BLE advertising channels. Red: common, non-overlapping WiFi channels. Orange: commonly used 4-channel hopping sequence in IEEE 802.15.4 TSCH.

networks allow city-wide networks with transmission distances of kilometers with data rates of a few kbit/s.

The work we present in this thesis builds upon the IEEE 802.15.4 standard for low-power wireless networks. Therefore, we focus on the first category of physical layers, operating in the license-free 2.4 GHz frequency range.

1.2.1 Communication in Low-Power Wireless Networks

In low-power wireless networks, devices often operate on battery. Therefore, communication has to be energy efficient. A common measure for achieving this is a device's maximum transmit power, which is also regulated by laws. However, the standard transmit power usually stays well below the legal limit. With a standard value of 0 dBm (equaling 1 mW), communication in the license-free 2.4 GHz band can only cover short distances. Therefore, a sensor cannot necessarily communicate directly with an actuator or with a data sink. This limited communication range is solvable by base stations, like in cellular networks (e.g., mobile phones) or Smart Home networks. These base stations relay data from one device to another or send the data over the internet to another base station to deliver the data. One downside of a system using base stations is the additional infrastructure necessary in addition to sensor and actuator nodes. Another approach for solving the limited communication range is mesh networks. In these networks, nodes between the data source and the destination act as a relay and receive and forward the data. Networks using this approach do not need additional infrastructure. They only require a sufficient amount of sensor or actuator nodes between potential senders and receivers.

1.2.2 Broadcast by nature

Mesh networking is specific to wireless networks, as these networks are broadcast by nature. Like with human speech, any wireless device in the sender's vicinity tuned to the correct frequency can at least sense the presence of communication. Using the correct wireless technology (or a software-defined radio (SDR)), it can even receive the sent data. Therefore, relaying this data to extend the communication range is possible. However, this also means that no communication is independent of other signals sent over the frequency portion of the medium. All signals will interfere with signals of other devices. Therefore, it is essential to schedule communication to avoid interference or find measures to successfully communicate even in the presence of unavoidable interference or interference outside of an application's control. Figure 1.1 shows the communication channels for three wireless technologies (BLE, IEEE 802.15.4, and WiFi) sharing the 2.4 GHz frequency band. While we can tackle interference within a network, we have no control over the communication of other applications.

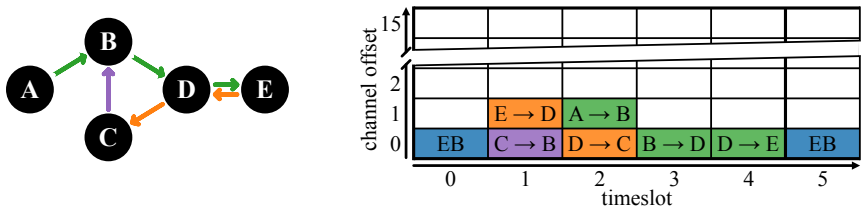
1.3 Communication in IEEE 802.15.4

After introducing the general area of low-power wireless networks and their communication, we continue with a closer look at communication in IEEE 802.15.4. IEEE 802.15.4 is a standard for low-rate wireless networks introduced in 2003 [7]. It defines different frequency ranges and modulation schemes. However, its 2.4 GHz physical layer with direct-sequence spread spectrum (DSSS) as modulation scheme is the most prominent one and the one we use. This layer defines 16 communication channels (11-26) between 2400 and 2483.5 MHz (see Figure 1.1) with a communication bitrate of 250 kbit/s. The direct-sequence spread spectrum modulation scheme uses a wider spectrum than necessary for the data rate and thus is less affected by interference than other modulation schemes.

Next to the definition of the physical layer, the IEEE 802.15.4 standard also defines the medium access control (MAC) layer. While the initial MAC layer was Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA), the 2012 IEEE 802.15.4 amendment proposed Time-Slotted Channel Hopping (TSCH) [8] as a new MAC layer. Since 2015 TSCH has been the standard MAC layer protocol for IEEE 802.15.4.

1.3.1 Time-Slotted Channel Hopping (TSCH)

Time-Slotted Channel Hopping (TSCH) is a MAC layer protocol for IEEE 802.15.4. Its design builds on the Time-Synchronized Mesh Protocol (TSMP), which is part of the previous industry standards ISA100.11a [9] and WirelessHART [10]. TSCH builds time-synchronized mesh networks. It combines the medium access methods of Time-division multiple access (TDMA) and Frequency-division multiple access (FDMA) and adds a pseudo-random channel hopping mechanism. Communication in TSCH happens in distinct time slots in one of the up to 16 frequency channels (see Figure 1.1), allowing up to 16 parallel communications in close vicinity to each other.



(a) Sample network. (b) Sample TSCH schedule.

Figure 1.2: Example of a TSCH schedule. The schedule contains three flows (A→E, E→C, and C→B) that use dedicated slots for communication and one shared beacon slot. The schedule has a slotframe length of 5 slots, therefore, slot 5 is a repetition of slot 0.

TSCH slots

TSCH time slots have a standard length of 10 ms. Each time slot has a specific role. It is either dedicated, shared, or empty. In certain implementations, like in the one of the Contiki-NG [11] operating system, a fourth role, beaconing (only), is available. In the case of a beaconing slot, nodes can send Enhanced Beacons (EB) containing control information for the network- and time-synchronization. These beacons are essential for maintaining the network's connections and time synchronization. In a dedicated slot, a node either receives or transmits data. The transmitting node can either communicate with a specific node (unicast) or with all nodes in range (broadcast). In the case of unicast transmissions, the receiver acknowledges the reception within the same slot. A shared slot allows any communication of the above, beaconing, unicast, and broadcast messages. As all possible senders share the same slot, these slots use the CSMA/CA back-off algorithm to limit collisions. Nodes that do not have anything to transmit in this shared slot listen for incoming packets. If a slot has none of the above roles, it is empty. In empty slots, the node's radio remains off to save energy.

TSCH groups slots in slotframes. Each of these slotframes has a priority. When two or more slotframes have an active slot simultaneously, TSCH chooses the one with the highest priority. For example, slotframes with beaconing slots often have the highest priority as they are necessary for maintaining the network. All other communication is distributed over the other slotframes. All slotframes are continuously repeated. As slotframes can have different sizes, not always the same slots of different slotframes overlap. All slotframes together form a TSCH schedule. Figure 1.2 shows an example of such a TSCH schedule.

Channel Hopping

Next to the availability of up to 16 frequency slots (FDMA) that allow the execution of up to 16 TSCH slots in parallel, TSCH uses the concept of channel hopping to withstand narrow-band interference. In channel hopping, all active channels cycle through a pseudo-random hopping sequence. Through this hopping sequence, every channel uses a different physical frequency at subsequent time slots. The hopping sequence has to include at least as many frequencies as parallelly active channels, with a maximum of 16 frequencies.

Thus, for every nth slot (n = number of frequencies), a channel uses the same physical frequency for transmission. For example, in a common hopping sequence of four channels, TSCH cycles through the channels 15, 25, 26, and 20. These channels belong to frequencies that do not overlap with frequencies of the three common non-overlapping WiFi channels (1, 6, and 11) (see Figure 1.1).

1.3.2 TSCH Scheduling

Scheduling in the context of TSCH is the allocation of active slots in a network. A schedule has to include slots for beaconing and one-to-one communication. The communication slots can be shared or dedicated. In the case of dedicated slots, a schedule has to ensure that relevant neighboring nodes can communicate to propagate data.

There are three classes of schedulers to create a TSCH schedule. These are centralized, autonomous, and distributed schedulers. Centralized schedulers use global knowledge to plan communications slots at the edge or cloud, separated from the actual network. Distributed schedulers, on the other hand, use local knowledge between neighboring nodes to negotiate communication slots. In contrast to both of these, autonomous schedulers operate without prior knowledge of the network topology and allocate slots for each node independent of its neighbors.

1.3.2.1 Centralized Scheduling

The oldest class of TSCH schedulers are centralized ones. Centralized schedulers combine the routing of traffic in a mesh network with the actual scheduling of communication slots. While we discuss routing in mesh networks in more detail in Chapter 1.3.3, we can define a route as the order of transmissions (hops) that are necessary for end-to-end communication between a specific sensor-actuator pair. This scheduling problem of allocating communication resources (i.e., sending and receiving slots) to meet an application's requirements is NP-hard (cf. [5]).

Centralized schedulers operate on global knowledge. Therefore, they collect information about the network, especially about the quality of the wireless links. We commonly measure the quality of a link using the link's packet reception rate, or its inverse, the expected transmission count (ETX) [12]. The latter is the expected number of transmissions necessary to transmit a packet over the specific link successfully. With this global knowledge of the network topology, the centralized scheduler can combine routing and scheduling to form a schedule accommodating the communication requirements of each node. Modern centralized schedulers commonly add retransmission slots for lossy links with an expected number of transmissions larger than one ($ETX > 1$). After computing the schedule, a central node disseminates it through the network to all nodes.

There are many different bodies of work proposing centralized scheduling algorithms for data collection and end-to-end communication. Many of these algorithms target different aspects like energy efficiency [13], or throughput maximization [14] but only provide simulation results. As our work focuses on scheduling systems with end-to-end communication on actual deployments, we limit ourselves to mostly discussing these. One of the first schedulers for TSCH

is TASA [4]. It is a traffic-aware protocol, but it assumes interference-free channels and does not include retransmissions in its schedule. In a second work, Palattella et al. [15] propose a mathematical analysis and method of computing the minimum number of active slots within a network. Other early works, building especially on WirelessHART [10], make the same assumptions. A noteworthy one is a paper introducing the C-LLF [5] scheduling algorithm. In this paper, Saifullah et al. [5] present a scheduling algorithm focusing on as high as possible schedulability for large amounts of deadline-aware communications. Moreover, they discuss and prove the scheduling problem to be NP-hard. Gunatilaka et al. [16] also focus on achieving more communications in a network by using the same channel for multiple communications in the same time slot if these communications happen physically far enough apart.

While the above protocols achieve high amounts of communications in a network, they rely on interference-free channels and thus do not use retransmissions. AMUS [17] is a protocol adding backup retransmission slots for vulnerable links in otherwise empty slots of the schedule. Gaillard et al. [18] propose another method of retransmissions as an extension to TASA. They create schedules with hop-based retransmissions, shifting the focus from schedulability towards reliability.

Darbandi et al. [19] propose the path collision-aware least-laxity first (PC-LLF) scheduling algorithm, another scheduling algorithm designed explicitly for TSCH networks which Rugamba et al. [20] implement as part of a central scheduler. Moreover, they describe a method of distributing such a schedule to nodes in a centrally scheduled network.

All the work above uses no retransmissions or slot-based retransmissions and designated pairs of nodes communicating with each other. The first one deviating from slot-based retransmissions is the flow-centric policy (FCP) [21]. That approach interprets end-to-end communications as a single entity (a flow), adding retransmissions wherever needed along the path. This approach has a certain design overlap with our first paper.

1.3.2.2 Distributed Scheduling

Instead of centralized schedulers operating on global knowledge, there are distributed ones operating on local knowledge. These schedulers negotiate communication cells between neighboring nodes during runtime. Moreover, they perform only the scheduling between neighbors and are independent of the routing of traffic.

Tinka et al. [22] present two distributed scheduling algorithms for networks constantly changing due to mobility. They present an algorithm for continuously announcing a node's presence and another one distributing scheduling information for quickly forming a network-wide schedule. Another early distributed scheduler is DeTAS [23], a traffic-aware distributed scheduler building collision-free schedules along a routing tree. Newer distributed TSCH scheduler built upon 6TiSCH, an IPv6 stack extension for IEEE 802.15.4 TSCH. The 6TiSCH minimal scheduling algorithm [24] uses a single cell in a schedule shared over all nodes. The Minimum Scheduling Function (MSF) [25] uses a minimal, shared cell for beaconing, autonomous cells for unicast communications to a specific cell when no other cell is available, and negotiated cells for the main

communication. The autonomous cells follow an approach of the autonomous scheduler Orchestra [3]. The Low-Latency Scheduling Function (LLSF) [26] schedules cells for multi-hop communication closer together to allow forwarding traffic immediately after reception. Domingo-Prieto et al. propose a PID-based scheduling solution that adds or removes cells from a schedule dependent on traffic demand and network state to counteract network changes and allow non-periodic and bursty [27] traffic. Low-latency Distributed Scheduling Function (LDSF) [28] achieve lower latency by splitting slotframes into shorter blocks and introducing retransmission options in consecutive blocks., focusing on improving latency in distributed TSCH. Palattella et al. [29] propose an algorithm matching the number of cells between nodes to the actual demand. Similarly, OST [30], another distributed scheduler with traffic awareness, allocates slots for each directional link and adapts its period according to the amount of traffic. While the previous works focus on low-latency and reliable communication, Jung et al. [31] propose a solution balancing latency, degree of activity of each node and without collisions to achieve a long network lifetime and high quality of service.

1.3.2.3 Autonomous Scheduling

The last class of TSCH scheduling algorithms is autonomous schedulers. While the other two scheduler classes either use global knowledge or local knowledge to make a scheduling decision, autonomous scheduling algorithms operate without knowing the network topology at all. Autonomous schedulers neither centrally plan communications and allocate resources nor negotiate resources between neighbors in a distributed fashion. Instead, they provide specific slots for each possible sender, receiver, or link in a network. By the availability of distinct slots for each node, autonomous schedulers can achieve high reliability of over 99.99%.

A prominent example of an autonomous scheduler is Orchestra [3]. Orchestra is a best-effort autonomous scheduler that uses sender-based or receiver-based communication within time slots reserved for certain groups of nodes. Orchestra requires a hash function to determine which nodes can send in which slot. DiGS [32, 33] adds autonomous scheduling to otherwise central WirelessHART networks adding robustness through path diversity introduced by devices selecting their own routing path. Oh et al. [34] propose Escalator, focusing on minimizing transmission delays in convergecast scenarios by allocating consecutive time slots along a packet's path. Moreover, contrary to Orchestra, Escalator uses multiple channels. Alice [35] deviates from the node-based slot allocation of Orchestra and uses link-based slots instead. Moreover, Alice uses multiple channels with link-based channel offsets instead of a single channel, and it reallocates all unicast slots after some time. The initial autonomous schedulers can achieve high reliability but are unaware of traffic flows and do not achieve the low latency of other scheduling approaches. However, TESLA [36] proposes a traffic-aware cell scheduling method to add adaptability to different traffic loads. TESLA adds and removes slots dependent on the traffic load of neighboring nodes. Jung et al. [37] propose a parameterized slot scheduler that adapts to the traffic load of nodes. The scheduler tries to find a trade-off between energy efficiency and reliability or latency by using shared slots for

nodes transmitting to a joint receiver if collisions are unlikely. Rekik et al. [38] present e-TSCH-Orch, an enhancement to Orchestra avoiding congestion by adaptively adding transmission slots for a node depending on the number of packets in the node's queue. ATRIA [39], another traffic-aware scheduling method allocates slots according to the traffic demand of each link. To improve network performance, ATRIA includes a method for selecting the optimal slotframe length and uses subslotframes to avoid slot conflicts.

1.3.3 Routing in low-power wireless mesh networks

Routing is the selection of a path data takes from a sender to a receiver. Routing determines the order of links over which data travels through the network, whereas scheduling concerns the allocation of the individual links that are available for routing. The two major routing approaches are tree-based routing or shortest-path routing. In tree-based routing, packets travel a tree upwards till a common ancestor of sender and receiver and then downwards the tree to the receiver. Distributed and autonomous TSCH schedulers commonly use this routing approach. The other approach is shortest-path routing which requires global knowledge of the network topology and uses algorithms like Dijkstra's shortest path algorithm [40] or the A* algorithm [41]. Shortest-path algorithms are a good choice for centralized routing and thus a good fit for a combination with centralized schedulers.

1.3.3.1 Tree-based routing: RPL

In most cases, routing and scheduling are separated into different layers. While several different routing protocols exist, the *Routing Protocol for Low-Power and Lossy Networks (RPL)* [42] tree routing protocol is generally accepted and widely used.

RPL is a best-effort routing protocol for low-power wireless networks susceptible to packet loss. For routing a path for a packet that needs to be transmitted, RPL uses a directed acyclic graph (DAG). Packets are forwarded (upwards routed) hop-by-hop from parent to parent until reaching the source of the graph. From there, the packets are routed downwards hop-wise until they reach their destination. That means all routing in RPL is a combination of upwards-routing to the tree's source and downwards routing to the packet's destination. A variation to this approach is a RPL mode storing information about a node's children. This variation routes traffic only upwards until meeting a common ancestor of the packet's source and destination. The path RPL uses for routing is based on the transmission-based shortest distance to the root using a metric like ETX.

1.3.3.2 Routing in TSCH

While TSCH is a MAC layer protocol and thus not responsible for routing, some work, especially those of centralized schedulers, usually merges scheduling and routing into one. A straightforward approach in the context of TSCH is computing the shortest path, the path with the fewest transmissions, using a shortest-path algorithm, like Dijkstra's algorithm [40]. Li et al. [43] deviate from this and instead perform an asymmetric routing approach. They apply

specific routing strategies for different communications in a network. Wu et al. [44] deviate from the standard shortest-path routing approach. Instead, they present a conflict-aware real-time routing approach. This approach takes path conflicts originating from scheduling decisions into account when making routing decisions.

Next to the integration of routing in centralized TSCH scheduling, there is the field of multi-path TSCH schedules. These are mainly distributed schedulers using the RPL routing protocol. However, instead of sticking to the single routing path, they route the traffic along multiple paths. They use triangular-based redundancy patterns [45], overhearing in RPL networks [46], and packet replication algorithms [47].

