Améliorer l'Internet industriel des objets grâce à la mise en réseau définie par logiciel : de la construction du réseau à la gestion des flux

Farzad Veisi

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE MATHÉMATIQUES, SCIENCES DE L'INFORMATION ET DE L'INGÉNIEUR

Laboratoire ICube – UMR 7357

THÈSE présentée par :

[Farzad VEISI GOSHTASB]

Défendu le : **20 Novembre 2023**

pour obtenir le grade de : **Docteur de l'Université de Strasbourg**

Discipline/ Spécialité : Informatique

Enhancing Industrial Internet of Things through Software-Defined Networking: from building the network to flow management

THÈSE dirigée par :
Dr. THEOLEYRE Fabrice — Directeur de recherche, CNRS, France

CO-ENCADRANT :
Dr. MONTAVONT Julien — Maitre de conférences HDR, Université de Strasbourg, France

RAPPORTEURS :
Prof. GUITTON Alexandre — Professeur, Université Clermont Auvergne, France
Dr. LOSCRI Valeria — Chargée de recherche HDR, Centre Inria de l'Université de Lille, France

---

AUTRES MEMBRES DU JURY :
Prof. CHRISMENT Isabelle — Professeur, Université de Lorraine, France
Dr. IOVA Oana — Maitre de conférences, INSA Lyon, France

Université de Strasbourg

École doctorale MSII

Enhancing Industrial Internet of Things through Software-Defined Networking: from building the network to flow management

THÈSE

pour obtenir le grade de

Doctorat de l'Université de Strasbourg

(mention informatique)

Defended on 20 November 2023

présentée par

Farzad VEISI GOSHTASB

Composition du jury

Directeurs de thèse: Dr. Fabrice THEOLEYRE, Directeur de recherche, CNRS, France
Dr. Julien MONTAVONT, Maitre de conférences HDR, Université de Strasbourg, France

Rapporteurs: Prof. Alexandre GUITTON, Professeur, Universite Clermont Auvergne, France
Dr. Valeria LOSCRI, Chargée de recherche HDR, Centre Inria de l'Université de Lille, France

Examinateurs: Prof. Isabelle CHRISMENT, Professeur, Universite de Lorraine, France
Dr. Oana IOVA, Maitre de conférences, INSA Lyon, France

Laboratoire des sciences de l'ingénieur, de l'informatique et de l'imagerie
ICUBE — UMR 7357

Acknowledgments

I would like to express my deepest gratitude and appreciation to all those who have contributed to the completion of this Ph.D. thesis. This journey, particularly starting in the middle of the Covid-19 pandemic, has been challenging yet immensely rewarding, and I owe my success to the support and encouragement of many individuals.

First and foremost, I extend my sincere thanks to my supervisors, Fabrice Theoleyre and Julien Montavont, for their unwavering guidance, insightful feedback, and constant support throughout the research process. Their expertise and mentorship have been invaluable in shaping the direction of my research.

I am grateful to the members of my doctoral committee, Alexandre Guitton, Valeria Loscri, Isabelle Christment, and Oana Iova, for their constructive feedback and valuable suggestions that significantly improved the quality of this work.

Not to forget my colleagues from the Network research group, including professors and students. A special thanks to the Ph.D. students who made my adaptation to the new country much easier. Renato Caminha, for all the unforgettable support that enabled me to gather essential administrative information to adapt to the conditions. Amaury Bruniaux, for all our discussions in the lab, and I remember well the idea of the second contribution of my thesis sparked when we had been discussing together about a research challenge. Jean Romain, for sharing his experiences, particularly in the preparation before the defense. Thomas Alfroy, Thomas Holterbach, Jean-Philippe Abegg, I wish the best for all of you. Also, for the new members of the team, Samir Si-Mohammed and Ahmad Mahmod, I wish the best for your research and future.

Special thanks to all of our dear friends with whom we have spent a very memorable time together in Strasbourg, especially our beloved couple friends: MohammadReza (Zolala) and Nastaran, MohammadReza (Dolatpoor) and Isun, Amir and Niloofar, Omid and Elnaz, Mohammad and Maliheh, Sina and Firoozeh, Daroush and Solmaz.

I extend my deepest gratitude to both my family and my wife's family for their unwavering support and encouragement throughout my Ph.D. journey. Their belief in my abilities and shared joy in my accomplishments have been instrumental in my success. I am truly fortunate to have such a supportive network of families, and I am grateful for their enduring presence in my life.

Last but not least, I want to express my deepest appreciation to my wife, Sanaz. Your unwavering support, encouragement, and enduring presence have been my greatest strength throughout the highs and lows of my Ph.D. journey. Your belief in me, coupled with your boundless support, has made all the difference. You are the golden gift of my life, and I am profoundly grateful for all the love and encouragement you have showered upon me. Thank you for being my rock and my source of inspiration. I cherish the endless support you have provided, and I am truly fortunate to have you by my side. Thanks, my love.

Farzad Veisi Goshtasb,
November 2023

Abstract

Industry 4.0 represents a transformation in the way industries operate, making them more efficient, adaptable, and responsive to application demands. It has the potential to drive significant advancements in manufacturing and industrial automation. In this context, Industrial Wireless Sensor Networks (IWSNs) play a pivotal role in enabling the Industry 4.0 revolution. The network infrastructure assumes a critical role in interconnecting sensors and actuators while adhering to Service Level Agreements (SLAs). Typically, each device hosts a critical application that generates data packets, necessitating high Quality of Service (QoS) in terms of reliability and latency.

Typically, IWSNs rely on deterministic protocols and mechanisms to minimize uncertainties, ensuring predictability and reliability. In the Medium Access Control (MAC) layer, time and frequency resources are dedicatedly allocated to devices for accessing the generally shared medium and sending their data without contention.

However, a network scheduler is needed to carefully manage resource allocation based on network demands. While distributed scheduling approaches fail to achieve optimal performance due to their partial knowledge of the network, centralized schedulings are able to define efficient schedules while considering applications' QoS requirements. Nevertheless, centralized scheduling approaches require a mechanism to gather network information and push the defined schedule to the network, making them challenging to implement practically.

Software Defined Network (SDN) presents high potential to enable centralized scheduling in IWSNs by collecting network statistics in an intelligent SDN controller and then pushing the defined scheduling rules to the devices. However, exploiting SDN in wireless networks presents additional challenges due to the lossy nature of wireless links, which can jeopardize communication between devices and the controller.

In this thesis, we present an SDN solution for IWSNs. Firstly, we develop a dedicated control plane configuration to ensure the reliability of control plane

communication. Then, we introduce a call admission system that enables devices to communicate their QoS requirements to the SDN controller, which then configures and allocates appropriate resources to each data plane flow.

Additionally, we propose an accurate and energy-efficient topology discovery and Link Quality Estimation (LQE) method, allowing the controller to construct a precise network topology view. This precise LQE approach is crucial for our scheduling algorithm, as it helps prevent overestimation or underestimation of resource provisioning, thereby avoiding bandwidth wastage and SLA violations.

Finally, we introduce an efficient solution for maintaining scheduled SDN networks due to the time-varying nature of wireless links. The controller monitors link quality changes and triggers reconfigurations when significant changes occur. This process involves both control and data plane updates, impacting schedules on the weak links. To tackle this, we propose an efficient mechanism that minimizes control traffic and reconfiguration time while preserving SLAs for critical flows.

List of Publications

Journal paper

- Farzad Veisi, Julien Montavont, and Fabrice Theoleyre. "Enabling Centralized Scheduling Using Software Defined Networking in Industrial Wireless Sensor Networks". In: IEEE Internet of Things Journal (2023).

Conference papers

- Farzad Veisi, Julien Montavont, and Fabrice Theoleyre. "SDN-TSCH: Enabling Software Defined Networking for Scheduled Wireless Networks with Traffic Isolation". In: 2022 IEEE Symposium on Computers and Com- munications (ISCC). IEEE. 2022, pp. 1–7.
- Farzad Veisi, Julien Montavont, and Fabrice Theoleyre. "Link Quality Estimation in Wireless Software Defined Network with a Reliable Control Plane". In: 2023 IEEE 9th International Conference on Network Softwarization (NetSoft). IEEE. 2023, pp. 10–18.
- Farzad Veisi, Julien Montavont, and Fabrice Theoleyre. "Efficient and Reliable Maintenance for SDN-based Scheduled Wireless Networks". In: 2024 IEEE Consumer Communications & Networking Conference (CCNC). IEEE. 2024.

We live on an island surrounded by a sea of ignorance. As our island of knowledge grows, so does the shore of our ignorance. (John Archibald Wheeler)

Contents

Chapter 1

Introduction

Contents

Wireless Sensor Networks (WSNs) merge sensor technology with wireless communication, creating a versatile platform with diverse applications [1]. They are designed to autonomously gather data from physical environments through plenty of sensor nodes, wirelessly transmitting this information to a central hub for more advanced analysis. This concept has found compelling applications across various domains. WSNs are used to monitor the environment for changes and hazards, including temperature, humidity, air quality, and pollution levels [2]. In smart cities, they enable tasks such as traffic management, waste management, energy efficiency, and public safety by providing timely data for decision-making [3]. Within healthcare facilities, WSNs support remote patient monitoring and improve the quality of life for the elderly people [4].

Relying on wireless communications as a key enabler brings a range of valuable features. One of the key benefits is cost reduction, as they eliminate the need for expensive wiring installation and maintenance [5]. Rapid installation, easy maintenance, and system reconfigurations further contribute to saving time and lowering operational costs. Additionally, wireless technologies provide a high level of flexibility, enabling easy addition and removal of sensor nodes and equipment [6].

WSNs are characterized by their resource limitations, including limited energy sources, constrained communication ranges, limited available bandwidth, and restricted processing and storage capabilities inherent to each sensor node [7]. These constraints present substantial technical challenges in designing and optimizing WSNs. Addressing these challenges involves innovative hardware design, energy-

efficient protocols, and advanced data management techniques to ensure optimal network performance.

Moreover, the design constraints of WSNs depend on the specific application [8] and the unique environment, influencing choices like communication protocols, node placement, data strategies, and power management. Therefore, effective design in WSNs requires a holistic understanding of both the resource constraints and the application-specific requirements to ensure the network operates efficiently and effectively.

1.1 Industrial Wireless Sensor Network Applications

Even more rigorously compared to the WSNs, Industrial Wireless Sensor Networks (IWSNs) must also guarantee specific Service Level Agreements (SLAs) of industrial applications. They demand specific characteristics in terms of ***reliability*** and ***latency*** that differentiate IWSNs from conventional WSNs [9, 10]. Additionally, due to longer operational periods between maintenance cycles and the potential lack of nearby power sources, low-power consumption is imperative for industrial wireless devices.

Industry 4.0 aims to enable automation and the employment of smart factories, thereby creating highly flexible and reconfigurable production lines [11]. By systematically collecting extensive data from the factory floor, a smart factory can achieve a high level of information transparency and enable more informed decision-making through data analysis. The primary motivation behind this analysis is to enhance industrial operations in several critical ways. Firstly, it enables ***predictive maintenance***, whereby historical and real-time sensor data is analyzed to predict equipment failures or performance degradation before they occur. By proactively scheduling maintenance only when needed, companies can reduce downtime, minimize repair costs, and extend the lifespan of machinery [9]. Secondly, data analysis aids in ***quality control*** by continuously monitoring sensor data for variations or deviations in production processes [12]. Any anomalies detected can trigger immediate adjustments to maintain product consistency and quality. Lastly, ***operational efficiency*** is improved as data analysis identifies inefficiencies or bottlenecks in industrial processes, allowing organizations to optimize their workflows and resource allocation, leading to cost savings and increased productivity.

As illustrated in Figure 1.1, a set of sensors and actuators are disseminated throughout the industrial environment to construct a wireless network [13]. Typically, multi-hop communication is essential in this context to overcome obstacles, extend coverage, and ensure reliable data transmission [14]. In industrial environments, where sensors are often distributed across expansive areas and may encounter

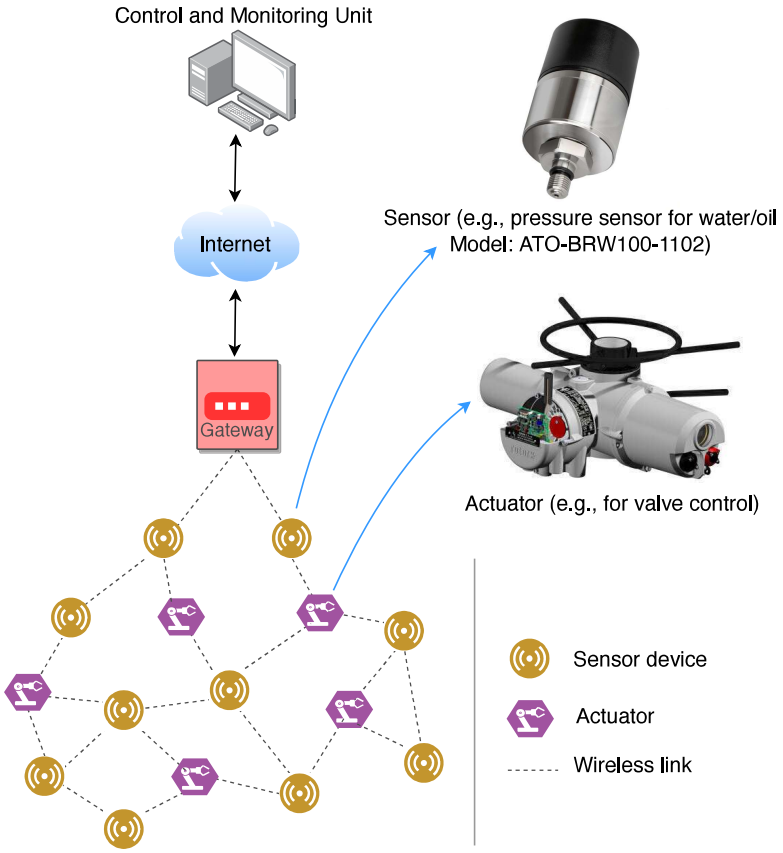

Figure 1.1: Industrial Wireless Sensor Network scenario

physical obstacles such as machinery, walls, or metal structures, multi-hop communication enables data to navigate through a series of intermediate nodes, effectively bypassing these obstacles. This capability not only extends the network's coverage to reach sensors in remote or obstructed locations but also ensures the reliability of data transmission. By offering redundancy, energy-efficient data relay, and adaptability to changing network conditions, multi-hop communication enhances the resilience and performance of IWSNs in demanding industrial settings.

As defined by the ISA100 committee (Table 1.1), industrial process automation applications are categorized into six classes, spanning from critical control to monitoring, each characterized by distinct latency and reliability requirements. Class 0 is the most time-sensitive, demanding less than 10 ms latency. Consequently, this class might not be suitable for transmission over a wireless multi-hop network [15]. Classes 1 to 3 are primarily designed for control loops, with required latencies ranging from 10 to 100 ms, depending on the specific application requirements. Monitoring applications are the least latency-sensitive, where data transmission delays can extend to several thousand milliseconds.

Table 1.1: Classes of industrial process automation applications [15]

Category	Class	Application	Latency	Criticality
Safety	0	***Emergency action:*** emergency shutdown, automatic fire control, Leak detection	10 ms maximum	Latency and reliably requirements increase
Control	1	***Closed-loop regulatory control:*** direct control of actuators, pumps, and valves, automated shutdown	10 - 100 ms based on application	
	2	***Closed-loop supervisory control:*** optimizing control loops, flow diversion		
	3	***Open loop control:*** annual adjustments	100 ms average	
Monitoring	4	***Alerting:*** event-based maintenance, vibration monitoring, motor temperature monitoring		
	5	***Logging:*** history collection, preventive maintenance record		

The sensory device utilizes built-in sensors to measure parameters such as temperature, humidity, pressure, and motion [16]. Adapting to different tasks in industrial automation, devices may generally employ different traffic generation behaviors. Periodic traffic is generated to ensure continuous machine health updates, enhancing efficiency through predictive maintenance [17]. For instance, closed-loop motion control for a conveyor belt uses position sensors to generate cyclical data, allowing a controller to adjust motor actions like adjusting the motor's speed and direction. Conversely, event-driven traffic promptly alerts sudden equipment problems [18]. For example, in a bottle manufacturing plant's closed-loop supervisory system, event-based sensor feedback regulates critical processes like bottle filling, capping, and defect detection, rather than relying on continuous monitoring.

Various traffic patterns can be explored within IWSNs, such as convergecast and point-to-point [19]. In the convergecast pattern, sensors transmit their measurements to a central computing control unit for subsequent processing. Point-to-point traffic enables direct data exchange between two specific nodes. Also, the controller can send a command to a sensor node or actuator to configure or trigger an action. A gateway device serves as a bridge, connecting the wireless network to external computing systems, typically through an IP network.

Achieving both high end-to-end reliability and bounded latency at the same time is crucial in IWSNs. This means that a certain percentage of generated packets must be received at the destination before a given deadline [20], enabling timely decisions and reactions. However, meeting these strict requirements becomes challenging due to the lossy and broadcast nature of wireless links, which can lead to packet failures

caused by various factors such as wireless channel conditions, collisions, or external interference [21].

1.2 Motivation and Contribution

Deterministic Medium Access Control (MAC) protocols are well-suited for IWSNs by minimizing uncertainties and providing predictable communication [22]. They achieve this through techniques like timeslot allocation, and frequency hopping, ensuring low-latency and reliable communication. However, a scheduler is essential to carefully allocate time and frequency resources to each transmitter, ensuring adherence to the defined SLAs (end-to-end Packet Delivery Ratio and latency requirements) of each application. Distributed schedulings cannot guarantee a collision-free schedule due to the suboptimal view of the network [23]. Indeed, each pair of devices has to locally negotiate the resources to use, which may create collisions with already established interfering flows. Alternatively, centralized scheduling algorithms can provide stringent guarantees by utilizing graph-based and coloring approaches. More precisely, thanks to the comprehensive knowledge of the network, determining the optimal resources within a central entity is straightforward. However, gathering the network information to define the schedule and pushing the schedule to the network remain challenging tasks.

Software Defined Network (SDN) paradigm proposes significant potential for enabling efficient centralized network management [24]. SDN involves the separation of the control plane (the intelligence part) and the data plane (the forwarding part) in network devices, introducing centralized control and programmability to network management. In this framework, SDN controller(s) (logically centralized) makes decisions in terms of rules governing how data packets should be forwarded and then pushes these rules to data plane devices. Forwarding devices only need to look up in their flow table to handle data packets efficiently, simplifying packet processing and eliminating the need for complex, distributed logic. The centralization empowers the SDN controller with a comprehensive view of the network, allowing for precise definition of forwarding rules. A southbound API serves as the communication interface between the SDN controller and forwarding devices. Indeed, devices need to regularly interact with the controller to inquire about new flows and share statistical information. In response, the controller configures the devices with the appropriate forwarding rules. The southbound API protocol should be reliable enough to ensure a timely communication between SDN controller(s) and devices.

In particular, the SDN concept can be applied in IWSNs to enable precise centralized scheduling, ensuring flow guarantees for critical applications [25]. The control plane is responsible for gathering network information and pushing the defined

schedule in the form of rules. However, in wireless networks, an SDN controller assumes more extensive role, which entails additional feature requirements compared to its classical tasks in wired networks. The controller needs to accommodate unreliable and shared wireless links, preventing collisions while establishing a robust control plane communication. It must also configure the data plane flows with sufficient radio resources to meet the SLAs of applications. This entails reserving extra radio resources for weak links, enabling nodes to retransmit packets in case of transmission failures.

To define the forwarding rules and schedule radio resources, the SDN controller relies on local information from neighboring nodes and the link qualities provided by each node to the controller. However, this entails additional procedures for the devices beyond their forwarding tasks to gather this information. They need to effectively discover their neighbors through control traffic exchange. This procedure must be energy-efficient to be compatible with battery-constrained devices in IWSNs. Additionally, the gathered information must be accurate enough to avoid misleading the controller's decisions. Erroneous link quality can cause the controller to either over-allocate or under-allocate resources, potentially resulting in bandwidth wastage or SLA violations, respectively.

Furthermore, the SDN controller should incorporate fault tolerance mechanisms for updating the network configuration when network conditions, such as link quality and traffic load change. Since changing link quality can affect multiple flows, it places a significant reconfiguration burden on the controller, necessitating updates to the routing and scheduling of all the impacted flows. This issue can impact both the reconfiguration time for critical flows and the energy consumption of the network.

Main Research Question

- How to establish a reliable control plane over multi-hop, lossy and shared wireless links?
- How to enable centralized scheduling for flow guarantees in critical IWSN applications?
- How to provide accurate and energy-efficient topology discovery and link quality estimation in software defined IWSNs?
- How to enable cost-effective and continuous network reconfiguration in software defined IWSNs?

We present three main contributions in this thesis to address those research gaps.

1.3 Structure of Thesis

In Chapter 2, we provide the essential background knowledge required to comprehend our contributions and provide insights into the state of the art solutions for SDN in IWSNs.

In Chapter 3, we present our first contribution, an SDN architecture for IWSNs, with a focus on ensuring flow guarantees. We provide a detailed explanation of how a reliable control plane is configured for a new node when joining the network. Additionally, we describe the process of admitting a data flow and configuring sufficient resources for it.

In Chapter 4, we present our second contribution on accurate and energy-efficient link quality estimation for software defined IWSNs. Additionally, we evaluate the efficiency of different control plane approaches in terms of energy consumption and reliability.

Chapter 5 extends the functionality of the SDN controller introduced in Chapter 3 to continuously update the network configuration as wireless link quality changes. We detail how we minimize the reconfiguration cost in scheduled SDN networks.

Finally, in Chapter 6, we present our conclusions from the research conducted during this doctoral program and discuss interesting research directions in the field of SDN scheduled networks.

Chapter 2

Background & State of the Art

Contents

This chapter provides an overview of industrial applications and standard technologies, with a particular focus on IEEE 802.15.4-TSCH [26] protocol. It follows by introducing the concept of Software Defined Network (SDN) and its exploration in the context of wireless sensor networks, specifically Industrial Wireless Sensor Network (IWSN). Furthermore, the chapter explores the discussion of state-of-the-art solutions in this field. Additionally, the chapter investigates the crucial role of topology discovery and link quality estimation in topology construction and influencing resource allocation within wireless SDN networks.

2.1 Industrial Wireless Sensor Network

As applications require a high guaranteed delivery ratio and low latency in IWSNs, they also encounter a critical challenge regarding their network lifespan. Due to the limited resources and small size of wireless nodes, the implementation of a complex and computationally exhaustive communication protocol becomes impractical [27]. So, to attain sufficient efficiency, multiple networking layers are engaged.

The growing number of industrial applications has led to the emergence of different wireless communication standard technologies, each designed to fulfill the specific needs of industrial environments. These technologies differ in features such as communication range, data rate, power consumption, and network reliability, providing options to address different industrial requirements.

2.1.1 Industrial Networking Stack

Various mechanisms and features must be considered across the network layers to overcome IWSN constraints and ensure efficient communication.

The Medium Access Control (MAC) layer plays a crucial role in managing access to the shared wireless medium among sensor nodes. It can employ different mechanisms, such as contention-based, contention-free approaches, and duty cycling techniques [28]. In contention-based protocols, nodes compete for access to the wireless channel when they have data to transmit. This competition is mainly done using Carrier Sense Multiple Access (CSMA) [29], where nodes listen to the channel before transmitting. If the channel is idle, they can transmit, else, they back off and try again later. While contention-based protocols are simple and efficient for low to moderate network loads, they can lead to collisions and increased energy consumption in high-density networks [30]. Therefore, they are not recommended for use in applications that require Service Level Agreements (SLAs) [31].

Contention-free protocols allocate dedicated resources such as timeslots or frequencies to different nodes, allowing them to transmit without contention. Time Division Multiple Access (TDMA) is a common example of contention-free MAC. TDMA assigns non-overlapping timeslots to nodes, reducing collisions and ensuring predictable access to the channel. However, contention-free MAC requires synchronization between nodes and may not be as flexible in handling dynamic traffic patterns [32].

Duty cycling is commonly used to conserve energy in Wireless Sensor Network (WSN)s by periodically putting nodes into a low-power sleep state [33]. The duty cycle is the ratio of time the node stays active to the total time. Instead of keeping nodes active all the time, they alternate between active and sleep periods. During the sleep period, a node turns off its radio and other power-consuming components,

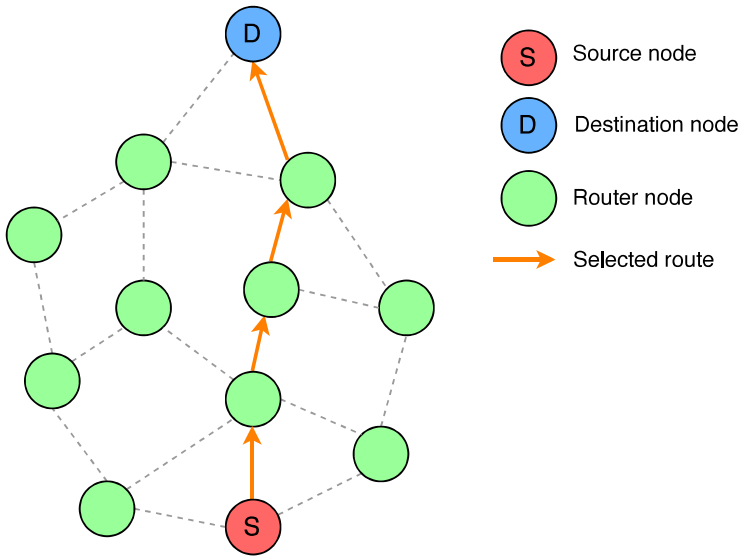

Figure 2.1: Routing scenario in wireless networks

which reduces energy consumption. When a node needs to transmit or receive data, it wakes up from the sleep state and engages in the network operation. It is worth noting that the duty cycle concept is applicable to both contention-based [34] and contention-free [35] approaches. ContikiMAC [36] exploits duty cycling while relying on a contention-based mechanism to access the channel. On the other hand, IEEE 802.15.4-TSCH [26] can allocate time-frequency blocks to different nodes, allowing them to awaken solely during their designated slots.

The routing protocol (Figure 2.1) plays a significant role in finding the most reliable path from the source to the destination in the multi-hop networks [37]. There are various routing protocols designed to meet the specific challenges and characteristics of WSNs. LOADng [38] emphasizes on-demand path establishment, DSDV [39] utilizes a table-driven approach, LEACH [40] focuses on energy efficiency with clustering, while RPL [41] is designed to facilitate convergecast routing in low-power and lossy networks. The choice of routing protocol depends on the application requirements, network size, energy constraints, and the desired trade-offs between factors like energy efficiency, latency, and reliability. For instance, energy-efficient protocols (*e.g.*, LEACH [40]) may prioritize energy consumption and residual energy [42], while real-time applications may emphasize low end-to-end delay and high throughput [43] (*e.g.*, RPL [41]).

Cross-layer network design plays a pivotal role in the realm of IWSN applications. It empowers the configuration, optimization, and adaptability of network protocols and functionalities to fulfill the stringent SLAs demanded by industrial applications.

2.1.2 Industrial Standard Technologies

Wireless standard technologies are purposefully designed with specific goals in mind to effectively meet the requirements of industrial applications (Table 2.1). For example, the Long Range Wide Area Network (LoRaWAN) is specifically designed for long-range communication with low data rates and low-power applications such as environmental monitoring, asset tracking, and smart agriculture. It follows a star topology, where end devices communicate directly with the gateway [44]. Devices exploit a random access method for upward data transmission, which can lead to frequent collisions [45].

Bluetooth Low Energy (BLE) [46] targets short-range and low-power applications such as healthcare and smart house applications. It employs a star topology where a central device, such as a smartphone or a computer, communicates with peripheral devices, like sensors or wearable devices, using short data packets and efficient power management mechanisms [47]. BLE utilizes Frequency Hopping Spread Spectrum (FHSS) for data transmission, enhancing its robustness against noise and interference. However, BLE's star network topology has limitations in terms of scalability and range. To address these challenges and enable larger-scale networks, Bluetooth Mesh was introduced [48]. By designating certain nodes as relays, it introduces a mesh networking topology that enables multi-hop communication, effectively extending range and coverage.

The IEEE 802.11 (WiFi) standard is commonly used for high-rate communication in enterprise and home networks. Its advanced version, IEEE 802.11ah, is specifically designed for low-power, long-range applications in the Internet of Things (IoT) domain. Unlike traditional Wi-Fi versions, IEEE 802.11ah operates in the sub-1 GHz frequency band, such as 900 MHz, which allows it to achieve extended coverage and better penetration through obstacles like walls and buildings, making it suitable for large-scale and outdoor IoT deployments [49]. Additionally, 802.11ah uses narrower channel bandwidths (1 MHz or 2 MHz) compared to regular Wi-Fi, which contributes to its improved energy efficiency [50]. These narrower channels reduce power consumption, enabling devices to operate on battery power for extended periods, making IEEE 802.11ah an excellent choice for energy-constrained IoT applications. Furthermore, 802.11ah supports a large number of nodes per network, making it scalable for scenarios with thousands of connected devices [51].

IEEE 802.11ax (Wi-Fi 6) exploits the advancements of IEEE 802.11ah, utilizing features like Basic Service Sets (BSSs) coloring and Target Wake Time (TWT) to enhance network efficiency and performance [52]. BSS coloring assigns distinct "colors" to BSSs within a frequency spectrum, reducing interference from neighboring networks and enhancing overall network performance. TWT coordinates device wake times, optimizing power efficiency by allowing devices to synchronize their

Table 2.1: Comparison of different standard technologies for IWSN

	MAC scheme	Data rate	Power consumption (mA)	Range	Topology	Frequency
LoRaWAN	ALOHA	50 Kbps	Tx:21-41 Rx:12	15 km	Star	Sub GHz (ISM)
BLE	TDMA	1 Mbps	Tx:39 Rx:37	50 m	Star, mesh	2.4 GHz (ISM)
WiFi (802.11ah)	CSMA/CA	150 Kbps	Tx:100-300 Rx:50-150	1 km	Star	Sub GHz (ISM)
WiFi6 (IEEE 802.11ax)	OFDMA	9.6 Gbps	Tx:100-400 Rx:50-200	240 m	Star	2.4/5/6 GHz (ISM)
NB-IoT	TDMA/FDMA	250 Kbps	Tx:100-300 Rx:50-150	10 km	Star	Sub GHz (LTE)
WirelessHART	FTDMA	250 Kbps	Tx:20 Rx:9	225 m	Mesh	2.4 GHz (ISM)
ISA100.11a	FTDMA/CSMA-CA	250 Kbps	Tx:27 Rx:15	100 m	Star, mesh	2.4 GHz (ISM)
TSCH	FTDMA/CSMA-CA	250 Kbps	Tx:18 Rx:20	100 m	Star, mesh, tree	Sub/2.4 GHz (ISM)

communication with access points, crucial for battery-operated devices. Wi-Fi 6 extends the benefits of Multi-User, Multiple Input Multiple Output (MU-MIMO), allowing both downlink and uplink MU-MIMO communication. This means the access point can simultaneously communicate with multiple devices using multiple spatial streams. By integrating these elements, Wi-Fi 6 effectively addresses the challenges posed by increasing device densities and data demands, offering higher performance, efficiency, and support for diverse environments such as stadiums, airports, and smart homes.

Narrowband Internet of Things (NB-IoT) [53] is a low-power standard technology enabling efficient and reliable communication in IoT devices via the existing Global System for Mobile (GSM) network and utilizing the licensed cellular spectrum of 4G LTE networks. NB-IoT operates in the sub-1 GHz frequency band, providing excellent coverage, range, and the ability to penetrate deep indoors and overcome obstacles like walls and buildings [54]. This unique capability makes NB-IoT well-suited for remote and hard-to-reach locations, offering reliable connectivity for IoT applications in challenging environments. In a star topology, IoT devices (end nodes) connect to and communicate with a central network element called the base station. The base station acts as a gateway to the core network, enabling data exchange between the IoT devices and the cloud or other servers.

WirelessHART [55], introduced by the HART (Highway Addressable Remote Transducer) foundation, is a widely adopted standard technology for industrial process automation, providing robust and secure communication. Its self-organizing mesh topology enables large-scale networks with redundant communication paths, enhancing reliability and fault tolerance [56]. To cope with the challenges of noisy industrial environments, WirelessHART employs frequency hopping mechanisms, dynamically changing channels to minimize interference and ensure consistent and

reliable data transmission across the network [57]. Additionally, it offers deterministic and time-synchronized communication, making it suitable for critical real-time applications in the industrial setting. The protocol's emphasis on low-power operation and extended battery life optimizes device performance and minimizes maintenance efforts in harsh industrial conditions.