Most distributed and autonomous schedulers use RPL for routing.

1.3.3.3 Opportunistic Routing

Opportunistic routing is an approach to improve the throughput and reliability of multi-hop wireless networks. Instead of routing traffic through one specific intermediate node, opportunistic routing addresses more than one potential receiver/forwarder [48–50]. For example, in ExOR [48] each packet is addressed to a set of potential forwarding nodes, prioritized by routing progress. Based on their priority, each node in the forwarder set is assigned a time slot for forwarding. It only utilizes this time slot if it did not overhear the forwarding of the packet in a previous time slot.

Later work such as ORW [51] and ORPL [52] introduce opportunistic routing to duty-cycled, low-power wireless networking. In ORW, the first receiver that successfully receives the packet and provides routing progress acknowledges and forwards the packet. ORPL combines the ideas of ORW and ExOR.

None of the protocols presented so far uses opportunistic routing in combination with TSCH. However, in recent years several works have addressed this. Huynh et al. [53], Hermeto et al. [54], and Hosni et al. [55] study the use of opportunistic routing or anycasts in TSCH and propose changes to TSCH to allow non-colliding acknowledgments from multiple receivers. BOOST [56] assigns different sending delays to the potential forwarders and lets the forwarders use carrier sense to ensure a single forwarder.

1.3.3.4 Lack of Routing – or – Flooding

The last category of routing protocols is the absence of an actual routing protocol. In network-wide flooding, the whole network takes part to propagate the information from sender to receiver. However, some flooding protocols limit the number of involved nodes and use only subsets of a network for directed floods which essentially are a kind of routing. We discuss these protocols further in Section 1.3.4 on synchronous transmissions.

1.3.4 Synchronous Transmissions

Signals in wireless networks are always broadcasts. When two closely located devices transmit simultaneously, their signals overlap and thus interfere with each other. This interference forms a combined, added physical wave. This

resulting wave is usually unintelligible for a receiver. However, there are circumstances under which a receiver can receive and decode just one of the transmitted signals. The field of low-power wireless networks knows two methods of concurrent or synchronous transmissions.

On the one hand, there is the capture effect which was initially observed for FM receivers [57]. If one signal is significantly stronger than the combination of all others, this can be successfully received and decoded nonetheless. For utilizing the capture effect in IEEE 802.15.4, the stronger signal has to have at least twice the power (+3 decibel (dB)) as the combined other signals [58]. Moreover, the stronger signal must not arrive later than 160 μs after the first signal [59].

On the other hand, there is non-destructively interfering communication. This communication requires a much higher degree of time synchronization ($< 0.5\,\mu s$ [60]) and the transmission of the same data by all transmitters. Initially, it was believed to be constructive interference. However, a recent work by Liao et al. [61] shows that it is mere non-destructive with DSSS helping to receive a packet successfully.

Synchronous transmission protocols

In 2011, Glossy [60] created the foundation for synchronous transmissions in low-power wireless IEEE 802.15.4 networks. Glossy uses synchronized flooding to disseminate data from one node to all others in a network. LWB [6] builds on Glossy floods by adding the capability of scheduling individual network floods for data collection. While LWB is not a real-time protocol, Blink [62] performs deadline-based real-time communication achieving high reliability on top of it. While the flooding protocols above perform network-wide flooding and thus involve the whole network for communicating data, works like WSNShape/Sparkle [63] and CXFS [64] limit the number of forwarders and perform directional flooding or essentially routing. CXFS, for example, is a method for concurrent transmission forwarder selection. LaneFlood [65] builds upon this forwarder selection to even allow the flooding of IPv6 traffic along a routed lane.

In contrast to all algorithms above, Chaos [59] builds upon the capture effect. It extends the design of Glossy to allow the concurrent transmission of different data for network-wide data aggregation.

1.4 Research Problem

The Industrial Internet of Things is gaining more and more traction in connecting devices in industrial settings to communicate sensor data and derive immediate action from this data. As these actions are often critical, the data communication must simultaneously be highly reliable and have low latency. However, the underlying low-power wireless communication is unreliable and covers only short ranges. Moreover, this communication should also be stable over longer periods without the need for in-network or external interceptions to rebuild the network's communication schedule.

Current solutions for the Internet of Things are not capable of fulfilling all of these requirements. Most offer high reliability or low latency, generally

without long-term stability in environments susceptible to dynamic interference levels. Some solutions offer both high reliability and low latency. However, they build upon network-wide flooding and thus involve the whole network or significant parts of a network to communicate data from one device to another. Flooding does not offer the flexibility of parallel communications and routed communications through smaller parts of the network. Consequently, this thesis focuses on answering the following questions:

RQ1: How can we dynamically adapt communication to ensure low-latency and high stability in centrally scheduled low-power wireless networks susceptible to changes in the wireless medium?

We answer this question mainly in our first paper (Paper A). In Paper A, we introduce a novel retransmission policy for centrally-scheduled low-power wireless networks. This policy allows the use of retransmissions wherever they are needed along a path. It ensures stable communication, even if certain links are more affected by changes in the wireless medium than anticipated, without introducing unnecessary delays.

RQ2: How can we ensure stable communication in networks susceptible to local wide-band interference without impacting latency?

Papers B and C answer this question. Paper B is an intermediate step relaxing the requirement for nodes to have a specific communication partner. Therefore, communication can skip nodes affected by interference. Paper C adds redundant communication paths, routing traffic around local interference sources. To achieve this, we combine opportunistic routing and synchronous transmissions with TSCH.

1.5 Contributions

In this section, we summarize the papers that build the second part of this thesis.

1.5.1 Paper A – Master: Long-Term Stable Routing and Scheduling in Low-Power Wireless Networks

In this paper, we address the problem of the stability of centrally scheduled low-power wireless networks susceptible to dynamic changes in the wireless medium. While existing retransmission algorithms tackle link changes in centrally scheduled networks, they are inflexible at handling unexpected amounts of interference at a link and introduce unnecessary delays where they are essentially not needed.

We introduce Master, an open-source central scheduler for TSCH. Moreover, we introduce a novel retransmission scheme, we name *Sliding Windows*, allowing more robust schedules while keeping low latency. Master consists of two parts: a central server component and a networking component. Its central server component runs on the edge of a sensor network or even in the cloud. Its networking component performs routing and schedule installation on the devices forming the sensor network. This twofold design splits the compute-heavy

scheduling task from the network operation of the usually resource-constraint devices in an IoT or sensor network.

While the scheduler MASTER lays the foundation for centrally scheduled communication, this paper also introduces the *Sliding Windows* retransmission scheme, which adds flexibility towards fluctuating levels of interference. In previous TSCH scheduling systems, retransmissions are strictly allocated for a single hop, thus, for communication between a specific sender and receiver. In these systems, each node has at each timeslot a specific role of either sender or receiver. *Sliding Windows* changes this by allowing a device to either receive or send in a particular slot, based on previously successful reception of a packet. With *Sliding Windows*, nodes actively listen for packets from the earliest possible time they can receive a packet and keep active for maximally the number of slots the retransmission budget allows. Thus, each hop can use between no and all of the shared retransmissions depending on its need.

We evaluate MASTER and *Sliding Windows* in testbed evaluations and show reliability of *Sliding Windows* of above 99% and long-term schedule stability over 24 hours.

Personal Contribution. I am the lead designer and implementer of MASTER and *Sliding Windows*. Additionally, I am the main designer of the evaluation and the main author of the paper. The chapter was published as a paper in the Proceedings of the 16th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2020 [66], and its source code is available on GitHub[1].

1.5.2 Paper B – (POSTER) Overtake: Opportunistic Routing and Concurrent Transmissions for TSCH

In Chapter 3, we have an initial look at strategies for overcoming local wide-band interference in centrally scheduled low-power wireless networks. While MASTER with *Sliding Windows* shows good performance in networks with varying interference levels on single channels, it is not capable of withstanding wide-band interference. In Paper B, we start from the assumption that especially in dense networks, not only the next scheduled node is in communication range, but potentially even the one after. To be able to skip nodes and thus overcome link failures, we introduce OVERTAKE. OVERTAKE allows a node to directly communicate with any other node of the respective communication flow. In OVERTAKE, we do not send packets to a specific node but a flow instead. When we do not send packets to a specific node, we have to solve the challenge of loops or packets traveling back to nodes we previously skipped. We solve this by introducing node ranks to ensure monotonic packet propagation. While this paper adds more flexibility to TSCH schedules than *Sliding Windows* and potentially even reduces latency further, it also proves another point: Concurrent transmissions of two neighboring nodes that both received the packet are feasible in TSCH.

[1] `https://github.com/ds-kiel/master-scheduler`

Personal Contribution. I am the lead designer and implementer of Overtake. Additionally, I am the main designer of the evaluation and the main author of the paper. The chapter was published as a poster paper in the Proceedings of the 16th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2020 [67].

1.5.3 Paper C - Opportunistic Routing and Synchronous Transmissions Meet TSCH

Chapter 4 more thoroughly discusses the question of stable communication in the presence of local wide-band interference. This paper shows that TSCH has sufficient time synchronization for synchronous transmissions using the capture effect. Moreover, we investigate the possibility of combining opportunistic routing and synchronous in TSCH networks and propose Autobahn, a protocol achieving long-term stable schedules in the presence of local wide-band interference. Autobahn starts like other TSCH schedulers with a single shortest path. Starting from this shortest path, Autobahn includes neighboring nodes in the end-to-end communication to add redundancy for the case of local interference along the shortest path. With these additional nodes, we have more than one path along which the communication can flow from sender to receiver. By using the concept of opportunistic routing, it is sufficient that a packet reaches its destination regardless of the actual path. That means that not each node has to take part in the communication to propagate a packet. As we might have multiple forwarders if multiple nodes receive a packet, we need to solve the challenge of selecting the best of them. We overcome this challenge using synchronous transmissions and therefore not choosing a single forwarder but synchronously forwarding the packet instead. Thus, we keep complexity low and do not increase the minimum latency compared to a single path schedule. Moreover, combining the three concepts of opportunistic routing, synchronous transmissions, and TSCH even allows a lower average latency in the case of local narrow-band interference. Our evaluation shows that Autobahn is capable of outperforming *Sliding Windows* and other single-path retransmission strategies both in the presence of and without interference. Moreover, Autobahn offers long-term stability with over 95% reliability over several days without the need for rescheduling.

Personal Contribution. I am the lead designer and implementer of Autobahn. Additionally, I am the main designer of the evaluation and the main author of the paper. The chapter was published as a paper in the Proceedings of the 46th IEEE Conference on Local Computer Networks (LCN), 2021 [68], and its source code is available on GitHub[2].

1.6 Conclusion and Future Work

In this thesis, we study different aspects increasing the robustness of centrally scheduled low-power wireless networks. With the growing amount of interconnected devices in industrial applications, the so-called Industrial Internet of

[2] `https://github.com/ds-kiel/autobahn`

Things, robust, reliable communication with low latency is required. While existing protocols achieve different aspects of these sufficient for the Internet of Things, there is thus far a lack of centrally scheduled and routed low-power wireless communication for industrial applications stable under dynamic amounts of interference. We propose three protocols achieving different levels of robustness depending on a network's need: MASTER, OVERTAKE, and AUTOBAHN. MASTER offers a centralized scheduler with retransmission flexibility along an end-to-end path. OVERTAKE addresses packets to a flow instead of a link to skip nodes if possible or necessary. Finally, AUTOBAHN widens the communication path, including neighboring nodes, to opportunistically and synchronously route traffic overcoming wide-band interference.

Future Work For the future, we see different trajectories continuing our work. For once, the scheduler we use still lacks a system for in-network neighbor data collection and schedule distribution. Thus, it is of interest to us to find effective ways to perform these tasks without (majorly) impacting the performance of the scheduled communication. Moreover, we see open research questions in the domain of other promising wireless technologies like BLE and LoRa. Especially, LoRa in the 2.4 GHz band as the basis for IoT applications is still underexplored. Next to using LoRa for communication, its ranging capabilities open a promising direction towards new means of localization.

2

Master: Long-Term Stable Routing and Scheduling in Low-Power Wireless Networks

---

Oliver Harms, Olaf Landsiedel

Proceedings of the 16th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2020, pp. 86–94.

PAPER A

Abstract

Wireless Sensor-Actuator Networks (WSANs) are an important driver for the Industrial Internet of Things (IIoT) as they easily retrofit existing industrial infrastructure. Industrial applications require these networks to provide stable communication with high reliability and guaranteed low latency. A common way is using a central scheduler to plan transmissions and routes so that all packets are delivered before a deadline. However, existing centralized schedulers are only able to achieve high reliability in the absence of interference. This limitation lowers the feasibility of using centralized schedulers in most environments susceptible to interference.

This paper addresses the challenge of stable, centrally scheduled communication in low-power wireless networks susceptible to interference. We introduce Master, a centralized scheduler and router, for IEEE 802.15.4 TSCH (Time-Slotted Channel Hopping). Master uses Sliding Windows, a novel transmission strategy, which builds on flow-based retransmissions instead of link-based ones. We show in our experimental evaluation that Master with Sliding Windows achieves routing and scheduling stability for over 24 hours with end-to-end reliability of over 99.6%. Moreover, we show that Master outperforms Orchestra, a state-of-the-art autonomous scheduler, in terms of latency by a factor of 8 while achieving similar reliability under a slight duty-cycle increase.

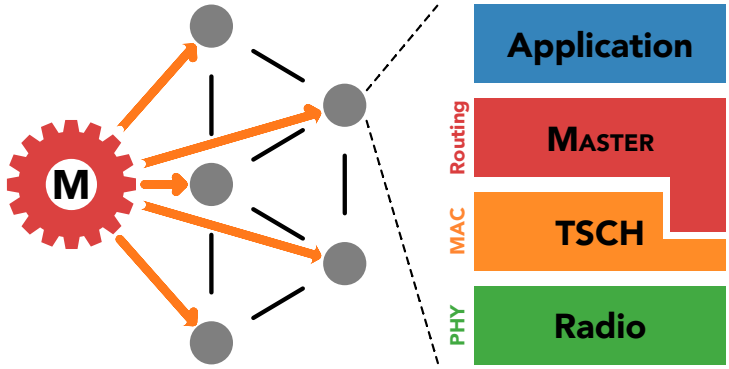


Figure 2.1: MASTER consists of an external centralized scheduler (M) and a routing layer. The external scheduler performs the global routing and scheduling and pushes the computed schedule onto the network. In each node, MASTER's routing layer implements the schedule in TSCH and performs the routing during runtime.

2.1 Introduction

For many applications in the Industrial Internet of Things (IIoT), it is essential that network traffic meets deadlines. To achieve this goal, commonly, a centralized scheduler collects information about the network topology and the wireless links. With this global knowledge, representing a major advantage over distributed solutions, the scheduler is able to compute optimal routes and transmission schedules of end-to-end communication (traffic flows). In IEEE 802.15.4, the scheduler assigns communication slots in the time and frequency domain to nodes, i.e., it employs Time-Slotted Channel Hopping (TSCH) [8]. However, due to wireless link dynamics, centralized schedulers have to account for the risk of packet losses and, therefore, usually include multiple retransmission slots for each link. These retransmission slots increase latency and reduce the available bandwidth, thus, causing an increased radio on-time.

Many recent centralized scheduling algorithms assume the availability of interference-free channels or at least a static amount of interference [5,16]. These assumptions do not hold in many of today's environments where IEEE 802.15.4 IIoT networks co-exist with an increasingly large number of WiFi and Bluetooth networks. This coexistence results in large amounts of interference and thereby limits the stability and reliability of those centralized solutions.

In this paper, we introduce MASTER, a centralized scheduler designed for TSCH. It combines the traditional steps of central scheduling and routing with a novel transmission strategy which we call Sliding Windows. Our Sliding Windows algorithm introduces the flexibility needed to accomplish long-term schedule stability and communication reliability while meeting the latency requirements of industrial applications. As a result, MASTER enables long-term stable schedules and thereby eliminates the need for frequent rescheduling, a key drawback of today's central schedulers. Furthermore, we design MASTER

as an open[1] and easily extendable platform to foster rapid experimentation with central scheduling policies.

Our evaluation shows that MASTER with Sliding Windows outperforms slot-based retransmission strategies of centralized schedulers. Moreover, it outperforms the low-power autonomous scheduler Orchestra [3] in terms of latency while achieving similar reliability and consuming not significantly more energy, making it particularly suitable for low-power systems. Overall, this paper makes the following contributions:

- We present MASTER, an open-source, centralized router and scheduler for TSCH-based networks designed with easy extendability in mind.
- We design Sliding Windows, a transmission strategy for MASTER to increase the flexibility, stability, and reliability of centrally scheduled communications.
- We propose flow-based queues as an extension to TSCH to enable the use of central scheduling algorithms.
- We implement MASTER as part of Contiki-NG and evaluate it in environments susceptible to interference. We show the long-term stability of schedules computed by MASTER in experiments of 24 hours. These experiments result in highly reliable (>99.6%), low-latency (<4.5 slots) communications.

The remainder of this paper is organized as follows. Section 2.2 gives the necessary background information on TSCH as well as TSCH schedulers. Section 2.3 introduces the design of MASTER, and Section 2.4 presents our testbed evaluation. Section 2.5 reviews related work, followed by the conclusion in Section 2.6.

2.2 Background

This section gives an overview of relevant concepts on (A) Time-Slotted Channel Hopping (TSCH), (B) the ETX metric, (C) scheduling, and (D) retransmissions.

2.2.1 Time-Slotted Channel Hopping

Time-Slotted Channel Hopping (TSCH) is one of the MAC-layer protocols defined in the IEEE 802.15.4e standard [8]. TSCH uses dedicated time- and frequency-slots (TDMA and FDMA) for accessing the wireless medium. These slots are standardized to a length of 10 ms, and each slot uses one out of maximally 16 channels. TSCH continuously cycles through a hopping sequence of all active channels. Thus, it is changing the channel every slot. Assigning different frequencies to slots allows TSCH to increase the network's resilience to interference. Slots dedicated to control-information, so-called Enhanced Beacon (EB) slots, provide broadcasts which support both network formation and time

[1] Available as open-source at: `https://github.com/ds-kiel/master-scheduler`

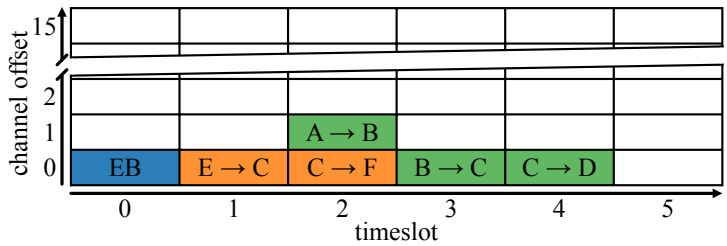


Figure 2.2: Sample TSCH schedule. Slot 0 is a shared slot for sending and receiving Enhanced Beacons (EB) while slots 1-4 are unicast slots with one transmission per channel at a time. This simple schedule contains two multi-hop communication flows, highlighted in green and orange. The channel offset is added on top of the usual hopping sequence.

synchronization, both essential for maintaining a schedule of synchronized transmissions as in TSCH.

Multiple TSCH slots are grouped into slotframes, and multiple slotframes form a TSCH schedule, see Figure 2.2. Each node has a custom TSCH schedule determining its behavior in each slot. Slots are either dedicated, shared, or empty: In a dedicated slot, a node either transmits or receives. In shared slots, nodes may broadcast or receive control information, such as Enhanced Beacons. Such slots are not assigned to individual nodes and have multiple nodes contending for transmissions. To limit collisions, these slots employ the CSMA-CA back-off algorithm. If a slot is neither dedicated nor shared, it is empty, and the radio remains off to save energy.

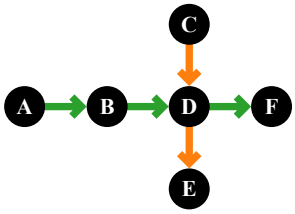
2.2.2 Link quality metric

Link quality metrics, such as the expected transmission count, ETX [12], represent the quality of a wireless link. ETX specifies the number of transmissions expected to transmit a packet successfully over a wireless link. The ETX value is the inverse of the packet reception rate (PRR) of a link ($ETX = 1/PRR$).

2.2.3 Scheduling

In the context of wireless communications, scheduling is the process of allocating resources for communications to meet all requirements such as release-time and deadline. Scheduling is an NP-hard problem, meaning, it is not optimally solvable in polynomial time (cf. [5]). Therefore, different heuristics and algorithms were developed to solve scheduling problems sufficiently well for specific scenarios.

TSCH scheduling: TSCH does not specify how communications are scheduled. Therefore, scheduling TSCH communications can be performed in a centralized, distributed, or autonomous manner. In the distributed case, subsets of the network perform cooperative scheduling (cf. 6TiSCH MSF [69]). The autonomous case, used by the well-known TSCH scheduler Orchestra [3], performs an autonomous mapping of links to resources. The centralized scheduling approach provides us with global topology knowledge, and we can allocate



(a) Sample Topology with two intersecting flows.

	1	2	3	4
A		TX		
B		RX	TX	
C	TX			
D	RX	TX	RX	TX
E		RX		
F				RX

(b) Baseline schedule without retransmissions.

	1	2	3	4	5	6	7	8
A			TX	TX				
B			RX	RX	TX	TX		
C	TX	TX						
D	RX	RX	TX	TX	RX	RX	TX	TX
E			RX	RX				
F							RX	RX

(c) A slot-based schedule with one transmission and one retransmission slot per hop.

	1	2	3	4	5	6	7	8	9
A				TX	TX	TX	TX		
B				RX	RXTX	RXTX	RXTX	TX	
C	TX	TX	TX						
D	RX	RXTX	RXTX	TX	RX	RXTX	RXTX	RXTX	TX
E		RX	RX	RX					
F						RX	RX	RX	RX

(d) Sliding Windows schedule with one transmission slot per hop and two (orange flow) or three (green flow) retransmission slots to be shared among the nodes of a flow.

Figure 2.3: Example: One flow originates at node A to end at node F while the second one originates at node C and ends at node E.

resources using established algorithms such as Dijkstra's Shortest Path First algorithm [40].

2.2.4 Retransmissions

As wireless communication links are unreliable, transmissions are never guaranteed to be received. To increase the reliability, schedulers commonly include retransmission slots to retry a failed transmission. A common way of adding retransmissions is the duplication of single slots. This slot-based approach increases the reliability, by including multiple tries per hop. In this paper, we introduce a new, flow-based transmission strategy to increase both performance and flexibility, see Section 2.3.2.

2.3 Design

In this section, we present the design of Master, our transmission strategy Sliding Windows, and the system architecture of Master.

2.3.1 Centralized Routing and Scheduling with Master

A fundamental building block of Master is its centralized scheduler. Its design is a three-step process to build a long-term stable, low-latency, reliable communication schedule. This process is a sequential top-down approach of (1)

centralized routing, (2) applying a transmission strategy, and (3) scheduling. The input to the process is (a) a set of traffic flows specified by source, destination, periodicity, and deadline, as well as (b) the network topology with long-term link reliability statistics. The application commonly provides the set of flows, and we derive the network topology from long-term link measurements, see Section 2.3.4.5.

2.3.1.1 Centralized Routing

Routing is the first step in MASTER and uses the previously specified flows and network link-reliability as input. To perform the routing, MASTER constructs a directed weighted graph using an ETX-based metric (ETX^n, $n \in \mathbb{N}$, usually $n = 2$), corresponding to the link reliability statistics. A higher ETX-power favors a higher number of highly reliable links over a lower number of links with lower reliability. Using this graph, we compute the shortest end-to-end routes. As shortest path routing finds the optimal path for each flow, the flow latency selected by the routing process stays minimal. The result of our routing is an extended set of flows that consists of a source, a destination, and the intermediate hops. In MASTER, we use Dijkstra's algorithm for shortest-path routing, but our modular design allows us to plug-in any routing algorithm and metric.

2.3.1.2 Transmission Strategies

After computing the route for each flow, we employ a transmission strategy to ensure reliable communication over unreliable wireless links. Thus, the transmission strategy adds retransmission slots to each flow to handle failed transmissions due to link dynamics and interference. The transmission strategy extends each flow by a specific number of slots. In the case of highly reliable links in an interference-free environment, we can employ a simple transmission strategy of assigning only one slot per hop. In practice, however, we add retransmission slots according to the expected link reliability of each hop. We employ either a slot-based transmission strategy (see Section 2.2.4) or our new approach of a flow-based transmission strategy (see Section 2.3.2 below).

2.3.1.3 Scheduling

After applying one of the transmission strategies, we pass the modified flows to the scheduler. The scheduler builds a communication schedule for all flows considering their periodicity.

For our application scenarios and to be comparable to Orchestra, we employ a non-deadline-based scheduling algorithm. It is especially suitable for best-effort, periodic, deadline-free systems. The algorithm is *Reverse Longest Path First (R-LPF)*, our own flavor of the *Shortest Path First (SPF)* scheduling algorithm. SPF is based on the process scheduling algorithm Shortest Job First (SJF) [70]. Contrary to starting with the shortest flow, our scheduler performs backward scheduling, starting with the end of the longest flow. This modification of the scheduling algorithm results, in our experience, in a lower number of unused slots within a flow. A lower number of unused slots corresponds with lower latency.

Figure 2.3 shows a schedule for two flows generated using no retransmissions, a slot-based retransmission strategy, as well as the transmission strategy of Sliding Windows with a transmission number based on Equation (2.3) and a scaling factor of 1. To generate the schedule of Figure 2.3d, we assume the ETX-value of each link to be between 1 and 2 ($ETX_{link} \in]1, 2[$).

Any scheduling algorithm, including deadline-based ones, can easily be implemented in Master. For the remainder of this paper, we use R-LPF.

2.3.2 Master's Flow-based transmission strategy

Our flow-based transmission strategy assigns a specific number of retransmissions to a flow instead of using a per-hop basis, as done traditionally, see Section 2.2.4. The flow-based retransmission slots allow the nodes of a flow to share these slots and use them as needed along the path, see Figure 2.3d. As a result, we can increase the communication reliability while potentially using minimally more slots in the final schedule (see Node D in Figure 2.3c and Figure 2.3d).

With this, we divert from the traditional scheme of two active nodes to one with multiple active nodes: Traditionally, at a single time-slot, frequency, and within a localized area, only one node transmits and another one receives. Instead, we now have more than two nodes awake that either transmit or receive. Our transmission strategy has the advantage of being adaptable to network changes, e.g., due to interference. Thus, during the journey of a packet, we can use the shared transmission slots in whichever part of the flow interference impacts communication. This adaptability is traditionally possible within distributed schedulers that can locally adapt to link changes. With Sliding Windows, we now enable such flexibility in centralized ones.

2.3.2.1 Window Size

The maximal number of transmission slots (TX_{max}, later denoted as $\#transmissions$) in a flow and the hop-count of the flow determine the window size which is calculated by

$$window\_size = 2 + TX_{max} - hops \tag{2.1}$$

This window size is the number of nodes maximally active in a slot of a flow. Moreover, it matches the maximum number of active slots of a node for a given flow. According to this relation, the window size is equal to the shared number of slots of a node for transmission or reception ($TX_{max} - hops$) plus its first and last slot allocated for reception and transmission, respectively.

In Master, we have two flow-based transmission policies: (1) fixed window size and (2) metric-based window size. For the first policy, we use the same window size for all flows independent of their length or link quality. For the second one, our scheduler determines the window size and number of transmissions depending on the flow's or link's ETX-values. The metric-based window size allows us to account for both the number of hops and the reliability of the individual links.

Using the link's ETX values, we can calculate the total number of transmissions of the flow with either

$$\#transmissions = n * \lceil \sum ETX_{link} \rceil, \quad n \in \mathbb{N} \tag{2.2}$$

or

$$\#transmissions = n * \sum \lceil ETX_{link} \rceil, \quad n \in \mathbb{N}, \tag{2.3}$$

including a scaling factor n. This scaling factor regulates the conservativeness of the scheduler. If we choose a scaling factor of 1 for Equation (2.3), the number of transmissions is equal to the one using an ETX-based, slot-based retransmission strategy (cf. Section 2.2.4). Equation (2.2) uses the end-to-end ETX-value of the flow, while Equation (2.3) uses the ETX-values of the individual links.

Throughout the remainder of this paper, we use the following naming scheme to refer back to these equations:

$$SW - < Equation\ number > [- < scaling\ factor\ n >]$$

SW denotes it as a Sliding Windows transmission strategy. The naming scheme includes the scaling factor only if referring to a specific representation of the strategy. When referring to the general strategy, it is not included.

Please note that for long flows, i.e., with many hops such a strategy could lead to a large window, and thereby too many nodes being awake at the same point in time. Too many active nodes lead to inefficiencies, and we counterbalance it by splitting a flow into sub-flows once it exceeds a limit N. The flow-based strategy is then applied to each sub-flow individually. In MASTER, we use a threshold of $N = 10$. Thus, for example, a flow of length 11 is split into two overlapping sub-flows of length 6.

2.3.2.2 Algorithm

In Algorithm 1, we present the algorithm for applying a flow-based transmission strategy. The algorithm takes as input a flow consisting of multiple nodes, the network's ETX graph, the strategy (SW-2 or SW-3), and the scaling factor. The algorithm starts calculating the flow's total ETX cost, as well as the flow's number of transmissions according to the given strategy (SW-2 or SW-3) and the window size according to Equation (2.1). From line 17 onward, the algorithm computes the active slots for each node of the flow and inserts the nodes into the respective slots of the new flow. For example, slot 6 of Figure 2.3d would be represented in the new flow as a list containing the elements A, B, D, F in this order.

2.3.2.3 Flow-based transmissions vs. Flow Centric Policy (FCP)

Recently, a paper by Brummet et al. [21] introduced a similar idea of moving from link-based to flow-based transmissions.

The main difference between Brummet's proposed Flow Centric Policy (FCP) and our Sliding Windows strategy are the rules for determining the optimal number of flow transmissions. FCP only defines fixed numbers of retransmissions with a maximum of up to 4 retransmissions for a flow. Sliding

Algorithm 1 Sliding Windows transmission strategy

Input: $flow$, $graph_{ETX}$, strategy, scaling factor n
Output: $flow_{new}$ (modified version of $flow$)
1: $cost_{total} = 0$
2: **for** $i = 0$ to $length_{flow} - 1$ **do**
3: $sender_{hop} \leftarrow flow[i]$
4: $receiver_{hop} \leftarrow flow[i + 1]$
5: **if** $strategy =$ "$SW - 2$" **then**
6: $cost_{total} = cost_{total} + graph_{ETX}[sender_{hop}][receiver_{hop}]$
7: **else if** $strategy =$ "$SW - 3$" **then**
8: $cost_{total} = cost_{total} + \lceil graph_{ETX}[sender_{hop}][receiver_{hop}] \rceil$
9: **end if**
10: **end for**
11: **if** $strategy =$ "$SW - 2$" **then**
12: $cost_{total} = \lceil cost_{total} \rceil$
13: **end if**
14: $\#transmissions \leftarrow n * cost_{total}$
15: $window\_size \leftarrow 2 + \#transmissions - length_{flow}$
16: $flow_{new} \leftarrow$ list of $\#transmissions$ lists
17: **for** $i = 0$ to $length_{flow} - 1$ **do**
18: **if** $i = 0$ **then**
19: $slots \leftarrow$ list $[0 \,..\, window\_size - 1]$
20: **else if** $i = (length_{flow} - 1)$ **then**
21: $slots \leftarrow$ list $[i - 1 \,..\, i + window\_size - 2]$
22: **else**
23: $slots \leftarrow$ list $[i - 1 \,..\, window\_size - 1]$
24: **end if**
25: **for** $slot$ in $slots$ **do**
26: extend $flow_{new}[slot]$ by $flow[i]$
27: **end for**
28: **end for**
29: **return** $flow_{new}$

Windows, on the other hand, allows choosing the number of transmissions based on a metric, in our case, the ETX metric. Moreover, Sliding Windows allows a different number of transmissions for each flow in the same network due to its use of the ETX metric. Because Sliding Windows is based on link qualities, we argue that it offers better adaptability to a network's link characteristics during the scheduling process.

2.3.3 Time Synchronization

Stable time synchronization is essential for TSCH networks. It ensures that clocks do not drift apart, and nodes wake-up for transmissions and reception within the guard times specified by TSCH. Master achieves this by building a clock synchronization tree from the root as part of the scheduling process. Similar to the routing of the flows, a minimal spanning tree with ETX as metric and with the coordinator of the TSCH network as root is computed using Dijkstra's algorithm. This tree assigns each node a parent node for clock synchronization.

2.3.4 System Design

Next, we detail on the system architecture of Master. It consists of both the external scheduler and the routing layer on each node (see Figure 2.1). Here we put a particular focus on the integration with TSCH and Contiki-NG [11].

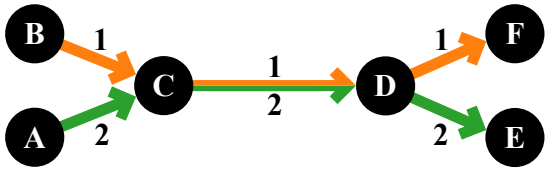


Figure 2.4: Example of 2 flows sharing a common link between nodes C and D.

2.3.4.1 Central Logic of Master

The central logic of MASTER consists of a centralized router and scheduler with all the functionality described above. We implement MASTER in Python to enable easy extendability and rapid experimentation of new routing, transmission, and scheduling strategies.

2.3.4.2 Schedule Distribution

For schedule distribution, MASTER can work together with most schedule distributors (e.g., plexi [71]), as scheduling and distributing the schedule are orthogonal. Moreover, it can also directly upload schedules via the serial port for rapid experimentation.

2.3.4.3 Per node routing layer

The routing layer of MASTER has multiple functions: it performs neighbor discovery (Section 2.3.4.5), implements the schedule, and adds a routing header to the communication payload to be compliant with the lower layers as well as relaying the packet to the next hop (Section 2.3.4.6). We place it in the Contiki-NG network stack above TSCH, see Figure 2.1, and implement it in C.

2.3.4.4 Contiki-NG/TSCH Extensions

To match the requirements of MASTER and its scheduling algorithm, we extend the elements of TSCH and its implementation in Contiki-NG: (1) the packet buffer implementation and (2) the TSCH queues.

In the packet buffer, we add fields to store the flow identifier and the time to live of a transmission. With these two fields, the TSCH stack and MASTER can map incoming packets to flows and thereby follow the global schedule on each node. We extend the TSCH queue to enable a transmission order differing from the reception order at a node, e.g., the forwarding of a packet to a specific neighbor before forwarding an earlier received packet to the same neighbor. To allow this behavior, we add flow-based queues, in addition to the neighbor-based queues of TSCH. We realize the flow-based queues through the use of virtual neighbors.