ISA100.11a [58], an industrial wireless communication standard developed by ISA and sharing similarities to WirelessHART, aims to provide wireless communication for a broader range of industrial automation and control applications. This includes applications in process industries, discrete manufacturing, and infrastructure [59]. Its PHY layer utilizes frequency hopping and adaptive channel blacklisting for interference mitigation [59]. The MAC sublayer combines TDMA and CSMA/CA approaches, supporting mesh and star topologies. More precisely, the system manager offers three operational alternatives: slotted hopping, slow hopping, and hybrid slotted/slow hopping [60]. Slotted hopping employs equal-duration channel-hopping timeslots for single transaction data exchange; slow hopping designates consecutive timeslots over 100-400 ms for a channel, intensifying power consumption due to continuous listening; and hybrid hopping combines these, allowing periodic data via slotted hopping and sporadic data on a contention basis through slow hopping, addressing the needs of devices with imprecise timing or intermittent network contact.

Finally, inspired by WirelessHART and ISA100.11a standard technologies, IEEE 802.15.4-TSCH stands out as a superior choice due to its adherence to an open standard. This open standard promotes interoperability, allowing for seamless integration with various devices and systems, and provides a broader range of vendor options. TSCH is specifically designed for low-power and constrained devices [61]. It offers high flexibility in defining an efficient communication schedule by leveraging the TDMA and frequency hopping features. IEEE 802.15.4-TSCH allocates specific timeslots to devices for data transmission, allowing for precise control over when and how often devices can communicate. This flexibility enables efficient coordination of communication and ensures deterministic and predictable data exchange.

In the realm of industrial automation applications, where short communication ranges prove sufficient [62], the need arises for a standard technology that prioritizes high reliability rather than high data rates (*e.g.*, WiFi), particularly for transmitting sensory data. Here, IEEE 802.15.4-TSCH emerges as a fitting candidate, presenting adaptable scheduling and energy-efficient attributes. Furthermore, it delivers superior deployment efficiency compared to its predecessors, namely WirelessHART and ISA100.11a.

2.2 IEEE 802.15.4-TSCH Background

IEEE 802.15.4 is a standard technology specifying the Physical Layer (PHY) and MAC layer protocols for Low-Rate Wireless Personal Area Networks (LR-WPAN). The initial release, IEEE 802.15.4-2003 [63], operates in 2.4 GHz and sub-GHz frequency bands. The MAC layer adopts a flexible protocol with a superframe structure containing an active and an inactive periods. Active period comprises the Contention Access Period (CAP) and the Contention-Free Period (CFP). During the CAP, nodes contend for channel access through Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), while the CFP is dedicated to time-slotted, deterministic, and contention-free data transmission for time-critical or guaranteed delivery applications. However, operating on a single radio channel can jeopardize the reliability of communication in interfering environments [64].

IEEE 802.15.4b-2006 [65] introduced enhancements, such as an optional PHY in the 868/915 MHz band with higher data rates [66]. Many existing standards, like ZigBee, WirelessHART, and ISA100.11a, build upon this standard with specific upper layer adaptations. Despite its wide use, the original IEEE 802.15.4 standard has limitations in supporting mission-critical applications in terms of high reliability and bounded latency [67].

To address these requirements, IEEE released IEEE 802.15.4e [68] in 2012. This enhanced version incorporates five improved MAC protocols, known as MAC behavior modes, making it suitable for applications like smart healthcare and industrial process automation and control [69]. Among these modes, Time Slotted Channel Hopping (TSCH) focuses on enhancing reliability in industrial environments susceptible to interference.

IEEE 802.15.4-TSCH [26] exploits a frequency-time division multiple access (FTDMA) MAC layer, with a channel hopping mechanism. Transmissions rely on an FTDMA matrix, that defines when a transmitter can start a transmission, and which channel it has to use. In the subsequent sections, we detail the key features and mechanisms of the TSCH network.

2.2.1 Medium Access

TSCH relies on a slotframe structure that is repeated over time. A slotframe can be represented as a scheduling matrix composed of cells (pairs of timeslots and channel offsets). Each node is configured with the list of cells where it is authorized to transmit or receive. Obviously, if no pair of transmitters is allocated to the same cell, no collision occurs. The standard defines two types of cells in the scheduling matrix:

a dedicated cell is allocated to one transmitter. The transmitter waits for a fixed

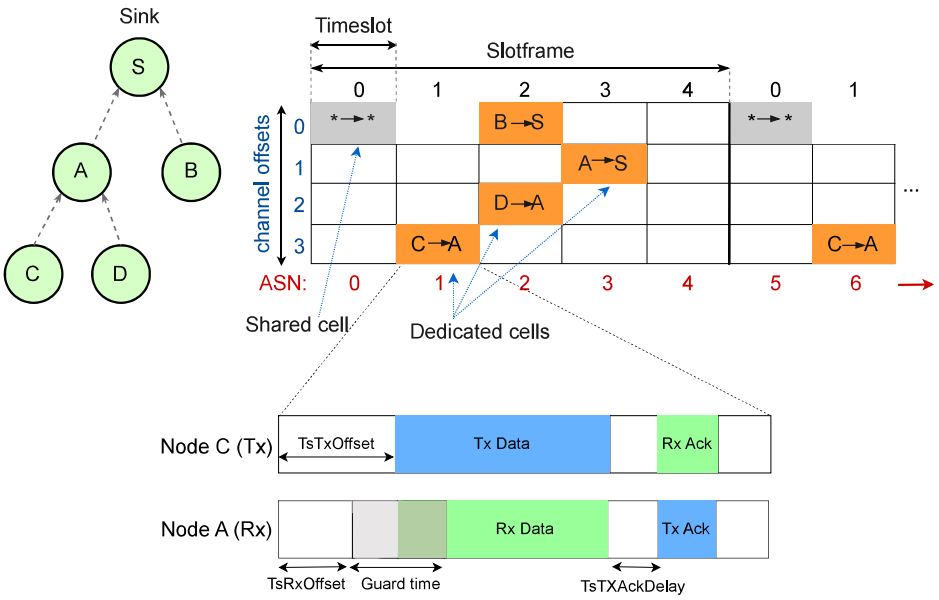

Figure 2.2: Simple TSCH schedule with shared and dedicated cells

offset from the beginning of the cell to accommodate clock drifts, and then sends its frame. If the same dedicated cell is assigned to interfering transmitters, collisions will be repetitive.

a shared cell can be allocated to more than one transmitter, so collision may happen between two concurrent transmissions. Thus, a contention resolution mechanism is applied for acknowledged packets. If no `ack` is received, a transmitter waits for a random number of shared cells to retransmit the same packet. Thus, collisions may be quite frequent in shared cells [70].

Figure 2.2 illustrates a simple TSCH schedule, including both shared and dedicated cells. Each node has a dedicated cell to communicate with its next hop toward the final destination. Additionally, all nodes can use the shared cell for broadcast traffic. For instance, nodes can transmit broadcast packets on the shared cell, allowing all neighbors to receive them, while critical data packets are sent over dedicated cells.

2.2.2 Association to TSCH Network

The association process begins with a new node entering an initial discovery phase. The new node continuously scans the available channels and listens for Enhanced Beacons (EBs) transmitted by already associated nodes in the TSCH network. These EBs contain Information Elements (IEs) about the network's parameters,

including the network's PAN ID (Personal Area Network Identifier), the network authentication information, and the TSCH schedule details.

TSCH utilizes Absolute Sequence Number (ASN) as a global clock within the network. The ASN represents the count of timeslots that have elapsed since the network bootstrapped. Extracting the ASN and schedule IEs (slotframe ID, size, and timeslot type) lets a node to identify the start of the slotframe and wake up during its designated timeslots and frequency channel.

The advertising policy of EBs is left unspecified by the standard. Balancing synchronization needs with energy efficiency is important [71]. Too frequent EB transmissions can lead to higher energy consumption, which might not be suitable for battery-powered devices.

In an environment with potential interference, adjusting the beacon interval can help mitigate collisions and enhance the reliability of communication in the network [72]. Some TSCH networks dynamically adjust the beacon interval based on network conditions [73]. If there are changes in network topology and traffic load, the EB interval might be adapted accordingly.

2.2.3 Synchronization

Device-to-device synchronization is necessary in TDMA-based networks to prevent collisions and ensure coordinated and efficient communication among devices sharing the same channel. In IEEE 802.15.4-TSCH network, each node selects a neighbor as time source to create globally a synchronization tree, rooted at the PAN coordinator. During the joining process, a node chooses the sender of the EB packet as its time source, thereby establishing a shared time reference for accessing its schedule within the slotframe.

However, due to imperfections in the crystal oscillators of devices, there can be slight variations in their frequencies. Over time, these variations can accumulate and cause clock drift in the time synchronization between a node and its time source [71]. To compensate for this clock drift, the standard defines two mechanisms: data frame-based resynchronization and acknowledgment-based resynchronization.

In data frame-based resynchronization (Figure 2.3), the transmitter serves as the time source. As the transmitter sends its frames after a fixed offset from the beginning of the timeslot, the receiver can identify the drift by subtracting the time it receives the data packet from the expected reception time. This information helps the receiver adjust its clock accordingly. On the other hand, in acknowledgment-based resynchronization, the receiver calculates the clock drift and includes it in the acknowledgment (`ack`) packet. The transmitter then utilizes this information to resynchronize its time with the receiver. The TSCH standard incorporates a keepalive mechanism to address situations where a node does not receive any packets

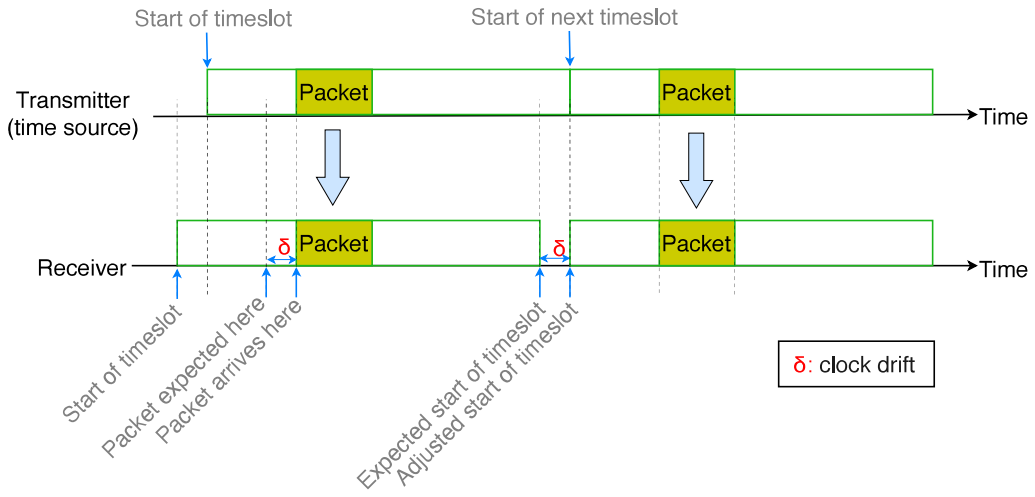

Figure 2.3: Frame-based synchronization schemes in TSCH network

from its time source for a while. This mechanism involves the periodic transmission of small control packets by the node to request synchronization information from its time source. The maximum period for these keepalive packets is determined by the guard-time and the maximum clock drifts [74]. When a node cannot synchronize itself with its time source after a given threshold time, it initiates disassociation. It flushes its synchronization-related parameters and scans the network for potential new time sources.

2.2.4 Frequency Hopping Mechanism

FTDMA helps to combat ***internal*** interference: when two transmitters may interfere, the network schedules their transmissions at different instants or frequencies. Frequency hopping mitigates multipath fading and ***external*** interference [75]. Indeed, other networks may be colocated, using the same unlicensed band, adding noise to some frequencies. Frequency hopping helps to reduce the probability of repetitive packet losses: link-layer retransmissions will use a different frequency. More precisely, the index of the channel to use is derived from its ASN as follows:

$$Channel = HSL\,[(ASN + CH\_Off)\,mod\,|HSL|\,] \tag{2.1}$$

The Hopping Sequence List (HSL) represents a sequence of $|HSL|$ channels available in the 2.4 GHz frequency band. As per the standard, there are 16 frequency channels defined, allowing for a maximum of 16 collision-free concurrent transmissions. The parameter CH_Off refers to the channel offset assigned to the transmitter during the timeslot.

Supported by extensive experimental research in interfering environments, [64] demonstrates how the channel hopping mechanism of TSCH can outperform the basic operation of IEEE 802.15.4 technology over a single channel. Elsts *et al.* [76] intend to enhance channel selection in the face of external interference. By comparing various methods, the authors highlight that utilizing Packet Reception Ratio (PRR) based channel assessment yields superior results. This approach reduces retransmissions, and maintains high Packet Delivery Ratio (PDR), even when interference is heavy. Gomes *et al.* [77] introduce decentralized algorithms, where receiver/transmitter pairs collaboratively establish a local blacklist (a list of cells that are not allowed to be used) by evaluating PDRs. They formulate channel quality assessment as a multi-armed bandit challenge, demonstrating that effective blacklists can be created without a distinct learning phase, yielding outcomes close to optimal performance.

2.2.5 Scheduling

The scheduling mechanism, *i.e.*, how to allocate cells in the slotframe, is not defined in the standard. This allows the network designers and implementers to tailor the scheduling approach based on their specific applications' requirements.

Different objective functions can be employed for scheduling tasks. Energy-aware scheduling strategically allocates timeslots by effectively coordinating active and passive timeslots, resulting in nodes with limited battery capacity being allocated less active timeslots [78]. Scheduling can prioritize allocating timeslots to nodes with higher data traffic demands, aiming to maximize total data throughput and optimize available network resources [79, 80]. When low communication latency is essential, scheduling can emphasize minimizing data traversal time within the network, offering advantages to time-sensitive applications like real-time control systems through reduced latency [81, 82]. More sensitively, within critical industrial applications, an efficient scheduling approach is imperative to ensure both the expected end-to-end reliability and bounded latency [83]. Specifically, a designated percentage of packets must be successfully delivered to the destination before a given deadline.

The 6TiSCH [84] protocol stack has standardized the 6P [85] protocol, which facilitates scheduling negotiation in TSCH networks. 6P enables neighboring nodes to add or delete TSCH cells on each other. A scheduling function in 6P determines cell additions and deletions, triggering 6P Transactions. The specification of the scheduling function remains unspecified. Depending on whether the scheduling function is performed in a distributed or centralized manner, we can achieve different levels of efficiency. TSCH supports both distributed and centralized algorithms [86]. We delve into the specifics of each in subsequent discussions.

2.2.5.1 Distributed Scheduling

In a distributed scheduler, each node executes a distributed algorithm to dynamically modify its local schedule. OST [87] presents an on-demand scheduling with traffic-awareness technique, which allocates per-link cells based on average traffic between nodes. It allocates some long-term cells based on average traffic between the link nodes, while additional temporary cells are allocated for bursty traffic. This approach uses existing data and `ack` packets for scheduling information exchange to minimize overhead. Temporary cell allocation piggybacks information onto data packets based on sender queue length. Successful packet reception allocates resources for subsequent packets in the queue. However, unreliable links can impact this technique, leading to increased packet losses and reduced reliability.

Numerous distributed algorithms have been proposed to function according to a routing tree topology, mainly built by RPL [41]. In DeTAS [88] scheduler, the network is organized as a tree in which every node knows the amount of traffic it will generate and receive from its children (convergecast traffic). This information is transmitted hop-by-hop to the root tree. The schedule is initiated by the root by scheduling the timeslots to receive the aggregated traffic from each child. To mitigate end-to-end delay and buffer overflows, DeTAS employs a strategy of alternating reception/transmission slots along the path toward the sink. Indeed, when a slot receives a packet, it is transmitted to the parent node in the subsequent slot. The main drawback of DeTAS lies in its static schedule, which lacks the ability to be updated in response to changing traffic conditions.

Wave [89] represents a distributed scheduling algorithm that exploits RPL information for its scheduling mechanism. It assumes that each node knows its set of conflicting nodes, acknowledging that collisions may arise during simultaneous transmissions. The protocol starts with the root node transmitting a start message to its child nodes. Upon receiving this message, each node selects a cell (a timeslot and a channel offset) for transmitting its initial packet. Then, it notifies this assignment to its conflicting nodes. It is worth noting that this entails a notable communication overhead resulting from the substantial number of exchanges for the scheduling procedure.

LDSF [81] enhances communication efficiency by introducing a structured approach to slotframe organization. Nodes positioned at even and odd hop distances from the border router (network gateway) are scheduled during alternating even and odd blocks, thereby effectively minimizing end-to-end delay. Through a sequential cell allocation strategy, nodes proactively anticipate packet arrival times, securing cells in upcoming blocks to minimize buffering delay.

Orchestra [90] is a very efficient solution to construct the IEEE 802.15.4-TSCH schedule in a distributed manner. Orchestra relies on RPL [41] to construct a route

toward the sink. Once RPL has converged, a device derives the cell used by each of its neighbors, using their IDs. More precisely, a pseudo-random function derives a cell coordinates (timeslot and channel offset) from the ID of each neighbor, in a slotframe. Orchestra relies on three different slotframes, to handle TSCH EBs, RPL packets, and data packets. Orchestra installs a dedicated broadcast slot from every node to its children for TSCH EBs in the first slotframe. Then, it installs a slot common for all nodes in the network for broadcast/unicast for RPL signaling in the second slotframe. Finally, it installs a dedicated unicast slot from every node to its RPL preferred parent in the third slotframe. Note that the slotframes possess the same order of priority as presented, and in cases of overlapping cells between two slotframes, the precedence is granted to the slotframe with the highest priority. Within these slotframes, Orchestra defines three types of cells:

- ***shared:*** it is one shared slot used by all nodes in the network for both Rx and Tx. The slot is installed at fixed coordinates, resulting in a behavior similar to slotted ALOHA.
- ***receiver-based:*** the cell (shared) coordinates are driven by the ID of receiver node, and all the neighbors use this cell to send packet to the receiver.
- ***sender-based:*** the cell (dedicated) coordinates are driven by the ID of sender node, and all the neighbors maintain a Rx cell accordingly to listen the sender.

As the Orchestra does not scale the schedule with the volume of traffic forwarded through a given radio link, it suffers from a poor network capacity, and most loaded nodes fail to send their packets. Additionally, as cells are defined according to a hash of the addresses, collisions may occur between some of the frames.

Minimal Scheduling Function (MSF) [91] is a popular standard that extends Orchestra [90] to mix per link and per node cell allocation. Orchestra is exploited for the control packet negotiation to reserve distributed cells on-demand with the 6P [85] protocol. This scheduling function provides a mechanism for each node to adjust its schedule to traffic variations, routing changes, and collisions. MSF operation relies on three types of cells:

- ***minimal cell:*** devices use a single shared cell to periodically send broadcast packets (such as Enhanced Beacons and routing packets) to bootstrap the network;
- ***autonomous cells:*** every node has a permanent receiver-based cell [90] used to send 6P [85] protocol packets;
- ***negotiated cells*** are dedicated cells negotiated by a pair of nodes to exchange traffic.

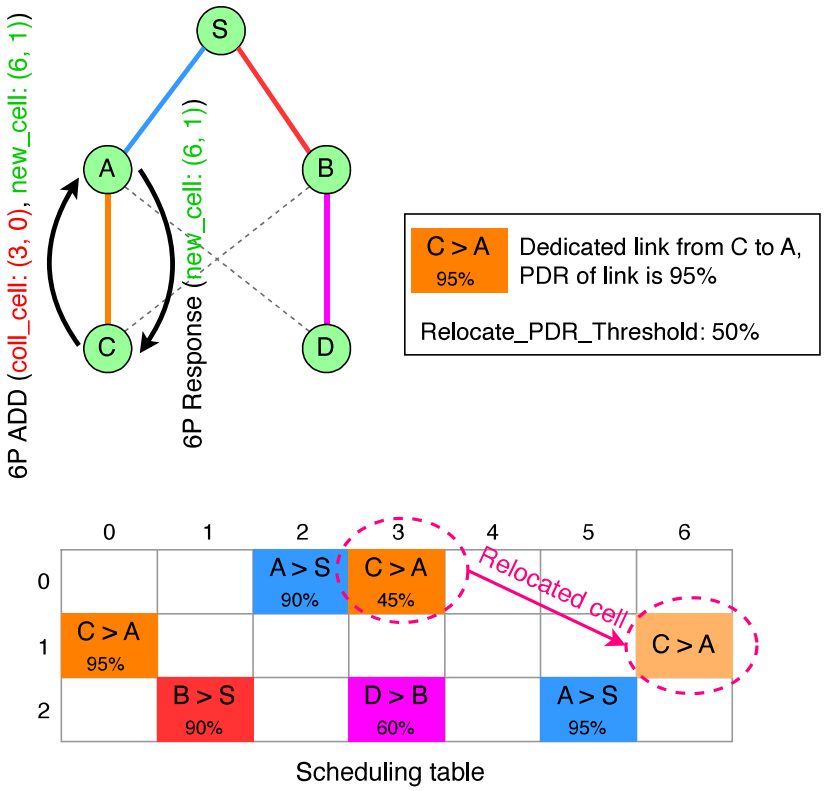

Figure 2.4: Relocation of colliding cell in MSF

MSF generally relies on RPL to build the network topology and derive autonomous cells. Upon RPL convergence, every node requests its parent for one negotiated cell using an autonomous cell. Then, a node can request additional negotiated cells if its current schedule cannot handle its current traffic load. Similarly, a node can remove negotiated cells in case of unused cells in its schedule. Also, MSF can detect colliding cells if they present low reliability compared with the other negotiated cells with the same neighbor. In that case, a relocation procedure allows the nodes to change the cell to use. Figure 2.4 illustrates an example in which node C detects that the PDR of cell $(3, 0)$ is lower than the `Relocate_PDR_Threshold` (50%). Thus, node C initiates a 6P packet to negotiate with node A to replace cell $(3, 0)$ with $(6, 1)$.

Distributed schedulings exhibit greater agility in responding to dynamic changes, as they do not rely on presumptions about radio topology or traffic volume, which allows schedules to be defined more promptly. In distributed schedulings, nodes lack comprehensive information about the network's status, traffic patterns, and schedules of neighboring nodes. This absence of global view can lead to collisions, interference, and suboptimal utilization of the available resources [23].

2.2.5.2 Centralized Scheduling

Centralized scheduling allows the scheduling algorithm to be executed on a centralized entity such as a Path Computation Element (PCE) [92]. A PCE has a complete view of the network topology, as well as the traffic requirements of nodes. Therefore, it can compute an efficient schedule while respecting guarantees (reliability and latency). The need for nodes to communicate with the central scheduler for schedule updates introduces significant communication overhead, consuming bandwidth and potentially causing delays in multi-hop lossy wireless networks [93]. In environments where network conditions change frequently, such as in mobile or highly dynamic scenarios, centralized scheduling algorithms might struggle to quickly adapt to these changes, leading to suboptimal performance [94].

AMUS [95] proposes a centralized scheduling scheme that allocates more cells to nodes closer to the sink, assuming those nodes have more traffic to forward, hence reducing the delay caused by interference or collisions.

TASA [96] aims to define a compact schedule. The algorithm prioritizes nodes with higher traffic loads, aiming to allocate bandwidth primarily to nodes facing greater limitations. This is achieved by employing distinct channels for conflicting links. For this purpose, it uses a combination of matching and coloring algorithms to build an appropriate scheduling.

Kausa [97] generates a schedule that adheres to the SLA. The allocation of slots considers the packet error rate (PER) of each link, determining the number of cells allocated. Moreover, the scheduling of transmissions follows the path while ensuring both reliability and latency requirements are met.

MABO-TSCH [77] integrates centralized scheduling scheme with local channel blacklisting. The scheduler, operating in a centralized manner, utilizes a coloring problem to arrange nodes based on their degree in the graph. To be reactive, a pair of nodes independently determines which physical channels to blacklist. To ensure consistency between the receiver and transmitter, they include the blacklist in the ACK frame along with a sequence number.

Choudhury *et al.* [98] introduce a centralized scheduling approach for TSCH at the cluster level, focusing on energy efficiency. The proposed mechanism generates a collision graph for each cluster in the network topology to arrange timeslots without overlap.

While many centralized scheduling algorithms have been proposed, they have been mostly evaluated with Monte-Carlo simulations, assuming that all the inputs are known accurately. However, collecting inputs and pushing scheduling decisions are challenging in wireless low-power networks: transmissions are unreliable. Software Defined Network can serve as a practical solution for this task.

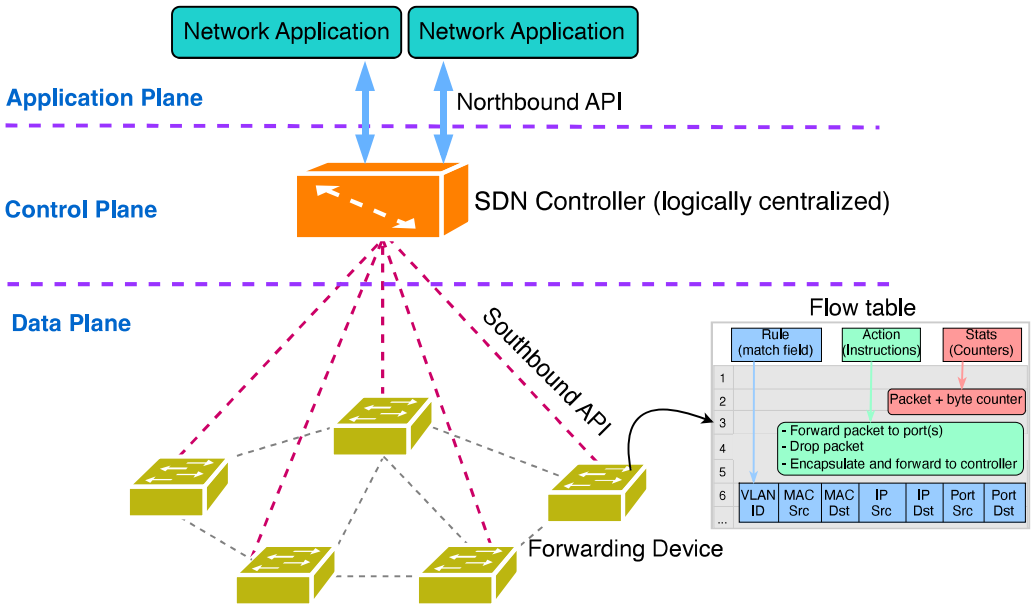

Figure 2.5: SDN architecture

2.3 Software Defined Networking for IWSN

SDN [99] decouples the intelligence part (control plane) of network devices from the forwarding part (data plane). It logically centralizes the network intelligence in the SDN controller, which has a complete view of the network, and makes optimal decisions on networking datapaths. The devices are kept as simple as possible to only perform the forwarding task.

Figure 2.5 illustrates the SDN architecture. Each forwarding device possesses a flow table(s) serving as a repository of flow entries that define distinct traffic flows. Flow entries, defined by SDN controller, dictate how these forwarding devices should process traffic flows. Each flow entry in the flow table includes matching rules and actions. When a packet arrives, the forwarding identifies the flow by matching header information, such as source and destination IP addresses, port numbers, and transport protocols, to flow entries. If a match is found, the corresponding action specified in the flow entry is executed. Actions can include forwarding the packet to a particular port, modifying packet headers, or sending the packet to an SDN controller for further processing.

The controller exploits a so-called southbound API to collect information on the network and push forwarding rules on the devices. OpenFlow [100] is a pioneering solution that focuses on the separation of control plane and data plane in network devices and provides a standardized way to manage and control the forwarding behavior of forwarding devices. OpenFlow-supported devices consist of two logical

components: a flow table that determines the forwarding rules, and an OpenFlow API responsible for managing communication between the OpenFlow switchs and SDN controller [101].

The SDN controller necessarily relies on a topology discovery mechanism to obtain comprehensive information about the link connectivity within the network and to construct an accurate representation graph of the network. This allows for efficient traffic routing, load balancing, fault detection, and overall network optimization. In wired SDN networks, the process of topology discovery relies on protocols like LLDP (Link Layer Discovery Protocol), OFDP (OpenFlow Discovery Protocol), and CDP (Cisco Discovery Protocol) to identify established connections [102].

Application plane plays a pivotal role by providing explicit network requirements and policies, which are subsequently translated into a set of forwarding rules [40]. These rules dictate the path and behavior of network traffic, effectively molding the actions of the underlying network framework. This translation process serves as a bridge between the high-level objectives set by the applications and the low-level configuration of network devices.

Numerous diverse scenarios leverage this translation mechanism to customize the network's behavior in response to precise operational requirements. For instance, dynamic traffic engineering optimizes network flows in real-time, Quality of Service (QoS) management ensures differentiated treatment for diverse traffic types, security and intrusion detection enforces comprehensive traffic controls, and network slicing achieves isolated environments for multiple tenants. Additionally, load balancing and application delivery ensure efficient resource utilization, while network monitoring and analytics provide insights into network performance and utilization patterns.

A northbound API acts as the gateway through which network applications establish communication with the SDN controller, enabling the exchange of commands, requests, and information. In this regard, various approaches have emerged, with many proposals advocating the utilization of RESTful APIs [103]. These APIs adhere to the principles of Representational State Transfer (REST), offering a standardized and interoperable means of interaction. However, it is important to note that while RESTful APIs provide a common foundation, the specifics of their implementation and design can vary based on the SDN architecture, controller platform, and the requirements of the network applications.

2.3.1 SDN for WSN: Limitations

The use of SDN in WSNs has emerged as a promising solution to alleviate the processing burden on constrained low-power devices and delegate it to a centralized controller [104, 105]. In conventional distributed WSNs, sensor nodes are responsi-

ble for both processing tasks and data transmissions. However, due to their limited computational capabilities, memory, and energy resources, efficiently handling complex processing tasks becomes challenging [106]. The controller, with its higher computational power, can effectively execute computationally intensive tasks, relieving the sensor nodes of this responsibility [107].

Nevertheless, the integration of SDN in wireless networks does introduce certain challenges, particularly related to communication overhead and latency [108]. In SDN-enabled WSNs, the sensor nodes primarily focus on forwarding and continuously communicate with the controller to access forwarding rules. This communication takes place over lossy wireless links, where successful transmission may require packet retransmissions. Consequently, the latency of control packets and the time required for flow rule configuration can be impacted.

While many related solutions adapt SDN for WSNs, the role of the SDN controller is often limited to establishing data aggregation and forwarding rules for devices, as well as determining the next hop. Consequently, the controller does not actively govern the behavior of the MAC layer.

Luo *et al.* [109] acknowledge several challenges that SDN faces in constrained environments. Significantly, they propose a template that allows OpenFlow flow tables to handle compact addressing schemes in non-IP WSNs. They suggest the inclusion of new OpenFlow forwarding rules, such as data aggregation, to enhance network performance. They highlight the potential overhead associated with the OpenFlow protocol in a wireless mesh and propose a solution suppressing retransmissions of SDN control messages from individual nodes. By providing sufficient time for the controller to respond, this approach ensures that nodes abstain from repeating control signaling when a controller response is imminent. While presenting potential solutions, this article is devoid of any supporting proof of concept.

SDWN (Software Defined Wireless Networks) [110] is developed to address the limitations of the SDN architectural model when applied to low-power networks (*e.g.*, IEEE 802.15.4), characterized by low capabilities in terms of memory, processing, and energy availability. The authors argue that certain fundamental aspects of the OpenFlow approach are incompatible with the diverse and intricate requirements of low-power wireless networks: OpenFlow assumes a network of high-speed Ethernet/MPLS switches and IP routers, primarily designed for reliable wired communication. Specifically, SDWN introduces a lightweight SDN architecture for IEEE 802.15.4 networks, enhancing performance and functionality while considering in-network data aggregation and adaptable rule definition. It incorporates a customizable flow table for low-power hardware by employing rules within a specific packet segment.

SDWN [110] led to the establishment of the widely recognized SDN-WISE [111].

This solution focuses on in-network data aggregation using a stateful flow table to apply rules per packet based on node state. The flow table contains Matching Rules, Actions, and Statistics. Matching Rules trigger Actions and update Statistics based on conditions. SDN-WISE incorporates node state in the flow table to reduce controller overhead, allowing nodes to make forwarding decisions based on the state of other nodes. This limits unnecessary packet transmissions and frequent controller consultation. The flow table deploys diverse actions depending on node state, enabling forwarding, dropping, modification of packets, radio deactivation, and in-network packet processing for data aggregation.

TinySDN [112] is an SDN architecture for low-power devices, utilizing TinyOS [113]. It introduces SDN-enabled sensor and controller nodes, where sensors serve as both switches and devices, forwarding data to controllers managing network flows. Multi-controller setup minimizes control latency, and sensors employ Collection Tree Protocol [114] to communicate with controllers via optimal link routes, forming collection trees for each root without needing awareness of multiple instances.

IT-SDN [115] enhances TinySDN with abstraction and separation, distinctively outlining the southbound controller, neighbor discovery, and controller discovery protocols. While it defines interfaces for neighbor and controller discovery, the specific protocols enabling these functionalities are not specified. Moreover, for addressing the overhead linked to flow installation, particularly in traditional OpenFlow networks [116], IT-SDN presents two approaches: "source routed" and "multiple flow." The former uses source-routed packets computed by the controller to avoid address flow table overload. The latter extends this by enabling a single packet to establish routes across multiple nodes simultaneously, eliminating the need for address flow tables.

SD6WSN [117] introduces an SDN architecture for 6LoWPAN networks, aiming to reduce latency by leveraging controller knowledge. It employs SDN-based agents in nodes, utilizing RPL topology and Constrained Application Protocol (CoAP) packets. SD6WSN achieves lower average latency than standard 6LoWPAN networks operating on RPL routing. The SDN controller employs a shortest path algorithm, resulting in more efficient peer-to-peer routes compared to RPL's sub-optimal paths [118]. However, the article lacks detailed information on the exact workings of the shortest path algorithm.

All the aforementioned solutions rely on shared mechanisms for accessing the wireless medium, leading to potential issues such as collisions, interference, and reduced overall network reliability. Given the inherent lossiness of wireless networks, the controller should also manage the behavior of the MAC layer by allocating sufficient resources for each hop along the path. Many existing schemes heavily rely on retransmissions as a means to enhance reliability [119, 120].

2.3.2 SDN for Scheduling Management

Industrial Wireless Sensor Network (IWSN) applications commonly utilize scheduled deterministic wireless networks to achieve high reliability and low latency. Therefore, it becomes necessary to adapt the SDN architecture to allocate time-frequency blocks as well [121].

uSDN [122] proposes a SDN architecture on TSCH networks with a primary emphasis on optimizing the signalling overhead in flow installation. When a node requests a flow, the controller installs the path (from the source and to the destination) as a flowtable action on the source node. Any upcoming packets within this flow are source-routed from the same source node. This approach allows the controller to avoid individual hop-by-hop flow rule installations, thus reducing the initial signalling overhead. Nonetheless, the overhead incurred by source routing piggybacking for each packet raises questions about its cost-effectiveness in comparison to the initial expense of flow installation. The control plane relies on autonomous algorithms, and on RPL that has been proved to exhibit oscillations under realistic conditions [123]. The authors of uSDN extend their work [122] to separate the radio resources of control traffic from data traffic to mitigate the effect of control overhead on normal network operation. For the control plane, a hop-by-hop scheduling mechanism is used to distributedly reserve a bunch of cells for a node toward the controller. However, the exact mec is worth noting that in their approach, the controller does not manage the scheduling of the data plane, solely defining forwarding rules for each flow.

Whisper [124] exploits a centralized controller that controls the behavior of RPL and 6P protocols. Their goal is to create a centralized management system with minimal modification in the network stack. The controller artificially manipulates the RPL rank of a node to force parent changes, setting up new paths. The controller also injects fake 6P commands to (de)allocate time-frequency blocks between two nodes. However, Whisper can not manage the routing and scheduling at per-flow level. It rather provides more flexibility for the RPL management system and is better suited for best-effort trafic scheduling.

Bello *et al.* [125] propose to modify SDN-WISE to support mobility in scheduled networks. In their scheme, the schedules of mobile nodes are already defined in the slotframe, and each mobile node has two dedicated upward and downward cells with each fixed neighbor. Depending on the network location of mobile nodes, the controller sends novel forwarding rules to the mobile nodes to let them know to which fixed node they should send data packets. The focus is rather on defining forwarding rules specifically when a node roams from one parent to another.

SDN primarily aims to dynamically reconfigure the network in response to changes [126, 127]. By continuously monitoring and analyzing network conditions,

such as traffic patterns, link quality, and topology changes, the SDN controller can dynamically reconfigure the network to adapt to these fluctuations [128]. The continuous reconfiguration capability of SDN ensures that the network remains resilient, and capable of meeting the evolving requirements of applications and services.

REACT [129] proposes a scheduling policy called ***gap-induced*** to reduce the latency and energy cost of network reconfiguration in industrial WirelessHART [55] networks. The policy introduces intentional gaps between transmissions of the same flow. If the scheduler needs to update the schedule, this flexibility avoids a complete reconfiguration. However, to push the new schedule, REACT uses individual ***DELETE*** and ***ADD*** command packets, modifying the schedule one by one on each device. Changes in link quality can impact multiple flows, leading to a storm of control packets that may disrupt convergence.

SDSense [130] presents a dynamic reconfigurable framework with a hierarchical controller architecture. This framework optimizes network components considering their slow and fast requirements. It assigns fast-changing functions to programmable sensors while maintaining slow-changing functions, like topology discovery, at the controller. An algorithm enhances TDMA resource utilization across the network. The architecture features a logically centralized controller handling topology control, TDMA scheduling, and data-rate allocation, ensuring efficient management of static or slow network elements from a global perspective. Additionally, modules for congestion control and data-rate reallocation adapt to local controller states on SDSense nodes, responding to network dynamics. Through simulations, the authors validate dynamic network reconfiguration's effectiveness, showcasing significant performance improvements over alternatives. However, the lack of detailed implementation hinders direct comparison with other SDN solutions.

SDNWISE-TSCH [131] extends SDN-WISE for scheduled TSCH networks, aiming to perform both efficient routing and scheduling tasks by SDN controller. Each flow has a specific priority and a deadline. The controller computes the shortest path through the Dijkstra algorithm. Paths are computed using a ***pressure*** metric defined as the amount of traffic scheduled for a link. The path that presents the minimum ***pressure*** value is selected for a flow. Then, the controller starts to define the schedule for the flow with the highest priority and the lowest deadline. It allocates one cell per hop and arranges flow cells back to back to respect the flow deadline. They enhance the OpenPath configuration packet of SDN-WISE to enable it to accommodate the TSCH schedule. However, they do not consider the actual quality of links in defining routes and schedules. Computed paths can have very low reliability, assuming the link qualities are perfect. Also, they do not define a flow isolation mechanism: a node may use any scheduled cell for any packet, impacting the flow guarantee. In addition, the control plane relies on shared cells that

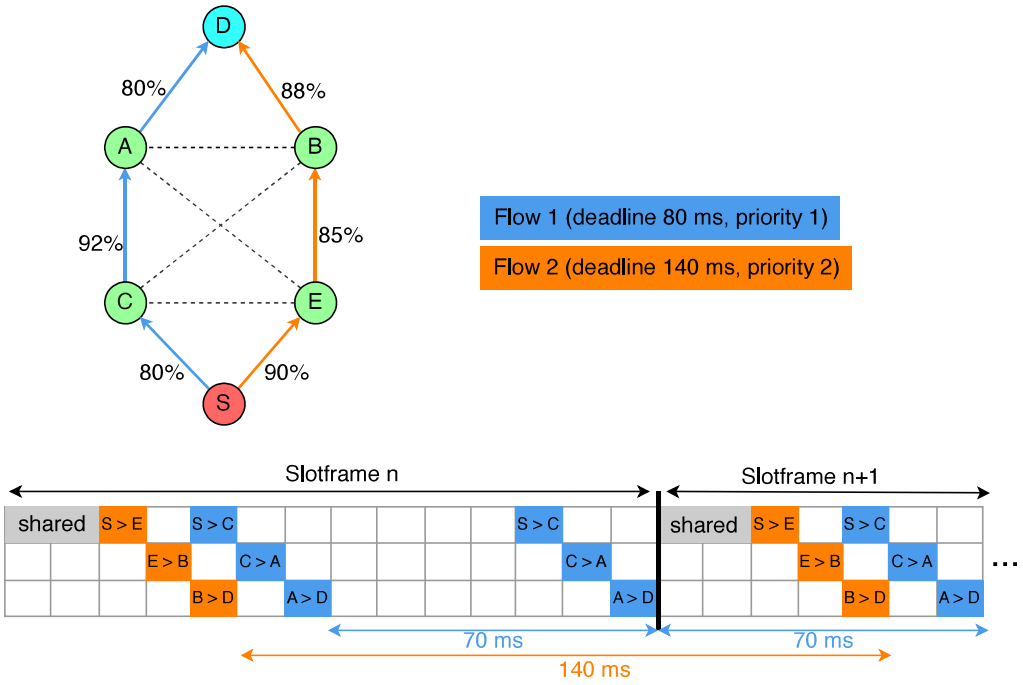

Figure 2.6: Scheduling flows with different deadlines and priority in SDNWISE-TSCH

can jeopardize the reliability of the contol traffic. Also, they extend this scheme to support multicast traffic for mobile nodes [132].

Figure 2.6 illustrates the operation of the SDNWISE-TSCH scheduler. Node S has two flows with different deadlines and priorities. The scheduler initially defines a schedule for flow 1 that has a higher priority. It allocates one cell to each hop, regardless of the PDR of the links, and repeats this schedule within the slotframe to meet the deadline. For flow 2, the scheduler selects the shortest path with the minimum scheduled flows, ensuring it does not choose a path through which flow 1 passes. Since the deadline of flow 2 equals the slotframe length, only one sequence of cells is scheduled for it within the slotframe.

To the best of our knowledge, and as overviewed here, no SDN solution supports per-flow guarantee in scheduled networks and allocates resources based on the link reliability. Table 2.2 presents the summary of the SDN solutions existing in the literature and their properties.

Table 2.2: Summary of related works with supported features

		Topology discovery		Link quality estimation		Reliability of the control/data planes			Continuous reconfiguration
		mode	mechanism	mode	metric	data/control separation	collision-free control plane	data flow isolation	
No scheduling	SD6WSN [117]	passive	RPL DIO	passive	ETX	no	no	no	no
	SDN-WISE [111]	active	extra broadcast probing	active	RSSI	no	no	no	yes
	TinySDN [112]	active	extra broadcast probing	active	ETX	no	no	no	yes
	IT-SDN [115]	active	extra broadcast probing	active	ETX	no	no	no	no
Scheduling	SDNWISE-TSCH [131]	active	extra broadcast probing	active	RSSI	no	no	no	no
	uSDN [122]	passive	RPL DIO	passive	ETX	yes	no	no	no
	Whisper [124]	passive	RPL DIO	passive	ETX	no	no	no	yes
	REACT [129]	passive	existing broadcast probing	passive	PRR	no	no	no	yes

2.3.3 Topology Discovery in Wireless SDN Networks

Topology discovery in wireless networks systematically identifies neighboring nodes and communication paths among nodes, forming a crucial foundation for network planning, optimization, and management [133, 134]. Typically, topology discovery can be achieved through active or passive schemes. In the active topology discovery approach, each node generates additional probing packets to announce its presence to neighbors and potentially receive responses in return. This method allows nodes to actively probe and identify connections with their neighboring nodes. However, this approach could result in elevated overhead due to the extra probing traffic, potentially affecting the overall energy consumption of the network.

On the contrary, the passive approach leverages regular network traffic that already exists. Nodes continuously monitor incoming data, analyzing the source nodes to identify their neighbors. This method avoids the overhead of generating probing packets but relies on the assumption that there is consistent traffic to gather sufficient information. However, it may not be as immediate in updating the neighbor list compared to the active approach.

Unlike the discovery process in wired networks, wireless SDN networks employ additional techniques like wireless signal analysis, neighbor discovery, and access point monitoring due to the dynamic nature of wireless links. Wireless topology discovery faces challenges stemming from mobility, signal interference, and the dynamic addition/removal of wireless nodes. On the other hand, wired networks offer consistent and predictable connections, facilitating simpler topology mapping through standard protocols.

SDN-WISE [111] and SDNWISE-TSCH [131] exploit extra beacon packets for topology discovery. A broadcast packet is sufficient to be detected possibly by all the neighbors, making this solution energy efficient. Nodes periodically send beacon packets to announce their hop counts to the sink node. Upon receiving a beacon, each node adds the sender of the beacon to its neighbor table and selects the neighbor with the minimum hop count towards the sink node. Each node periodically sends its neighbor table to the controller through the next hop. In TinySDN [112], each node sends a probing packet to its neighbors. Several solutions [117, 122, 124] exploit the RPL neighbor list instead of performing topology discovery. However, it is proved that RPL has inconsistencies in its routing table [135].

2.3.4 Link Quality Estimation in Wireless SDN Networks

Link Quality Estimation (LQE) in wireless networks is the technical process of evaluating the reliability and performance of communication links between wireless devices. It involves assessing various metrics defined based on the received signal

or data packet characteristics. LQE is crucial for optimizing network performance, efficient routing, and QoS provisioning. LQE facilitates adaptive behavior by enabling devices to dynamically adjust the allocation of radio resources based on the assessed link quality. Additionally, it informs decisions regarding routing paths and the prioritization of QoS.

2.3.4.1 Metrics

Link quality metrics are used to quantify the performance and reliability of wireless communication links in a network. These metrics provide critical insights into the integrity of the wireless channel, guiding decisions related to data transmission, routing, and network optimization. Link quality metrics have been broadly investigated in the literature and can be categorized to hardware-based and software-based metrics [136].

Hardware-Based metrics: these metrics are directly obtained from the radio transceiver, eliminating the need for any additional computational processes.

- ***LQI (Link Quality Indicator):*** LQI is a value calculated by hardware in wireless device systems. It may be implemented by means of receiver energy detection, signal to-noise ratio estimation, or a combination of these methods [137]. LQI values range from 0 to a maximum value (often 255) and indicate the assessed quality of the received signal. Higher LQI values typically correspond to better link quality. However, LQI's accuracy diminishes for intermediate link qualities, and its readings might vary across different hardware implementations [138].

- ***RSSI (Received Signal Strength Indicator):*** RSSI is a simple metric that measures the strength of the received signal. It indicates the power level of the signal at the receiver. A higher RSSI value generally suggests a stronger signal, which often correlates with better link quality. However, RSSI alone does not provide a comprehensive picture of link quality since it does not account for noise or interference [139]. Also, RSSI can be affected by factors like distance and obstacles [140].

- ***SNR (Signal-to-Noise Ratio):*** SNR represents the ratio of measured power between the received signal and the background noise level. A higher SNR value indicates a higher quality signal, as the signal is stronger compared to the noise. SNR can be a reliable indicator of link quality for both high-quality and low-quality links. However, it might not be as accurate for intermediate link qualities due to its sensitivity to hardware differences and environmental factors [141].

Hardware-based LQE have limitations. They measure metrics only for successfully received packets, potentially causing an overestimation of link quality in the presence of excessive packet losses. Additionally, while these metrics offer a quick and cost-effective way to classify links as good or bad, they lack the ability to provide detailed link quality assessments [142].

Software-based metrics provide a more precise insight into link behavior, which is essential for applications requiring specific reliability insights into network performance. Nevertheless, their advanced calculations might introduce processing overhead.

- ***PRR (Packet Reception Ratio):*** is calculated by measuring the ratio of successfully received packets to the total number of packets sent within a specific time frame or over a certain number of transmission attempts. Typically, a Window Mean with Exponentially Weighted Moving Average (WMEWMA) technique is employed to gain insights from historical PRR data, enabling the estimation of link quality over time [143]. WMEWMA calculates a weighted average of past PRR values, giving more weight to recent measurements. This provides a balance between responsiveness to changing link conditions and stability against fluctuations. Adjusting the time window size allows tuning the estimator's sensitivity.

- ***RNP (Required Number of Packet transmissions):*** metric quantifies the reliability of a link by measuring how many times a packet needs to be transmitted before it is successfully received. A higher RNP indicates lower link quality and reliability. However, RNP exhibits the drawback of being highly unstable and unable to provide a dependable estimation of link packet delivery, primarily due to the presence of link asymmetry [144].

  As one of the techniques aiming to approximate the RNP, the Expected Transmission Count (ETX) [145] actively monitors link quality. The process entails calculating the inverse of the product of forward (Packet Reception Ratio) and backward delivery ratios (Acknowledgment Reception Ratio), while accounting for link asymmetry. This renders the ETX metric highly effective for optimizing high-throughput routing in multi-hop wireless networks. Passive monitoring-based ETX encounters challenges in congested networks, causing interruptions in routing due to restricted packet reception and leading to compromised network throughput.

  Four-bit [146] is a versatile metric serving as a link quality estimator and aiding routing protocols. It compiles information from different layers, swiftly identifying high-quality links through its white bit, signaling acknowledgment reception via the ack bit, and employing the pin and compare bits for net-

work layer tasks. Four-bit leverages a fusion of RNP and metrics derived from WMEWMA to estimate link quality, serving both sender and receiver sides, accounting for link asymmetry, and utilizing both passive and active monitoring approaches.

- ***Score-Based:*** offers assessments that go beyond direct references to physical phenomena like packet reception or retransmission. Instead, these LQEs provide scores or labels within a specific range, allowing for more abstract and flexible link quality evaluations.

  MetricMap [147] employs a classification approach to assess link quality. It takes multiple metrics (e.g., RSSI, channel load, node depth) as inputs and uses a classification algorithm to categorize links into different classes, such as "Good," "Fair," or "Poor." The classification is usually performed during a training phase, where the algorithm learns the relationships between metrics and link quality classes. This approach provides more detailed insights into link quality but may require significant computational resources.

  F-LQE (Fuzzy Link Quality Estimator) [148] combines four link properties (Smoothed Packet Reception Ratio (SPRR), link stability factor (SF), link asymmetry level (ASL), and channel average signal-to-noise ratio (ASNR)) using fuzzy logic. Each property is assigned linguistic terms like "Very Low," "Low," "Medium," "High," and "Very High." Fuzzy logic rules then combine these terms to generate a score representing the overall link quality.

Choosing a link quality metric in wireless networks should align precisely with the application's goals. This involves selecting a metric that suits the desired outcomes, like reliable data transmission, energy efficiency, low latency, or scalability.

2.3.4.2 Measurement

Link quality measurement has been broadly investigated in the literature [141] with active versus passive techniques. Active techniques involve sending extra packets, which requires radio resources but can lead to energy wastage in scheduled networks. Passive techniques estimate link quality using existing packets, yet unbalanced traffic and collisions might reduce accuracy.

The SDN controller needs precise quality measurements for each link. Each node estimates the link quality with its neighbors locally, collecting relevant metrics, and subsequently reports these measurements to the SDN controller. The controller possesses the flexibility to either directly employ the individual metrics or aggregate them to formulate more sophisticated metrics. For instance, it could assimilate unidirectional PRRs of a link to calculate a bidirectional PRR within the network's

graph representation. In this context, the LQE value assumes the role of the edge value in the constructed graph.

In [111, 131], the controller exploits the RSSI value of the last beacon reported by each device. However, a high RSSI does not directly lead to a high packet reception ratio [149]. TinySDN [112] uses probing packets. Each node transmits a broadcast probing packet and waits for the unicast reply of its neighbors. In this way, a node calculates the ETX toward each of its neighbors. The solution presented in [117] relies on the routing protocol (RPL) and more specifically on the ETX metric maintained by RPL. Each node periodically sends unicast probing packets to neighbors and calculates the corresponding ETX. However, sending probing packet increases the communication overhead. Also, in overloaded networks, many nodes fail to compute the ETX because they cannot receive packets.

Different link quality schemes and metrics may serve well for routing decisions in wireless networks, where a comparative assessment suffices to differentiate between routes. However, precise PDR values become imperative for the controller to make optimal scheduling decisions and efficiently manage radio resources. By considering the PDR, the controller can allocate radio resources, such as time-frequency blocks, based on the required QoS for each flow [83].

2.4 Summary

Scheduled standard technologies like IEEE 802.15.4-TSCH are designed for industrial applications to provide reliable link layer communications. However, enabling a flow guarantee necessitates a centralized scheduling system, as distributed solutions alone fail to achieve this goal due to the lack of collision free schedule.

SDN offers a promising approach for implementing centralized management systems. Upon reviewing the literature, many solutions have been suggested to integrate SDN into wireless networks. However, many of these solutions have focused solely on efficient network layer forwarding rule definition. Only a limited number of them have extended their scope to include centralized scheduling within scheduled wireless networks. Despite the variety of objectives pursued by these solutions, such as reducing control overhead or relying on non-realistic network conditions, none of them have been able to ensure flow guarantees in IWSN, establishing stable SLAs.

Given the inherent lossiness of wireless links, the robustness of the control plane remains relatively underexplored in existing literature. Control plane operations typically rely on shared medium access, which can be susceptible to the high collisions, particularly under high loads of control traffic.

To meet SLA requirements like reliability and latency, the controller must schedule tasks according to real-time wireless link conditions. Existing SDN solutions lack

scheduling with LQE consideration, so accurate LQE information, like PDR per link, is vital to prevent resource allocation errors. As link quality changes, the controller continually updates forwarding and scheduling to maintain network performance.

Our work extensively investigates these areas to enable stable flow guarantees in IWSNs. We compare our solution with the MSF IETF standard solution as it proposes an adaptive distributed scheduler for managing traffic load and collisions. Additionally, we compare our solution with SDNWISE-TSCH as the most advanced and relative solution to illustrate per-flow scheduling in centralized scheduling and the reliability of the control plane.

Chapter 3

SDN-TSCH: Enabling SDN for IWSN with Traffic Isolation

Contents

Industry 4.0 employs IWSN for automating industrial processes [150], with critical applications in each device demanding strong QoS in terms of high reliability and bounded latency. As wireless links can be lossy, the communication network must uphold these SLAs. IEEE 802.15.4-TSCH [26] stands out for industrial networks, utilizing a TDMA-based MAC layer and frequency hopping to ensure robust communication. To meet application-defined SLAs, enough time-frequency blocks must be allocated to each flow.

SDN represents a promising solution to enable centralized scheduling in IWSNs. However, in wireless networks, the controller has a broader role and needs to i) maintain the network synchronized to exploit the FTDMA matrix without collision, ii) support unreliable links in the control and data planes to avoid communication bottleneck, iii) admit data flows and install forwarding rules. SDNWISE-TSCH [131]

represents a pioneering piece of work to support a centralized controller. However, it still suffers from collisions, as demonstrated in Section 6.2, resulting in packets that exceed QoS limits which are unacceptable for most industrial applications.

We assume that each application possesses one or more critical flows that require specific end-to-end PDR and deadline parameters. By communicating their QoS requirements to the SDN controller, it calculates efficient forwarding rules and schedules resources for each flow, proceeding to install them in the network. While the convergecast traffic pattern is the most commonly used case in IWSN, involving the transmission of sensory data to a central control unit, we also consider the possibility of peer-to-peer communication in our deployment. We assume the presence of a controller located outside the wireless network but in close proximity to manage the network efficiently. However, before the configuration of the data plane, we need a reliable control plane to secure the communication between the SDN controller and wireless devices over lossy and multi-hop wireless links.

In this chapter, we propose a solution called SDN-TSCH to implement an SDN architecture in IWSN, aiming to address the specified requirements.

3.1 SDN-TSCH Overview

SDN-TSCH enables SLAs in IWSNs. SDN-TSCH operates on top of IEEE 802.15.4-TSCH scheduled MAC layer, separating clearly the control and the data planes with dedicated radio resources. The SDN controller assigns radio resources to maintain the control plane such that any device has a collision-free path from and to the controller. Furthermore, the controller can allocate radio resources in the data plane to ensure compliance with per-flow reliability and latency requirements. Packet forwarding employs a label-switching approach to isolate per-flow resources, enabling nodes to efficiently select suitable radio resources for forwarding using a simple table.

Through an efficient discovery process (Figure 3.1), new nodes identify their neighbors by listing them and accurately estimating their link qualities. Following that, a joining process begins, allowing new nodes to notify their presence to the SDN controller through an already configured and reliable control plane. The controller then proceeds to configure the control plane for the new nodes, allocating dedicated radio resources. Once the control plane is configured, each node applies for admission of its critical data flow by submitting its desired QoS requirements. If the necessary resources are available, the controller orchestrates an end-to-end configuration for the flow.

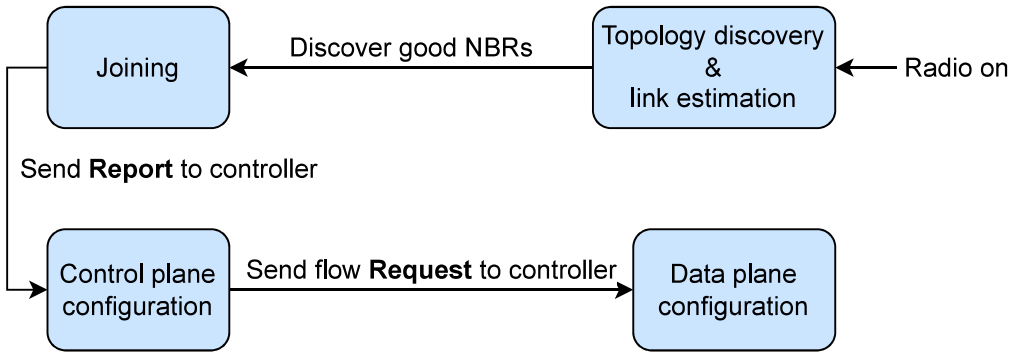

Figure 3.1: The configuration process of a new node in SDN-TSCH, (NBR = neighbor)

3.2 Label Switching for SDN

SDN-TSCH exploits a label switching technique to simplify and to speed up the packet forwarding and more importantly enable flow isolation in the network [151]: each flow has its own distinct radio resources that are exclusively used for transmitting its own packets, without being shared with other flows in the network. In fact, the SDN controller defines a global flow-id for each flow and accordingly populates the flow tables of all the nodes along the path to the destination. The flow-id is a unique identifier that enables devices to differentiate and manage flows effectively. More precisely, each Transmission (Tx) cell, associated with a particular flow in the slotframe, is tagged with a corresponding flow-id. At the beginning of a Tx cell, a node picks the first packet from the queue associated with the matching flow-id for transmission. This mechanism provides flow isolation for data traffic because two flows can not compete for the same cell if they are assigned to different flow-ids. In Figure 3.2, while cells 3 and 5 are scheduled for transmission between $A \rightarrow S$, they are exclusively used for forwarding the packets associated with their respective flow-ids.

We piggyback the flow-id, represented as a two-byte integer, in the packet header as an IE. When receiving a packet, a device extracts the flow-id from the header and looks up the flow table to find the matching rule to forward the packet. If there is forwarding action, the packet is stored in the outbound queue corresponding to the respective flow-id.

Particularly, two flow-ids are defined for the control plane:

"to-controller" for upward control packets generated by nodes;

"from-controller" for downward packets generated by a controller;

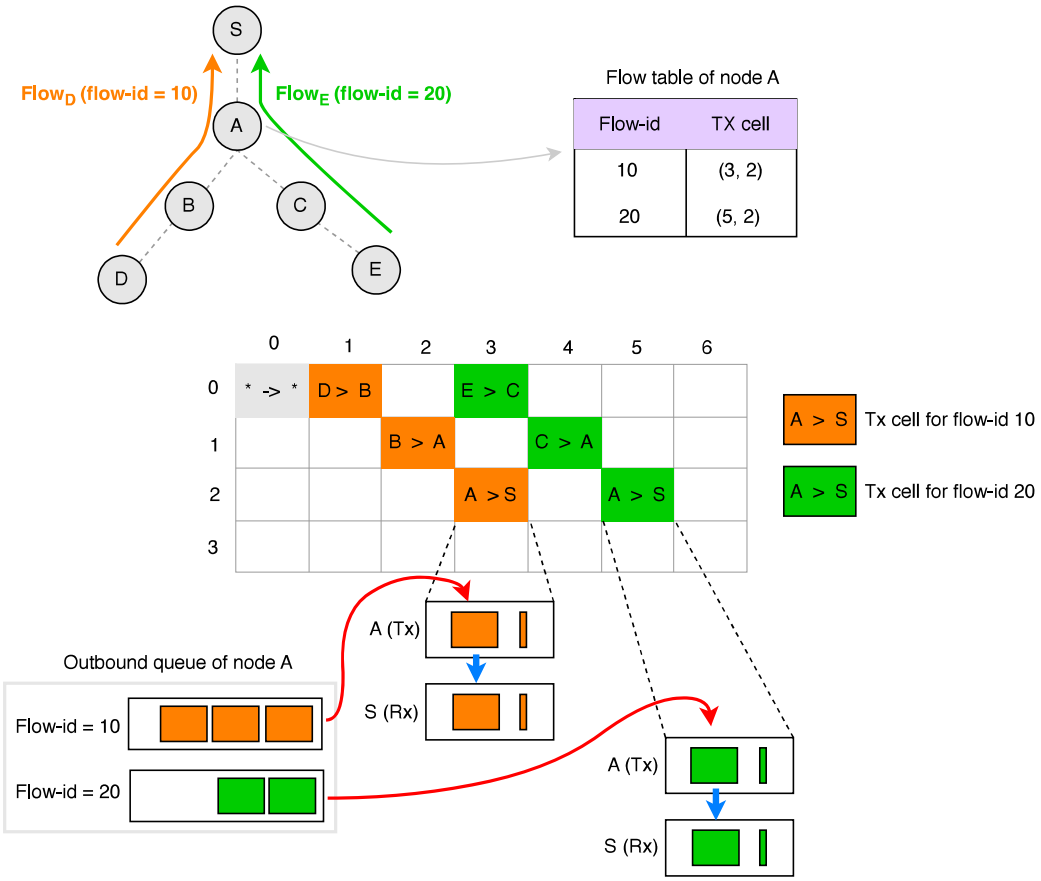

Figure 3.2: Label switching operation in SDN-TSCH

3.3 Slotframe and Schedule Organization

We isolate the radio resources for the control and data planes. More precisely, the slotframe (*e.g.*, Fig. 3.3) regroups:

shared cells are used for broadcast traffic *e.g.*, EB transmissions and initial joining request packets. They are installed as soon as a device is synchronized with the network, using the information piggybacked in the EBs.

dedicated control cells are used for the control plane: one cell (in green) toward the parent to reach the controller (`report` packets), and one cell (in yellow) toward the children (`config` packets). It is worth noting that no collision can arise: one single transmitter is active at a time;

dedicated data cells are assigned by the SDN controller for each data flow. Forwarding rules guarantee flow isolation. For the sake of conciseness, the data cells were not depicted in Fig. 3.3.

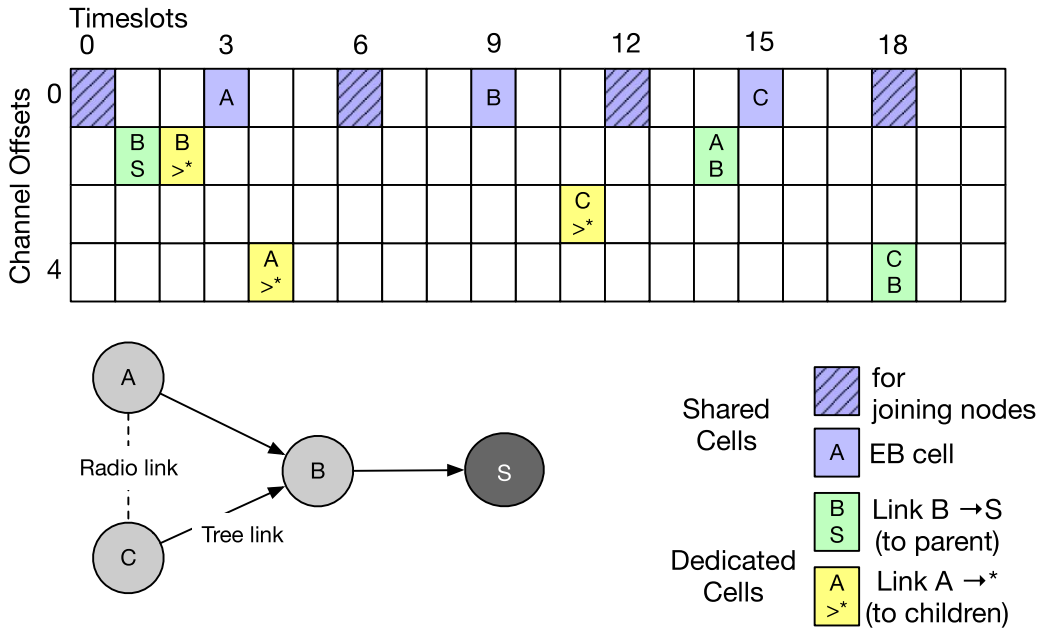

Figure 3.3: Organization of the slotframe

3.4 Discovery Process

The discovery process enables a new node to identify its neighbors and obtain an estimation of their link qualities. This process subsequently facilitates an efficient configuration of the control plane for the new node by the controller, which associates the new node with its optimal neighbor.