Figure 2.4 illustrates why neighbor-based queues as used by Contiki-NG cannot be practicably used by MASTER. If packet 2 is received by node C first, but packet 1 has an earlier deadline, packet 1 will be stuck behind packet 2 until the first is transmitted to node D. With flow-based queues, packets 1 and 2 will be added to different queues at C. Therefore, they are independent of each other and packet 1 can be forwarded first.

This new queue design increases the schedulability of the presented scheduler, which is crucial for deadline-dependent systems. It also decreases the latency in networks that are not deadline-critical by reducing congestion at bottlenecks of the network. Moreover, it allows us to use scheduling algorithms initially developed for process scheduling, a domain without these congestion problems.

2.3.4.5 Neighbor Discovery and Bootstrapping

Before MASTER can build any schedule, it requires information about all links between the nodes in the network. Thus, to bootstrap and collect topology information with MASTER, we deploy a custom, topology agnostic schedule only designed for neighbor discovery. In this schedule, we use one independent transmission slot per node present in the network. This neighbor discovery schedule is similar to the sender-based operation mode of the autonomous scheduler Orchestra [3]. Each node sends a numbered broadcast in its active slot and listens in all other slots for broadcasts of other nodes in its surroundings.

Please note that this schedule only serves for bootstrapping. After deployment of the actual transmission schedule, the task of probing neighbors becomes part of the normal TSCH beaconing process. Nodes collect this information for any potential later update of the schedule.

2.3.4.6 Header format

MASTER routes packets based on flows, and as a result, we add a custom routing header. The routing layer of MASTER adds a 7-byte routing header to each packet. This header contains a flow identifier (1 byte), a sequence number (2 bytes), the time-to-live (TTL) (2 bytes), and the earliest TSCH transmission slot (2 bytes). The header is necessary for nodes to know whether they are the receiver of the packet or a forwarder. Moreover, the header specifies, where to forward the packet to, and whether there is still time left for forwarding. In practice, our header replaces the IPv6 header which we could use instead in a system using the full IPv6 stack.

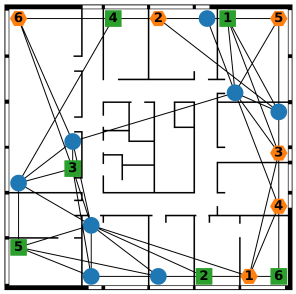
2.4 Evaluation

In this section, we evaluate the performance of MASTER and compare it to the state-of-the-art. We begin by evaluating our newly proposed flow-based scheduling policy and compare it to state-of-the-art scheduling policies, including a baseline strategy without retransmissions (cf. TASA [4]) and a slot-based transmission strategy (cf. AMUS [17]). Next, we compare MASTER to Orchestra, the default autonomous scheduler in Contiki-NG, which also builds on TSCH. Finally, we evaluate MASTER's ability to compose long-term stable schedules.

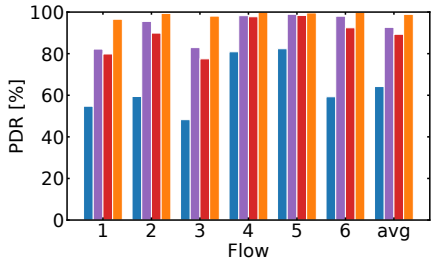
2.4.1 Evaluation Setup

2.4.1.1 Testbed

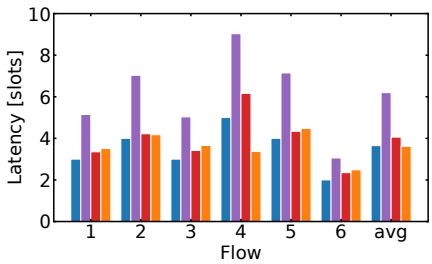
We run on a 20 node testbed deployed in offices and student lab rooms, see Figure 2.5a. It is located on the top most floor of a university building with



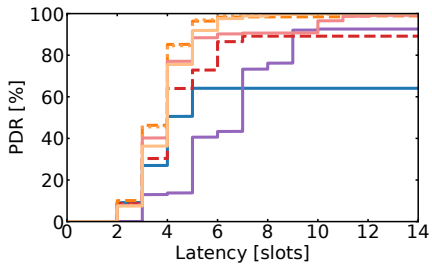
(a) 500 m^2 testbed of 20 nodes at Kiel University. Source nodes: orange hexagons; Sink nodes: green squares; Relay-only nodes: blue circles; Numbers: corresponding flow



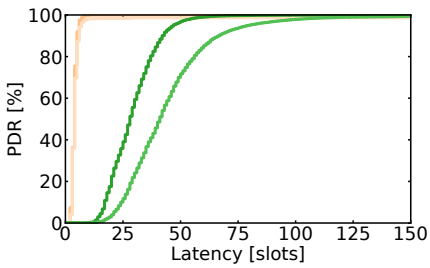
(b) Reliability of MASTER's transmission strategies: baseline, slot-based, SW-2-1 and SW-3-1.



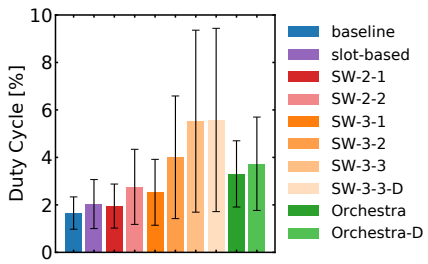
(c) Latency of MASTER's transmission strategies: baseline, slot-based, SW-2-1 and SW-3-1.



(d) Combined latency and reliability CDF of MASTER's transmission strategies.



(e) Combined latency and reliability CDF of MASTER's transmission strategy SW-3-3 and Orchestra at nighttime and daytime.



(f) Duty cycle of MASTER and Orchestra.

Figure 2.5: Evaluation of MASTER's transmission strategies and comparison to Orchestra. SW-3 outperforms all other strategies reliability-wise and outperforms Orchestra latency-wise. We display the legend of figures 2.5b - 2.5f in Figure 2.5f.

spanning an area of 500 m^2. The testbed shares the wireless spectrum with WiFi and Bluetooth communications outside of our control. Due to this, the testbed is exposed to high levels of interference, especially during work hours.

2.4.1.2 Metrics, Comparison, and Duration

We evaluate our scheduler in terms of end-to-end reliability, end-to-end latency, as well as network energy consumption. We measure these metrics for different centralized scheduling approaches with and without retransmissions. Moreover, we compare our scheduler with the autonomous scheduler Orchestra [3]. These comparisons are based on 2-hour experiments for each strategy, except for the long-term stability evaluation in Section 2.4.5, which has a duration of 24-hours per experiment.

2.4.1.3 Implementation

We implement MASTER for Contiki-NG [11]. We target the Zoul Firefly platform, featuring a 32 MHz 32-bit CC2538 Cortex-M3 CPU, 32 KB of RAM, 512 KB of flash, with an IEEE 802.15.4 compatible radio.

2.4.1.4 Channels

Due to the high levels of interference, we use only the four channels (15, 20, 25, and 26), defined in the standard four-channel TSCH hopping sequence. Furthermore, Orchestra uses by default only these four channels as well.

2.4.1.5 Application Payload and Overhead

For all experiments, we include a 64-byte randomly generated data payload, a medium packet size supported by TSCH. In addition to this data payload, MASTER adds its 7-byte routing header independent of the specific scheduling policy. Orchestra, on the other hand, uses the IPv6 headers and requires additional network layer control traffic.

2.4.1.6 Notations

Throughout the evaluation, we use the following naming scheme: The baseline strategy without retransmissions we call *baseline*, and the slot-based retransmission strategy (as used by many state-of-art schedulers) with $\lceil ETX_{link} \rceil$ transmissions per link we label *slot-based*. The Sliding Windows strategies use the naming scheme we present in Section 2.3.2.1. Experiments performed during daytime are extended by the marker *-D*.

2.4.2 Baselines

We compare MASTER's Sliding Windows policies to three other scheduling policies. These are MASTER's baseline strategy without retransmissions, MASTER's slot-based retransmission strategy, and the autonomous scheduler Orchestra [3]. The design of the baseline strategy is based on the transmission policy used in, e.g., TASA [4], and uses one distinct slot per hop. The slot-based strategy is inspired by policies presented in several recent publications, including AMUS [17].

Contrary to most of these, our design performs all possible retransmissions of a hop before proceeding to the next hop, which favors high reliability over low latency contrary to AMUS's approach. Moreover, to be in line with our Sliding Windows strategies, MASTER's slot-based strategy uses an ETX-based number of retransmissions per link ($\lceil ETX_{link} \rceil$). Lastly, we use Orchestra to compare our centralized routing and scheduling solution to distributedly routed and autonomously scheduled solutions to verify the adaptability of MASTER to dynamic environments predestined for distributed policies.

2.4.3 Performance of Master's transmission strategies

We first evaluate the performance of different transmission strategies supported by our scheduler. We compare the Sliding Windows transmission strategy with a baseline strategy without retransmissions and with the traditional slot-based retransmission strategy mentioned above. We run experiments with six scheduled flows, a number of flows used at a recent EWSN dependability competition [72]. The flows have a length of 2 to 4 hops each. Each flow has a sole source and destination node. Each source node generates a packet roughly every second with a configured time to live of one second. The length of the communication slotframes of 1 second corresponds roughly with 101 slots.

Figure 2.5b shows the reliability of transmission approaches scheduled with MASTER. The transmission approaches include the baseline and slot-based strategy, as well as Sliding Windows transmission strategies SW-2-1 and SW-3-1; see Section 2.3.2.1 for notations. The latter of the two Sliding Windows strategies has the same number of transmissions per flow as the slot-based strategy.

All strategies with retransmissions clearly outperform the baseline without retransmissions, which shows the presence of interference in the used channels. The slot-based strategy reaches an average reliability of 92.7% whereas the Sliding Windows strategies reach average reliabilities of 89.3% and 98.9%, respectively. The SW-2-1 strategy has for all flows lower reliability than the slot-based strategy, but the number of scheduled slots per flow is only by one larger than the baseline number of slots, see Table 2.1. The SW-3-1 strategy outperforms all other strategies while using no more slots per flow than the slot-based strategy. Its least reliable flow achieves a packet delivery rate (PDR) of 98.1% while the slot-based strategy drops as low as 82.2%.

We can model this superiority of SW-3-1 over SW-2-1 and over the other strategies mathematically using the probability mass function of the binomial distribution [73]:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{2.4}$$

This probabilistic model also explains the lower reliability of SW-2-1 compared to the slot-based strategy.

As an example, we consider a flow of three hops ($n = 3$), e.g., the green flow in Figure 2.3a, with the same ETX value for each link of 1.2 ($p = \frac{5}{6}$). Thus, the number of transmissions for SW-2-1 and SW-3-1 are 4 and 6 slots, respectively. The expected PDRs for SW-2-1 and SW-3-1 are $P(X = 3) + P(X = 4) \approx 0.868$ and $P(X = 3) + P(X = 4) + P(X = 5) + P(X = 6) \approx 0.991$, respectively. Likewise, the expected PDR for the baseline, is $P(X = 3) \approx 0.579$. The

Table 2.1: Summary of the results plotted in Figure 2.5c: Maximum latency (slots) for each flow and for flow 4 maximum number slots active in parentheses.

Flow	Baseline	Slot-Based	SW-2-1	SW-3-1
1	2	4	3	4
2	3	6	4	6
3	3	6	4	6
4	5 (3)	10 (6)	7 (4)	12 (6)
5	4	8	5	8
6	4	8	5	8

slot-based strategy can be seen as 3 independent, subsequent chains of two binomial trials each ($n = 2, k \geq 1$). This results in an expected PDR of $(P(X = 1) + P(X = 2))^3 = 0.919$. These mathematical results confirm the trend we see in Figure 2.5b.

Latency-wise, both Sliding Windows strategies perform much better than the slot-based strategy. Moreover, their latency is minimally higher than the latency of a strategy without retransmissions (see Figure 2.5c), which, in turn, has a high packet loss rate. It appears that SW-3-1 has a lower latency for flow 4 than the baseline. Contrary to all other flows, flow 4's schedule contains more slots than active slots throughout all strategies. Due to the flow-based approach of SW-3-1 and a large enough number of continuous active slots at the beginning of the schedule, most packets were received within a few slots, leading to a latency lower than the baseline's one. Table 2.1 shows that the maximal number of active slots is still smaller for the baseline strategy.

Figure 2.5d visualizes the latency and reliability of a wider range of transmission strategies. Solid lines represent the baseline, the slot-based, the SW-2-2, and the SW-3-3 strategies. For the Sliding Windows strategies SW-2-1 and SW-3-1, the figure uses dashed lines, and for the SW-3-2 strategy, it uses a dotted line. The figure shows that the slot-based strategy is the worst latency-wise. The SW-3 Sliding Windows strategies are superior to the other Sliding Windows strategies (SW-2). The superior strategies with a scaling factor of 2 and 3, both perform well. The strategy with the higher scaling factor reaches the maximal possible reliability. Therefore, we use the Sliding Windows strategy SW-3-3 for the following comparison to Orchestra.

The duty-cycle evaluation in Figure 2.5f shows a higher radio on-time for a higher number of scheduled slots. SW-3-3 has a radio on-time of up to 11.95% for a node with a lot of traffic.

2.4.4 Master vs. Orchestra

We now evaluate the performance of Master in comparison to Orchestra, the default, autonomous scheduler of TSCH in Contiki-NG. We use Orchestra as is, with a receiver-based schedule of length 7 in non-storing mode. We schedule the same six flows used before. As transmission strategy for Master, we use the one with the highest reliability of those presented above (SW-3-3). To provide detailed information on the performance, we present runs of both Master and Orchestra during nighttime as well as during office hours in the daytime. Figure 2.5e shows the latency and reliability of the four experiments. Master's

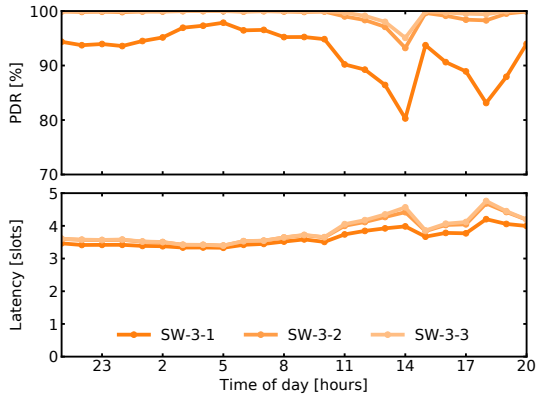


Figure 2.6: Reliability and latency evaluation of Sliding Windows according to Equation (2.3) for all 3 scaling factors. Each value corresponds with the hour, that started at the given time. Note, that the y-axis of the PDR plot does not begin at zero.

latency is drastically shorter than the latency of Orchestra with a mean latency of 3.9 and 4.2 slots compared to 25.9 and 40.9 slots during nighttime and daytime, respectively, while reaching similar reliability. The four rightmost columns in Figure 2.5f show the duty cycle for the experiments included in this section of the evaluation. Orchestra has on average a two percentage points lower duty cycle than Master (3.52% vs. 5.55%) and the maximum duty cycle of a node of four percentage points lower (7.73% vs 11.95%). As each node in Orchestra is only able to use every seventh slot, the possible duty cycle is automatically lower than the one for Master. However, this lower duty cycle results in much higher latency, as presented above.

2.4.5 Long-term stability of Master

In the last part of our evaluation, we investigate Master's long-term stability. In Figure 2.6, we present the reliability and latency of the SW-3 Sliding Windows strategies for 24 hours (Day 1, 21:00 - Day 2, 21:00) during workdays. During the night and the early morning, both SW-3-2 and SW-3-3 reach a PDR of above 99.99% and an average latency of around 3.5 slots. Between 14:00 and 15:00, the reliability drops for all strategies to 95%, 93.2%, and 80.3%, respectively, under a slight latency increase. During this time, a group of students entered the lab, leading to a drastic increase in WiFi and BLE traffic and thereby an interference level increase. Another reliability drop, mainly for SW-3-1, is visible at the end of the working day. Over the whole period of 24 hours, the average reliability of SW-3-1, SW-3-2, and SW-3-3 is 99.6%, 99.2%, and 92.5%, respectively. The high average reliability, as well as the reliability recovery after times of high interference, validates Master's long-term stability.

2.5 Related Work

We first discuss centralized schedulers and algorithms, followed by a discussion of autonomous scheduling solutions.

After the introduction of TSCH, TASA [4] was one of the first central scheduling algorithms proposed. It is traffic aware, yet like other papers focusing on scheduling algorithms like C-LLF [5], it assumes the availability of interference-free channels and, therefore, does not include retransmissions. Saifullah et al. [5] and Gunatilaka et al. [16] focus in their work on the highest possible schedulability for a large amount of communications meeting deadlines but not much on the network reliability. AMUS [17] is one of the protocols for TSCH that includes slot-based retransmissions. It schedules additional resources for vulnerable links and allocates backup slots in empty cells of the scheduler. Rugamba et al. [20] build another centralized scheduler based on a path collision-aware least-laxity first scheduling algorithm by Darbandi et al. [19]. Moreover, Rugamba et al. describe a method of distributing a centrally computed schedule. The first approach of moving from slot-based retransmissions to flow-based ones is the flow-centric policy (FCP) [21]. The authors present a dynamic approach of retransmissions not fixed to specific links. This approach is similar to the transmission strategy of Sliding Windows presented in this paper. We discuss the differences between the two in Section 2.3.2.3.

Besides the advances regarding scheduling, Wu et al. [44] present advances in the field of centralized routing in combination with central scheduling. The authors present a conflict aware real-time routing approach, that is aware of scheduling decisions and the possible conflicts of routed paths. Li et al. [43] take a different, asymmetric approach in routing by applying different routing strategies for different communications in one network.

Related to these central scheduling and routing approaches, are systems focusing on network softwarization. plexi [71] is a framework exposing TSCH network resources through a web interface and allowing the rescheduling of communications. Similarly, Baddeley et al. [74] and Galluccio et al. [75] present SDN solutions for Wireless Sensor Networks for network monitoring and reconfiguration. These SDN solutions are conceptually in line with central schedulers calculating schedules externally. Moreover, a combination of our work with SDN solutions is imaginable.

Next to the centralized approaches, a significant focus of recent work is on autonomous scheduling, a concept introduced by Orchestra [3]. Orchestra, as well as Alice [35] and DiGS [32] are autonomous solutions for TSCH, as they do not require neither any central infrastructure nor the exchange of data to build a schedule and achieve high reliabilities of 99.999%. However, autonomous schedulers are not able to achieve this reliability with latency guarantees necessary for many industrial applications as they have no knowledge on the underlying topology.

2.6 Conclusion

This paper introduces Master, a central scheduling solution for TSCH networks. Master introduces a novel Sliding Windows transmission strategy

and achieves high reliability independent of knowing the optimal amount of retransmissions per link. Instead, it schedules a number of retransmissions for a flow that can be used at all links of a flow where necessary. The key idea is enabling centralized schedulers to adapt to interference changes without the need for rescheduling while keeping the lowest possible latency. Thus, eliminating a significant overhead of traditional central schedulers.

We implement MASTER in Contiki-NG and evaluate it extensively on a testbed in an environment susceptible to interference. We demonstrate MASTER's practicality and ability to keep stability for over 24 hours and achieve latencies much smaller than Orchestra while achieving similar reliability.

As part of future work, we plan to investigate the challenges of neighbor data collection and schedule distribution to provide a comprehensive central scheduling solution. Moreover, we are planning to evaluate the use of centralized schedulers in harsh wireless environments, such as the ones used in the EWSN dependability competitions [72].

3

(POSTER) Overtake: Opportunistic Routing and Concurrent Transmissions for TSCH

Oliver Harms, Olaf Landsiedel

Proceedings of the 16th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2020, pp. 141–143.

PAPER B

Abstract

In this paper, we present Overtake, an opportunistic routing protocol for Time-Slotted Channel Hopping (TSCH). Overtake combines (1) opportunistic routing, (2) concurrent transmissions and (3) TSCH. We show that this novel combination enables low-latency, central scheduling withstanding node failures. Our initial results show its ability to withstand node failures of up to 40% of nodes of a flow while keeping minimal latency.

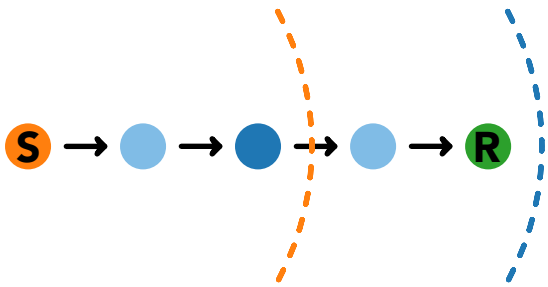


Figure 3.1: Nodes included in each slot. The dashed lines represent the transmit ranges of the sender and the second relay node, respectively. If every node can reach the next two nodes, the minimal number of communication slots is 2 resulting in the first 3 nodes active in the first slot and the remaining nodes active from the second slot.

3.1 Introduction

Context: Applications in the context of the Industrial Internet of Things (IIoT) require high communication reliability and low latency. To achieve this goal, commonly, a central scheduler for TSCH [8] computes optimal routes (single-path) and schedules end-to-end communication flows incorporating possible retransmissions.

Challenge: These single-path strategies (e.g. Sliding Windows (MASTER) [66]) achieve the required reliability under usual amounts of interference. However, if a node of a scheduled flow fails, existing protocols fail. Moreover, central schedules are generally computed for worst-case situations, meeting worst-case latencies. In many situations, a communication flow could reach its destination with a fewer number of hops. Using less hops would reduce the average latency and the consumed energy.

Protocols overcoming the challenge of node failures do not forward a packet to a single node but apply an anycast approach instead. Known approaches are opportunistic routing (cf. Landsiedel et al. [51]) as well as flooding-based protocols (e.g., LWB [6]). Opportunistic routing allows flexibility in the forwarding process, as no fixed link has to communicate. A problem of opportunistic routing is choosing a node forwarding the packet. Flooding-based protocols, on the other hand, cover the whole network with a communication. They use highly time-synchronized concurrent transmissions eliminating the selection of a forwarder.

Approach: In this paper, we introduce OVERTAKE, an opportunistic routing strategy for Wireless Sensor-Actuator Networks using concurrent transmissions. We specifically target networks utilizing the IEEE 802.15.4 TSCH (Time-Slotted Channel Hopping) MAC protocol [8]. OVERTAKE combines the three approaches of opportunistic routing, concurrent transmissions, and TSCH. Through this combination, we can create single-path-based schedules withstanding node failures.

We build OVERTAKE on top of the centralized scheduler MASTER [66] and its Sliding Windows transmission strategy. The combination of opportunistic

routing mentioned above with flow-based retransmissions allows Overtake not only to withstand node failures but offers a significant robustness improvement over Master's current performance.

Our initial evaluation shows that with Overtake, Master can schedule communication even more stable than in its default configuration. Moreover, Overtake reduces the end-to-end latency of communication in environments susceptible to interference. Overall, we make three contributions:

- We present Overtake, a single-path-based concurrent, opportunistic-routing strategy, extending Sliding Windows, achieving lower latency and resilience to node failures.
- We propose acknowledgment supported opportunistic transmissions for TSCH to enable reliable routing of concurrent communication.
- We implement Overtake as a module of Master. We present initial results showing Overtake's latency improvement on Master's schedules. Moreover, we show the resilience of Overtake to node failures.

3.2 Design

In this section, we present Overtake's design. Moreover, we discuss our design modifications of Master enabling concurrent transmissions, as well as the TSCH modifications concerning opportunistic routing.

3.2.1 Overtake

Overtake brings concurrent, opportunistic routing to TSCH. Overtake extends Master's transmission strategy of Sliding Windows with concurrent transmissions. We achieve opportunistic routing by sending a packet not to a single specified neighbor, but to all active participants of a flow instead. A flow is the set of nodes a packet traverses from its sender to its receiver. All nodes of a flow that successfully receive a packet, transmit it in the next slot concurrently.

We introduce our opportunistic routing approach in more detail through the following example. In Fig. 3.1, we show an example flow of 5 nodes with the most cost-effective path taking four hops. However, the communication range of each node covers the next two nodes, as represented by the dashed lines. With opportunistic routing and a schedule like the one given in Fig. 3.2, a successful transmission might be possible within two slots. The given schedule includes three retransmissions. If a packet overtakes a node, so to say, it arrives at a node earlier than taking each hop of the flow, the remaining section of the flow dynamically gains a retransmission per overtake. These additional retransmissions increase the end-to-end reception likelihood and add flexibility.

As we are combing opportunistic routing with concurrent transmissions, we do not have to select a specific forwarder if several nodes received a packet, nor do we have to use a collision avoidance mechanism.

	1	2	3	4	5	6	7
S	TX	TX	TX	TX			
1	RX	RXTX	RXTX	RXTX	TX		
2	RX	RXTX	RXTX	RXTX	RXTX	TX	
3		RX	RXTX	RXTX	RXTX	RXTX	TX
R		RX	RX	RX	RX	RX	RX

Figure 3.2: Example Schedule for Overtake with 3 retransmissions according to the topology presented in Fig. 3.1. A node stays in receive mode (RX) until it receives a packet and transmits (TX) it starting the next slot.

3.2.2 System Design

We design the Overtake routing strategy for Master and TSCH. As both of them do not natively support opportunistic routing or concurrent transmissions, we discuss, in this section, the integration of Overtake into Master and TSCH:

3.2.2.1 Master extensions

We extend Master's routing header by a rank uniquely identifying a node's position in a flow. This rank allows a receiver to determine whether the received packet is further down the flow or not. Moreover, Master sends packets using Overtake directly to a flow address instead of a specific node address. These extensions enable the TSCH anycast described below.

3.2.2.2 TSCH extensions

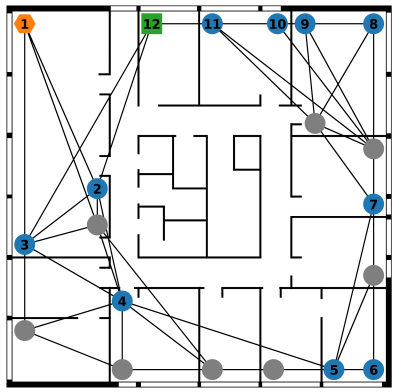
TSCH generally supports broadcast as well as unicast communication. However, to perform opportunistic routing, it is necessary to have support for anycast followed by acknowledgments.

We realize anycast communication in TSCH by transmitting to a flow instead of a neighbor. As multiple nodes belong to a flow, each active node of the flow receiving the packet accepts and potentially acknowledges its reception. If multiple nodes receive a packet, only nodes that are closer to the flow's receiver, then the current sender should acknowledge the reception to ensure a successful packet forwarding. We base the decision on the node's unique rank determined by the central scheduler Master.

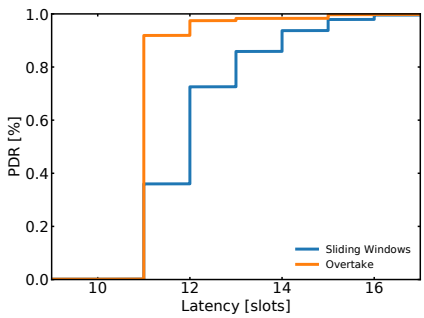
The multiple nodes accepting the packet, and the modified acknowledgment behavior enable successful anycast communication in TSCH.

3.3 Evaluation

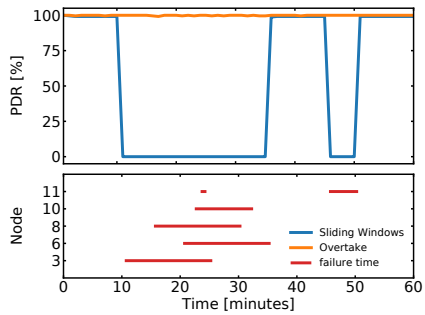
In this section, we evaluate the performance of Overtake in comparison with the Sliding Windows transmission strategy. Moreover, we show Overtake's possibility to withstand node failures.



(a) A testbed of 20 nodes at Kiel University. Source node: orange hexagon; Sink node: green square; Relay nodes: blue circles; Non participating nodes: grey circles; Numbers: Position in flow; Lines: connectivity > 70%.



(b) Combined latency-reliability CDF of Sliding Windows and OVERTAKE. OVERTAKE outperforms Sliding Windows latency-wise and achieves a 0.25 percentage points higher reliability of 99.80%.



(c) Reliability over time of Sliding Windows and OVERTAKE under node failures. OVERTAKE is capable of withstanding all node failures while Sliding Windows is not able to withstand any node failure.

Figure 3.3: Evaluation showing OVERTAKE's superiority over the Sliding Windows transmission strategy.

3.3.1 Evaluation Setup

3.3.1.1 Metrics and Comparison

We evaluate the two transmission approaches, OVERTAKE and Sliding Windows, concerning end-to-end latency and end-to-end reliability. For the proof of concept of withstanding node failures, we present the end-to-end reliability for various numbers of node failures.

3.3.1.2 Implementation

We implement OVERTAKE for Contiki-NG [11]. We target the Zolertia Zoul Firefly platform featuring a 32 MHz 32-bit CC2538 Cortex-M3 CPU, 32 KB of

RAM, 512 KB of flash, including an IEEE 802.15.4 compatible radio.

3.3.1.3 Testbed

We evaluate on a 20 node, 500 m^2 testbed deployed in student lab rooms and offices, see Fig. 3.3a. The testbed is exposed to interference, as it shares the wireless spectrum with Bluetooth and WiFi communication outside of our control.

3.3.1.4 Channels and Application Payload

Due to the significant amount of interference in the testbed's environment, we use the default four channel hopping sequence of TSCH (channels 15, 20, 25, and 26). As we use only one flow, the channel offset for all communication is the same. In addition to the frame headers, and an 8-byte routing header, we include a 64 byte randomly generated application payload, a medium packet size supported by TSCH.

3.3.2 Overtake vs. Sliding Windows

We begin our evaluation by comparing the performance of Overtake with the performance of Master's default transmission strategy Sliding Windows. Throughout this evaluation, we use the following scenario. We run experiments with a single flow covering the entire floor in a counterclockwise circle. The flow consists of 11 hops from node 1 to 12 (see Fig. 3.3a), using a transmit power of -7 dBm. To allow easy comparison between Overtake and Sliding Windows, we use the same schedule for both strategies, created and optimized for Sliding Windows with six possible retransmissions. Contrary to the schedule presented in Fig. 3.2, the schedule used here, starts with two active nodes and increases the number of active nodes from hop to hop, until it reaches up to 7 active nodes in slot 7. It allows up to 7 simultaneously active nodes and a minimum latency of 11 slots. With this schedule, only a small latency improvement is possible. However, allowing more than 7 active nodes at a time would lead to unrealistic latencies and hop counts due to the circular shape of the scheduled flow. All experiments have a run-time of 1 hour and send a new packet every 250 ms.

Fig. 3.3b shows the performance of Sliding Windows and Overtake for the presented scenario. While Overtake offers only a small reliability improvement over Sliding Windows of 0.25 percentage points (99.80% vs. 99.55%), Overtake's average latency of 11.1 slots is almost one slot smaller than the average latency of Sliding Windows. Moreover, a significant number of packets were received after the minimal possible latency of 11 slots. As the schedule was not optimized for Overtake, an optimal schedule should result in even lower latency.

3.3.3 Overtake under node failures

After comparing the general performance of Overtake and Sliding Windows, we evaluate their respective performance in case of node failures. The general scenario is the same as described above. However, up to 5 nodes cease operation

at different times, leading to up to 5 non-responding nodes at a given time. Fig. 3.3c shows the results of this experiment. While Sliding Windows is highly affected by any node failure, OVERTAKE is, in the given scenario, not at all influenced by the occurring node failures. It shows that there are at all times enough longer communication distances available, overtaking one or two nodes, which can be used by OVERTAKE while Sliding Windows has to stick to its single-path schedule. If no or not enough node overtakes were possible, OVERTAKE would be affected by node failures as well. Nevertheless, it is clearly visible that OVERTAKE offers much higher stability compared to Sliding Windows, as long as each node can reach more nodes than its immediate neighbors.

3.4 Conclusion & Future Work

This paper introduces OVERTAKE an opportunistic routing and concurrent communication strategy for (centrally scheduled) TSCH networks. Instead of relying on traditional unicast or broadcast communication, OVERTAKE introduces concurrent opportunistic communication in TSCH. OVERTAKE is capable of reaching high reliability and low latency with early results showing a significant latency decrease compared with Sliding Windows. Moreover, we show OVERTAKE's feasibility of withstanding node failures.

We implement OVERTAKE for MASTER, a centralized scheduler for TSCH based on Contiki-NG. We demonstrate the practicality of concurrent transmissions in the context of TSCH networks and its ability to increase the stability of schedules created by MASTER.

As future work, we intend to extend MASTER by a multi-path communication strategy building upon OVERTAKE called AUTOBAHN. This strategy should not only be able to withstand node failures but even withstand higher levels of interference present in harsh wireless environments.

4

Opportunistic Routing and Synchronous Transmissions Meet TSCH

Oliver Harms, Olaf Landsiedel

Proceedings of the 46th IEEE Conference on Local Computer Networks (LCN), 2021, pp. 107–114.

PAPER C

Abstract

Low-power wireless networking commonly uses either Time-Slotted Channel Hopping (TSCH), synchronous transmissions, or opportunistic routing. All three of these different, orthogonal approaches strive for efficient and reliable communication but follow different trajectories. With this paper, we combine these concepts into one protocol: Autobahn.

Autobahn merges TSCH scheduling with opportunistically routed, synchronous transmissions. This opens the possibility to create long-term stable schedules overcoming local interference. We prove the stability of schedules over several days in our experimental evaluation. Moreover, Autobahn outperforms the autonomous scheduler Orchestra under interference in terms of reliability by 13.9 percentage points and in terms of latency by a factor of 9 under a minor duty cycle increase of 2.1 percentage points.

4.1 Introduction

Within the past 20 years, research on low-power wireless networking resulted in a multitude of different protocols. They fall into three prominent fields: Time-Slotted Channel Hopping (TSCH), opportunistic routing, and synchronous transmissions. So far, all three of these fields have little to no overlap, while all strive for a common goal of stable, reliable communication in low-power wireless networks.

In the first field of protocols, the IEEE 802.15.4 Time-Slotted Channel Hopping (TSCH) [8] MAC layer protocol forms the basis for many routed communication protocols. This protocol is standardized and dominates the industry. One category of TSCH protocols uses centralized schedulers, separating the network communication from the routing and scheduling. In recent works [21,66], centralized schedulers show high reliability and stability. Another category are autonomous schedulers with Orchestra [3] as a prominent example.

TSCH protocols offer stability regarding narrow-band interference. However, long-term stable schedules that are prone to wide-band interference are an open challenge. Wide-band interference likely leads to link failures or even node failures heavily affecting routed communication.

The other two fields can overcome these challenges. Opportunistic routing [48, 51, 52] utilizes anycasts instead of unicasts to add forwarding flexibility by addressing a packet to multiple potential forwarders. It increases the possibility of successful reception in the presence of wireless link dynamics. Protocols building upon synchronous transmissions [6, 59, 60] allow multiple nodes to transmit packets concurrently, commonly by network-wide flooding.