The estimation of link quality is accomplished through passive means, involving the reception of EBs from neighbors during specific shared cells. Our approach leverages a collision-free EB schedule, allowing us to determine the PDR of each link by merely counting the EBs received from individual neighbor within a designated time period.

Let us focus on the discovery process: a collection of nodes has already joined the SDN-TSCH network, and the new node turns its radio on to join the network. By engaging in continuous channel scanning, the node receives an EB and then proceeds to synchronize its clock with the TSCH network. This synchronization is primarily achieved by utilizing the source of the EB as a time synchronization reference. The joining node also extracts the IEs from the EBs to acquire the TSCH parameters (such as slotframe length, number of shared cells, etc.) and installs the shared cells into its slotframe.

Then, by listening on the shared cells, the joining node discovers its neighbors (senders of EBs) and computes the link PDR. If a neighbor presents a good link

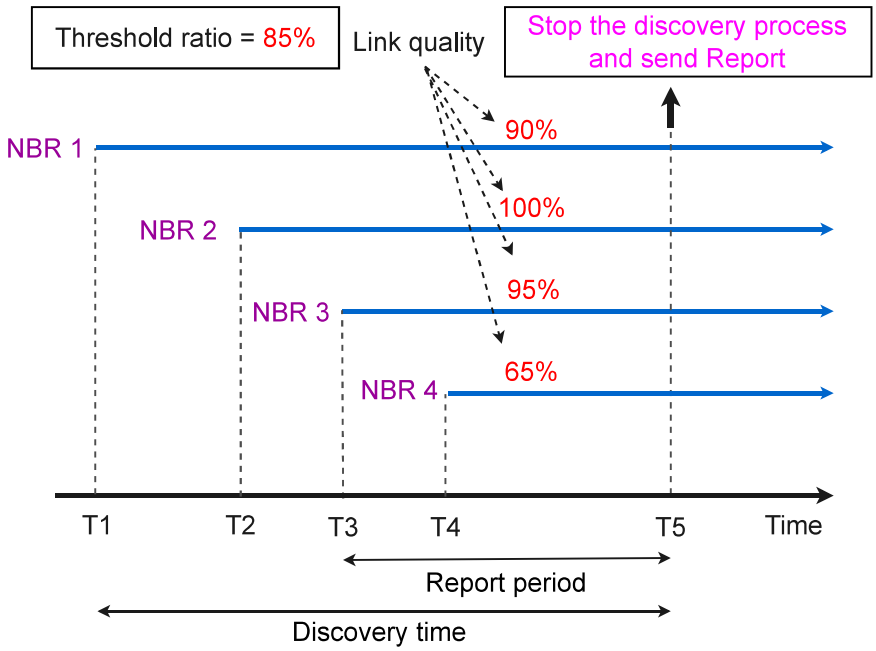

Figure 3.4: Discovery time of a joining node

quality, it should be validated for a designated time window to ensure its stability. Thus, the discovery process stops when all good neighbors (neighbors for which the link quality is above a predefined threshold) have been seen for at least one T_{report} period. More precisely, the process stops if Eq. 3.1 is satisfied:

$$\forall n \in \mathcal{N} | \widehat{PDR}(n) \geq PDR_{min}, t_0(n) + T_{report} \geq t \tag{3.1}$$

with $t_0(n)$ the receiving time of the first EB from n, $\mathcal{N}$ the set of neighbors, $\widehat{PDR}(n)$ the measured Packet Delivery Ratio (PDR) for n, and t the current time.

As illustrated in Figure 3.4, the joining node stops the discovery process only when the link quality of all good neighbors ($NBR1$, $NBR2$, and $NBR3$) is measured for at least one report period. Namely, the process is not stopped at $T4$ because $NBR2$ and $NBR3$ are presenting good link quality, while they have not been measured for a sufficient amount of time. This guarantees a stable margin in the measured link quality. The joining node does not consider the discovery time of $NBR4$ as it proposes a PDR lower than the threshold ratio.

It is worth noting that during the discovery period, the joining node consistently adjusts its time source neighbor to the node presenting the best link quality. This minimizes the probability of desynchronization.

The SDN controller needs real-time updates on the evolution of link qualities, and hence, computed PDRs are continuously transmitted to the controller every

T_{report} once the discovery process is completed.

3.5 Joining Process

When a joining node has performed the discovery process, it needs to join the SDN network. For this purpose, it creates a `report` packet including the list of neighbors and their associated PDR. Then, it sends the packet in unicast to the neighbor with the highest link quality to maximize the transmission success probability. Since the joining node has no configured control plane, it must use a shared cell to transmit its `report` packet. Then, the `report` packet traverses hop-by-hop the network to reach the controller in a reliable manner using the ***"to-controller"*** flow-id.

When receiving a report packet, the controller verifies if the report packet comes from a new node. In that case, the controller registers the node in the joined list and selects the neighbor with the best link quality as the parent to maximize the reliability of the control plane. We also employ the parent as the time source in TSCH after the joining process. This approach minimizes the probability of desynchronization due to the strong link quality.

Then, the controller selects randomly two dedicated cells in the slotframe for the control plane: one for sending upward control packets to the parent node and one for receiving downward control packets from the parent node. Obviously, allocated cells must correspond to an unused timeslot for the parent (half-duplex condition) and cannot be allocated to another interfering link (collision-free condition).

Finally, the controller constructs two `config` packets: one for the upward direction (***to-controller***) and one for the downward direction (***from-controller***). To reach the joining node, the controller can use the flow-id ***from-controller*** already configured in the rest of the network (except the last hop). However, several children may exist at each hop, and the `config` packets should be routed in the correct subtree. Hopefully, the controller knows the complete topology and can compute a path to the joining node. Thus, we implement source-routing: the whole path is piggybacked in each `config` packet.

It is worth noting that we always configure a flow from the destination toward the source. This approach allows for a generic scheme that enables the configuration of data flows in a reliable manner (see Section 6.2).

When a node n receives the config packet, it exploits the route and the schedule to update its configuration:

1. it extracts the flow-id and looks for its position in the route. n searches for a pair of nodes in the route that corresponds to the address of the transmitter of the `config` packet, and its own address. We denote by i the position of

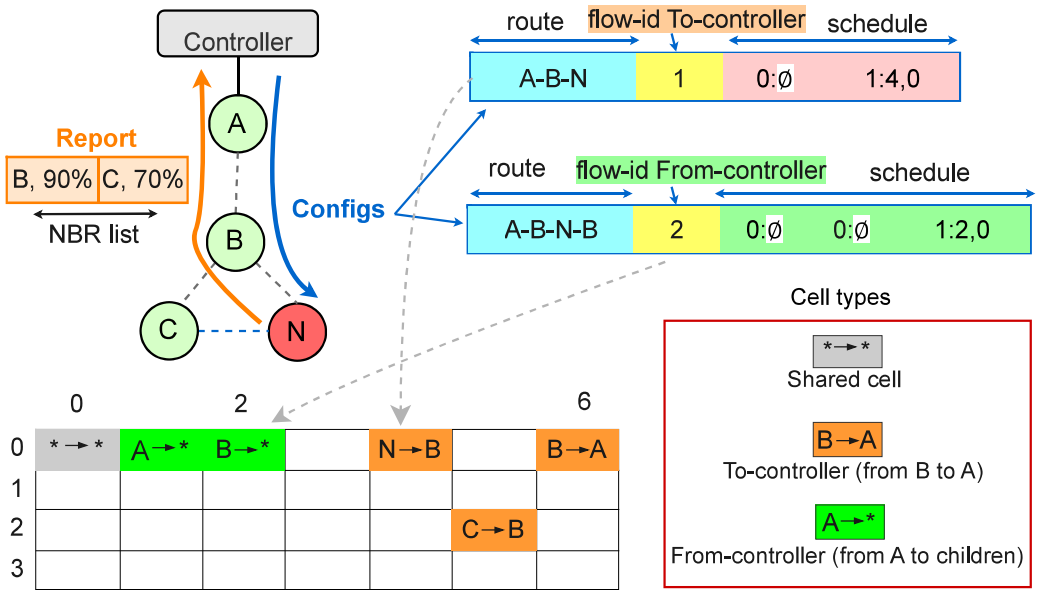

Figure 3.5: Dedicated control plane in SDN-TSCH

its address in the route ($i \in [0, k[$, k being the number of nodes in the source routing path);

2. if $i > 0$, it installs the $(i-1)^{th}$ element of the schedule as the TX cells in its scheduling table. As the node with the position $i = 0$ can either have RX cells to install or nothing in the case of a relay node;

3. if $i < k - 1$, it installs the i^{th} element of the schedule as RX cells in its scheduling table. As the node with the position $i = k - 1$ is the last node in the route and the source of the flow, it only needs to install TX cells;

4. n forwards the `config` packet to the next hop following the source routing path. If the `config` packet is for configuring the ***"to-controller"*** flow-id, nodes use the ***"from-controller"*** flow-id if $i < k - 2$, or the next available shared cell if $i = k - 2$. If the `config` packet is configuring the ***"from-controller"*** flow-id, nodes use the ***"from-controller"*** flow-id if $i < k - 3$, or the next available shared cell if $i = k - 3$, or the ***"to-controller"*** flow-id if $i = k - 2$. Indeed, the ***"to-controller"*** flow-id is installed first.

Figure 3.5 illustrates this process. Node N needs to join the network and sends a report packet to the controller through the neighbor B. Upon reception, the controller selects node B as the parent of node N since it is the neighbor with the highest link quality. It is worth noting that the controller can consider another metric (*e.g.*, load balancing) to select a parent, which is not the focus of our research.

The controller creates two `config` packets including the addresses for source routing and the schedule. More specifically, we encode the schedule of each hop as <`number_of_cells:list_of_cells`>. Notably, the first forwarding nodes (node A in Figure 3.5) do not have any cell to install (`number_of_cells` $= 0$).

3.6 Resource Allocation for the Data Plane

Any critical application opens a socket connection, describing its QoS requirements (*i.e.*, end-to-end minimum reliability and maximum latency). The node engages a call admission by sending a `flow-request` to the controller through the ***"to-controller"*** flow-id. When receiving a `flow-request`, the controller constructs a schedule that respects the QoS requirements:

1. the controller computes a path from the source to the destination using the tree topology (through the node/parent links);
2. the controller allocates backup dedicated cells for retransmissions for weakest links. Additional backup cells are assigned until the minimum end-to-end reliability is respected [152];
3. the controller schedules cells back to back (or as close as possible) to minimize the end-to-end delay.

If the schedule computation is impossible, the request is rejected, and the controller sends a negative (empty) `config` packet to the source node. If the computation is successful, the controller defines a new flow-id for the flow and constructs a `config` packet including the new schedule and the corresponding flow-id. The controller uses a single `config` packet to configure the whole path, which is composed of:

1. subpath from the root to the destination: the schedule of this part is empty, and the node only forwards the `config` packet with the ***from-controller*** cells;
2. subpath from destination to source: the configuration starts from the destination node toward the source node. Each node in this part extracts and installs the flow-id and the corresponding cells. When the source node receives the `config` packet, the whole path is configured, and it starts sending packets without delay.

Notably, the technique to identify the position of a forwarding node (cf. Section 3.5) is still valid when a node is present in both subpaths. Indeed, we individually identify each **link** in the whole path when installing the schedule.

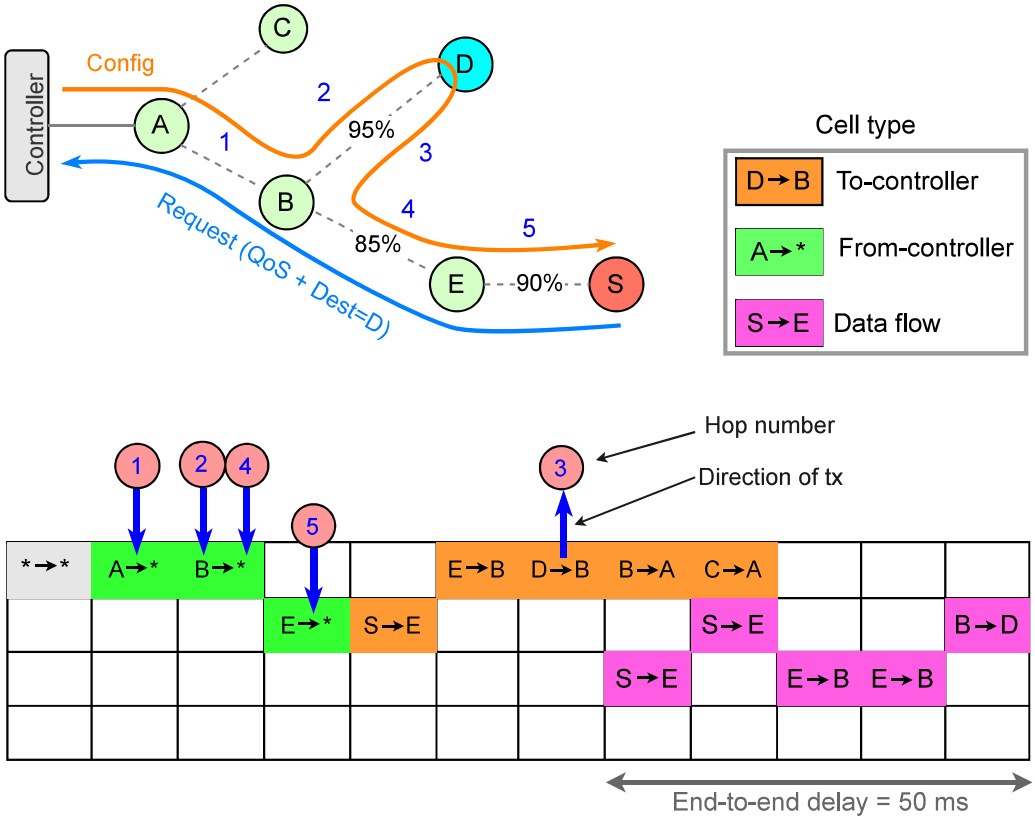

Figure 3.6: Data flow Configuration in SDN-TSCH

For the second part of the path, each node has to select either the ***"to-controller"*** or ***"from-controller"*** flow-ids to forward the `config` packet. To determine the direction, each node checks if the address of the next hop is also present in the list of nodes preceding its position in the source routing. If this is the case, the next hop is an upper-hand node, and the `config` packet uses the ***"to-controller"*** flow-id. Otherwise, the node uses the ***"from-controller"*** flow-id to forward the `config` packet.

Figure 3.6 illustrates a scenario where S is the source node. The `flow-request` packet describes the flow requirements (*e.g.*, PDR = 90%, end-to-end delay = 70 ms) and the destination D. In return, the `config` packet uses source routing, first from A to D and then from D to S. The controller allocates more cells to weak links (2 cells for S→E and 2 cells for E→B) to meet the end-to-end PDR of flow (90%). The controller allocates cells sequentially to consider the end-to-end delay of flow (70 ms). The route $A - B - D - B - E - S$ is piggybacked in the `config` packet. Thus, D knows B is an upper-hand node since B is both present after and before D: it has to use the ***"to-controller"*** flow-id to forward the `config` packet.

Table 3.1: Simulation parameters

Simulation environment	OS: Contiki-ng (version 4.7) Simulator: Cooja https://github.com/Farzadv/Contiki-ng-SDN-TSCH.git Simulation time: 2.2h Propagation model: Unit Disk Graph Medium (UDGM)[2] Tx range = 100 m Interference range = 150 m Rx success = proportional to distance (100% - 0%) Initial energy = 2400 mAH (AAA battery) Energy consumption model = Energest tool in Contiki-ng (tx_curr = 17.7mA, rx_curr = 20mA)
Topology	Distribution: Random Network sizes: 10, 20, 30, 40, and 50 nodes
SLAs	Number of data flow: 1 flow per node Traffic pattern: Convergecast Traffic rate: Constant Bitrate, 1 packet every 5s
Applications	Requested PDR: 99% Requested deadline: 2s
SDN-TSCH	TSCH EB period: 15s SDN report period: 5 min `flow-request` timeout: 50s

Inversely, S is a downward node for E since S is only present after E: it has to use the ***"from-controller"*** flow-id.

It is important to note that our primary focus is not on the scheduling algorithm itself, as we currently use a greedy approach. However, our architecture allows for the integration of any centralized algorithm if desired.

3.7 Performance Evaluation

We implement SDN-TSCH in Contiki-ng and the Cooja simulator to assess its performance. We compared SDN-TSCH with two state-of-the-art approaches:

SDNWISE-TSCH [131] is a variant of SDNWISE to cope with TSCH networks. SDNWISE-TSCH enables the SDN architecture for Industrial Wireless Sensor Networks. It computes a schedule taking into account the deadline of each flow.

MSF [91] is the IETF standard for distributed scheduling[1], combining autonomous and negotiated cells to avoid collisions and allocate resources dynamically;

[1] `https://github.com/alexrayne/contiki-ng.git`

We simulate networks with sizes of 10, 20, 30, 40, and 50 nodes. Each node has a critical data flow to transmit to the sink node (*i.e.*, convergecast traffic). We consider critical applications that require an end-to-end PDR larger than 99% and with a deadline of 2 seconds [153]. We use the default values for the parameters of MSF [91]. A controller allocates a novel flow-id and a set of cells to each critical flow to meet the requirements. Table 3.1 represents our different parameters.

3.7.1 Comparison of SDN-TSCH and SDNWISE-TSCH

We compare SDN-TSCH with SDNWISE-TSCH to compare two different SDN-based approaches. More precisely, we i) evaluate the convergence time and energy efficiency of the control plane, and ii) assess the performance of the flow guarantees in the data plane.

SDNWISE-TSCH employs shared cells for both EBs and SDN control packets. According to [131], a network of 10 nodes and slotframe size of 19 uses 2 shared cells. To accommodate different network sizes without impacting the reliability of the control plane of SDNWISE-TSCH, we propose to maintain constant the time per node between two shared cells. Thus, we define the default number of shared cells as follows:

$$N_{shared-cells} = \frac{2 * SF_{length}}{19} * \frac{N}{10} \approx N * SF_{length} * 0.01 \tag{3.2}$$

with $N_{shared-cells}$ the number of shared cells, N the number of nodes, and SF_{length} the slotframe length.

3.7.1.1 Convergence time and energy efficiency of the control plane

Figure 3.7 shows the convergence time of each approach for different network sizes. We define convergence as the duration from the network's bootstrap until the last node is admitted by the SDN controller. We test two ratios of shared cells in the shared control plane: i) $15N$ (eq. 3.2) used by SDNWISE-TSCH and ii) $2N$, which serves as a lower bound value. With 2N shared cells, the number of collisions becomes very high for medium-sized networks. In the worst conditions, the network never converges. With 15 shared cells per slotframe, SDNWISE-TSCH succeeds to converge. However, the convergence time is still larger for SDNWISE-TSCH compared with SDN-TSCH. Indeed, shared cells receive a peak of control traffic, and the loss of control packets significantly impacts the convergence.

Using shared cells also impacts energy consumption, as illustrated in Figure 3.8. We focus here on the power consumption of the network during the convergence

[2] https://github.com/contiki-ng/cooja/blob/06863708772e33504b3a397c225db9466082fcaf/java/org/contikios/cooja/radiomediums/UDGM.java

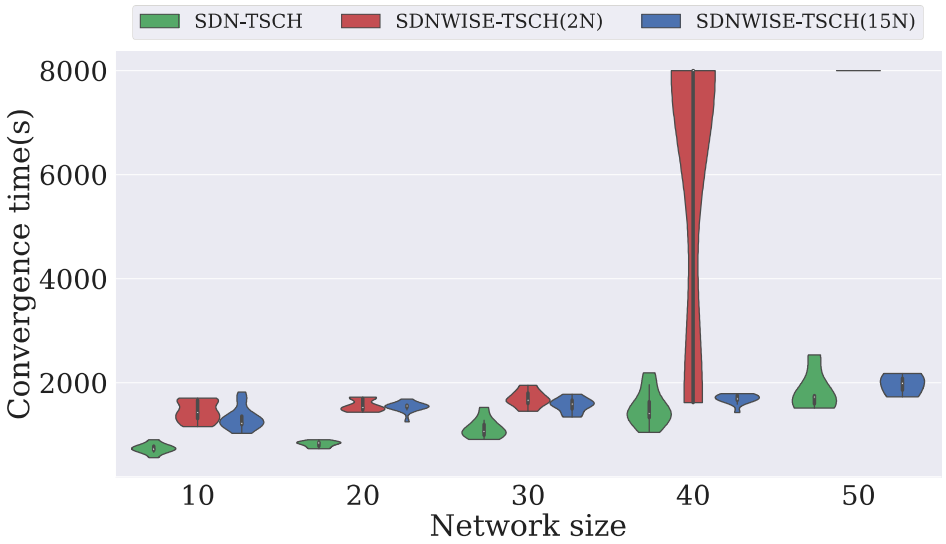

Figure 3.7: Network convergence time

period. This is calculated by dividing the energy consumed during the convergence time by the convergence time. Using only 15N shared cells consumes much energy: all the nodes wake up during these slots. On the contrary, using only 2N shared cells is much more energy efficient (but with an impact on reliability). Only SDN-TSCH is able to converge fast while providing a very reasonable energy consumption: using dedicated cells is much more efficient, even for control traffic. Indeed, the nodes have to wake up less frequently, and the transmissions are more reliable since we cannot create collisions.

3.7.1.2 Data flow handling

We focus now on the scenario illustrated in Figure 3.9 to assess the performance of the solution to identify good paths. Nodes A, B, C, and S host critical flows to the destination node D, and node S has two possible paths to reach it. Path 1 has an end-to-end link quality of 45% ($0.8 \times 0.7 \times 0.8 = 0.45$), while path 2 has an end-to-end link quality of 25% ($0.5 \times 0.5 = 0.25$).

To configure the flow of node S, the SDNWISE-TSCH controller selects path 2 based on the shortest path criteria. It only allocates one cell for each hop and schedules cells back to back to respect the flow deadline. However, most of the transmissions fail because of low link quality. Besides, buffering delays increase, and a few packets that are successfully transmitted arrive after the deadline. Moreover, due to the lack of flow isolation, node B uses any cell scheduled toward node D to send its packets. While it allows using cells unused by the other flow, it also impacts the reliability of the competing flows (from S). To sum up, 78% of packets

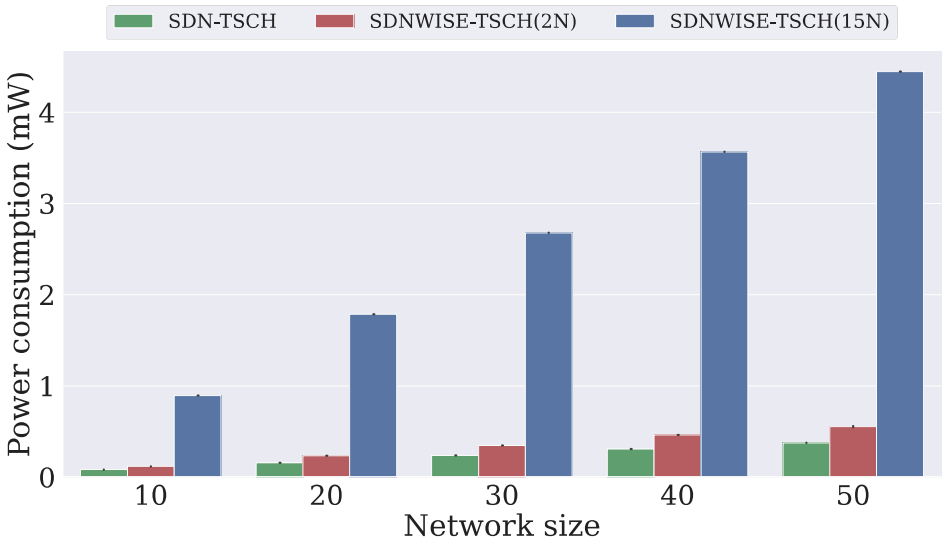

Figure 3.8: Power consumption of nodes in joining period

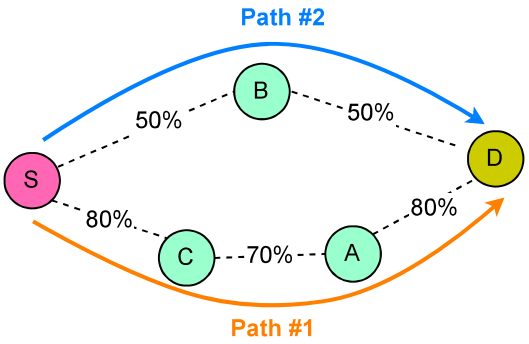

Figure 3.9: Data flow handling in SDN-TSCH and SDNWISE-TSCH

are lost along the path, and 22% of packets are received at the destination after the deadline.

Let us focus on the behavior of SDN-TSCH. The controller selects path 1 when it admits the flow of S: the path uses links with better quality, and fewer retransmissions would be required to reach the same level of reliability. Besides, the controller allocates a sufficient number of cells (including the backup cells): 100% of the packets of node S are received by the destination before the deadline. SDN-TSCH selects efficient data paths and allocates enough resources while guaranteeing flow isolation.

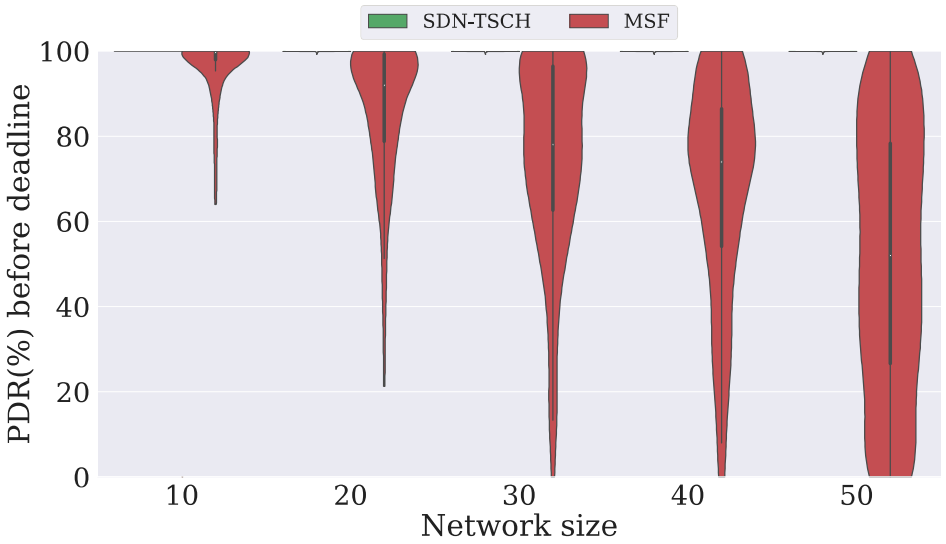

Figure 3.10: PDR of flows before deadline

3.7.2 Comparison of SDN-TSCH and MSF

We compare SDN-TSCH with MSF to assess the differences between a centralized solution and a distributed solution in terms of both reliability and energy efficiency, with a specific emphasis on data plane efficiency. Figure 3.10 illustrates the end-to-end PDR of each flow. Clearly, MSF provides very low reliability. With 20 nodes, the average end-to-end PDR is larger than 95%, but some flows exhibit a PDR of only 50%. The reliability is even worse with 50 nodes: more flows are forwarded, and the region around the sink becomes a bottleneck. Thus, MSF fails to provision enough backup cells to provide high reliability. On the contrary, SDN-TSCH achieves perfect end-to-end reliability, whatever the conditions. Even better: the reliability is equal to 100% even in the worst case, which is an expected property for industrial networks.

Figure 3.11 illustrates the end-to-end delay. We plot the 2s deadline (horizontal line) to see its impact. MSF tends to deliver packets very close to the deadline, but too many packets are received after the deadline, particularly for large networks. On the contrary, the SDN controller provisions enough resources in SDN-TSCH. While the latency increases because packets are forwarded through longer paths, the deadline is still respected. The packet losses correspond mostly to statistical outliers.

We compare the convergence of MSF and SDN-TSCH in Figure 3.12. We define the convergence time for a flow as the duration it takes for the flow's Packet Delivery Ratio (PDR) to reach the desired PDR. In MSF, increasing the network size leads to longer convergence times: each hop of the path needs to negotiate cells

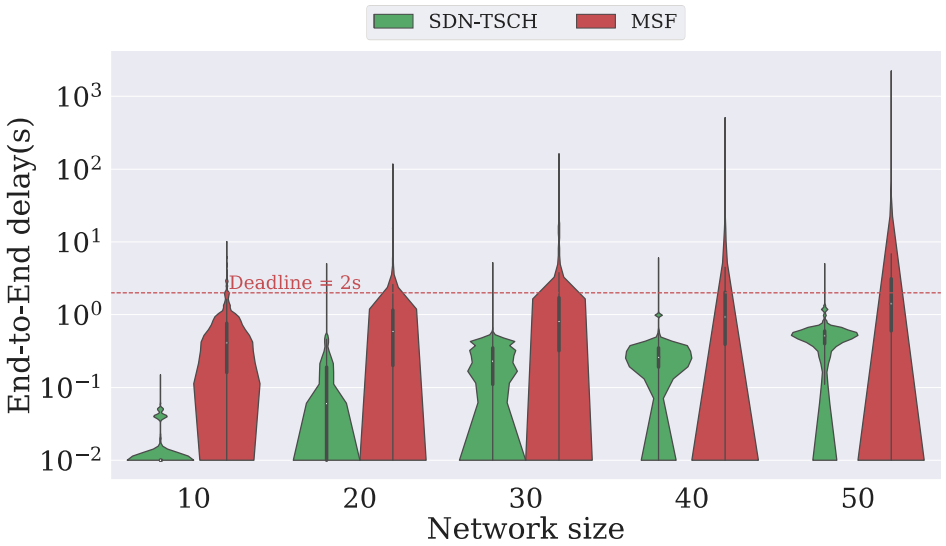

Figure 3.11: End-to-end delay

reactively. In addition, the arrival of new flows impacts the performance of the ongoing ones because MSF does not provide flow isolation. This observation is clearer on Figure 3.13, which illustrates the instantaneous PDR of a given flow while MSF converges. As we can see, the PDR value fluctuates significantly between 40% and 100% because a new flow is admitted in the network and competes for the same resources, or when a collision arises temporarily and the corresponding cell has to be relocated. As a result, relying on distributed algorithms, such as MSF, impacts convergence and PDR. In SDN-TSCH, the flow convergence time is effectively reduced to zero because the controller pre-configures the entire path for a flow. Thus, we have an efficient call admission scheme: the flow starts only when and if enough resources can be scheduled all along the path.

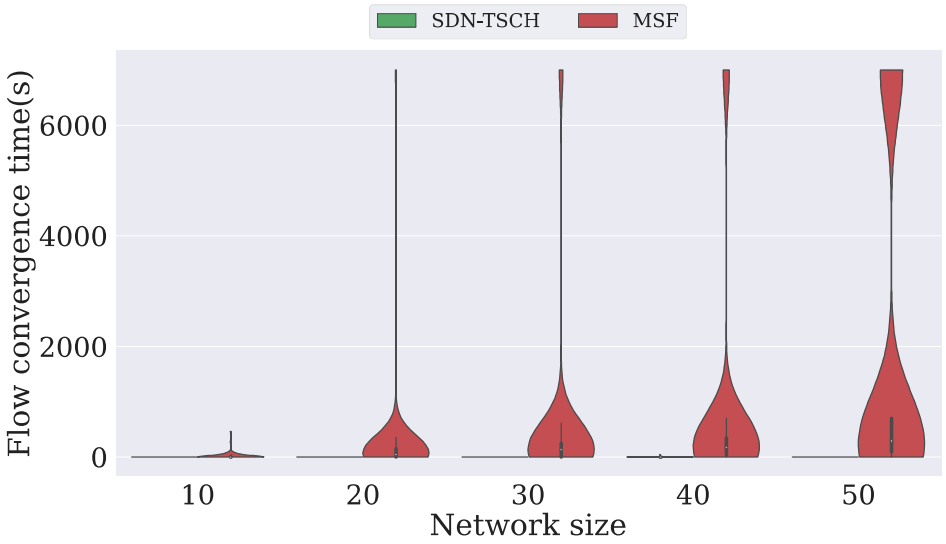

Figure 3.12: Data flow convergence time

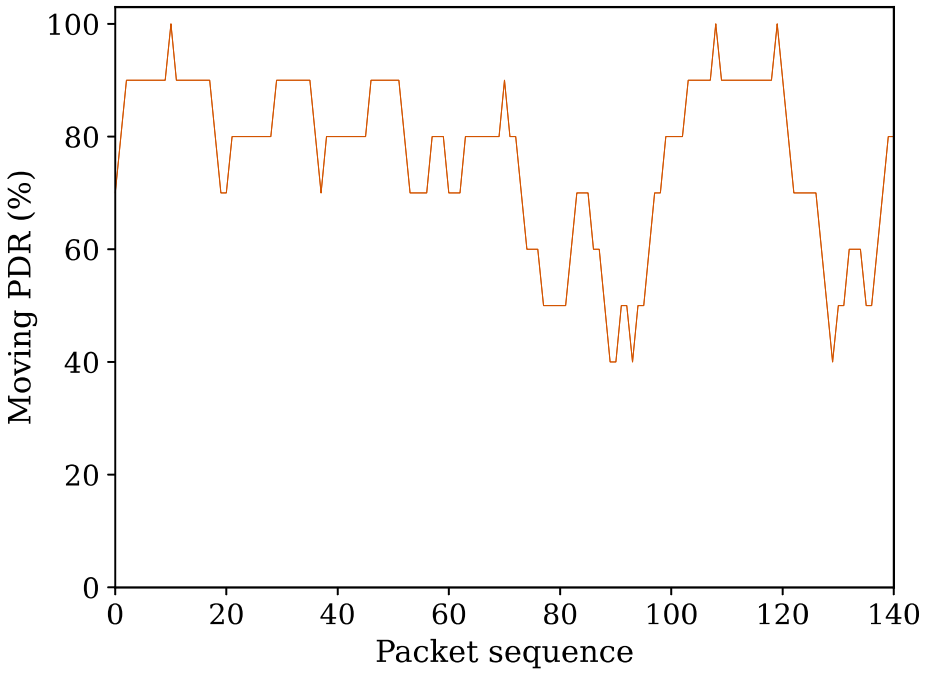

Figure 3.13: Instantaneous PDR of a given flow during a time interval using MSF

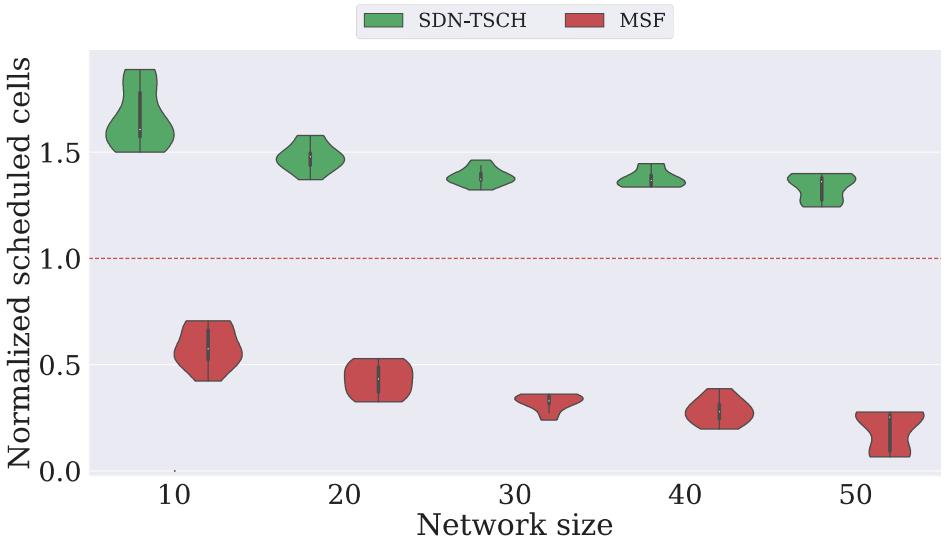

Figure 3.14: Ratio of scheduled cells

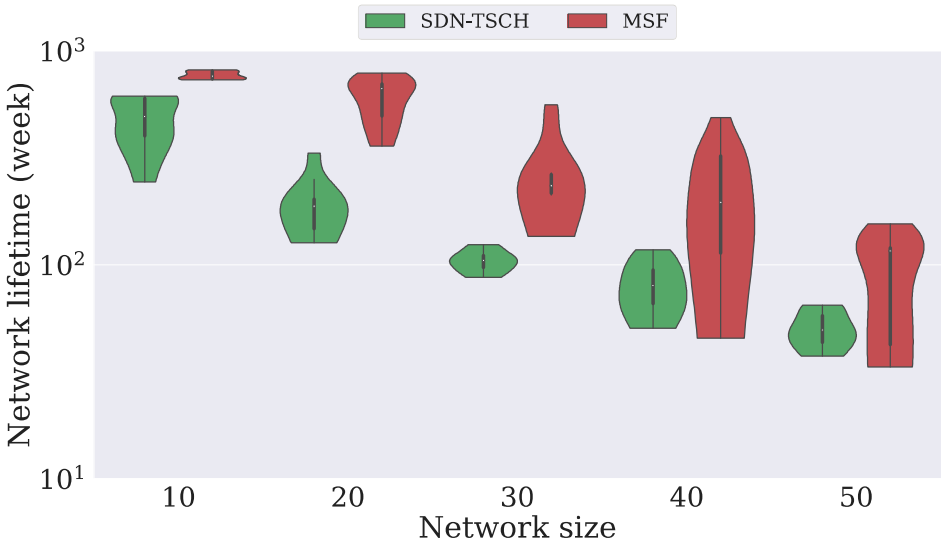

Figure 3.15: Network lifetime

Figure 3.14 focuses on resource allocation efficiency. We measured the ratio of the cells that are reserved for each link. To measure the efficiency, we normalize this amount of cells by the required number of cells computed directly from the parameters of the PHY layer. We can note that MSF is under-provisioning: the ratio is smaller than 1.0, which means that the number of cells is insufficient to provide the expected end-to-end reliability. In contrast, SDN-TSCH is based on an accurate link quality estimation and allocates cells according to this estimation. SDN-TSCH is over-provisioning, allocating more cells than the strict minimum. However, this safety margin compensates for the possible over-estimation of the link qualities, and SDN-TSCH finally provides the expected end-to-end reliability.

We finally measured the network lifetime (Figure 3.15), defined as the time until the first node dies due to depleted energy. MSF provides the longest network lifetime: since fewer cells are reserved, nodes have to wake up less frequently. Thus, they maximize their sleeping time, but at the cost of poor reliability. SDN-TSCH achieves a shorter but comparable lifetime and respects all the reliability and latency constraints. Obviously, ensuring strong Service Level Agreements has an impact on energy consumption. It corresponds to the price to pay for a communication infrastructure dedicated to critical applications.

3.8 Conclusion and Future Works

We demonstrated in our work with SDNWISE-TSCH that the utilization of shared cells significantly affects the reliability of control plane communication. Furthermore, we underscored the critical importance of incorporating link qualities into the scheduling definition. Additionally, we emphasized the necessity of clearly segregating radio resources designated for a specific flow from other flows, thereby minimizing inter-flow resource contention. Given that SDNWISE-TSCH relies on the shortest path approach and lacks per-flow resource isolation, it exhibits suboptimal performance on links with low PDR, ultimately impacting overall PDR and delay.

In this chapter, we introduced SDN-TSCH, an SDN-based solution tailored for IWSNs to address these challenges. Our approach leverages a label switching technique to enable effective flow isolation. The SDN controller establishes a collision-free control plane, allowing devices to reliably connect to the controller through lossy wireless links. Each data flow is admitted into the network. The controller maintains complete control over the network and allocates sufficient dedicated resources for critical flows. SDN-TSCH meets end-to-end flow requirements and converges promptly for new flows. Our solution also outperforms MSF with faster convergence, sustained flow constraints, improved stability, and consistent PDR for each

flow. However, this comes with higher energy consumption.

Our solution is also compatible with point-to-point traffic. As a future work, it would be interesting to evaluate the efficiency of the data plane for this type of traffic. Our frame formats and control plane structure allow us to configure the data plane using various paths. By traversing the control plane topology, the `config` packet may contain settings for a node or simply use a node as a relay, enabling the configuration of the desired data flow path.

As our scheduler relies on link quality measurements to allocate resources, the accuracy of link quality estimation becomes imperative to avoid over or under-resource provisioning. In the next chapter, we present an accurate link quality estimation scheme relying on shared cells while ensuring collision-free communication.

Moreover, since a dedicated control plane may raise questions about energy consumption, as it requires nodes to wake up during timeslots labeled ***to-controller*** and ***from-controller*** flow-ids even when there is no traffic. We have already shown that a shared control plane is insufficient. However, a hybrid control plane may still provide reliability but at a lower energy cost. Thus, in the next chapter, we will also provide a more extensive investigation of this research question.

Chapter 4

SDN Architecture Improvements in Link Quality Estimation & Control Plane

Contents

In lossy networks, the SDN controller can allocate additional radio resources for retransmissions to compensate for packet loss. However, the controller may inaccurately estimate the link qualities. Under-estimation leads to over-provisioning (energy wastage), and over-estimation leads to under-provisioning (unreliability). Active techniques rely on probes (aka control packets) transmitted regularly to estimate the link quality toward its neighbors [154]. However, a node must dedicate resources for each neighbor, which represents an unacceptable amount of energy and

bandwidth, especially in a TDMA network. Passive techniques are less expensive but may inaccurately estimate the link quality if packets collide [155]. Moreover, the Link Quality Estimation must be measured for all neighbors without any bias in measurements, necessitating regular traffic with each neighbor. This enables the controller to apply various routing strategies.

As mentioned in the Chapter 6.2, our link quality estimation relies on EB transmissions in the shared cells, enabling the identification of all neighbors. In this chapter, we provide detailed information about our LQE solution, presenting how we prevent collisions in link quality measurements and offer an accurate and energy-efficient scheme.

As also mentioned in Chapter 6.2, we have utilized a fully dedicated control plane for SDN-TSCH. However, we measured that this strategy is energy-consuming. Considering that devices are energy-constrained in IWSNs, a question may arise regarding the high energy cost associated with maintaining a fully dedicated control plane. In this regard, this chapter also includes a broad evaluation of the performance of shared, dedicated, and mixed control planes in the SDN-TSCH network. This evaluation aims to quantify the impact of realistic conditions (such as lossy links) on the SDN network. To the best of our knowledge, this issue has not been addressed previously in scheduled wireless networks.

4.1 Discussion on Link Quality Estimation

We need to measure accurately the link quality to make the wireless SDN network efficient. Basically, two different approaches may be applied to estimate the link quality in wireless networks [141]:

active monitoring: it involves a node observing its neighboring links through the dispatch of probe packets. These packets are transmitted using either broadcast or unicast methods. Unlike unicast probe packets, broadcasting probe packets does not necessitate acknowledgments or retransmissions at the link level. Those probes are typically sent at specific rates, introducing a tradeoff between energy efficiency (at lower rates) and accuracy (at higher rates). An adaptive beaconing rate could potentially achieve a favorable balance in this aspect. Broadcast-based active link monitoring is relatively straightforward to implement and introduces a minor additional load compared to unicast-based approaches. This is why many network protocols and mechanisms choose to utilize it. Conversely, unicast-based active link monitoring allows for more precise measurements due to its resemblance to actual data transmission across the link. However, it remains a resource-intensive mechanism in WSNs due to the communication overhead it introduces.

passive monitoring: is a technique that exploits the existing traffic of a network. It can leverage common packets or their acknowledgments to estimate link quality between nodes. This form of monitoring is widely adopted in low-power wireless networks due to its energy efficiency, avoiding the need for additional communication overhead. While passive link monitoring is energy-efficient, its accuracy can be influenced by the irregularity and fluctuation of network traffic patterns.

More particularly, we need to consider the following challenges that are specific to scheduled wireless networks:

all-neighbors discovery: to build a complete routing topology, the SDN controller should be aware of all good links and their qualities in the network. Unfortunately, in scheduled networks, a node can only estimate the link quality of the neighbor with which it exchanges packets. Moreover, the utilization of shared cells results in collisions, leading to both inaccurate link estimation and neighbor detection failures.

accurate estimation: link quality misestimation has a severe impact on the network capacity and on reliability. In the case of underestimation, a controller will allocate more bandwidth for the link and waste both the network capacity and the energy. By contrast, in the case of overestimation, not enough bandwidth is allocated, and some packets will be dropped or received after the deadline.

continuous estimation: due to the dynamic nature of wireless networks, link qualities may change. Thus, the link estimation system should continuously inform a controller of the last changes in the network.

energy efficiency: exploiting active monitoring in scheduled wireless network is costly. Allocating one dedicated cell to each neighbor for probing consumes high bandwidth and energy. To have an energy-efficient link estimation system, Hermeto *et al.* [156] proved that accurate link quality estimation can be achieved by regularly transmitting existing broadcast packets.

4.2 Accurate Link Quality Estimation

We propose a passive link monitoring scheme to fulfill the requirements outlined in the preceding section. We leverage the EB packets of the TSCH network to estimate the quality of the links. Each node counts the number of received EBs from each neighbor in a given period of time. Because the EB packets are sent periodically, a

node knows how many EBs it should receive from a neighbor during a given amount of time. A controller calculates the PDR of EB packets and uses it as a link quality metric. More precisely, it is computed according to Eq. 2:

$$\widehat{PDR}(n) = \frac{counter(n) * T_{EB}}{T_{report}} \tag{4.1}$$

with $counter(n)$ the EB counter of neighbor n, T_{EB} the period of EBs, $\widehat{PDR}(n)$ the measured PDR for n, and T_{report} the period of report.

Since EBs are periodically transmitted within the TSCH network, maintaining a continuous link quality estimation is energy-efficient and eliminates the need for generating new probing packets. This provides the controller with a consistently updated estimation of link quality. Thus, if there is a change in link quality, the controller can reconfigure the network by defining new rules.

Furthermore, since all neighbors of a node must be awake to receive an EB from the node, we use shared cells for EB transmissions. More importantly, this allows link quality to be calculated and updated for unused or undiscovered links.

Nonetheless, the problem of collisions can arise between two concurrent EB transmissions or between EBs and other broadcast packets sent via shared cells. We propose to solve this link quality misestimation by scheduling more appropriately the EBs. We keep on exploiting shared cells to optimize the discovery process and the continuous link quality re-estimation, while removing collisions.

4.3 Organization of the Shared Cells in the Control Plane

As described in Chapter 6.2, We need to send control packets in the control plane (joining, configuration, etc.): Novel nodes cannot use dedicated cells for `report` packets due to no reserved resources. Keep-alive packets are also needed initially to synchronize with the parent if no data packet is forwarded.

EBs also use shared cells for the network discovery. Thus, to avoid collisions among EBs, we propose that the SDN controller allocates cells for EBs. A controller allocates a given shared cell to a single node for its EB transmissions. Thus, we cannot have anymore EB collisions. Additionally, to prevent collisions between EBs and non-EB control traffic, we establish a clear distinction between the shared cells used for EB transmissions and those used for non-EB traffic.

More precisely, at the beginning of a shared cell, a node makes the following verifications:

1. if the shared cell is the EB cell dedicated to the node, it engages the transmission of its EB;

2. if the shared cell is the EB cell of a neighbor node, the node keeps awake to receive possibly its EB;

3. if the shared cell is not an EB cell, the node transmits the first control packet in its queue. If none is present, it stays awake to receive possibly solicitations from neighbors.

In any case, a node in listening mode can turn off its radio after a fixed offset if no preamble is detected on the medium.

4.4 Schedule of EBs Shared Cells

We propose that each node uses a given shared cell in slotframe to transmit its EBs. To enable this, we assign unique ID to each shared cell in the slotframe. In the joining process, a controller specifies which shared ID must be used by a new node. A controller allocates shared cell IDs sequentially, in the order of the arrival of the nodes. The last joined node has the maximum shared cell id ($slot_{EBmax}$). In that way, we can make a distinction between the EB and non-EB parts of the slotframe. Since the $slot_{EBmax}$ value is announced in the EBs, the novel value is pushed hop by hop in the network, and after a given amount of time, all the nodes are aware of the novel cells reserved for the EBs.

We allocate some of the shared cells for non-EB traffic, and also separate the shared cells of EBs from non-EBs. To avoid collision with non-EB packets, nodes should only use the shared cells that are not used for any EB: if the shared slot ID is smaller than $slot_{EBmax}$, the shared cell is reserved for EBs, else, the cell can be used by any unassociated node.

Meanwhile, the distribution of non-EB shared cells can impact the amount of collision in non-EB traffic. If the shared cells for non-EB traffic are grouped contiguously in the slotframe, the nodes experience a high collision rate [157]. Indeed, the nodes that generate their non-EB packets at the beginning of slotframe have to wait until the end of slotframe to send their packets. Then, all the nodes try to use the first coming non-EB shared cell which leads to a cumulative collision.

To minimize the collision probability, the EB cells should be distributed in the slotframe. More precisely, we assign to each shared cell a unique ID (designated as ***shared-id***). Then, we use a recursive algorithm (Figure 4.1) to assign ***shared-id***s regularly in the slotframe. Recursively, the controller assigns the ***shared-id***s to split the remaining available space into two equal parts. For instance, in the first step, the ***shared-id***1 is located in the middle of the slotframe (between 0 and the slotframe length). As illustrated in Figure 4.1, the ***shared-id***1 corresponds to the timeslot 30. Then, the subsequent ***shared-id***s are positioned in the middle of their

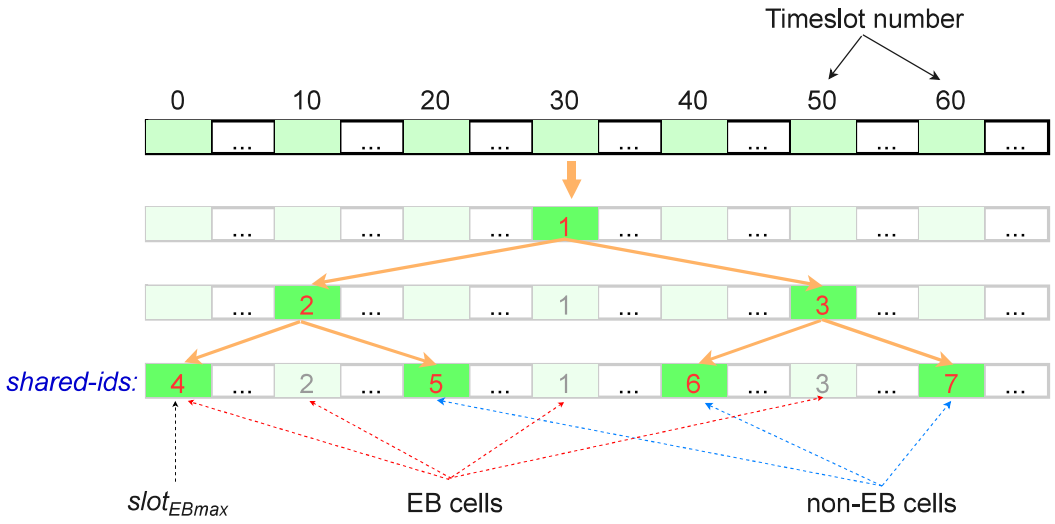

Figure 4.1: Shared cell ID allocation in slotframe for the control plane

respective halves. Each node can apply the same recursive algorithm to identify EB and non-EB cells using $slot_{EBmax}$.

4.5 Numerical Estimation of Shared Cells for non-EB Traffic

We use the slotted Aloha model to estimate the number of shared cells in a slotframe for both EB and non-EB traffic. We assume that for joining the SDN network, nodes wake up at any time and generate a join request when they received a sufficient amount of EBs from already attached neighbors. We assume the control traffic in shared cells (keep-alive and `report` packets) follows a Poisson distribution with a mean of λ. We calculate the collision probability through:

$$\begin{aligned} P(collision) &= 1 - P(idle) - P(success) \\ &= 1 - e^{-\lambda} - \lambda e^{-\lambda} \end{aligned} \tag{4.2}$$

$P(idle)$ denotes the probability of zero transmission, and $P(success)$ is the probability of one unique transmission in the timeslot. By fixing the maximum acceptable value of $P(collision)$, we obtain the λ value. With the number of generated packets in the slotframe, we find the number of required shared cells:

$$Non\_EB\_cell = \lambda \cdot Num\_packet \tag{4.3}$$

Num_packet is the sum of all non-EB packets for all the nodes in the slotframe.

Indeed, to provide a safety margin, we assume the worst-case situation in which all the nodes are in the transmission range of each other, and a shared cell may be used by any node in the network. Moreover, for EB transmissions, each node needs one shared cell. Hence, the total number of shared cells is finally:

$$Num\_shared\_cell = Non\_EB\_cell + Node\_number \tag{4.4}$$

4.6 Resource Allocation and Configuration of Control Plane

We may allocate a large number of dedicated cells to create a reliable control plane. However, the more cells we allocate, the more we increase energy consumption. Indeed, unused cells consume a fixed amount of energy [158]: a receiver has to stay awake during a fixed offset from the beginning of the timeslot to accommodate clock drifts.

We investigate here three different methods to implement the control plane:

shared: all the control packets use the shared cells, that are mutualized in the network to reduce the amount of resource allocated to the control plane;

dedicated: each node maintains one dedicated cell to send control packets to its parent, and another one to its children. While no collision can arise among control packets, more resources are allocated to the control plane. We employed this strategy for SDN-TSCH in the previous chapter.

hybrid: each node maintains one dedicated cell to send control packets to its parent and sends other control packets through shared cells. Downward transmissions may be adjusted by a controller, and we may expect a low volume of collision in downlink.

The cost of these different architectures for the control plane will be evaluated in the following section.

4.6.1 Dedicated Control Plane

We propose in this variant to use only dedicated cells when the node has joined the network. Thus, control packets are protected against collisions, and we can upper-bound the delay to reconfigure a network (that depends on the size of each subtree rooted at the sink). When admitting a novel node, a controller allocates two dedicated cells:

dedicated-up: is reserved for the control traffic sent to the parent (for the flow-id ***"to-controller"***);

dedicated-down: is reserved for the control traffic sent to any child (for the flow-id ***"from-controller"***). Practically, all the children have to stay awake for this dedicated cell. Only the link-layer destination will acknowledge the control packet.

However, exploiting dedicated cells has a cost: each node has to stay awake during 4 dedicated cells (*i.e.*, to and from children, to and from parent) per slotframe, even if no control traffic is transmitted. In particular, the cells corresponding to the flow-id ***"from controller"*** are uniquely used for `config` packets and are thus unused after the network has converged.

4.6.2 Shared Control Plane

It may be energy efficient to exploit only the shared cells to handle all the control traffic [159]. We face a burst of control packets when the network bootstraps, and all the shared cells are mutualized to send all the control traffic. We could expect statistical multiplexing to make the number of collisions reasonable. However, as illustrated in the performance evaluation in the previous chapter (Figure 3.7), using shared cells impacts the convergence time of the network: collisions may delay or even prevent nodes to exchange admission request/response packets.

Moreover, a shared control plane may affect the recovery time of critical applications. In particular, an event may occur locally that triggers flow reconfiguration. For instance, the decrease in the quality of a critical link may force a controller to reconfigure all the flows that pass through this faulty link, changing the paths, or rescheduling the whole flow to respect deadline constraints. A shared control plane may jeopardize the re-convergence of the network. In conclusion, a shared control plane is not agile enough to deliver traffic in a timely manner.

4.6.3 Hybrid Control Plane

We propose a hybrid variant to combine dedicated cells in uplink and shared cells in downlink. Indeed, `report` packets are only transmitted uplink, through the ***"to controller"*** flow-id. The rest of the control packets that may compete correspond to i) `config` packets from a controller, ii) keep-alive packets to maintain the synchronization, iii) `report` packets from novel nodes. It is worth noting that the EBs cannot collide with other control packets, since shared cells are allocated by a controller to EBs and separated from the non-EB cells.

Keepalive packets are only used when the network bootstraps. Indeed, nodes use any packet (including the data ones) for re-synchronization. Thus, when the network has converged and when each node forwards or generates data packets, the network

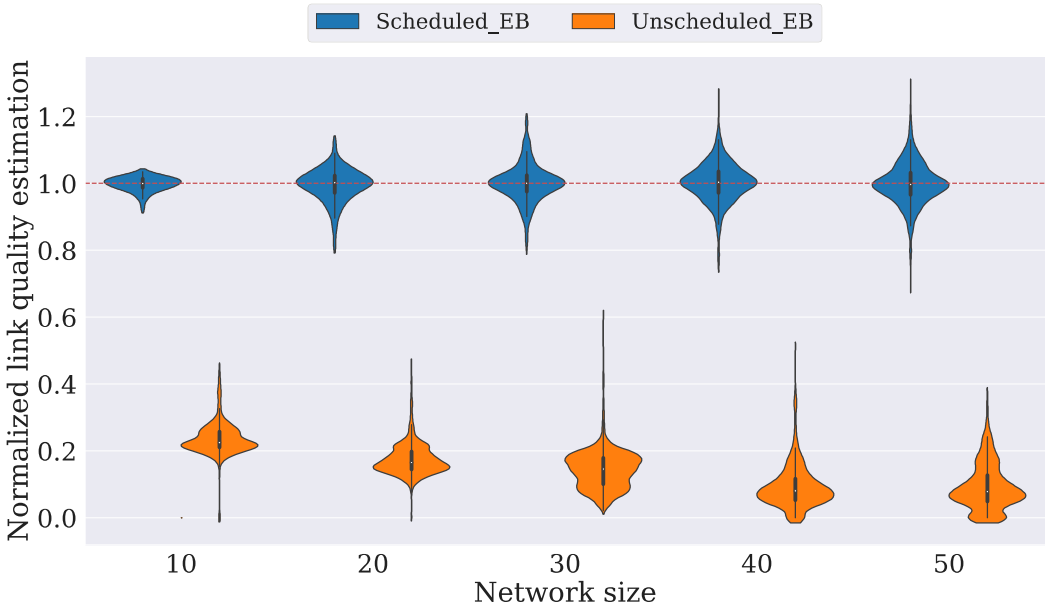

Figure 4.2: Link quality estimation

is maintained synchronized. Keepalive packets are only required to maintain the synchronization before the arrival of the `config` packet from a controller.

In conclusion, we would expect a lower amount of collisions compared with the full shared control plane, but the convergence delay has to be finely measured to quantify the cost of using shared cells in the downlink.

4.7 Performance Evaluation

We first assess the accuracy of our link quality estimation solution. Then, we evaluate the performance of our three different control plane architectures. We use the same setup outlined in Table 3.1.

4.7.1 Accuracy of the Link Quality Estimation

We measure the accuracy of our link quality estimation for our scheduled EB approach and compare it with the non-scheduled EB approach. In the non-scheduled EB approach, EB cells are not scheduled by the controller, allowing any node to utilize any shared cell for transmitting its EBs. We measured in particular the normalized link quality, which is the ratio between the PDR measured by the approach, and the actual PDR as modeled by the simulator (Figure 4.2). An ideal estimation would always return quality of 1. Values below 1 mean that the PDR is under-estimated, and values above 1 mean that the PDR is over-estimated.

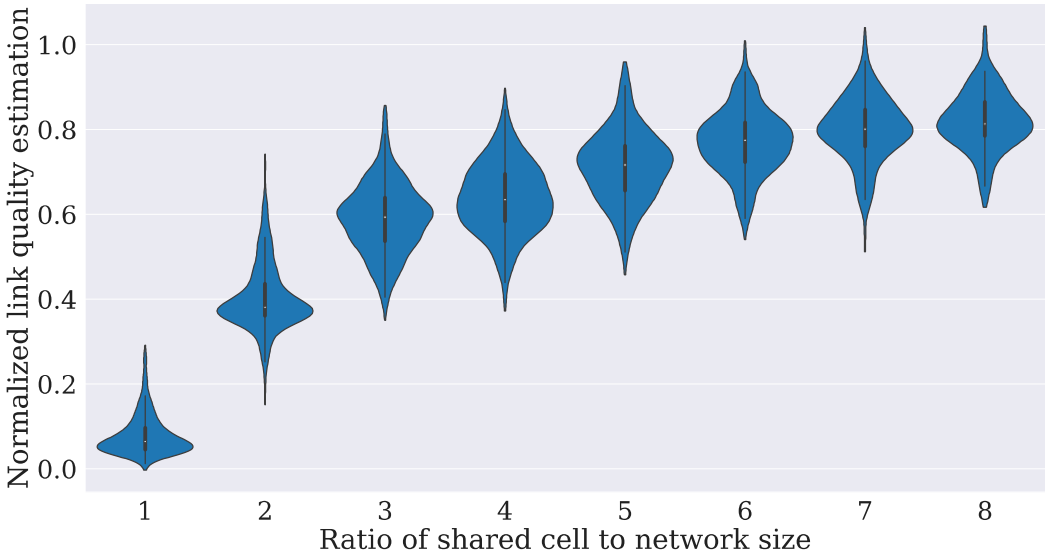

Figure 4.3: Accuracy of link quality estimation in SDN-TSCH

We use the same number of shared cells for both scheduled EB and unscheduled EB solutions. As shown in the scheduled EB approach, link qualities are estimated with high accuracy regardless of the network size. The error seems following a normal distribution centered on the real value. By contrast, the quality is highly under-estimated when EBs are not scheduled: an EB may not be received because of a low SNR value, or because of a collision. Even worse, the error increases for larger network sizes, with a normalized quality tending toward zero for the largest network sizes. This misestimation leads to high energy consumption: a controller allocates too much resources for all the flows, wasting bandwidth and energy.

Figure 4.3 illustrates the collision rate of EBs in unscheduled mode, that measures the accuracy of the link quality when varying the ratio of the number of cells and the number of nodes (network size). As shown, EBs still collide even when allocating a large number of shared cells, resulting in an inaccurate link quality estimation. It keeps on underestimating the link quality, biased by the collisions of EBs. Thus, this can lead to significant overprovisioning in resource allocation.

Our scheduled-based solution is very efficient since we can estimate very accurately the PDR of each link, that is reported in `report` packets. Thus, the controller can allocate the radio resources accordingly.

4.7.2 Efficiency of Dedicated and Hybrid Control Planes

As the performance of the shared control plane has already been evaluated in a previous chapter, we here compare the performance of dedicated and hybrid control

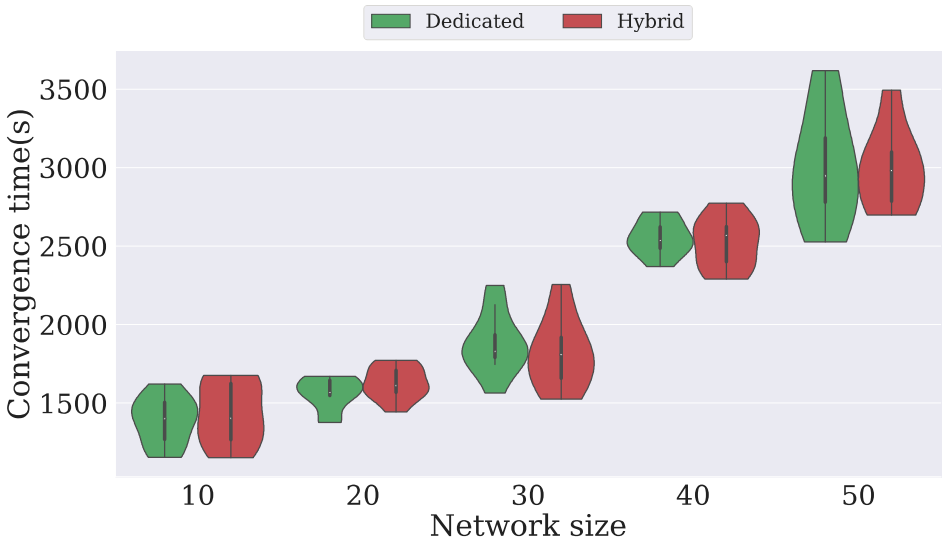

Figure 4.4: Network convergence time

planes. We consider a single controller that allocates all the resources. In light of our previous results, we use the scheduled EB transmission on separated shared cells for all approaches.

We first focus on the convergence (Figure 3). We define the convergence time as the time required from the bootstrap of the network until the last node to join the control plane (*i.e.*, receives the `config` from a controller). Both the dedicated and hybrid control planes maintain a reasonable convergence time, which appears to be linear with the number of nodes Indeed, the density is kept constant, and a larger network size means a larger mean hop distance from the sink. Thus, the nodes need to wait longer that the previous hops synchronize and join the network.