Synchronous transmissions achieve high reliability even in the presence of wide-band interference. However, they have an impact on all nodes in a network. If, for example, in a 1000 node network, two nodes two-hops apart want to communicate, the whole network is involved. In a routed network, only a fraction of these nodes needs to communicate.

In this paper, we ask the following question: Can we combine the benefits of opportunistic routing, synchronous transmissions and centralized TSCH scheduling? For this, we introduce Autobahn: a hybrid routing scheme that combines the best of these worlds: centrally scheduled flows and one-to-one routing of packets as in traditional networking combined with the reliability and robustness of opportunistic routing and synchronous transmissions.

The basic concept of Autobahn is as follows: Its central scheduler schedules a flow along a wider path and allows neighboring nodes to transmit concurrently the same data at the same timeslot and frequency. Thus, a node forwards a packet opportunistically to multiple neighboring nodes, which in turn, in the next slot, concurrently forward opportunistically to their neighbors. In our evaluation, we show that by combining these three approaches, Autobahn efficiently provides reliable, low-latency packet delivery even when links fail, and its schedules are stable for days even in the presence of dynamic interference.

Overall, this paper makes the following contributions:

- We are the first to combine the concepts of opportunistic routing, synchronous transmissions, Time-Slotted Channel Hopping (TSCH) into a single protocol to achieve long-term stable routed communication.

- We design AUTOBAHN, a robust scheduling and routing policy that withstands link and node failures in the presence of interference.
- We implement AUTOBAHN for Contiki-NG [11] and evaluate it in environments susceptible to interference. We show the long-term stability of schedules using AUTOBAHN over 12 days and under various interference levels for 25 hours. These experiments achieve reliability under interference of 96.8% and latency of 4.2 slots outperforming both the central scheduler MASTER [66] and the autonomous TSCH scheduler Orchestra [3].

The remainder of this paper is organized as follows. Section 4.2 gives the necessary background information and reviews related work on TSCH as well as the concepts combined in AUTOBAHN. In Section 4.3, we introduce the design of AUTOBAHN. In Section 4.4 we evaluate AUTOBAHN's performance experimentally, followed by the conclusion in Section 4.5.

4.2 Background & Related Work

In this section, we introduce the necessary background on TSCH, opportunistic routing, and concurrent transmissions and discuss the relevant related work.

4.2.1 Time-Slotted Channel Hopping (TSCH)

The MAC protocol Time-Slotted Channel Hopping (TSCH) [8] is a combined TDMA and FDMA MAC protocol. It uses 10 ms long time slots with up to 16 frequency channels at each time slot. All active channels follow a pseudo-random hopping sequence that is cycled through, using a different channel at each timeslot to counteract narrow-band interference.

TSCH groups communication slots in continuously repeated slot-frames. All slot-frames together form the TSCH schedule. A TSCH schedule is generated by a centralized, autonomous, or distributed scheduler.

Centralized Scheduling: Central schedulers use global knowledge about the network topology (esp. wireless link quality) to build a schedule and disseminate the schedule into the network. Many early ones, such as TASA [4] and others [5,16], assume interference-free wireless channels without lossy links and, therefore, do not include retransmissions in their schedules. Later work focuses on increasing reliability in the presence of fading channels while ensuring end-to-end latency requirements of each flow. They achieve this by adding retransmissions, i.e., slot-based retransmissions, as used by AMUS [17], to the schedule. As interference can rarely be linked to a specific location beforehand, some recent works by Brummet et al. [21], and MASTER [66] introduce a new approach to retransmissions in TSCH scheduling: they introduce flow-based retransmissions achieving lower latency and a higher degree of adaptability to local interference level.

Autonomous/Distributed Scheduling: Next to these centralized TSCH protocols, a significant amount of work concentrates on autonomous scheduling, a concept introduced by Orchestra [3] and extended by others [32, 35]. Distributed scheduling on the other hand builds on 6TiSCH with its default

scheduling function MSF [76], as well as, LLSF [26] and LDSF [28], focusing on improving latency in distributed TSCH.

Multipath TSCH: For multi-path communication in TSCH, several algorithms [45, 77] were studied for distributed and centralized scheduling scenarios. To some extent, these works propose similar ideas as AUTOBAHN, yet they clearly stay within the specifications of TSCH and do not apply opportunistic routing or synchronous transmissions. Moreover, their evaluation results are solely based on simulation.

4.2.2 Opportunistic Routing

Opportunistic routing is a routing approach to improve network throughput, communication reliability and efficiency in wireless multi-hop mesh networks. Instead of performing unicast communication as established TSCH schedulers do, opportunistic routing builds upon anycasts. By this, opportunistic routing sends each packet to a set of receivers. If any of them receives the packet, the transmission is successful. As multiple receivers might receive the packet, opportunistic routing has to overcome the challenge of selecting a unique forwarder. This forwarder selection has to wait until after the transmission [48–50].

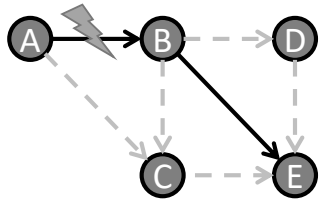
While initial works do not use duty-cycled, low-power wireless networking, later works such as as ORW [51] and ORPL [52] bring opportunistic routing to these. Nonetheless, these protocols are not built for TSCH. Huynh et al. [53], Hermeto et al. [54], and Hosni et al. [55] study the use of opportunistic routing or anycasts in TSCH and propose changes to TSCH to allow non-colliding acknowledgments from multiple receivers. BOOST [56] introduces forwarder selection through sending delays with carrier sense in TSCH. In contrast to these approaches, AUTOBAHN does not use any preferred forwarder selection method. Instead, we overcome this challenge by using synchronous transmissions.

4.2.3 Synchronous Transmissions

Synchronous transmission protocols allow multiple nodes to transmit packets simultaneously. With precise timing, these packets do not collide destructively, allowing protocols to achieve high communication reliability [6, 60]. As a result, protocols employing synchronous transmissions do not maintain routes by selecting parent nodes, announcing routing metrics, discovering neighbors, and maintaining routing tables as traditional routing protocols.

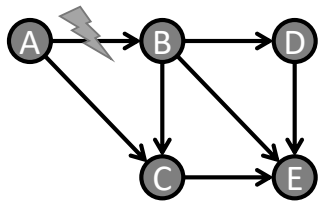
For receiving such a packet, the senders must not significantly differ in timing. One common option of receiving synchronous transmissions is the so-called Capture Effect [57]. According to the capture effect in IEEE 802.15.4, a stronger signal must not arrive later than $160\mu s$ after the first signal [59]. When sending the same data, non-destructive interference is achievable if the time offset between multiple senders is within a bound of $0.5\mu s$ [60].

Synchronous transmissions are well studied. Glossy [60] laid the foundation for synchronous transmissions in wireless sensor networks. Since Glossy's introduction, many protocols including Chaos [59] and LWB [6] followed. They all are protocols that use network-wide flooding without a concept of routing. Protocols like WSNShape/Sparkle [63], CXFS [64] and LaneFlood [65] divert



	1	2	3	4
A	TX	TX		
B	RX	TXRX	TX	
C				
D				
E		RX	RX	

(a) Established central scheduling approaches employ a single routing path. Their schedule will fail if one of the links fails, such as the link between nodes A and B in this example.



	1	2	3	4
A	TX	TX		
B	RX	TXRX	TX	
C	RX	TXRX	TX	
D		RX	RXTX	TX
E		RX	RX	RX

(b) AUTOBAHN utilizes multi-path routing and thereby provides redundant options in case routes fail. In this example, packets can travel via node C to destination E.

Figure 4.1: AUTOBAHN compared to established centralized TSCH scheduling approaches. In this example, we assume a topology of five nodes, with node A as source and node E as destination. We show both the scheduled paths and the TSCH schedule, using RX, RXTX, and TX slots as typical for flow-based retransmission schemes (with a retransmission window of two). Grayed-out slots present slots where reception and transmission are not possible due to previously failed interfered receptions.

from network-wide flooding and use flooding with some notion of routing along a path of forwarders.

All of these protocols operate without a routing layer, whereas AUTOBAHN follows the principle of combining synchronous transmissions and TSCH as envisioned by Chang et al. [78]. Gomes et al. [79] study an initial approach of flooding-based routing in TSCH. This approach relies fully on broadcasts (no acknowledgements) and uses shorter TSCH slots. Baddeley et al. [80] present a hybrid between TSCH and synchronous transmissions by replacing some TSCH slots with synchronously transmitted BLE packets for exchanging control information.

While some protocols explore the field of combining TSCH and synchronous transmissions, AUTOBAHN explores it further by combining synchronous transmissions with TSCH, including both synchronous transmissions as well as synchronous acknowledgments in combination with opportunistic routing.

4.3 Design

We continue with the design of AUTOBAHN. We begin with a simple example to present the basic idea of AUTOBAHN. Then, we introduce (1) general node

selection requirements, (2) the forwarder selection through node ranks, and (3) the active nodes in each slot. After discussing these main points of the design, we present the system design, including the Contiki/TSCH extensions to allow anycast communication, and AUTOBAHN's routing layer adaptations.

4.3.1 Autobahn: General Idea

As an example, we assume the network of five nodes in Figure 4.1 where Node A communicates with node E. Further, let us assume that the link between nodes A and B fails due to interference.

To illustrate the benefits of AUTOBAHN, we first discuss how established centralized scheduling approaches suffer from link failures. Established approaches commonly employ a single routing path. Their schedule will fail if one of the links fails, such as the link between nodes A and B in this example, see Figure 4.1a. Retransmissions, as scheduled in the example, usually happen on a different channel and thereby protect the protocol against narrow-band interference. Wide-band interference, however, can break links and result in packet loss. Eventually, the scheduler has to deploy an updated schedule. If this is done frequently, this adds significant overhead to the communication scheme.

The general idea behind AUTOBAHN is to add redundancy to the routing path, see Figure 4.1b. In the example of AUTOBAHN, node A sends a packet that will be received by nodes B and C. These two forward the packet synchronously to nodes D and E, which receive one of the two transmissions due to the capture effect. Lastly, node D sends the packet to node E as well. In case of interference, node B is not reachable. That means that only node C receives the packet from node A. Node C then forwards the message to node E. Redundant routing paths in AUTOBAHN add non-neglectable overhead to the duty cycle of the network. In our evaluation, we, however, show that this overhead is justifiable in the interference-free case, and in the case of interference, it is essential for reliable communication.

4.3.2 Routing Set

Centralized schedulers have global knowledge over the network topology through long-term link quality metrics. They commonly route traffic along a single path, using single forwarders. In contrast to that, AUTOBAHN addresses packets to multiple forwarders (anycast). To achieve this, we employ a routing set with redundancy instead of a single path.

We define a routing set to consist of all nodes we use for end-to-end communication. A routing set $\{rs\}$ is a set of nodes $n_1, ..., n_k \in \{rs\}$ responsible for routing a data packet from the source node n_1 to the sink node n_k. This routing set contains the nodes forming the shortest path from source to sink as well as additional nodes used for opportunistic, anycast routing in AUTOBAHN, which adds path redundancy.

To build a routing set, we start with the shortest path from source to destination employing the ETX-metric [12] and Dijkstra's shortest-path algorithm [40]. Next, we add routing redundancies by including neighboring nodes along the path. For this, we introduce three schemes: (i) neighbor-based, (ii)

hop-based, and (iii) cost-based selection of routing sets. Especially in dense networks, the number of these neighboring nodes for each of the three schemes is likely to be high and leads to the inclusion of massive parts of the network. Therefore, we specify a node overhead factor (scaling the number of additional nodes) and a cost overhead factor (scaling the max. allowed ETX cost of nodes of an end-to-end path).

Neighbor-based participant selection starts with determining all nodes neighboring at least one node of the shortest path. From these nodes, we continue with three different subsets: (a) all selected nodes that do not exceed the cost overhead, (b) a subset of (a) forming a second shortest path, and (c) a subset of (a) forming a shortest path from each node of the original path to the destination node.

After selecting the respective nodes, we check whether the node overhead is too large. If so, we refine our selection only to include the allowed number of nodes with the lowest cost.

The hop-based selection possibility includes additional nodes with a similar combined distance to the shortest path's source and destination while not exceeding the cost overhead. Equation 4.1 shows the general idea of this selection strategy, that the combined hop distances from source to forwarder (d_{sf}) and forwarder to destination (d_{df}) must not exceed the direct distance d_{sd} plus a slack value s. The slack value has to be a natural number. If the node overhead is too large, nodes with the lowest hop count are preferred.

$$d_{sf} + d_{df} \leq d_{sd} + s \tag{4.1}$$

The cost-based selection possibility follows the same equation. However, instead of hop-based distances, we use ETX-based distances and an ETX-based slack (s), which can be a positive real number. However, the maximum slack value for this strategy equals the maximum cost overhead allowed according to the cost overhead factor. If, after node selection, the node overhead is too large, we rank the nodes regarding their cost and take those with the lowest cost. In addition, we exclude all other nodes that have the same cost as one of the already excluded nodes.

4.3.3 Anycast forwarding in Autobahn

In anycast routing, we address a packet to a set of neighboring nodes, i.e., the ones making sufficient progress towards the destination. Thus, for each transmission in Autobahn, this set of possible recipients listens for the packet. Practically, we introduce node ranks: Each node has a rank according to its distance to the destination of a flow from source to destination. The sender of a packet has rank 0, and the rank increases towards the receiver, with the receiver having the highest rank.

If a node receives a packet, it compares its rank to the sender's rank, which we include in the packet header. If the own rank is higher, it acknowledges the packet and forwards it. Otherwise, it stays silent and acknowledges for itself that the packet has passed. The sender of a packet performs a similar action. If it receives an acknowledgment from a node with a higher rank, it concludes that the opportunistic anycast succeeded and stops forwarding this packet. This way, we ensure that only nodes closer to the packet's destination

acknowledge the reception of the packet and forward the packet; thus, we avoid loops and packets stuck mid-flow.

In traditional opportunistic routing, packet duplicates are often a challenge [48,51,52]: There is always a risk that multiple forwarders receive a packet, and each individually forwards the packet, adding additional load on the network. In AUTOBAHN, all packets – including duplicates – are forwarded synchronously, and their spatial diversity is the basis for the reliability of our design in the presence of interference. Thus, duplicates are (i) inherently part of the design and (ii) do not add the overhead as in traditional designs.

4.3.4 Active slots in Autobahn

AUTOBAHN uses flow-based retransmission schemes such as Sliding Windows introduced by MASTER [66], with multiple nodes possibly active in one slot. AUTOBAHN extends this by activating all nodes in a slot that are reachable by any previously active node.

In the first slot of a flow, the sender of a packet and all receivers in range are active. For each of the following slots, we include all additional nodes reachable by any previously active node. From this, we derive the first active slot of a node, i.e., the first point in time a packet in a flow can reach a node along one of the different paths employed by AUTOBAHN. We determine a node's last active slot based on the node's hop distance to the flow's receiver and the flow's number of total transmissions.

Due to the opportunistic nature of AUTOBAHN, a high network duty-cycle is expectable. Nevertheless, to still keep the energy consumption as low as possible, each node stays only active until we no longer need it for forwarding the packet. As we explain above, a node determines whether it is still needed through the received rank of other participants.

The schedules in Figure 4.1 illustrate the difference between the active slots of a flow-based central scheduler (Figure 4.1a) without opportunistic routing and AUTOBAHN (Figure 4.1b).

4.3.5 System Integration

For the design of AUTOBAHN, we devise a TSCH implementation with support for opportunistic anycasts and a good enough time synchronization for synchronous transmissions. In our evaluation, we show that the TSCH implementation of Contiki-NG is sufficient for synchronous transmissions. However, as it does not support anycasts, we have to realize these ourselves. AUTOBAHN itself can be implemented on top of any centralized scheduler. We choose MASTER, a centralized scheduler implemented for Contiki-NG as our basis. We implement AUTOBAHN to replace MASTER's central routing and retransmission logic while keeping its scheduling module. Below, we discuss the integration into MASTER's Contiki routing layer and the extension of Contiki-NG/TSCH to allow opportunistic anycasts.

4.3.6 Integration in Master's routing layer

We extend MASTER's routing layer to have access to a node's rank and relay a packet back to the correct flow address instead of a neighbor address.

The routing layer is also responsible for the routing-specific header. In addition to the existing 7-byte routing header, AUTOBAHN requires one additional byte. The existing 7 bytes are a flow identifier (1 byte), a sequence number (2 bytes), the time-to-live (TTL) (2 bytes), and the earliest transmission slot (2 bytes) of a packet. AUTOBAHN adds the node's rank to the packet to allow the receiver to make its forwarding decision according to our description above.

4.3.6.1 Contiki-NG/TSCH extensions

The TSCH implementation of Contiki-NG does not support anycast communication. To add support for anycasts, we extend it with (1) the capability of accepting packets from any neighbor of a flow, as well as (2) using this flow as a sender and receiver simultaneously. Moreover, we (3) define a flow-specific sequence number to accept acknowledgments successfully.

With our modification, TSCH accepts packets from a flow address if the receiving node is a member of the respective flow. As we no longer need the sender's and receiver's addresses, we replace them with the flow address.

Besides accepting packets from any flow participant, TSCH needs the capability to accept acknowledgments from any possible forwarder of the flow. Therefore, we include the receiver's rank in the acknowledgment. If a node receives an acknowledgment acknowledging a different synchronous sender, it still needs to be accepted. Therefore, our routing layer replaces the TSCH sequence number with a flow and packet-specific end-to-end sequence number.