Then, we measure in Figure 4.5 the time required to configure a novel data flow in the network – the time between the transmission of the `flow-request` and the reception of the corresponding `config` packet. The dedicated solutions provide the lowest configuration time: resources are dedicated to send the `flow-request` and the `config` packets without collisions. On average, less than 5 seconds are required for the call-admission and the configuration of the whole path, even with 50 nodes. The hybrid control plane provides also a very reasonable configuration time for any network size. However, the worst-cases may exhibit a large configuration duration. When many devices send their `flow-request` in a short time window, the sink is unable to transmit the corresponding `config` packets in a timely fashion, due to the collision happening between the `config` packets themselves or between the `config` and keep-alive packets. Some nodes still have not been configured yet, and send keep-alive packets to keep synchronized with their parents. Also, in the burst of

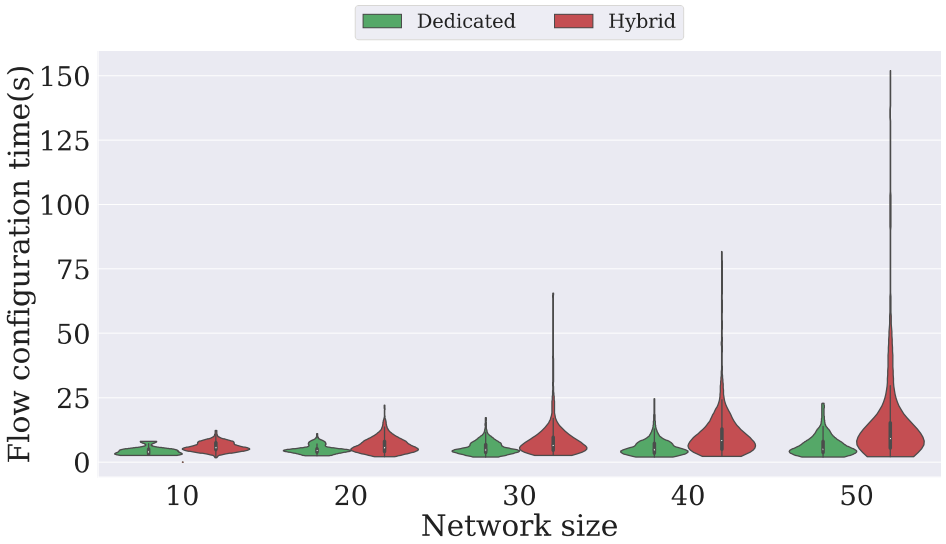

Figure 4.5: Data flow configuration time

collision, the sink node sometimes cannot dequeue all of its queued `config` packets and drops some of them due to the queue size limitation. This forces the source nodes to retransmit another `flow-request` after the timeout (50s in the simulations).

Then, we measure the power consumption of devices, focusing specifically on the period before the network has converged (Figure 4). Thus, we report the power consumption for this period – the energy consumed divided by the time elapsed before receiving the first `config` packet. Dedicated and hybrid control planes have the same level of power consumption. The power consumption linearly increases with the network size. Indeed, based on the proposed model for the estimation of shared cells, we increase the number of shared cells by increasing the network size so that nodes wake up in more timeslots and consume more power.

Finally, we measure the network lifetime (Figure 4.7). We assume the lifetime is given by the first death. To extrapolate our simulations, the lifetime is estimated as the initial energy divided by the power consumption of the most constrained node. Since the configuration is done once, we consider the power consumption after the node has joined the network. Thanks to the use of shared cells for downward control traffic, the hybrid control plane provides a higher lifetime than a dedicated control plane. However, the difference is not substantial. For instance, in the network size of 50, the hybrid control plane improves the lifetime by 7% on average.

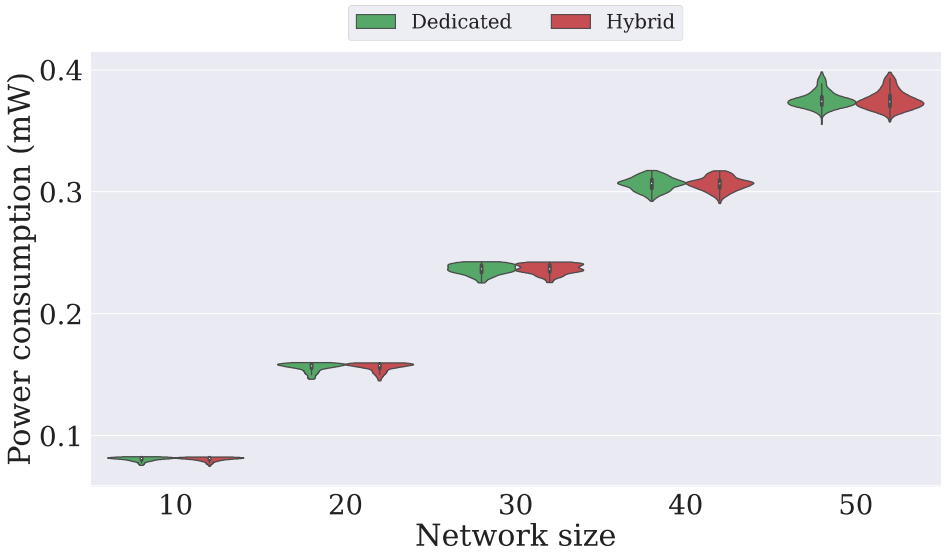

Figure 4.6: Power consumption of nodes in joining period

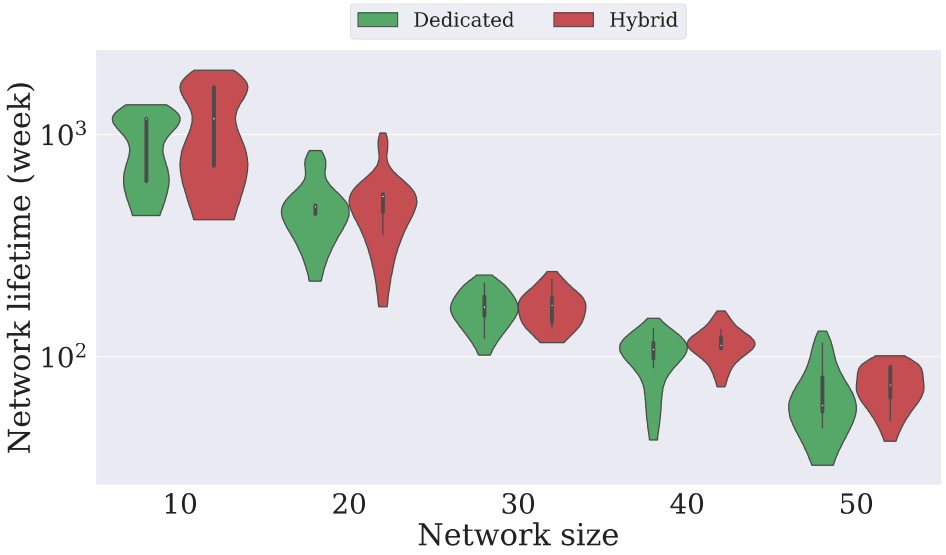

Figure 4.7: Network lifetime

While the hybrid control plane impacts the time of flow admission, it also does not improve the lifetime impressively. Moreover, we may later need to extend the control plane to support more functionalities. Thus, the controller has more control packets to send. From our point of view, the dedicated control plane can provide a higher reliability to handle control packets in a timely fashion, with a small increase in energy consumption compared with the hybrid control plane.

4.8 Conclusion and Future Works

We have introduced an accurate and energy-efficient link quality estimation method for scheduled SDN networks. Making use of Enhanced Beacons on the TSCH network, we passively assess the quality of each link in terms of PDR. By exploiting the global vision of the controller, we designate specific shared cells for collision-free EB transmissions, while also inducing a separation between EB and non-EB shared cells to avoid collisions EBs and non-EB traffic. Our simulation demonstrates the highly accurate estimation of link qualities in different network sizes. Furthermore, our detailed analysis of control plane systems highlights the advantages of using a fully dedicated control plane to increase overall reliability. It is agile enough to configure the network in a timely manner with only a slight increase in energy overhead compared to the hybrid control plane.

As a prospective endeavor, validating our approach in a real-world setting will provide insight into its effectiveness and potential challenges that may not be apparent in simulation settings. Also, Since our LQE relies on periodic transmission of EBs, it would be interesting to investigate situations in which EBs are sent in dynamic period.

Given the dynamic nature of link quality in wireless networks, the SDN controller must consistently reconfigure the network to update nodes routing and scheduling rules. In the upcoming chapter, we further enhance the controller's capabilities to efficiently reconfigure the flows when needed.

Chapter 5

Maintenance of Software Defined IWSN

Contents

SDN demonstrates its potential to efficiently manage routing and scheduling parameters of wireless devices. However, it is important to acknowledge that the quality of wireless links can change dynamically due to various factors like signal interference and environmental channel conditions [160]. This can impact the reliability of both the control and data planes in such networks. This challenge becomes even more challenging in scenarios where IWSNs' applications require high reliability, as a specific number of time-frequency blocks (e.g., IEEE 802.15.4-TSCH schedule) are allocated for a flow to achieve a certain delivery ratio and a latency deadline. Thus, the SDN controller must continuously monitor changes in link quality for wireless connections and compute appropriate forwarding and scheduling rules to address these variations effectively.

The SDN controller faces the substantial challenge of swiftly adapting to network dynamics. It needs to gather device connectivity details and then establish routing paths and a novel schedule [129]. Thus, this freshly computed schedule is disseminated across the network using a multi-hop wireless mesh arrangement. This process of updating the schedule can lead to considerable communication overhead, particularly within networks characterized by numerous intermediary hops.

In this chapter, we introduce mechanisms for the efficient maintenance and update of wireless scheduled SDN networks. We present a method enabling a controller to identify failing links by utilizing the link quality reports from individual nodes. Additionally, we propose techniques to update both the control and data planes, facilitating the redirection of flows through new subpaths when necessary.

5.1 Scheduled SDN Reconfiguration Overview

In the Chapter 6.2, we propose an architecture for admission of critical industrial applications with a focus on fulfilling SLAs. We extend here SDN-TSCH to support a real-time network reconfiguration whenever the conditions evolve. When the link quality significantly deteriorates below a specified threshold, the controller initiates reconfiguration. However, the reconfiguration process is not straightforward: the control plane uses specific cells that have to be relocated if the topology changes. Besides, all the flows have to be redirected in the data plane.

Our solution is:

energy efficient : we can readjust bandwidth while over-provisioning leads to energy inefficiency, and under-provisioning may result in SLA violations. Besides, a `config` packet can configure a whole path: there is no need to send an end-to-end `config` packet to each forwarding node.

reliable: we prevent any inconsistent state when the network is reconfigured. The forwarding rules are consistent and no data packet is dropped in the data plane, even if a SDN control packet needs to be retransmitted, or is dropped.

Let us consider the scenario illustrated in Fig. 5.1. If the quality of the link $F \rightarrow B$ decreases from 95% to 40%, the network must be reconfigured. The controller needs to i) detect the ***fault***, ii) reconfigure the control plane so that F has dedicated cells with its new next hop toward the border router (C), iii) redirect data flows (*e.g.*, $Flow_D$) by allocating resources in the data plane for the new path.

5.2 Fault Detection & Parent Selection

As explained in Chapter 6.2, each node sends periodically `report` packets to the controller, piggybacking the number of EBs received from its neighbors. This allows the controller to estimate the PDR of each link. In particular, the link between any node with its parent must present a sufficient link quality to limit the number of retransmissions and packet drops. Thus, for each node, the controller continuously compares the PDR of its current parent, and the PDR of its best neighbor which could serve as new parent. Additionally, the controller verifies that the candidate

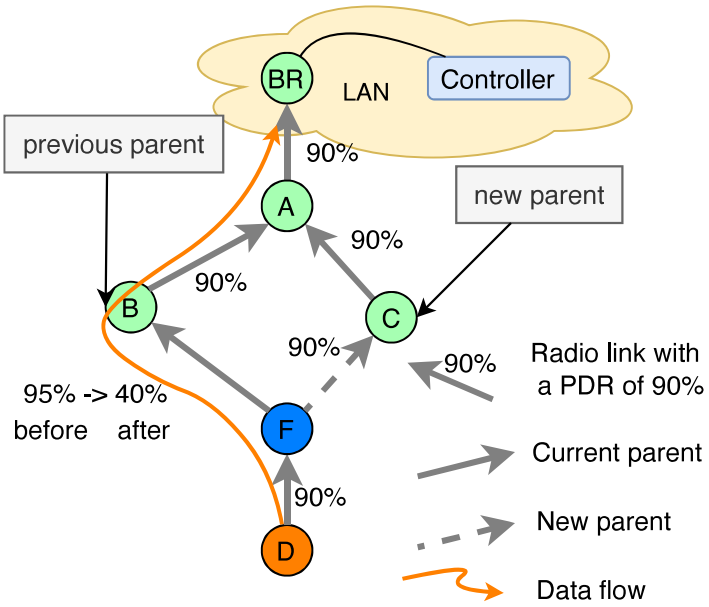

Figure 5.1: A controller needs to reconfigure the control and data planes to redirect the flows through the link (F→C) when the PDR of the link (F→B) decreases from 95% to 40%

for new parent is not a node within the subtree of the target node (indicating the need for a parent change), thereby preventing loops in the network topology. Thus, a new parent is selected if the following conditions hold:

$$\begin{aligned} &\text{Condition 1:} \quad PDR(current\_parent) \leq \alpha * PDR(best\_parent) \\ &\text{Condition 2:} \quad best\_parent \notin subtree\_nodes \end{aligned} \tag{5.1}$$

where $PDR()$ denotes the PDR value, $best\_parent$ is the neighbor with the largest PDR and which is not a descendant (no-loop condition), and $current\_parent$ is the current parent. $0 < \alpha < 1$ is the parameter defining the sensitivity of the reconfiguration rule (a larger value means more oscillations). $subtree\_nodes$ is the list of nodes in the subtree of the target node through which their control traffic passes.

Before triggering the reconfiguration, the controller also verifies that there is no active reconfiguration process in the network. This deterministic approach prevents possible conflicts between the updating processes and ensures consistent reconfiguration. For example, if a reconfiguration process is active for the $best\_parent$ and its parent when the $best\_parent$ is selected for the reconfiguration process of the target node, the concurrent reconfiguration process is paused until the current process is finished.

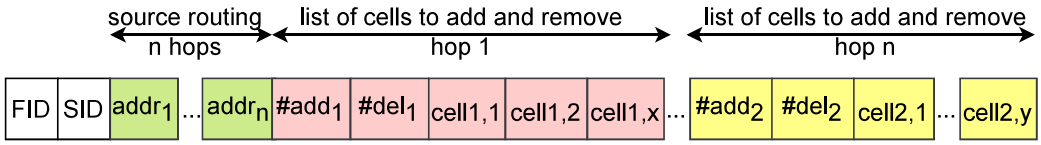

Figure 5.2: Enhanced format of `config` packet

5.3 Control and Data Planes Update

The controller needs first to reconfigure the control plane. More precisely, the path between the target node and the border router has to be updated to pass through the new parent. The controller has to allocate cells between the target node and its new parent (respectively F and C in Fig. 5.1). More precisely:

to controller: a new dedicated cell has to be reserved between the target node and its new parent. If the timeslot between the target node and its old parent is free in the schedule of the new parent (half-duplex condition), the same cell is reused. Else, another cell is selected randomly to avoid collisions;

from controller: the new parent has already a transmitting cell to its children. Thus, the controller just needs to add the corresponding RX cell in the schedule of the target node.

The controller forges two `config` packets to push the new configuration, one per direction (to and from the controller). To route `config` packets to the target node, we employ the same source-routing mechanism used to configure the flows in the SDN-TSCH (Chapter 6.2). The `config` packet is forwarded through the dedicated cells in the `from controller` direction. In particular, the new parent will receive the packet to forward and will update its own schedule. For the last hop, a shared cell is used to reach the target node since no dedicated cell is present in the control plane for this specific (new) link.

To optimize the overhead of control packets, we enhance the `config` packet format to use the same packet to both install and remove cells (Fig. 5.2). Basically, the `config` packet piggybacks i) the route to follow (n nodes), ii) the list of cells to insert and to remove for each hop.

When the target node finally receives the `config` packet, it sends an end-to-end `ack` to the controller. It is worth noting that the target node stops using its previous parent as soon as the cell is installed in its schedule.

After a timeout, the controller retransmits the `config` packet if no `ack` is received. This way, we maintain the global consistency of the schedule of the control plane.

When the target node has dedicated cells to and from its new parent for the control plane, the controller needs then to redirect the data flows through the new parent. Thus, it needs to send one `config` packet for each of the data flows. A `config` packet is forwarded through the control plane, which has already been reconfigured, using a similar method as before. The `config` packet follows the new path from the destination toward the source and exploits only dedicated cells to avoid collisions. As soon as the target node receives the `config` packet, data packets start to be transmitted through the new path for the corresponding data flow. It is worth noting that we guarantee flow isolation even during the convergence. We cannot have any inconsistent state since the path is configured up to the source node (upstream).

5.4 (Re)-scheduling Algorithm

Computing a new schedule from scratch is expensive since we would have to remove all the previous cells and to install a sequence of new cells all along the path. We propose rather a rescheduling algorithm that tries to minimize the number of schedule updates. Obviously, the problem is NP-complete, and we propose here a heuristic.

To minimize the changes, we focus on the subset of the path which differs between the previous and the new path. For instance, the subpath to update corresponds to the links $F \rightarrow C$ and $C \rightarrow A$ in Fig. 5.1. To reduce the end-to-end delay, we need to schedule the cells back-to-back, minimizing the buffering time. Moreover, it is more costly to change the schedule of nodes farther from the border router: the `config` packet has to be forwarded farther, possibly retransmitted, etc. Thus, we keep the same schedule from the source node to the target node and update only the rest of the schedule.

We adopt a greedy approach to compute a new schedule for the subpath to minimize the end-to-end delay [152]. First, the controller computes the number of cells to allocate for each hop, depending on the link quality. Similar to our approach in Chapter 6.2, it greedily increases the number of cells on the weakest link in the subpath until the minimum end-to-end reliability is respected. Then, it selects free cells for each hop of the subpath. To minimize the forwarding delay, consecutive cells are selected preferentially (else, the closest cell is selected).

When the schedule of the subpath is computed, the controller may encounter two different cases:

valid: the last cell in the subpath is scheduled before the first cell of the rest of the path. In that case, the schedule is valid and is applied without modification;

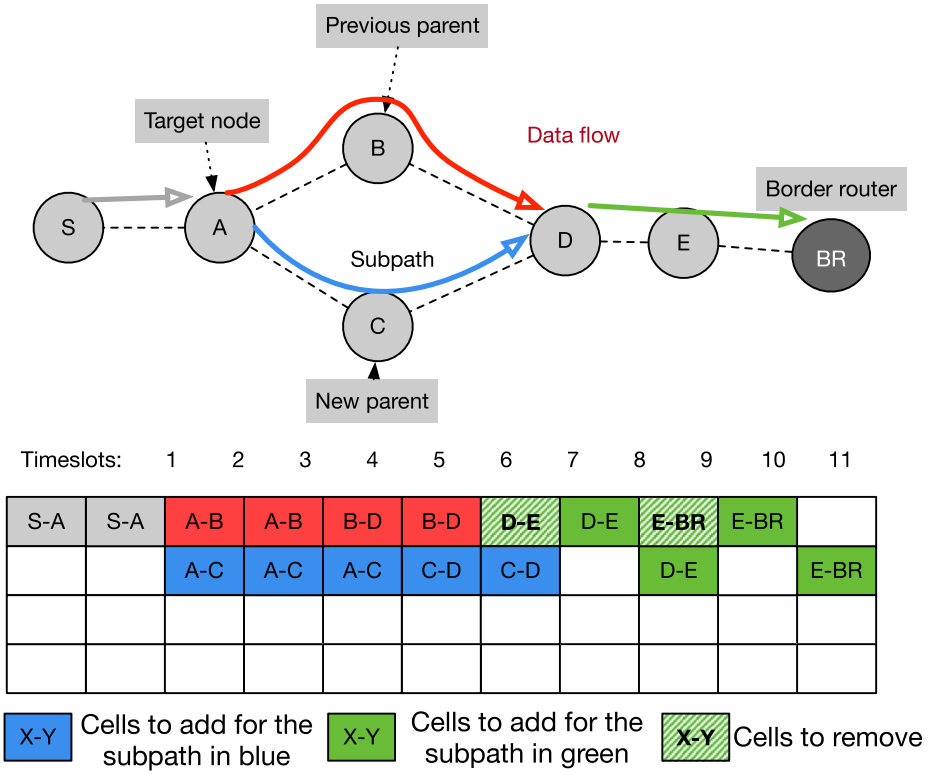

Figure 5.3: Schedule update when redirecting a data flow

overlap: the last cell of the subpath is scheduled after the first cell of the rest of the path. In that case, the controller updates the rest of the schedule hop-by-hop toward the border router. If the last cell of the current hop is scheduled **before** the cells for the next hop, the controller stops the updates. In the worst case, the schedule is updated up to the border router.

For the sake of clarity, we describe here only the case where the controller is the destination. However, this approach is obviously generalizable to any destination in the network.

Algorithm 1 presents the re-scheduling approach of the new path. Starting with the schedule of the subpath, it updates only the parts of the schedule that need to be changed to ensure that the end-to-end reliability and deadline are respected. More specifically, in lines 1 to 8, the algorithm assigns cells to the subpath. In line 9, it verifies whether the last cell scheduled for the subpath is after the first cell of the hop following the subpath. If this condition holds, it necessitates the relocation of cells after the subpath. This relocation continues until the newly defined cell is placed before the next already defined cell, and the condition of back-to-back is met.

Let us consider the scenario illustrated in Fig. 5.3. The first two cells correspond to the path before the target node (A): they will not change. Then, the scheduler

Algorithm 1: Re-scheduling of novel path

Data: $prev\_path\_num\_hops$, $prev\_schedule$,
$prev\_path\_num\_cells\_hop$, $subpath\_num\_hops$,
$subpath\_num\_cells\_hop$, $last\_cell\_of\_hop\_before\_subpath$,
$first\_cell\_of\_hop\_after\_subpath$,
$first\_hop\_index\_after\_subpath$
Result: $schedule$

```
1  lastCell ← last_cell_of_hop_before_subpath
2  for i = 1 to subpath_num_hops do          // schedule cells for subpath
3  |  for j = 1 to subpath_num_cells_hop[i] do
4  |  |  cell ← Pick first free cell after lastCell
5  |  |  schedule ← cell
6  |  |  lastCell ← cell
7  |  end
8  end
9  if lastCell > first_cell_of_hop_after_subpath then   // back-to-back
     condition is not met
10 |  for i = first_hop_index_after_subpath to
   |   prev_path_num_hops do
11 |  |  for j = 1 to prev_path_num_cells_hop[i] do
12 |  |  |  cell ← Pick first free cell after lastCell
13 |  |  |  schedule ← cell
14 |  |  |  lastCell ← cell
15 |  |  |  if lastCell < prev_schedule[i][j + 1] then
16 |  |  |  |  PushSchedule(schedule) // Condition is met:  stop scheduling
17 |  |  |  end
18 |  |  end
19 |  end
20 end
21 else
22 |  PushSchedule(schedule)                  // Push the final schedule
23 end
```

assigns consecutive cells for the new subpath (A,C,D) in blue. It is worth noting that the controller can assign the same timeslots through the old and new paths: it is sufficient to use different channel offsets to avoid collisions during the convergence.

This example shows an overlapping schedule: the last cell for the link $C \rightarrow D$ and the first cell of the link $D \rightarrow E$ are the same (timeslot 7). Thus, the controller has to assign hop-by-hop new cells (the cells in green) to update the rest of the schedule up to the border router. Symmetrically, the overlapping cells have to be deallocated (with diagonal hatching).

5.5 Obsolete Cells Removal

It is worth noting that some cells have been allocated and are not used anymore through the previous path. However, the energy consumption of unused cells is quite low [158]. Indeed, an unused TX cell has no cost: the transmitter does not wake up when no packet in its buffer has to be forwarded through the corresponding cell. Inversely, the receiver has to wake up, but it can turn its radio off after a fixed offset if no activity is measured on the medium.

We argue that the cost of unused cells is much smaller than sending and forwarding explicit `config` packets that have to be acknowledged end-to-end, retransmitted, etc. Thus, we propose a simple, timeout-based, schedule management. If no cell is used for a particular flow for a certain period of time, the corresponding cells are silently removed from the schedule. Each node maintains one timer per data flow, rearming the timer when it forwards a packet corresponding to the flow. Typically, the cells in red in Fig. 5.3 will be automatically removed from the schedule of B and D after the timeout.

We need to ensure consistency between the view of the controller and the nodes in the old path with obsolete cells. Thus, when the controller receives the `ack` of a `config` for a flow-id on the new path, it sets a timer (with the same value as the nodes with obsolete cells) to trigger the removal of the cells from old path in the schedule. Obviously, the controller knows the timer value as it has already set it for the nodes.

5.6 Performance Evaluation

We implement our solution in contiki-ng and Cooja and compare the following solutions:

SDN-TSCH-orig is the original operation of SDN-TSCH that does not update the schedule when a flow is configured;

SDN-TSCH-reconf is the extension of SDN-TSCH-orig that detects when a reconfiguration is required and which reconfiguration is needed;

MSF is the distributed scheduling function for TSCH[3]. It proposes an adaptive solution for network changes;

We simulate the scenario illustrated in Fig. 5.1 representative of a topology with changing conditions. This straightforward scenario allows us to verify that the reconfiguration process operates properly as intended. To more precisely analyze the

[3] `https://github.com/alexrayne/contiki-ng.git`

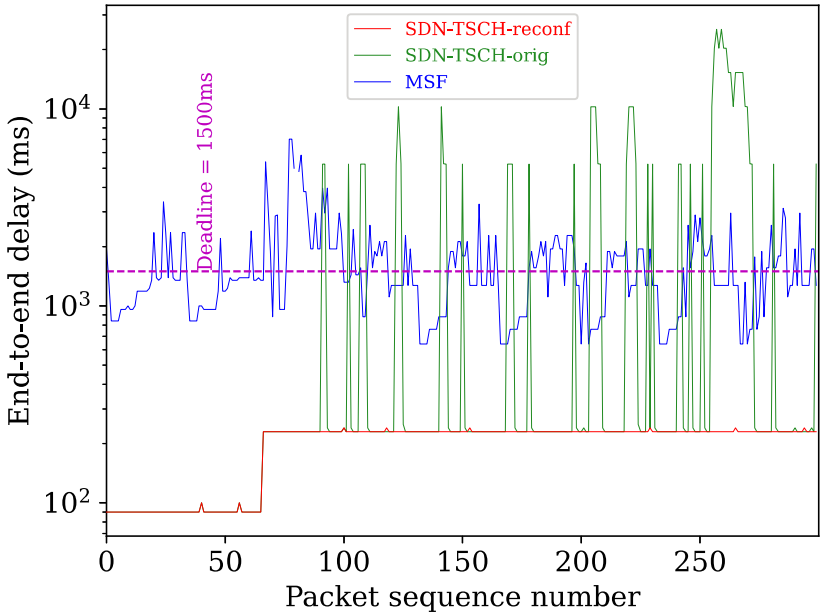

Figure 5.4: Per-packet end-to-end delay of $Flow_F$ in each solution

impact of a reconfiguration, we consider a single link change. A real deployment may imply multiple reconfigurations by the controller but leading to the same conclusions. Each node hosts an application which requires an end-to-end PDR larger than 99% and an end-to-end delay lower than 1500ms. For the remaining simulator parameters, we use the same setup as presented in Table 3.1.

5.6.1 Results and Comments

We measure first the end-to-end delay of $Flow_F$ (Fig. 5.4). The quality of the link $F \rightarrow B$ is manually changed when D generates the packet sequence number 90. SDN-TSCH-orig is able to deliver only some of the packets before the deadline: many of them require retransmissions or are dropped because of a buffer overflow. On the contrary, SDN-TSCH-reconf is able to reconfigure the network very efficiently, and all the packets are delivered before the deadline. It is worth noting that the delay increases after sequence number 90 because more retransmitting cells are required to compensate for the slightly lower PDR offered by the new path. Finally, MSF cannot provide any delay guarantee since it has not been designed for this purpose.

Fig. 5.5 reports the PDR of two data flows ($Flow_F$ and $Flow_D$) that have to be redirected through the link ($F \rightarrow C$). When the link quality degrades, SDN-TSCH-orig is not able to reconfigure the network, causing the end-to-end PDR to fall below the expected 99% level. The number of cells is not sufficient to cope with the required number of retransmissions. By contrast, SDN-TSCH-reconf is

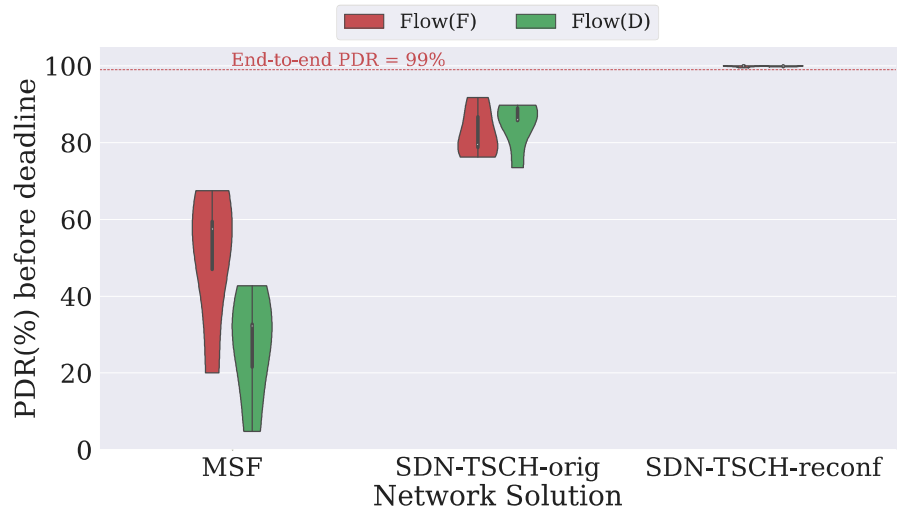

Figure 5.5: Per-flow PDR

able to update the schedule, and it keeps on providing ultra high-reliability after the topology change. It is worth noting that MSF is not able to provision enough resources quickly, since many oscillations may arise. Thus, it provides a very low PDR after the quality of the link $F \rightarrow B$ is reduced.

We finally measure the reconfiguration time (Fig. 5.6) and the overhead during the reconfiguration (Fig. 5.7). We define the reconfiguration time as the time it takes to initiate the reconfiguration of a node and its subtree to adapt to the new path and schedule. SDN-TSCH-reconf implements an average reconfiguration of 14s. Indeed, one `config` packet is forwarded to or from the controller in one slotframe, which lasts 2.5 seconds. Since the control and data planes have to be reconfigured for two data flows (4 `config` packets), we need consequently several slotframes (at least one slotframe per `config` packet if we consider possible retransmissions). MSF needs on average 12 packets to readjust the bandwidth for the incriminated links, which remains very reasonable. However, we need also to wait for the convergence of RPL to have consistent routing tables and then update the local schedules. Possibly, RPL may experience temporary oscillations during such a convergence. Thus, MSF converges much slower than SDN-TSCH-reconf.

In addition to the reconfiguration time, the network should also detect when a significant change occurs. MSF experiences very long detection times, ranging from 1.5 to 18 minutes. Considering SDN-TSCH-reconf, a reconfiguration is triggered upon reception of a `report` packet. In the worst case, the change occurs just after F sends its `report`, so the detection time is bounded by the `report` period set to 5min in our simulations.

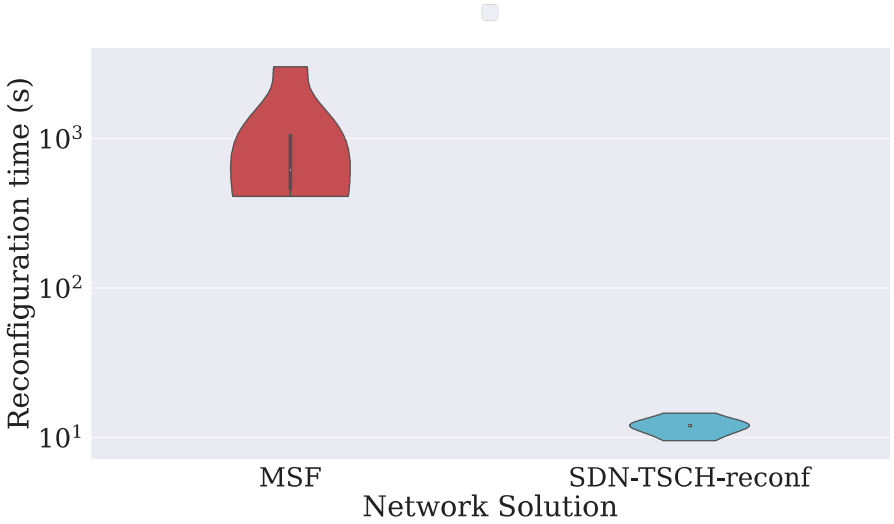

Figure 5.6: Reconfiguration time

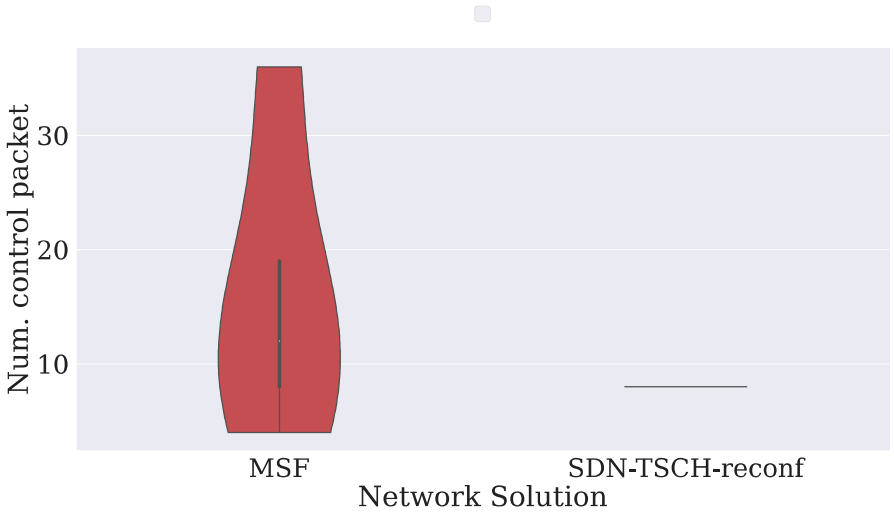

Figure 5.7: Control overhead for reconfiguration

5.7 Conclusion and Future Works

We have presented here all the mechanisms to reconfigure a SDN scheduled wireless network. We reduce the overhead, with a single control packet to configure a whole path, and present how we can prevent any inconsistent state, even temporarily. We also present a scheduling algorithm to minimize the amount of reconfiguration to make the maintenance energy efficient, and to minimize the impact of unreliability.

As future work, it would be interesting to enhance fault detection for quicker operation in various scenarios. Since the fault detection mechanism relies on report packets, the reporting time can affect the speed of fault detection. This concern becomes more significant in cases where the link between a node and its parent is severely degraded, and the node cannot transmit its report packet. Therefore, it is worth conducting a thorough investigation to address these concerns.

Moreover, it is worth investigating the network behavior in more advanced scenarios where several reconfigurations are needed simultaneously in the network. Typically, before starting each reconfiguration, the controller must prioritize the pending reconfiguration processes based on their dependencies and criticality. For instance, if a node and one of its predecessors in a subtree need reconfiguration, it is more reliable to handle the configuration of the predecessor node first, as it paves the way for reconfiguring the grandchild node. Additionally, different metrics such as convergence time and data flow distribution are interesting to measure.

Chapter 6

Conclusion and Future Research Directions

This thesis has explored critical flow management in scheduled IWSNs through the SDN paradigm. Since IWSN applications demand strong SLAs in end-to-end PDR and latency, we have leveraged SDN as a centralized management system to effectively control the packet forwarding and the scheduling of resources in the network. However, exploiting SDN in wireless networks faces its own challenges due to the lossy nature of wireless links. The following summarizes the key findings and contributions of this research.

Our first contribution (Chapter 6.2): As we cannot adapt the classical SDN protocols (e.g., OpenFlow) to wireless scheduled networks due to resource-constrained devices and wireless links, we have firstly developed a robust control plane communication for wireless scheduled networks. We configure each node with dedicated resources for upward and downward control directions, which eliminates collisions.

Unlike regular WSN applications with best-effort traffic, IWSN applications demand end-to-end resource provisioning to fulfill strict SLAs. We have proposed a call admission system that allows a device to submit its QoS requirements to the SDN controller, and in return, the controller defines proper forwarding rules and dedicates resources per each flow.

We advocate for the necessity of this centralized approach in meeting application SLAs, whereas an adaptive state of the art distributed protocol fails to achieve the same level of performance. In addition, we support the notion that resource allocation for flows should be based on the quality of network links, ensuring that sufficient resources are provisioned for each hop to meet the end-to-end SLAs. Furthermore, allocated resources for each flow must be isolated from other flows to

prevent inter-flow resource contentions.

Our simulation results illustrate that our solution can perfectly meet the SLA requirements of the critical applications, whereas state-of-the-art solutions fail to support them due to the lack of efficient schedule definition.

Our second contribution (Chapter 6.2): As the SDN controller relies on the link quality measurements to perform the scheduling, we proposed an accurate LQE approach for software-defined scheduled networks. This helps to avoid over-provisioning (resource wastage) or underprovisioning (SLA violations) of resources in the network. To be energy efficient, we exploit a passive technique by monitoring the existing synchronization beacons of the network to calculate the delivery ratio of links. We exploit the global knowledge of the SDN controller to manage the beacons transmissions, providing a collision-free communication and allowing to estimate the link quality accurately. Hence, while it is possible to use existing metrics to measure link quality, achieving high precision while maintaining energy efficiency remains challenging without leveraging the complete view of the controller.

Also, we evaluated the efficiency of different variants of the control plane (dedicated, shared) in terms of reliability and energy efficiency. We have demonstrated that a dedicated control plane provides high efficiency with only a slightly higher energy consumption cost. In addition, it can secure the reliability of higher control plane traffic if we aim to extend the control plane to support more advanced functionalities.

Our third contribution (Chapter 6.2): Due to the time-variant nature of wireless link qualities, we introduce an efficient network maintenance solution tailored for scheduled SDN networks. The controller continuously monitors changes in link quality within the network. Whenever a significant change is detected between a node and its next hop, it initiates the reconfiguration of both the control plane and all data plane flows for the target node, establishing a new path.

Given our reliance on scheduled networks, the controller is also responsible for reconfiguring schedules for all affected flows on the weak link, requiring a significant exchange of control traffic. To address this challenge effectively, we have proposed an efficient mechanism that minimizes both the volume and time required for reconfiguration without compromising the SLAs of critical flows. Therefore, the initial configuration of the SDN controller is insufficient, and there is a need for continuous network maintenance. Additionally, our solutions should be energy-efficient and compatible with constrained devices in IWSNs.

6.1 Short Term Research Direction

We propose several short term directions that can be directly taken from the work in this thesis.

6.1.1 Parent Selection Criteria

The strategy of selecting a neighbor node as the next hop to reach the controller can impact both the reliability and energy consumption of the network. However, attaching a large number of nodes in a node's subtree can significantly drain the battery and, consequently, affect the overall lifetime of the network. Even more challenging, to simplify data plane configuration and minimize overhead, the same routes used in the control plane may be configured for data flows. Thus, these factors emphasize the significance of carefully optimizing route selection strategies in wireless SDN networks. In general, when the route selection mechanism is conducted by the SDN controller, it becomes more straightforward to choose the best strategy as it has complete knowledge.

In our proposition, we discussed the construction of the control plane, where a routing parent is selected for a joining node by the controller. We utilize the most reliable link quality as the metric for selecting a parent node for the joining node. Nevertheless, since several neighbors of the joining node may offer a good link quality, we also can apply different metrics, such as load balancing or battery level, to select the parent node. In fact, the controller can choose the neighbor with the lowest number of scheduled cells or flows among the list of good neighbors as the parent. In this way, we can maximize the network lifetime while ensuring the reliability of the control plane by selecting the parent among good neighbors.

Our framework can be easily expanded to implement and evaluate this approach. The evaluation can demonstrate how extending the network's lifetime can be improved when applying a load balancing scheme.

6.1.2 Centralized Channel Blacklisting

An external transmitter may utilize the shared frequency spectrum, potentially causing interference with the operation of our nodes on certain channels. Channel blacklisting is a common approach to exclude interfering channels and improve network performance. Many solutions have been proposed for enhancing channel usage, both in centralized and distributed manners. In a distributed manner, pairs of nodes negotiate with each other to identify interfering channels and agree not to use them in their operations. Conversely, in a centralized approach, a central entity is responsible for identifying optimal channel usage based on the gathered channel quality information and then pushing the list of blacklisted channels to devices. Based on

the literature review, distributed solutions often fail to achieve optimal performance due to the high complexity of finding a collision-free list of channels and negotiation overhead [161]. On the other hand, centralized approaches are often considered impractical due to the lack of a mechanism to gather and push the list of interfering channels to the network [77].

Our SDN solution addressed the lack of communication mechanism between devices and the central controller, which can be exploited to efficiently optimize channel usage in the network. Thereby, different centralized channel blacklisting mechanisms can be integrated into the framework.

Also, we leverage the periodic transmissions of Enhanced Beacon packets in the IEEE 802.15.4-TSCH network to calculate the PDR for each link. Specifically, in each consecutive slotframe, a node utilizes different frequency channels to send its EBs, and we compute the PDR based on all these transmissions combined. However, having a per-channel frequency PDR allows the controller to implement channel blacklisting and maximize capacity usage.

Additionally, this information can be beneficial in situations where link quality degradation is detected, triggering network reconfiguration (as discussed in Chapter 6.2). If the link quality degradation is attributed to low PDR in a specific frequency channel colonized by an external interference, the controller can take a more efficient approach. Instead of redirecting the entire control and data plane to a new parent, the controller can block the list of interfering channels between the node and its parent, resulting in lower reconfiguration costs and overhead.

Our work is flexible enough to be extended for this proposition. It is necessary to adapt the frame formats to accommodate per channel PDR. The evaluation would be even more valuable if the experiments are carried out in the presence of an external interference source.

6.1.3 Exploration of Benchmark Scheduling Schemes

It would be interesting to incorporate a comprehensive analysis of the performance exhibited by different scheduling approaches within our SDN-TSCH solution. Many centralized scheduling approaches are proposed in the literature with specific goals. Incorporating diverse scheduling strategies into the SDN controller is a straightforward process. The defined schedule can be installed in the network using SDN-TSCH control plane. This analysis would encompass an evaluation of energy consumption patterns and fault tolerance capabilities.

Also, our architecture also naturally supports multipath, since a given flow-id may be associated with multiple TX cells, to different neighbors.

6.2 Long Term Research Direction

To efficiently organize, maintain, and optimize networking systems, increased intelligence is required. However, due to the inherently distributed nature of traditional networks, the application and deployment of Machine Learning (ML) techniques for network control and operations have proven challenging. SDN presents new opportunities for embedding intelligence within networks [162]. SDN's capabilities such as high computational power, logically centralized control, a global network view, software-based traffic analysis, and dynamic gathering the network statistics and updating of forwarding rules, facilitate the application of ML techniques. Relying on a proactive approach of ML equips the SDN controller with the capability to take preemptive actions. More specifically, the ML can be employed for the tasks of traffic classification, QoS prediction, resource management and security.

Traffic classification: it would be interesting to employ QoS-aware traffic classification to optimize network resource allocation based on their desired QoS parameters. Various metrics such as flow lifetime, delay sensitivity, reliability, and traffic load can be employed to categorize flows and assign priority levels. Initially, ML techniques are employed for traffic flow classification. Subsequently, the centralized SDN controller can leverage the classification results to configure the flows. This approach aids the SDN controller in effectively managing resources and optimizing the accommodation of critical flows within the network.

QoS prediction: SLA fulfillments are closely tied to various network Key Performance Indicators (KPIs), including packet size, link quality, hop distance, queue length, energy resources, etc. Uncovering the quantitative relationships between these KPIs and SLA parameters can enable QoS management while extending network lifetime. ML can forecast optimal network parameters, such as routing decisions and resource requirements, enabling the controller to execute efficient configurations.

Security: is a fundamental concern for network operators, and intrusion detection plays a critical role in protecting networks. An Intrusion Detection System (IDS) is either a device or software application with the primary goal of monitoring network events and identifying potential attacks [163]. IDS empowers network operators to take proactive measures to mitigate threats before they escalate. Anomaly-based IDS often leverages machine learning techniques to differentiate normal network activities from intrusions [164]. The capabilities of SDN facilitate ML-based intrusion detection, thereby fortifying network security [165]. The SDN controller's global network perspective simplifies the collection and analysis of network traffic, and SDN's programmability enables rapid responses to detected network attacks.

By addressing the communication barriers between devices and the SDN con-

troller, our deployments can be leveraged for extensive research on using ML potential with SDN controllers to optimize IWSN networks. In addition to our existing data, we can also incorporate extra statistical information via the reliable control plane of SDN-TSCH, enriching the controller's understanding of the network's state.

List of Abbreviations

ASN Absolute Sequence Number.
BLE Bluetooth Low Energy.
CoAP Constrained Application Protocol.
CSMA Carrier Sense Multiple Access.
CSMA/CA Carrier Sense Multiple Access with Collision Avoidance.
EB Enhanced Beacon.
IoT Internet of Things.
IWSN Industrial Wireless Sensor Network.
KPI Key Performance Indicator.
LoRaWAN Long Range Wide Area Network.
LQE Link Quality Estimation.
LR-WPAN Low-Rate Wireless Personal Area Networks.
MAC Medium Access Control.
ML Machine Learning.
MSF Minimal Scheduling Function.
PDR Packet Delivery Ratio.
PHY Physical Layer.
QoS Quality of Service.
SDN Software Defined Network.
SLA Service Level Agreement.
TDMA Time Division Multiple Access.
WSN Wireless Sensor Network.

List of Figures

List of Tables

Améliorer l'Internet industriel des objets grâce à la mise en réseau définie par logiciel : de la construction du réseau à la gestion des flux

Les réseaux de capteurs sans fil (WSNs) intègrent la technologie des capteurs à la communication sans fil, fournissant ainsi une plate-forme polyvalente pour des applications telles que la surveillance environnementale, les villes intelligentes et les soins de santé. Ils collectent de manière autonome les données des nœuds de capteurs et les transmettent sans fil à un hub central pour analyse [1].

Les avantages de la communication sans fil dans WSNs incluent la réduction des coûts, la flexibilité et la facilité de maintenance des nœuds. Cependant, ces réseaux sont confrontés à des défis dus aux limitations des ressources, qui influencent la conception du matériel, les protocoles économes en énergie et la gestion des données [5].

Les réseaux de capteurs sans fil industriels (IWSNs) répondent à des SLAs spécifiques aux applications industrielles de l'Industrie 4.0. Ils prennent en charge la maintenance prédictive, le contrôle qualité et l'efficacité opérationnelle grâce à l'analyse des données [9].

Dans les environnements industriels, la communication multi-sauts surmonte les obstacles et garantit une transmission fiable des données dans IWSNs. Classés selon les exigences de latence, les modèles de trafic incluent le trafic périodique et déclenché par des événements pour les tâches d'automatisation industrielle [13].

Garantir la fiabilité et la latence limitée dans IWSNs est un défi en raison des caractéristiques des liaisons sans fil. Des protocoles MAC déterministes et une planification minutieuse sont essentiels pour répondre à SLAs [20].

Le paradigme SDN dans IWSNs sépare les plans de contrôle et de données, offrant une gestion centralisée efficace du réseau. Les défis incluent des liaisons sans fil peu fiables, la découverte de topologies économes en énergie et la tolérance aux pannes pour la reconfiguration du réseau [24].

Le doctorat. la thèse aborde des questions clés :

Établir un plan de contrôle fiable sur les liaisons sans fil multi-sauts, avec perte et partagées. Activation de la planification centralisée pour les garanties de flux dans les applications IWSNs critiques. Fournir une découverte de topologie précise et économe en énergie et une estimation de la qualité des liens dans IWSNs défini par logiciel. Permettre une reconfiguration du réseau rentable et continue dans IWSNs défini par logiciel. Les contributions incluent une architecture SDN pour IWSNs axée sur les garanties de flux, l'estimation précise et économe en énergie de la qualité des liens et la reconfiguration continue du réseau.

État de l'art

Cette section fournit un aperçu des applications industrielles, en se concentrant sur le protocole IEEE 802.15.4-TSCH, présente le SDN dans le contexte de l'IWSN et explore les solutions de pointe.

Contexte IEEE 802.15.4-TSCH

IEEE 802.15.4-TSCH est un mode opérationnel conçu pour les applications industrielles de faible consommation. IEEE 802.15.4-TSCH combine l'accès multiple par répartition dans le temps (TDMA) avec un mécanisme de saut de fréquence pour fournir des communications fiables et économes en énergie. Un mécanisme de planification détermine quand un émetteur doit commencer sa transmission (créneau horaire) et quelle fréquence utiliser (décalage de canal). Les transmissions sont organisées en slotframe, c'est-à-dire une matrice de cellules sous forme de paires de tranches de temps et de décalages de canal. Chaque nœud possède dans son planning une liste de cellules TX (respectivement RX) pendant lesquelles il doit rester éveillé pour transmettre (respectivement recevoir) des paquets.

IEEE 802.15.4-TSCH exploite deux types de cellules dans le slotframe :

une cellule dédiée est allouée à un seul émetteur et un ou plusieurs récepteurs. Ainsi, l'émetteur peut engager sa transmission sans conflit ;

une cellule partagée est allouée à un ensemble d'émetteurs, ce qui entraîne des collisions potentielles. Si `ack` est attendu mais n'est pas reçu, l'émetteur attend un nombre aléatoire de cellules partagées pour retransmettre le paquet. Les collisions peuvent être fréquentes même avec une faible intensité de trafic [70].

Dans IEEE 802.15.4-TSCH, les nœuds se synchronisent à l'aide de balises améliorées (EB) périodiques. Le paquet s'appuie sur des éléments d'information (IE), qui sont un mécanisme extensible pour échanger des informations au niveau de la sous-couche MAC. En particulier, le numéro de séquence absolu (ASN) sert d'horloge globale dans le réseau. Puisque le timing est fixe (pas d'intervalle) dans un intervalle de temps, le récepteur peut ajuster son horloge après toute réception de paquet.

SDN pour la gestion de la planification dans les IWSN

L'intégration du SDN dans les WSNs émerge comme une stratégie prometteuse pour soulager les contraintes de traitement sur les dispositifs à faible puissance en centralisant le contrôle via un contrôleur dédié [104, 105]. Dans les WSNs distribués classiques, où les nœuds capteurs gèrent à la fois le traitement et les transmissions de données, les limitations en puissance de calcul posent des défis [106]. Les contrôleurs centralisés, dotés de capacités de calcul plus élevées, exécutent efficacement des tâches computationnelles intensives, allégeant la charge sur les nœuds capteurs [107].

Les applications des réseaux de capteurs sans fil industriels (IWSN) reposent fréquemment sur des réseaux sans fil déterministes planifiés pour atteindre une haute fiabilité et une faible latence. En conséquence, il devient impératif d'adapter l'architecture SDN pour allouer des blocs temps-fréquence [121].

uSDN [122] propose une architecture SDN pour les réseaux TSCH, mettant l'accent sur l'optimisation des frais généraux de signalisation lors de l'installation des flux. En installant des chemins au niveau du nœud source, le contrôleur atténue les installations individuelles de règles de flux hop-by-hop, réduisant les frais généraux initiaux de signalisation. Cependant, des préoccupations surgissent quant à la rentabilité du piggybacking du routage source, et la dépendance à des algorithmes autonomes et à RPL, connu pour ses oscillations [123]. uSDN étend son travail [122] pour séparer les ressources radio entre le trafic de contrôle et de données, utilisant un mécanisme de planification hop-by-hop pour la réservation distribuée de cellules vers le contrôleur. Notamment, le contrôleur définit uniquement des règles de transmission pour chaque flux, sans gestion de la planification du plan de données.

Whisper [124] utilise un contrôleur centralisé pour gouverner les protocoles RPL et 6P, aspirant à établir un système de gestion centralisé avec des modifications minimales de la pile réseau. Cependant, Whisper manque de capacités de routage et de planification par flux, le rendant mieux adapté aux scénarios de trafic best-effort.

REACT [129] introduit une politique induite par des écarts pour réduire les coûts de latence et d'énergie lors de la reconfiguration du réseau dans les réseaux WirelessHART. Bien qu'offrant une flexibilité en évitant une reconfiguration complète,

des défis surgissent de modifications individuelles de paquets (commandes DELETE et ADD), potentiellement causant des perturbations en raison des changements de qualité de liaison.

SDSense [130] présente un cadre reconfigurable dynamique avec une architecture de contrôleur hiérarchique, optimisant les composants du réseau pour les besoins lents et rapides. Malgré des améliorations significatives des performances dans la reconfiguration dynamique du réseau, le manque de mise en œuvre détaillée entrave une comparaison directe avec d'autres solutions SDN.

SDNWISE-TSCH [131] étend SDN-WISE pour les réseaux TSCH planifiés, visant une routage et une planification efficaces par le contrôleur SDN. Tout en priorisant les flux selon des critères spécifiques, SDNWISE-TSCH néglige la considération de la qualité réelle des liaisons dans la définition des itinéraires et des horaires, impactant la fiabilité. De plus, l'absence d'un mécanisme d'isolation des flux et la dépendance à des cellules partagées dans le plan de contrôle posent des défis potentiels.

En conclusion, la revue de littérature met en évidence une lacune dans les solutions SDN pour la garantie par flux dans les réseaux planifiés et l'allocation de ressources basée sur la fiabilité des liaisons. Les solutions SDN résumées et leurs propriétés sont présentées dans le Tableau 2.2.

Découverte de topologie et estimation de la qualité des liaisons dans les réseaux SDN sans fil

La découverte de la topologie dans les réseaux sans fil identifie systématiquement les nœuds voisins et les chemins de communication, constituant une base cruciale pour la planification, l'optimisation et la gestion du réseau [133, 134]. Elle peut être réalisée de manière active ou passive.

Dans l'approche active, chaque nœud génère des paquets de sonde supplémentaires pour annoncer sa présence aux voisins et potentiellement recevoir des réponses en retour. Bien que cette méthode permette une identification active des connexions entre les nœuds voisins, elle peut entraîner une surcharge due au trafic de sondage supplémentaire, affectant potentiellement la consommation d'énergie globale du réseau.

À l'inverse, l'approche passive exploite le trafic réseau existant. Les nœuds surveillent en continu les données entrantes pour identifier leurs voisins. Cette méthode évite la surcharge des paquets de sondage mais repose sur l'hypothèse d'un trafic constant pour recueillir suffisamment d'informations.

Dans les réseaux SDN sans fil, la découverte de la topologie utilise des techniques supplémentaires telles que l'analyse des signaux sans fil, la découverte des voisins et la surveillance des points d'accès en raison de la nature dynamique des liaisons sans fil. Contrairement aux réseaux câblés offrant des connexions constantes et

prévisibles, les réseaux sans fil font face à des défis liés à la mobilité, aux interférences et à l'ajout/suppression dynamique de nœuds sans fil.

SDN-WISE [111] et SDNWISE-TSCH [131] utilisent des paquets de balises supplémentaires pour la découverte de la topologie. Chaque nœud envoie périodiquement des paquets de balises pour annoncer son nombre de sauts au nœud puits. Les nœuds mettent à jour leurs tables de voisins et envoient ces informations au contrôleur. TinySDN [112] utilise des paquets de sondage où chaque nœud envoie un paquet de sondage à ses voisins. Plusieurs solutions [117, 122, 124] exploitent la liste des voisins RPL au lieu de la découverte de la topologie, bien que RPL puisse présenter des incohérences dans sa table de routage.

L'évaluation de la qualité de liaison (LQE) dans les réseaux sans fil est le processus technique d'évaluation de la fiabilité et des performances des liaisons de communication entre les appareils sans fil. Elle est cruciale pour optimiser les performances du réseau, le routage efficace et la fourniture de la qualité de service (QoS).

Les métriques matérielles telles que LQI (***Link Quality Indicator***), RSSI (***Received Signal Strength Indicator***), et SNR (***Signal-to-Noise Ratio***) fournissent des informations sur la qualité de la liaison. Cependant, elles ont des limitations. Les métriques basées sur le matériel mesurent uniquement les paquets reçus avec succès, ce qui peut entraîner une surestimation de la qualité de la liaison en cas de pertes excessives de paquets. D'autre part, les métriques basées sur le logiciel offrent un aperçu plus précis du comportement de la liaison mais peuvent introduire des coûts de traitement supplémentaires.

Le choix d'une métrique de qualité de liaison dans les réseaux sans fil doit être aligné sur les objectifs de l'application, tels que la transmission fiable des données, l'efficacité énergétique, la faible latence ou la scalabilité.

La mesure de la qualité de liaison est cruciale pour la prise de décision du contrôleur SDN. Les techniques actives impliquent l'envoi de paquets supplémentaires, nécessitant des ressources radio, tandis que les techniques passives utilisent les paquets existants mais peuvent manquer de précision en cas de trafic déséquilibré.

Le contrôleur SDN collecte les mesures de qualité de liaison de chaque nœud, évaluant les métriques comme RSSI. Cependant, un RSSI élevé ne garantit pas nécessairement un taux de réception de paquets élevé. D'autres solutions utilisent des paquets de sondage ou exploitent les protocoles de routage comme RPL. Cependant, l'envoi de paquets de sondage peut augmenter la surcharge de communication. La précision des protocoles comme RPL peut être limitée par des incohérences dans leurs tables de routage.

Les technologies planifiées telles que IEEE 802.15.4-TSCH dans les IWSNs manquent d'une planification centralisée pour garantir les flux. Les solutions SDN existantes se concentrent sur les règles réseau, négligeant les plans de contrôle robustes

dans les liens sans fil. Notre travail comble cette lacune en mettant l'accent sur une planification consciente de la qualité des liens, comparée à MSF et SDNWISE-TSCH pour l'évaluation des performances.

SDN-TSCH : Permettre le SDN pour les IWSN avec Isolation du Trafic

L'Industrie 4.0 utilise les IWSN pour automatiser les processus industriels [150], avec des applications critiques exigeant un fort QoS en termes de fiabilité et de latence bornée. IEEE 802.15.4-TSCH [26] est une solution privilégiée pour les réseaux industriels, assurant une communication robuste via une couche MAC TDMA et un saut de fréquence. Cependant, pour respecter les SLA, il est crucial d'allouer suffisamment de blocs temps-fréquence à chaque flux.

SDN semble prometteur pour une planification centralisée dans les IWSN, mais les solutions existantes présentent des lacunes, notamment en termes de collisions et de garanties de flux dans IWSN. Notre proposition, SDN-TSCH, vise à combler ces lacunes en offrant une architecture SDN dans les IWSN, avec une planification centralisée, une gestion efficace des ressources radio, et une isolation de flux pour des performances stables.

SDN-TSCH fonctionne avec IEEE 802.15.4-TSCH, séparant les plans de contrôle et de données avec des ressources radio dédiées. Le contrôleur SDN assure un plan de contrôle sans collision et alloue des ressources dans le plan de données pour répondre aux exigences de fiabilité et de latence. Le transfert de paquets utilise une approche d'étiquetage pour isoler les ressources par flux, facilitant la sélection efficace des ressources radio.

Avec un processus de découverte efficace, les nouveaux nœuds notifient leur présence au contrôleur SDN, qui configure ensuite les plans de contrôle et de données. Chaque nœud peut demander l'admission de son flux critique, et si les ressources sont disponibles, le contrôleur orchestre une configuration de bout en bout pour le flux.

Notre solution SDN-TSCH se distingue par sa capacité à assurer des garanties de flux dans les IWSN, surpassant les limitations des solutions existantes, et sert de base pour des applications industrielles fiables et à faible latence.

Commutation d'étiquettes pour SDN

SDN-TSCH utilise la commutation d'étiquettes pour isoler les flux dans le réseau [151]. Chaque flux possède des ressources radio dédiées, marquées avec un identifiant de flux (flow-id), garantissant une transmission exclusive. Le flow-id est ajouté à l'entête des paquets, permettant une gestion efficace des flux. La figure 1 illustre cette

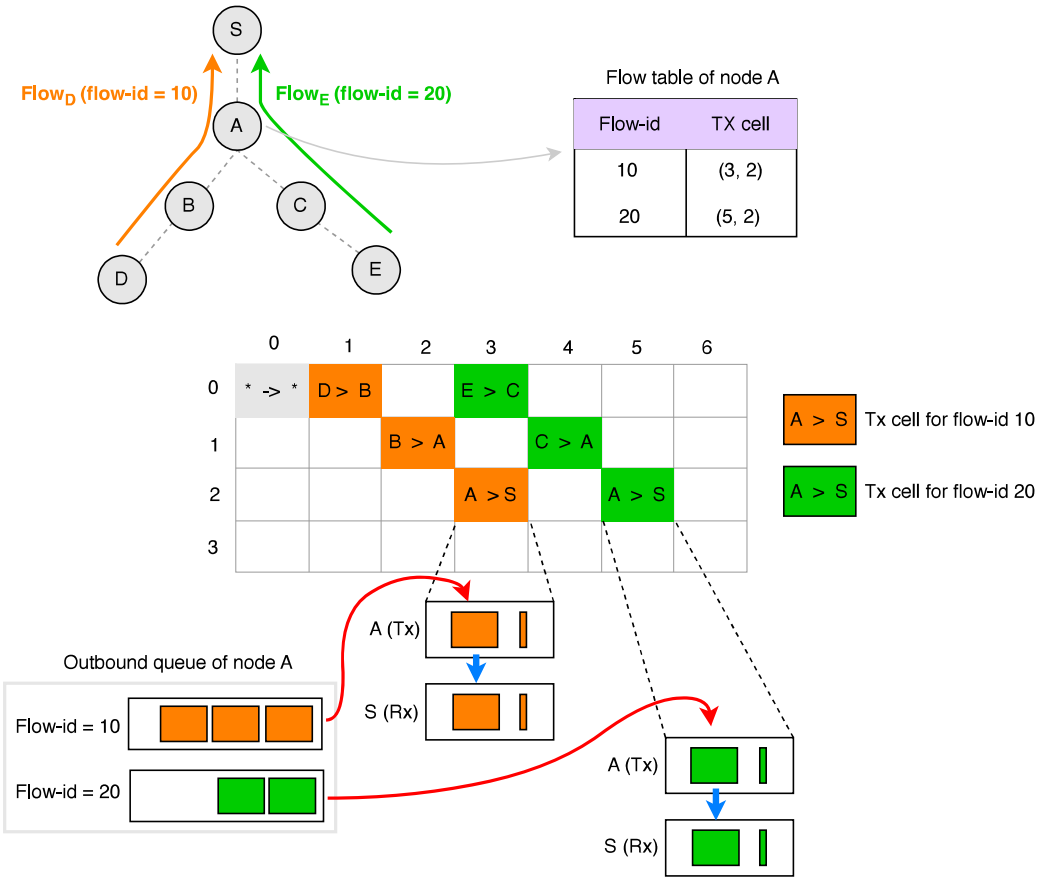

Figure 1: Label switching operation in SDN-TSCH

opération.

Deux flow-id sont définis pour le plan de contrôle : "to-controller" pour les paquets montants et "from-controller" pour les paquets descendants. Cette approche améliore l'efficacité du traitement des paquets, assure l'isolation des flux et alloue des ressources en fonction des exigences de qualité de service (QoS).

Processus de Découverte

Le processus de découverte permet à un nouveau nœud d'identifier ses voisins et d'estimer leurs qualités de lien, facilitant ainsi une configuration efficace du plan de contrôle par le contrôleur.

L'estimation de la qualité des liens se fait passivement en recevant des EBs pendant des créneaux spécifiques. Notre approche utilise un calendrier EB sans collision pour déterminer le PDR de chaque lien en comptant simplement les EBs reçus de chaque voisin dans une période définie.

Le processus commence lorsque le nouveau nœud reçoit un EB, synchronise son

horloge avec le réseau TSCH, extrait les IEs pour obtenir les paramètres TSCH, et découvre ses voisins en écoutant sur les créneaux partagés.