4.4 Evaluation

In this section, we evaluate AUTOBAHN's performance and compare it to the state-of-the-art. We start by showing the feasibility of synchronous transmissions in the context of TSCH. After that, we evaluate AUTOBAHN's different routing set selection choices and compare those to MASTER in scenarios with and without interference. Afterward, we compare AUTOBAHN's best routing selection algorithm against Orchestra, the default autonomous scheduler in Contiki-NG. We conclude our evaluation with long-term stability analysis of schedules in AUTOBAHN.

4.4.1 Evaluation Setup

4.4.1.1 Testbed and Platform

We run our experiments on a 20-node testbed at our local university. This testbed (Figure 4.2) covers the top floor of a university building with offices and student lab rooms and thus shares the wireless spectrum with WiFi and Bluetooth communication outside of our control.

4.4.1.2 Metrics, Comparison, and Duration

We evaluate AUTOBAHN in terms of end-to-end reliability, end-to-end latency, and network energy consumption (network duty cycle). We measure these metrics for different routing and retransmission approaches for MASTER and

AUTOBAHN under different interference levels. Moreover, we compare AUTOBAHN with Orchestra according to these metrics. We include six flows we give in Figure 4.2. The duration of each experiment in sections 4.4.5, and 4.4.4 is 75 minutes, with each flow sending 100 packets per minute. In section 4.4.6 we run 75-minute experiments with 60 packets per minute and flow. For the long-term evaluations from Section 4.4.8 onward, we specify the duration as part of the specific experiment.

4.4.1.3 Implementation

We implement AUTOBAHN for Contiki-NG [11] and target the Zolertia Firefly Platform. This platform features a CC2538 Cortex-M3 CPU (32-bit, 32 MHz) with 32 KB of RAM, 512 KB flash storage, and an IEEE 802.15.4 compatible radio.

4.4.1.4 Channels and Interference

We perform most of our experiments under interference. To ensure comparable levels of interference for all tested protocols, we generate these ourselves in a repeatable manner using JamLab [81]. If not stated otherwise, we use an interference level of 10% channel occupancy. We use five interference sources depicted in Figure 4.2. Two of the interference sources are in a central position surrounded by several nodes, while the other three are each in close vicinity to a forwarding node in the network. As our testbed only provides the capability of generating interference on one channel at a time, we use only a single channel (channel 26) for all experiments. As we target networks susceptible to wide-band interference, evaluating on only one channel is not a problem. Wide-band interference, such as WiFi, would cover multiple IEEE 802.15.4 channels, eliminating channel hopping advantages. Therefore, it is more realistic to use one channel with interference than multiple channels with interference on only one of them. Moreover, using only one channel lets us compare the worst-case performance of the discussed protocols.

4.4.1.5 Application Payload and Overhead

We send packets with a 64-byte randomly generated payload for all experiments, a medium packet size for TSCH. Additionally to this data payload, we include 7-byte and 8-byte routing headers for MASTER and AUTOBAHN, respectively. Orchestra uses IPv6 headers instead and requires additional network layer control traffic.

4.4.1.6 Routing Sets

We include three AUTOBAHN routing sets marked as neighbor-based, hop-based, and cost-based. The neighbor-based one is option (c) of the neighbor-based routing sets in Section 4.3.2, the one with an alternative path from each node through all neighbors. For the hop-based routing set, we use a slack value of 2. For the cost-based routing set, we use the maximum possible slack value, equaling the maximum cost overhead. This slack value is potentially different for each flow. This value ensures that we include all nodes, with an end-to-end

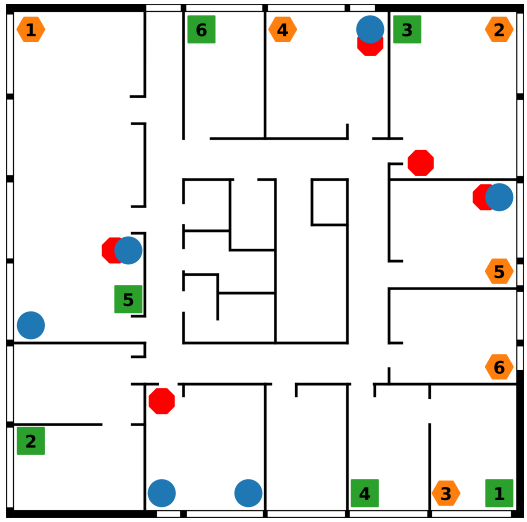


Figure 4.2: Local testbed of $\mathbf{500m^2}$. Source nodes: orange hexagons; Sink nodes: green squares; Relay-only nodes: blue circles; Red octagons: interferer; Numbers: corresponding flow

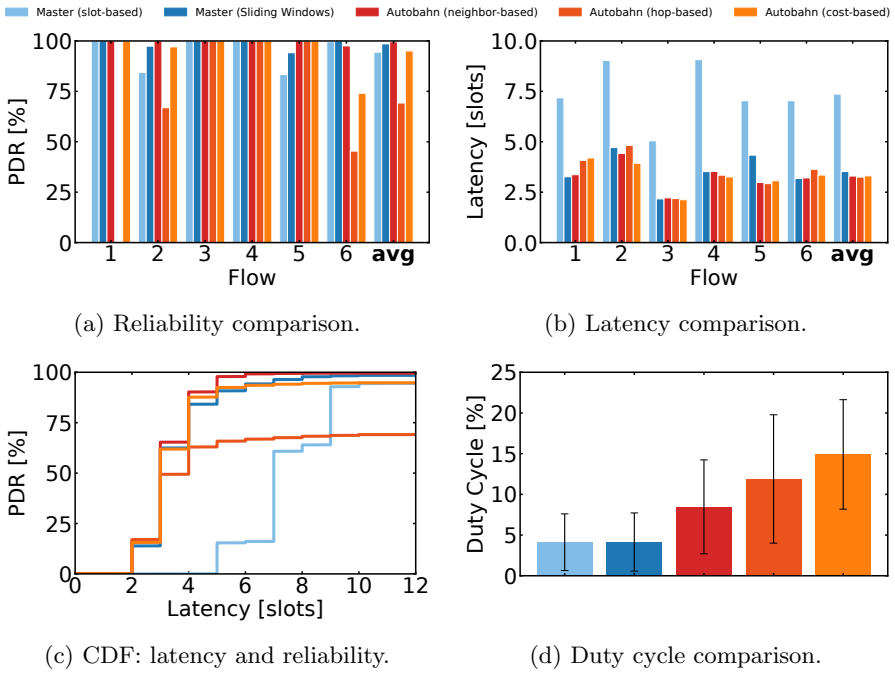
ETX value not exceeding the cost overhead factor. We use overhead factors of 2 and 2.5 as node-overhead factor and cost overhead factor, respectively.

4.4.2 Baselines

We compare AUTOBAHN's routing-set algorithms to three other TSCH scheduling policies. Two of these are MASTER's slot-based retransmission strategy and MASTER's flow-based transmission strategy called Sliding Windows. MASTER's slot-based retransmission strategy follows the traditional concept of replicating slots of single hops, done in several recent publications, including AMUS [17]. We use MASTER, as it provides us an implementation for Contiki-NG. As the last baseline we use Orchestra, to set AUTOBAHN into relation to a well-known protocol. Orchestra [3] is an autonomous scheduler for TSCH included in Contiki-NG [11]. It autonomously maps links to resources, e.g., determines a node's send or receive slot based on a hash function.

4.4.3 Possibility of Synchronous Transmissions in TSCH

Before starting our main evaluation, we investigate the quality of synchronization in TSCH for synchronous transmissions. With a desk setup of 4 nodes, we can identify the feasibility of synchronous transmissions. Our data shows an average offset between two synchronously transmitting nodes of 16.4μs with a standard deviation of 16.8μs and a maximum offset of 65.7μs. This offset clearly shows that the degree of synchronization in Contiki's TSCH implementation is by far not good enough for constructive interference (offset bound of 0.5μs [60]). However, the offset stays below the maximum offset for capture effect of 160μs [59]. TSCH generally does not require synchronization



(a) Reliability comparison.

(b) Latency comparison.

(c) CDF: latency and reliability.

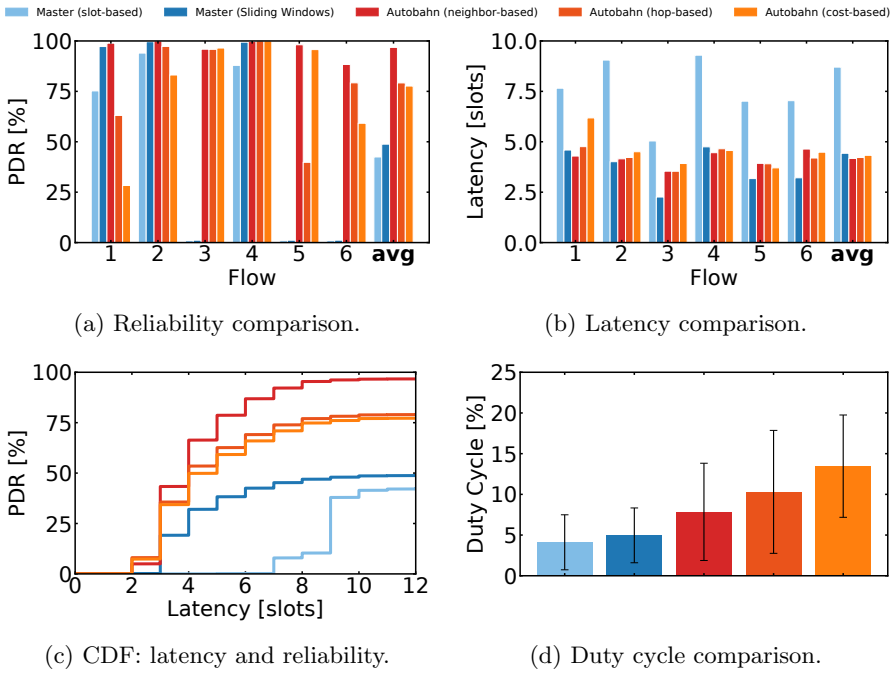
(d) Duty cycle comparison.

Figure 4.3: AUTOBAHN and MASTER without interference. AUTOBAHN's neighbor-based strategy outperforms MASTER while increasing the duty cycle by 4.3 percentage points.

as strict as Glossy and therefore does not include additional physical layer time synchronization measures. Nonetheless, our results show that synchronous transmissions are possible due to the capture effect.

4.4.4 Performance without Interference

We begin our evaluation by comparing the performance of AUTOBAHN's different routing sets with the performance of MASTER's retransmission strategies. For this evaluation, we do not generate any interference. When comparing the reliability of the different routing sets (see Figure 4.3a), we see a generally better performance of the neighbor-based routing set in comparison with the hop-based or cost-based ones. Especially the difference between the neighbor-based and hop-based routing sets is visible for flows 1 and 6. The hop-based routing set has too many simultaneously active nodes at similar distances to a forwarder or receiver. With this number of active nodes, no signal is strong enough for reception through the capture effect. For flow 1, we even see a destruction of the signal. The cost-based strategy has better reliability but generally does not achieve the high reliability of the neighbor-based routing set. The baseline strategies are not exposed to in-flow interference and achieve almost the reliability of neighbor-based AUTOBAHN, with a slight advantage for Sliding Windows over the slot-based retransmission strategy. However, if the network's link qualities are not perfect for a flow (flows 2 and 5), MASTER is



(a) Reliability comparison.

(b) Latency comparison.

(c) CDF: latency and reliability.

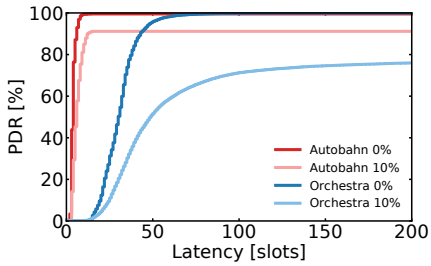
(d) Duty cycle comparison.

Figure 4.4: AUTOBAHN and MASTER under interference. AUTOBAHN has a much better performance than MASTER with the best performance using the neighbor-based routing set.

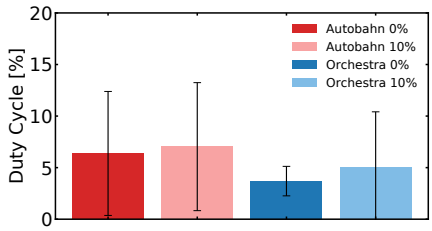
more strongly affected. Latency-wise (see Figure 4.3b, 4.3c), AUTOBAHN has a small advantage over MASTER, while both AUTOBAHN and Sliding Windows are better than the slot-based retransmission strategy. The reliability improvement comes at a cost of a higher network duty cycle (see Figure 4.3d). With more active nodes, the duty cycle increases significantly. Nevertheless, the best performing routing set of AUTOBAHN leads to the least increase in duty cycle by, on average 4.3 percentage points.

4.4.5 Performance under Interference

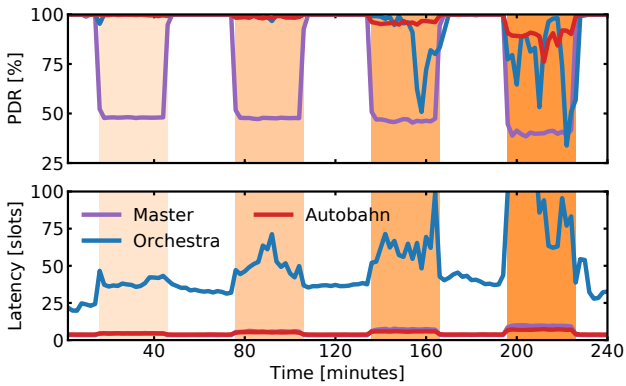
Next, we compare the same strategies as before under induced interference. From Figure 4.4a it is visible that the neighbor-based routing set once again performs best. The other routing sets still offer average reliability of around 80%. However, for some flows, these routing sets achieve very low reliability (e.g., flow 1). This low reliability indicates that too many nodes are active simultaneously plus the additional interference heavily impacting a successful capture effect. Comparing AUTOBAHN to MASTER shows that all routing sets of AUTOBAHN have higher average reliability than MASTER's strategies. This is due to MASTER being heavily impacted by interference in certain flows (flows 3, 5, and 6). In these flows, the shortest path passes closely to an interference source. Thus, we see that additional nodes offered by AUTOBAHN are necessary to route traffic around interference sources opportunistically. The latency



(a) Latency and reliability comparison between Orchestra and AUTOBAHN with and without interference.



(b) Network duty cycle of Orchestra and AUTOBAHN with and without interference.



(c) Performance under and recovery from interference over time. (Interference levels from left to right: **5%**, **10%**, **15%**, and **25%**)

Figure 4.5: Comparison of AUTOBAHN and Orchestra with and without interference. Figure 4.5c includes the recovery performance of MASTER's Sliding Windows strategy as an additional baseline.

differences (see Figure 4.4b) follow the same trend as without interference, just slightly higher. As the latency comparison only includes received packets, we also show the combination of latency and reliability in Figure 4.4c. AUTOBAHN requires a latency of 4 slots to reach a 50% reliability, while MASTER cannot reach this network-wide reliability at all. The higher overall reliability comes at the cost of a higher network duty cycle (see Figure 4.4d). The cost is similarly high as for the interference-free case. However, especially in the presence of interference, an increase of network duty cycle of 2.89 percentage points for the best routing set should be acceptable if reliability has high priority.

4.4.6 Autobahn vs. Orchestra

Next, we compare AUTOBAHN's best-performing routing set (neighbor-based) with another baseline, the autonomous scheduler Orchestra. As Orchestra is a best-effort protocol and is therefore not limited to a deadline, i.e., a certain number of slots to successfully transmit a packet, we, therefore, relax these limitations for AUTOBAHN as well. However, as we send a new packet

every second, AUTOBAHN's schedule should not exceed this value. We achieve this by using a higher scaling factor of three instead of two to determine the number of transmission slots. That means that we use for each flow 50% more transmissions than in the previous experiments. The results of this comparison (see Figure 4.5a-4.5b) show that Orchestra achieves slightly higher reliability than AUTOBAHN without the presence of interference (99.96% vs. 99.51%). However, in the presence of interference, AUTOBAHN clearly outperforms Orchestra with a 13.89 percentage points higher packet delivery rate (PDR). Latency-wise, AUTOBAHN clearly outperforms Orchestra. In case of interference, by a factor 9. However, we need to attribute some of that to the fact that all flows communicate actively simultaneously in Orchestra, while in AUTOBAHN's schedule, the flows communicate one after another. Energy-wise, we can see a similar trend as with the comparison of AUTOBAHN and MASTER (Figure 4.3d and Figure 4.4d). AUTOBAHN uses more active nodes and therefore has a higher duty cycle. In contrast both MASTER and Orchestra follow a single path, with MASTER's duty cycle being the lowest, while Orchestra occupies the middle ground.

4.4.7 Recovery from interference

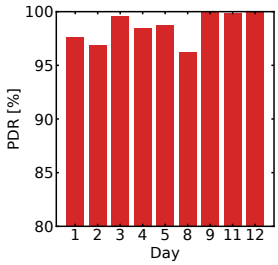
Next, we compare how well MASTER, AUTOBAHN, and Orchestra perform under interference over time and how good they are at recovering from interference. Figure 4.5c shows that all three algorithms are influenced by interference but are successful at recovering independently of the interference level. MASTER and AUTOBAHN have similar latency responses, while Orchestra has a less uniform curve as the rerouting in case of interference takes some time. Reliability-wise, Orchestra keeps high reliability for quite a long time but has to drop packets towards the end of an interference block. MASTER is generally hit the hardest by interference, while AUTOBAHN clearly performs best under interference.

4.4.8 Long-term stability of Autobahn

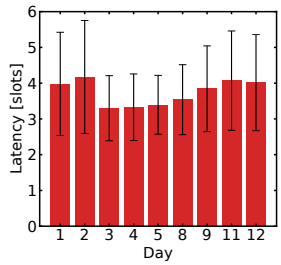
After comparing AUTOBAHN to various baselines under different interference levels, we evaluate AUTOBAHN's long-term stability. As before, we use AUTOBAHN's neighbor-based routing set. For this section, we use a schedule generated with a neighbor discovery on day one and run this schedule almost daily over 12 days. Moreover, after 12 days, we run a final 25 hour experiment with three-hour blocks of 5%, 10%, 15%, and 25% interference, respectively. While the performance varies within the days, reliability always stays above 96% (see Figure 4.6a). Latency fluctuates slightly, yet it remains around the level of the first days, whereas the duty cycle shows an upwards trend in the beginning (see Figure 4.6b-4.6c). The performance analysis over 25 hours (see Figure 4.6d) shows that even an almost two-week-old AUTOBAHN schedule is still able to perform under interference and quickly recover from it.

4.5 Conclusion

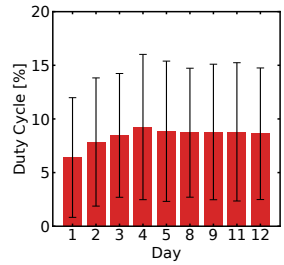
Centrally scheduled networks are sensitive to wireless link dynamics, esp. wideband interference. AUTOBAHN addresses this by adding spatial redundancies



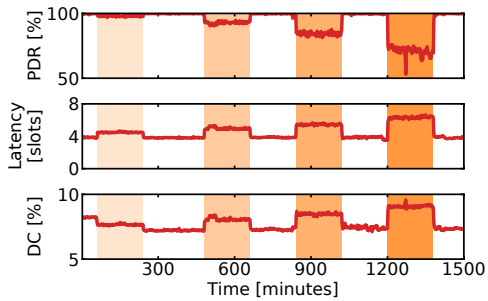
(a) Reliability over the course of 12 days.



(b) Latency over the course of 12 days.



(c) Energy consumption (duty cycle) over the course of 12 days.



(d) Performance over 25 hours with 3 hour long interference blocks (interference levels from left to right: **5%**, **10%**, **15%**, **25%**).

Figure 4.6: Long-term stability evaluation of AUTOBAHN. Figures 4.6a - 4.6c present (almost) daily runs of AUTOBAHN with the same schedule over the course of 12 days. Figure 4.6d shows the performance of the same 12-day old schedule under different interference levels and its recovery from interference.

via combining TSCH with synchronous transmissions and opportunistic routing.

We show that AUTOBAHN offers reliability of 95% and more under interference while mildly increasing the duty cycle by 4.3 percentage points. Moreover, experiments over 12 days show the long-term stability of AUTOBAHN's schedules with 98.6% reliability.

Bibliography

[1] *nRF52840 Product Specification*, Nordic Semiconductor, 2019, 4413_417 v1.1. [Online]. Available: https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.1.pdf

[2] *CC2538 Powerful Wireless Microcontroller System-On-Chip for 2.4-GHz IEEE 802.15.4, 6LoWPAN, and ZigBee® Applications.*

[3] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *ACM SenSys*, 2015.

[4] M. Palattella, N. Accettura, M. Dohler, L. Grieco, and G. Boggia, "Traffic-Aware Time-Critical Scheduling in Heavily Duty-Cycled IEEE 802.15.4e for an Industrial IoT," in *IEEE PIMRC*, 2012.

[5] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-Time Scheduling for WirelessHART Networks," in *IEEE RTSS*, 2010.

[6] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *ACM SenSys*, 2012.

[7] "IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPAN)," *IEEE Std 802.15.4-2003*, 2003.

[8] R. F. Heile, R. Alfvin, P. W. Kinney, J. P. K. Gilb, and C. Chaplin, "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer," IEEE, Tech. Rep., 2012.

[9] ISA, *ISA-100.11a-2011 - Wireless Systems for Industrial Automation: Process Control and Related Applications.* ISA, 2011.

[10] HART Communication Foundation, *WirelessHART Specification 75: TDMA Data-Link Layer. HCF_SPEC-75.* HART Communication Foundation, 2008.

[11] "Contiki-NG: The OS for Next Generation IoT Devices," 2020. [Online]. Available: http://www.contiki-ng.org/

[12] D. S. J. De Couto, "High-Throughput Routing for Multi-Hop Wireless Networks," PhD thesis, Massachusetts Institute of Technology, 2004, https://pdos.lcs.mit.edu/papers/grid:decouto-phd/thesis.pdf.

[13] M. Ojo, S. Giordano, G. Portaluri, D. Adami, and M. Pagano, "An energy efficient centralized scheduling scheme in TSCH networks," in *IEEE ICC Workshops)*, 2017.

[14] M. Ojo and S. Giordano, "An efficient centralized scheduling algorithm in ieee 802.15.4e TSCH networks," in *IEEE CSCN*, 2016.

[15] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler, and T. Engel, "On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH," *IEEE Sensors Journal*, vol. 13, no. 10, 2013.

[16] D. Gunatilaka and C. Lu, "Conservative Channel Reuse in Real-Time Industrial Wireless Sensor-Actuator Networks," in *IEEE ICDCS*, 2018.

[17] Y. Jin, P. Kulkarni, J. Wilcox, and M. Sooriyabandara, "A centralized scheduling algorithm for IEEE 802.15.4e TSCH based industrial low power wireless networks," in *IEEE WCNC*, 2016.

[18] G. Gaillard, D. Barthel, F. Theoleyre, and F. Valois, "High-reliability scheduling in deterministic wireless multi-hop networks," in *IEEE PIMRC*, 2016.

[19] A. Darbandi and M. K. Kim, "Path Collision-aware Real-time Link Scheduling for TSCH Wireless Networks," *KSII Transactions on Internet & Information Systems*, vol. 13, no. 9, 2019.

[20] J. P. G. Rugamba, D. L. Mai, and M. K. Kim, "Implementation of a Centralized Scheduling Algorithm for IEEE 802.15.4e TSCH," in *Intelligent Computing Methodologies*, D.-S. Huang, Z.-K. Huang, and A. Hussain, Eds. Springer Int. Pub., 2019, vol. 11645.

[21] R. Brummet, D. Gunatilaka, D. Vyas, O. Chipara, and C. Lu, "A Flexible Retransmission Policy for Industrial Wireless Sensor Actuator Networks," in *IEEE ICII*, 2018.

[22] A. Tinka, T. Watteyne, and K. Pister, "A Decentralized Scheduling Algorithm for Time Synchronized Channel Hopping," in *Ad Hoc Networks*, J. Zheng, D. Simplot-Ryl, and V. C. M. Leung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[23] N. Accettura, M. R. Palattella, G. Boggia, L. A. Grieco, and M. Dohler, "Decentralized traffic aware scheduling for multi-hop low power lossy networks in the internet of things," in *IEEE WoWMoM*, 2013.

[24] X. Vilajosana, K. Pister, and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration," RFC 8180, May 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8180.txt

[25] T. Chang, M. Vučinić, X. Vilajosana, S. Duquennoy, and D. R. Dujovne, "6TiSCH Minimal Scheduling Function (MSF)," RFC 9033, May 2021. [Online]. Available: https://rfc-editor.org/rfc/rfc9033.txt

[26] T. Chang, T. Watteyne, Q. Wang, and X. Vilajosana, "LLSF: Low Latency Scheduling Function for 6TiSCH Networks," in *IEEE DCOSS*, 2016.

[27] M. Domingo-Prieto, T. Chang, X. Vilajosana, and T. Watteyne, "Distributed PID-Based Scheduling for 6TiSCH Networks," *IEEE Communications Letters*, 2016.

[28] V. Kotsiou, G. Z. Papadopoulos, P. Chatzimisios, and F. Theoleyre, "LDSF: Low-Latency Distributed Scheduling Function for Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 7, 2020.

[29] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel, "On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks," *IEEE Sensors Journal*, vol. 16, no. 2, 2016.

[30] S. Jeong, H.-S. Kim, J. Paek, and S. Bahk, "OST: On-Demand TSCH Scheduling with Traffic-Awareness," in *IEEE INFOCOM*, 2020.

[31] J. Jung, D. Kim, T. Lee, J. Kang, N. Ahn, and Y. Yi, "Distributed slot scheduling for qos guarantee over tsch-based iot networks via adaptive parameterization," in *ACM/IEEE IPSN*, 2020.

[32] J. Shi, M. Sha, and Z. Yang, "DiGS: Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks," in *IEEE ICDCS*, 2018.

[33] ——, "Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, 2019.

[34] S. Oh, D. Hwang, K.-H. Kim, and K. Kim, "Escalator: An Autonomous Scheduling Scheme for Convergecast in TSCH," *Sensors*, vol. 18, no. 4, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/4/1209

[35] S. Kim, H.-S. Kim, and C. Kim, "ALICE: autonomous link-based cell scheduling for TSCH," in *IEEE/ACM IPSN*, 2019.

[36] S. Jeong, J. Paek, H.-S. Kim, and S. Bahk, "TESLA: Traffic-Aware Elastic Slotframe Adjustment in TSCH Networks," *IEEE Access*, vol. 7, 2019.

[37] J. Jung, D. Kim, J. Hong, J. Kang, and Y. Yi, "Parameterized slot scheduling for adaptive and autonomous TSCH networks," in *IEEE INFOCOM WKSHPS*, 2018.

[38] S. Rekik, N. Baccour, M. Jmaiel, K. Drira, and L. A. Grieco, "Autonomous and Traffic-aware Scheduling for TSCH Networks," *Computer Networks*, vol. 135, Apr. 2018.

[39] X. Cheng and M. Sha, "ATRIA: Autonomous Traffic-Aware Scheduling for Industrial Wireless Sensor-Actuator Networks," in *IEEE ICNP)*, 2021.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Section 24.3: Dijkstra's algorithm," in *Introduction to Algorithms*, 2nd ed. MIT Press and McGraw-Hill, 2001.

[41] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, 1968.

[42] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, and T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, Mar. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6550.txt

[43] B. Li, Y. Ma, T. Westenbroek, C. Wu, H. Gonzalez, and C. Lu, "Wireless Routing and Control: A Cyber-Physical Case Study," in *ACM/IEEE ICCPS*, 2016.

[44] C. Wu, D. Gunatilaka, M. Sha, and C. Lu, "Real-Time Wireless Routing for Industrial Internet of Things," in *IEEE/ACM IoTDI*, 2018.

[45] P. Minet, I. Khoufi, and A. Laouiti, "Increasing reliability of a TSCH network for the industry 4.0," in *IEEE NCA*, 2017.

[46] T. Lagos Jenschke, R.-A. Koutsiamanis, G. Z. Papadopoulos, and N. Montavont, "Multi-path Selection in RPL Based on Replication and Elimination," in *Ad-hoc, Mobile, and Wireless Networks*, 2018, vol. 11104.

[47] A. C. Estrin, T. Lagos Jenschke, G. Z. Papadopoulos, J. Ignacio Alvarez-Hamelin, and N. Montavont, "Thorough Investigation of multipath Techniques in RPL based Wireless Networks," in *IEEE ISCC*, 2020.

[48] S. Biswas and R. Morris, "ExOR: opportunistic multi-hop routing for wireless networks," in *ACM SIGCOMM*, 2005.

[49] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," *ACM SIGCOMM Computer Communication Review*, vol. 37, 2007.

[50] R. R. Choudhury and N. H. Vaidya, "MAC-layer anycasting in ad hoc networks," *ACM SIGCOMM Computer Communication Review*, vol. 34, 2004.

[51] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low Power, Low Delay: Opportunistic Routing meets Duty Cycling," in *ACM/IEEE IPSN*, 2012.

[52] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the tree Bloom: scalable opportunistic routing with ORPL," in *ACM SenSys*, 2013.

[53] T. Huynh, F. Theoleyre, and W. Hwang, "On the interest of opportunistic anycast scheduling for wireless low power lossy networks," *Comput. Commun.*, vol. 104, 2017.

[54] R. T. Hermeto, A. Gallais, and F. Theoleyre, "Is Link-Layer Anycast Scheduling Relevant for IEEE 802.15.4-TSCH Networks?" in *IEEE LCN*, 2019.

[55] I. Hosni and F. Théoleyre, "Adaptive k-cast Scheduling for High-Reliability and Low-Latency in IEEE802.15.4-TSCH," in *Ad-hoc, Mobile, and Wireless Networks*, 2018, vol. 11104.

[56] Y. Jin, U. Raza, and M. Sooriyabandara, "BOOST: Bringing Opportunistic ROuting and Effortless-Scheduling to TSCH MAC," in *IEEE GLOBECOM*, 2018.

[57] K. Leentvaar and J. Flint, "The Capture Effect in FM Receivers," *IEEE Transactions on Communications*, vol. 24, 1976.

[58] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, "Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless," in *ACM SenSys*. New York, NY, USA: Association for Computing Machinery, 2010.

[59] O. Landsiedel, F. Ferrari, and M. Zimmerling, "Chaos: versatile and efficient all-to-all data sharing and in-network processing at scale," in *ACM SenSys*, 2013.

[60] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *ACM/IEEE IPSN*, 2011.

[61] C.-H. Liao, Y. Katsumata, M. Suzuki, and H. Morikawa, "Revisiting the So-Called Constructive Interference in Concurrent Transmission," in *IEEE LCN*, 2016.

[62] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele, "Adaptive Real-Time Communication for Wireless Cyber-Physical Systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 1, no. 2, Feb. 2017.

[63] D. Yuan, M. Riecker, and M. Hollick, "Making 'Glossy' Networks Sparkle: Exploiting Concurrent Transmissions for Energy Efficient, Reliable, Ultra-Low Latency Communication in Wireless Control Networks," in *Wireless Sensor Networks*, 2014, vol. 8354.

[64] D. Carlson, M. Chang, A. Terzis, Y. Chen, and O. Gnawali, "Forwarder Selection in Multi-transmitter Networks," in *IEEE ICDCS*, 2013.

[65] M. Brachmann, O. Landsiedel, and S. Santini, "Concurrent Transmissions for Communication Protocols in the Internet of Things," in *IEEE LCN*, 2016.

[66] O. Harms and O. Landsiedel, "MASTER: Long-Term Stable Routing and Scheduling in Low-Power Wireless Networks," in *IEEE DCOSS*, 2020.

[67] ——, "(POSTER) Overtake: Opportunistic Routing and Concurrent Transmissions for TSCH," in *IEEE DCOSS*, 2020.

[68] ——, "Opportunistic Routing and Synchronous Transmissions Meet TSCH," in *IEEE LCN*, 2021.

[69] T. Chang, M. Vučinić, X. Vilajosana, S. Duquennoy, and D. R. Dujovne, "6TiSCH Minimal Scheduling Function (MSF)," Internet Engineering Task Force, Internet-Draft draft-ietf-6tisch-msf-14, 2014, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-msf-14

[70] A. S. Tanenbaum and H. Bos, "Section 2.4.2: Scheduling in Batch Systems - Shortest Job First," in *Modern Operating Systems*, 4th ed. Pearson Education, Inc., 2015, pp. 157–158.

[71] G. Exarchakos, I. Oztelcan, D. Sarakiotis, and A. Liotta, "plexi : Adaptive re-scheduling web-service of time synchronized low-power wireless networks," *Journal of Network and Computer Applications*, 2017.

[72] C. A. Boano and M. Schuß, "EWSN 2019 Dependability Competition Logistics Information, rev. 1," Jan. 2018. [Online]. Available: https://iti-testbed.tugraz.at/fileupload/static/fileupload/EWSN2019_DC_Logistics_1.pdf

[73] L. Råde and B. Westergren, *Mathematics Handbook for Science and Engineering*. Lund: Studentlitteratur AB, 2004.

[74] M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, and D. Simeonidou, "Evolving SDN for Low-Power IoT Networks," in *IEEE NetSoft*, 2018.

[75] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *IEEE INFOCOM*, 2015.

[76] T. Chang, M. Vučinić, X. V. Guillén, D. Dujovne, and T. Watteyne, "6TiSCH minimal scheduling function: Performance evaluation," *Internet Technology Letters*, vol. 3, 2020.

[77] E. Mozaffari Ahrar, M. Nassiri, and F. Theoleyre, "Multipath aware scheduling for high reliability and fault tolerance in low power industrial networks," *Journal of Network and Computer Applications*, vol. 142, 2019.

[78] T. Chang, T. Watteyne, X. Vilajosana, and P. H. Gomes, "Constructive Interference in 802.15.4: A Tutorial," *IEEE Communications Surveys Tutorials*, vol. 21, 2019.

[79] P. H. Gomes, T. Watteyne, P. Gosh, and B. Krishnamachari, "Competition: Reliability through Timeslotted Channel Hopping and Flooding-Based Routing," in *EWSN*, 2016.

[80] M. Baddeley, A. Aijaz, U. Raza, A. Stanoev, Y. Jin, M. Schuß, C. A. Boano, and G. Oikonomou, "6tisch++ with Bluetooth 5 and Concurrent Transmissions," in *EWSN*, 2021.

[81] C. A. Boano, T. Voigt, C. Noda, K. Römer, and M. A. Zúñiga, "JamLab: Augmenting Sensornet Testbeds with Realistic and Controlled Interference Generation," in *IEEE IPSN*, 2011.