La découverte s'arrête lorsque tous les bons voisins (avec une qualité de lien supérieure à un seuil prédéfini) ont été détectés pendant au moins une période T_{report}. Le nouveau nœud ajuste également sa source de temps au voisin présentant la meilleure qualité de lien pour minimiser la désynchronisation. Les PDR calculés sont ensuite continuellement transmis au contrôleur toutes les T_{report} une fois le processus de découverte terminé.

Processus d'Adhésion

Une fois qu'un nœud nouvellement découvert a effectué le processus de découverte, il doit rejoindre le réseau SDN. À cet effet, il crée un paquet `report` contenant la liste des voisins et leur PDR associé, puis l'envoie en unicast au voisin présentant la meilleure qualité de lien pour maximiser la probabilité de transmission réussie. Comme le nœud n'a pas de plan de contrôle configuré, il doit utiliser une cellule partagée pour transmettre son paquet `report`. Le paquet `report` traverse ensuite le réseau de nœud en nœud pour atteindre le contrôleur de manière fiable en utilisant l'identifiant de flux ***"to-controller"***.

À la réception d'un paquet `report`, le contrôleur vérifie si le paquet provient d'un nouveau nœud. Dans ce cas, le contrôleur enregistre le nœud dans la liste des membres et sélectionne le voisin présentant la meilleure qualité de lien comme parent pour maximiser la fiabilité du plan de contrôle. Le parent est également utilisé comme source de temps dans TSCH après le processus d'adhésion, minimisant ainsi la probabilité de désynchronisation en raison de la forte qualité de lien.

Ensuite, le contrôleur sélectionne au hasard deux cellules dédiées dans le slotframe pour le plan de contrôle : une pour l'envoi de paquets de contrôle ascendante au nœud parent et une pour la réception de paquets de contrôle descendant du nœud parent. Les cellules allouées doivent correspondre à un créneau non utilisé pour le parent (condition de demi-duplex) et ne peuvent pas être attribuées à un autre lien interférant (condition sans collision).

Enfin, le contrôleur construit deux paquets `config`: un pour la direction ascendante (***to-controller***) et un pour la direction descendante (***from-controller***). Pour atteindre le nœud rejoignant, le contrôleur peut utiliser l'identifiant de flux ***from-controller*** déjà configuré dans le reste du réseau (sauf le dernier saut). Cependant, plusieurs enfants peuvent exister à chaque saut, et les paquets `config` doivent être routés dans le sous-arbre correct. Heureusement, le contrôleur connaît la topologie complète et peut calculer un chemin jusqu'au nœud rejoignant. Ainsi, nous implémentons une méthode de routage source : le chemin complet est intégré dans chaque paquet `config`.

Il est à noter que nous configurons toujours un flux du destinataire vers la source. Cette approche permet un schéma générique qui facilite la configuration des flux de données de manière fiable.

Lorsqu'un nœud n reçoit le paquet `config`, il exploite le chemin et l'ordonnancement pour mettre à jour sa configuration.

Allocation des Ressources pour le Plan de Données

Chaque application critique ouvre une connexion de socket, décrivant ses exigences en matière de QoS (*c.-à-d.*, fiabilité minimale de bout en bout et latence maximale). Le nœud initie une admission d'appel en envoyant une `flow-request` au contrôleur via l'identifiant de flux ***"to-controller"***. À la réception d'une `flow-request`, le contrôleur élabore un emploi du temps respectant les exigences de la QoS :

1. le contrôleur calcule un chemin de la source à la destination en utilisant la topologie en arbre (via les liens nœud/parent);
2. le contrôleur attribue des cellules dédiées de secours pour les retransmissions des liens les plus faibles. Des cellules de secours supplémentaires sont assignées jusqu'à ce que la fiabilité minimale de bout en bout soit respectée [152];
3. le contrôleur planifie des cellules dos à dos (ou aussi proches que possible) pour minimiser le délai de bout en bout.

Si le calcul de l'emploi du temps est impossible, la demande est rejetée, et le contrôleur envoie un paquet `config` négatif (vide) au nœud source. Si le calcul est réussi, le contrôleur définit un nouvel identifiant de flux pour le flux et construit un paquet `config` comprenant le nouvel emploi du temps et l'identifiant de flux correspondant. Le contrôleur utilise un seul paquet `config` pour configurer tout le chemin, qui se compose de :

1. sous-chemin de la racine à la destination : l'emploi du temps de cette partie est vide, et le nœud ne fait que transmettre le paquet `config` avec les cellules ***from-controller***;
2. sous-chemin de la destination à la source : la configuration commence du nœud de destination vers le nœud source. Chaque nœud dans cette partie extrait et installe l'identifiant de flux et les cellules correspondantes. Lorsque le nœud source reçoit le paquet `config`, tout le chemin est configuré, et il commence à envoyer des paquets sans délai.

Il est à noter que la technique pour identifier la position d'un nœud de transmission est toujours valable lorsqu'un nœud est présent dans les deux sous-chemins.

En effet, nous identifions individuellement chaque **lien** dans tout le chemin lors de l'installation de l'emploi du temps.

Pour la deuxième partie du chemin, chaque nœud doit sélectionner soit les identifiants de flux ***"to-controller"*** ou ***"from-controller"*** pour transmettre le paquet `config`. Pour déterminer la direction, chaque nœud vérifie si l'adresse du prochain saut est également présente dans la liste des nœuds précédant sa position dans le routage source. Si c'est le cas, le prochain saut est un nœud amont, et le paquet `config` utilise l'identifiant de flux ***"to-controller"***. Sinon, le nœud utilise l'identifiant de flux ***"from-controller"*** pour transmettre le paquet `config`.

La Figure 2 illustre un scénario où S est le nœud source. Le paquet `flow-request` décrit les exigences du flux (*p.ex.*, PDR = 90%, délai de bout en bout = 70 ms) et la destination D. En retour, le paquet `config` utilise le routage source, d'abord de A à D, puis de D à S. Le contrôleur attribue plus de cellules aux liens faibles (2 cellules pour S→E et 2 cellules pour E→B) pour atteindre le PDR de bout en bout du flux (90%). Le contrôleur attribue des cellules séquentiellement pour tenir compte du délai de bout en bout du flux (70 ms). Le chemin $A-B-D-B-E-S$ est intégré dans le paquet `config`. Ainsi, D sait que B est un nœud amont puisque B est à la fois présent après et avant D : il doit utiliser l'identifiant de flux ***"to-controller"*** pour transmettre le paquet `config`. Inversement, S est un nœud descendant pour E puisque S est uniquement présent après E : il doit utiliser l'identifiant de flux ***"from-controller"***.

Évaluation des performances

Nous avons implémenté SDN-TSCH dans Contiki-ng et le simulateur Cooja pour évaluer ses performances, le comparant avec deux approches de pointe :

MSF [91] est la norme de l'IETF pour la planification distribuée[4], combinant des cellules autonomes et négociées pour éviter les collisions et allouer dynamiquement des ressources ;

SDNWISE-TSCH [131] est une variante de SDNWISE adaptée aux réseaux TSCH. SDNWISE-TSCH permet l'architecture SDN pour les réseaux industriels de capteurs sans fil. Il calcule un programme en tenant compte de la date limite de chaque flux.

Nous avons simulé des réseaux de 10, 20, 30, 40 et 50 nœuds. Chaque nœud a un flux de données critique à transmettre au nœud collecteur (trafic de convergence). Nous considérons des applications critiques nécessitant un PDR de bout en bout supérieur à 99% et avec une date limite de 2 secondes [153]. Nous utilisons les

[4] `https://github.com/alexrayne/contiki-ng.git`

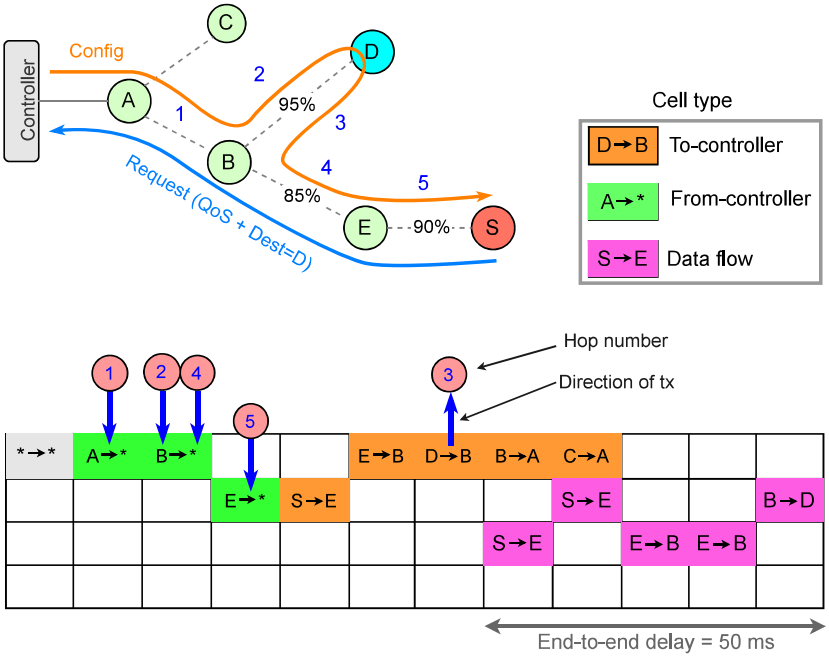

Figure 2: Data flow Configuration in SDN-TSCH

valeurs par défaut pour les paramètres de MSF [91]. Un contrôleur alloue un nouvel identifiant de flux et un ensemble de cellules à chaque flux critique pour répondre aux exigences.

Comparaison de SDN-TSCH et SDNWISE-TSCH

Nous comparons SDN-TSCH avec SDNWISE-TSCH pour évaluer deux approches SDN différentes. Plus précisément, nous i) évaluons la fiabilité et l'efficacité énergétique du plan de contrôle, et ii) évaluons les performances des garanties de flux dans le plan de données.

SDNWISE-TSCH utilise des cellules partagées pour les EB et les paquets de contrôle SDN. Selon [131], un réseau de 10 nœuds avec une longueur de slotframe de 19 utilise 2 cellules partagées. Pour accommoder différentes tailles de réseau sans impact sur la fiabilité du plan de contrôle de SDNWISE-TSCH, nous proposons de maintenir constant le temps par nœud entre deux cellules partagées. Ainsi, nous définissons le nombre par défaut de cellules partagées comme suit :

$$N_{shared-cells} = \frac{2 * SF_{length}}{19} * \frac{N}{10} \approx N * SF_{length} * 0.01 \quad (1)$$

avec $N_{shared-cells}$ le nombre de cellules partagées, N le nombre de nœuds, et SF_{length} la longueur de la slotframe.

La Figure 3 montre le temps de convergence de chaque approche pour différentes

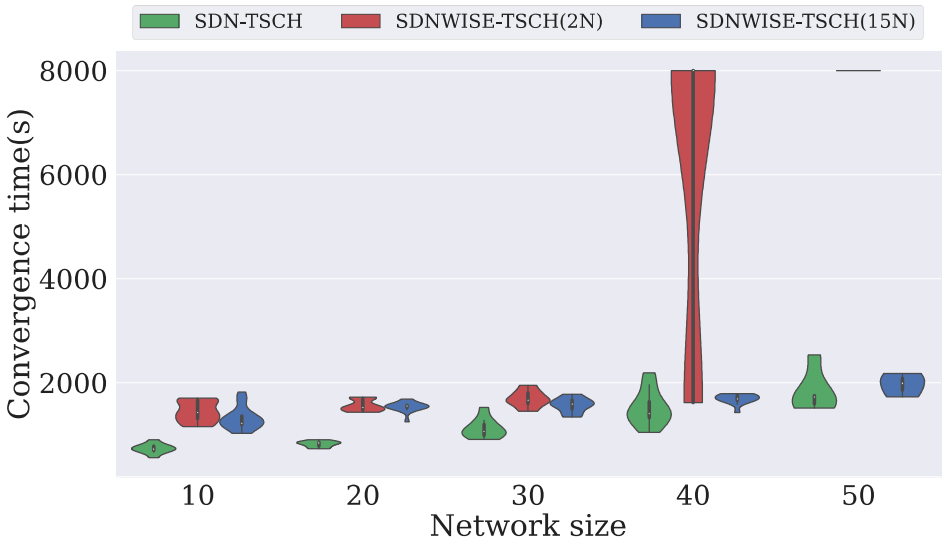

Figure 3: Temps de convergence du réseau

tailles de réseau. Nous définissons la convergence comme le moment où le dernier nœud du réseau est admis par le contrôleur SDN. Nous testons deux ratios de cellules partagées dans le plan de contrôle partagé : i) $15N$ (éq. 1) utilisé par SDNWISE-TSCH et ii) $2N$, qui sert de valeur limite inférieure. Avec 2N cellules partagées, le nombre de collisions devient très élevé pour les réseaux de taille moyenne. Dans les pires conditions, le réseau ne converge jamais. Avec 15 cellules partagées par slotframe, SDNWISE-TSCH réussit à converger. Cependant, le temps de convergence est encore plus long pour SDNWISE-TSCH comparé à SDN-TSCH. En effet, les cellules partagées reçoivent un pic de trafic de contrôle, et la perte de paquets de contrôle impacte significativement la convergence.

L'utilisation de cellules partagées impacte également la consommation d'énergie, comme illustré dans la Figure 4. Nous nous concentrons ici sur la consommation d'énergie du réseau pendant la période de convergence. L'utilisation de seulement 15N cellules partagées consomme beaucoup d'énergie : tous les nœuds se réveillent pendant ces créneaux. Au contraire, l'utilisation de seulement 2N cellules partagées est beaucoup plus économe en énergie (mais avec un impact sur la fiabilité). Seul SDN-TSCH est capable de converger rapidement tout en offrant une consommation d'énergie très raisonnable : l'utilisation de cellules dédiées est beaucoup plus efficace, même pour le trafic de contrôle. En effet, les nœuds doivent se réveiller moins fréquemment, et les transmissions sont plus fiables car nous ne pouvons pas créer de collisions.

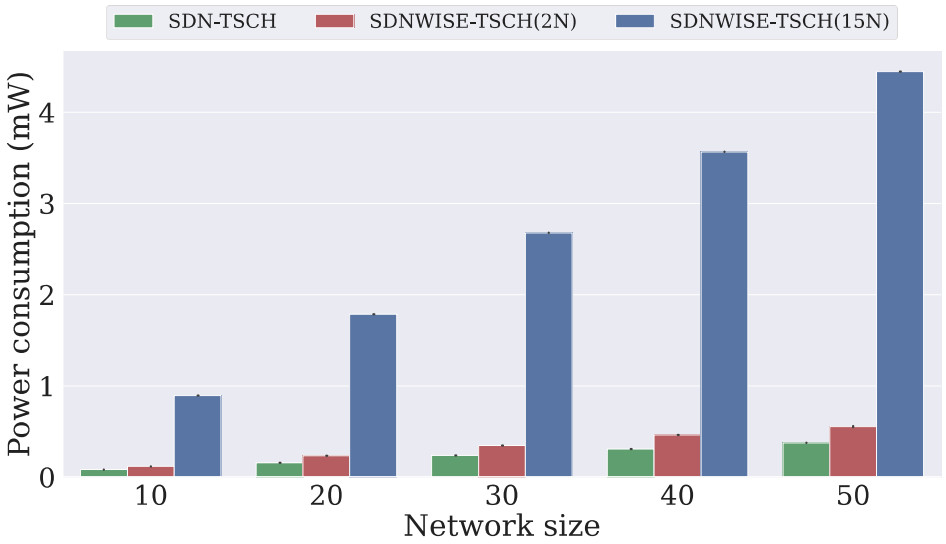

Figure 4: Consommation d'énergie des nœuds pendant la période de convergence

Comparaison de SDN-TSCH et MSF

Nous comparons SDN-TSCH avec MSF pour évaluer les différences entre une solution centralisée et une solution distribuée en termes de fiabilité et d'efficacité énergétique. La Figure 5 illustre le taux de remise de bout en bout de chaque flux. De manière évidente, MSF offre une fiabilité très faible. Avec 20 nœuds, le PDR moyen de bout en bout est supérieur à 95%, mais certains flux présentent un PDR de seulement 50%. La fiabilité est encore pire avec 50 nœuds : plus de flux sont acheminés, et la région autour du puits devient un goulot d'étranglement. Ainsi, MSF échoue à fournir suffisamment de cellules de secours pour garantir une fiabilité élevée. Au contraire, SDN-TSCH atteint une fiabilité de bout en bout parfaite, quelles que soient les conditions. Mieux encore : la fiabilité est égale à 100% même dans le pire des cas, ce qui est une propriété attendue pour les réseaux industriels.

La Figure 6 illustre la latence de bout en bout. Nous traçons la date limite de 2 secondes (ligne horizontale) pour en voir l'impact. MSF a tendance à livrer les paquets très près de la date limite, : il est très compliqué de respecter les délais de manière distribuée. mais trop de paquets sont reçus après la date limite, en particulier pour les grands réseaux. Au contraire, le contrôleur SDN provisionne suffisamment de ressources dans SDN-TSCH. Bien que la latence augmente car les paquets sont acheminés via des chemins plus longs, la date limite est toujours respectée. Les pertes de paquets correspondent principalement à des valeurs aberrantes statistiques.

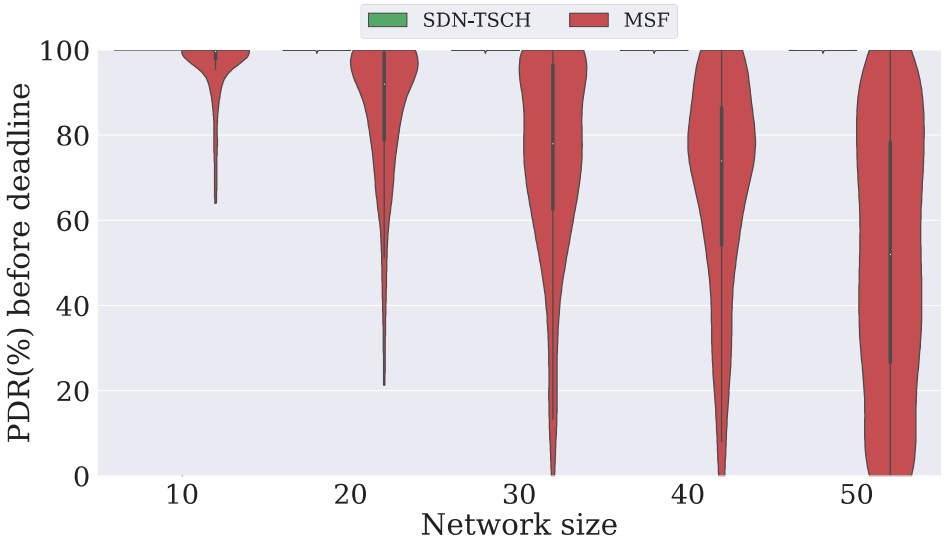

Figure 5: Taux de remise de paquets (PDR) des flux avant la date limite

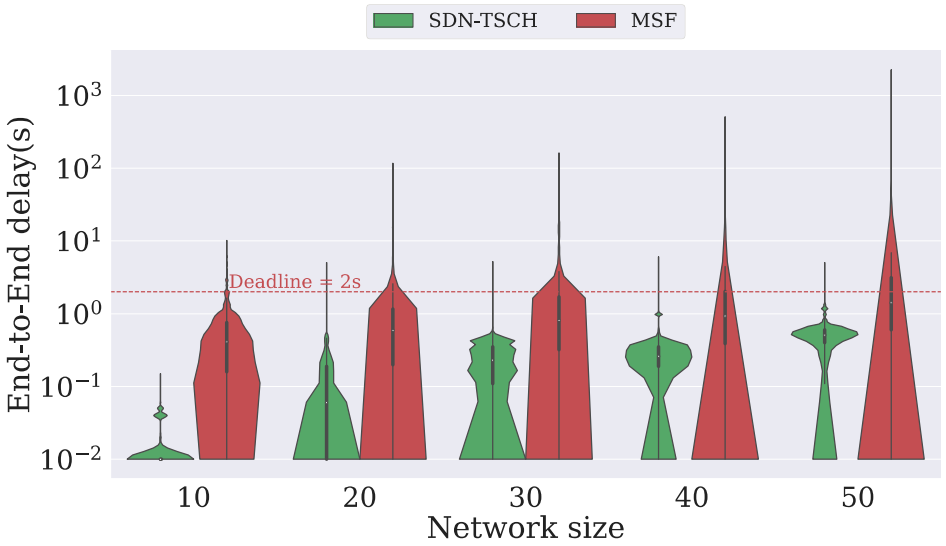

Figure 6: Latence de bout en bout

Estimation de la Qualité de Liaison dans les Réseaux SDN sans Fil Planifiés

Dans les réseaux sujets aux pertes, le contrôleur SDN peut allouer des ressources radio supplémentaires pour les retransmissions afin de compenser la perte de paquets. Cependant, le contrôleur peut estimer de manière incorrecte les qualités de liaison, entraînant une surprovision (gaspillage d'énergie) ou une sous-provision (manque de fiabilité). Les techniques actives nécessitent des sondes régulières, tandis que les passives sont moins coûteuses mais peuvent être imprécises en cas de collisions.

Notre estimation de la qualité de liaison dans SDN-TSCH repose sur les transmissions EB dans les cellules partagées, permettant l'identification de tous les voisins. Ce chapitre détaille notre solution, expliquant comment nous évitons les collisions pour une estimation précise et économe en énergie.

Nous avons utilisé un plan de contrôle entièrement dédié pour SDN-TSCH, mais cette approche s'est avérée énergivore. Nous évaluons les performances des plans de contrôle partagés, dédiés et mixtes dans SDN-TSCH, tenant compte des conditions réalistes telles que les liaisons sujettes aux pertes.

Cette évaluation vise à quantifier l'impact sur le réseau SDN, une question peu explorée dans les réseaux sans fil planifiés, où les dispositifs sont énergétiquement contraints. Ce chapitre offre une compréhension approfondie des compromis entre l'efficacité énergétique et la fiabilité dans un environnement de réseau sans fil planifié.

Estimation Précise de la Qualité de Liaison

Nous proposons un schéma passif de surveillance de liaison pour répondre aux exigences énoncées dans la section précédente. Nous tirons parti des paquets EB du réseau TSCH pour estimer la qualité des liaisons. Chaque nœud compte le nombre de EB reçus de chaque voisin pendant une période donnée. Le contrôleur calcule le taux de remise des paquets EB (PDR) et l'utilise comme métrique de qualité de liaison, calculée selon l'équation 2 :

$$\widehat{PDR}(n) = \frac{counter(n) * T_{EB}}{T_{report}} \tag{2}$$

Avec $counter(n)$ le compteur EB du voisin n, T_{EB} la période des EB, $\widehat{PDR}(n)$ le PDR mesuré pour n, et T_{report} la période de rapport.

Comme les EB sont envoyés périodiquement, une estimation continue de la qualité de la liaison est écoénergétique et élimine le besoin de générer de nouveaux paquets de sondage. Cela fournit au contrôleur une estimation constamment mise à jour de la qualité de la liaison. Ainsi, en cas de changement, le contrôleur peut reconfigurer le réseau en définissant de nouvelles règles.

De plus, comme tous les voisins d'un nœud doivent être éveillés pour recevoir un EB du nœud, nous utilisons des cellules partagées pour les transmissions EB. Cela permet de calculer et de mettre à jour la qualité de la liaison pour des liaisons non utilisées ou non découvertes.

Cependant, des collisions peuvent survenir entre deux transmissions EB concurrentes ou entre les EB et d'autres paquets diffusés via des cellules partagées. Nous proposons de résoudre ce problème de mauvaise estimation de la qualité de liaison en planifiant de manière plus appropriée les EB. Nous continuons à exploiter les cellules partagées pour optimiser le processus de découverte et la re-estimation continue de la qualité de la liaison, tout en évitant les collisions.

Organisation des Cellules Partagées dans le Plan de Contrôle

Dans le plan de contrôle, divers paquets, notamment ceux de l'adhésion et de la configuration, nécessitent des cellules partagées. Les nouveaux nœuds, dépourvus de cellules dédiées pour les paquets `report`, requièrent des cellules partagées pour les paquets de maintien de connexion afin de se synchroniser avec le parent.

Pour une découverte efficace du réseau, les EBs utilisent également des cellules partagées. Pour éviter les collisions d'EB, nous proposons que le contrôleur SDN attribue des cellules spécifiques pour les EBs, chacune étant assignée à un seul nœud. Une distinction claire est établie entre les cellules partagées pour les transmissions d'EB et celles pour le trafic de contrôle non-EB.

Chaque nœud vérifie les cellules partagées au début : 1. S'il s'agit de la cellule EB dédiée, le nœud transmet son EB. 2. S'il s'agit de la cellule EB d'un voisin, le nœud reste éveillé pour éventuellement recevoir l'EB. 3. Si ce n'est pas une cellule EB, le nœud transmet le premier paquet de contrôle ou reste éveillé pour recevoir des sollicitations.

Les nœuds utilisent des identifiants uniques pour les cellules partagées dans le slotframe. Lors de l'adhésion, un contrôleur attribue séquentiellement un identifiant partagé à un nouveau nœud. Le dernier nœud rejoint a le maximum d'identifiant de cellule partagée ($slot_{EBmax}$). Cette distinction aide à séparer les parties EB et non-EB du slotframe, et $slot_{EBmax}$ est diffusé dans les EB, informant tous les nœuds des cellules réservées aux EB.

Certaines cellules partagées sont allouées au trafic non-EB, et les cellules pour les EB sont séparées. Pour éviter les collisions avec les paquets non-EB, les nœuds n'utilisent que les cellules partagées non réservées pour les EB en fonction de l'identifiant partagé.

La distribution des cellules partagées non-EB impacte les taux de collision. Des cellules non-EB regroupées au début entraînent des collisions élevées, car les nœuds attendent la fin pour envoyer des paquets, provoquant une collision cumulative.

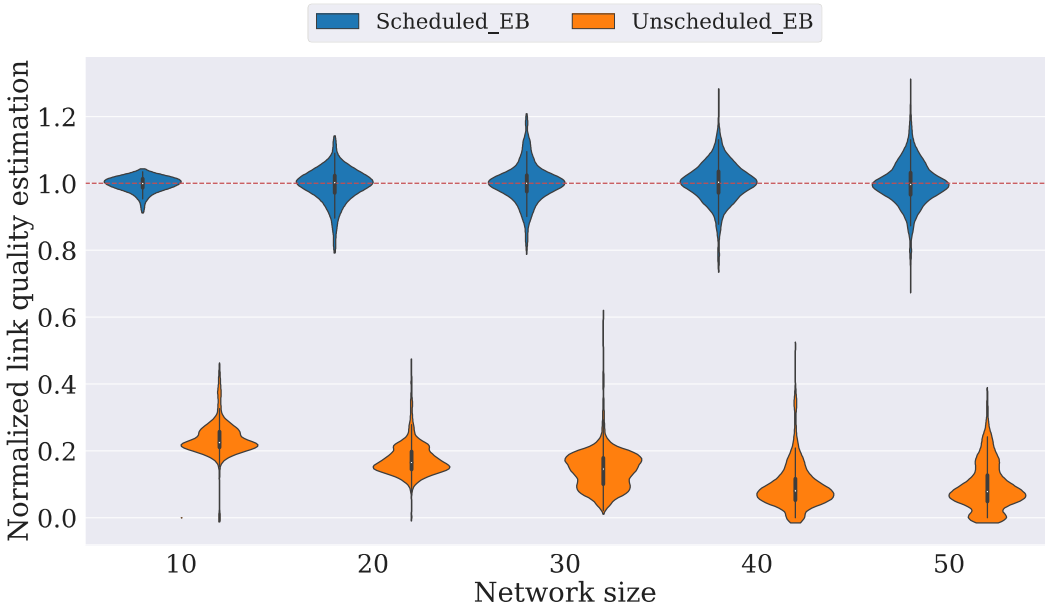

Figure 7: Estimation de la qualité de liaison

Pour minimiser les collisions, nous distribuons les cellules EB dans le slotframe, en attribuant des identifiants uniques (***shared-id***). Un algorithme récursif attribue les ***shared-id***s de manière équitable, minimisant la probabilité de collision. Les nœuds peuvent appliquer le même algorithme pour identifier les cellules EB et non-EB en utilisant $slot_{EBmax}$.

Évaluation des Performances

Nous évaluons la précision de notre estimation de la qualité de liaison en comparant notre approche basée sur les EB planifiés avec l'approche sans planification des EB. Dans l'approche sans planification des EB, les cellules EB ne sont pas planifiées par le contrôleur, permettant à chaque nœud d'utiliser n'importe quelle cellule partagée pour transmettre ses EB. L'évaluation se concentre sur la qualité de liaison normalisée, calculée comme le rapport entre le taux de livraison de paquets estimé (PDR) et le PDR réel modélisé par le simulateur (Figure 7).

En utilisant un nombre égal de cellules partagées pour les solutions EB planifiées et non planifiées, l'approche EB planifiée démontre une estimation précise de la qualité de liaison, indépendamment de la taille du réseau. Les erreurs d'estimation semblent suivre une distribution normale centrée sur la valeur réelle. En revanche, l'approche EB non planifiée sous-estime considérablement la qualité, surtout pour les réseaux plus importants. La qualité normalisée tend vers zéro, indiquant une sous-estimation importante entraînant une allocation inefficace des ressources et

une augmentation de la consommation d'énergie par le contrôleur.

Notre solution basée sur la planification se révèle très efficace, fournissant des estimations précises de PDR rapportées dans les paquets `report`. Cela permet au contrôleur d'optimiser efficacement l'allocation des ressources radio.

Maintenance des IWSN Définis par Logiciel

Le SDN gère efficacement les paramètres des périphériques sans fil, mais la qualité des liens peut changer dynamiquement, affectant la fiabilité des plans de contrôle et de données. Cela devient critique lorsque la fiabilité est essentielle, nécessitant une surveillance constante par le contrôleur SDN et des règles adaptatives.

Le contrôleur SDN doit s'adapter rapidement aux dynamiques du réseau, ce qui entraîne une communication accrue. Nous introduisons des mécanismes pour maintenir et mettre à jour efficacement ces réseaux. Cela inclut l'identification des liens défaillants grâce aux rapports de qualité des nœuds et des techniques pour mettre à jour les plans de contrôle et de données, permettant la redirection des flux lorsque nécessaire.

Nous présentons une architecture pour l'admission d'applications industrielles critiques, étendant SDN-IEEE 802.15.4-TSCH pour une reconfiguration en temps réel lorsque la qualité du lien diminue. Notre solution est économe en énergie et fiable, évitant les pertes de données lors de la reconfiguration.

Détection de Défaillance & Sélection du Parent

Chaque nœud envoie périodiquement des paquets `report` au contrôleur, transportant le nombre d'EB reçus de ses voisins. Pour garantir une qualité de lien suffisante, le contrôleur compare régulièrement le PDR entre le parent actuel et le meilleur voisin pouvant servir de nouveau parent. Le changement de parent est déclenché si les conditions suivantes sont remplies :

$$\begin{aligned} &\text{Cond. 1:} \quad PDR(parent\_actuel) \leq \alpha * PDR(meilleur\_parent) \\ &\text{Cond. 2:} \quad meilleur\_parent \notin noeuds\_sous - arbre \end{aligned} \tag{3}$$

$0 < \alpha < 1$ est le paramètre définissant la sensibilité de la règle de reconfiguration. Avant de déclencher la reconfiguration, le contrôleur vérifie également l'absence de processus de reconfiguration actif, évitant ainsi les conflits potentiels entre les processus de mise à jour.

Mise à Jour des Plans de Contrôle et de Données

Le contrôleur doit reconfigurer le plan de contrôle en allouant des cellules dédiées entre le nœud cible et son nouveau parent. Pour cela :

vers le contrôleur : une nouvelle cellule dédiée est réservée, réutilisant si possible la plage temporelle existante ou en en sélectionnant une nouvelle aléatoirement ;

du contrôleur : la cellule RX correspondante est ajoutée dans l'emploi du temps du nœud cible.

Deux paquets `config` sont envoyés pour mettre à jour la configuration, un pour chaque direction. Le contrôleur utilise un mécanisme de routage source pour acheminer les paquets `config`. Pour minimiser la surcharge, le format du paquet `config` est optimisé, transportant le chemin à suivre et les cellules à insérer/supprimer.

Après réception, le nœud cible envoie un `ack`, assurant la cohérence globale. En cas de non-réception, le contrôleur retransmet le paquet `config`.

Ensuite, le contrôleur redirige les flux de données vers le nouveau parent en envoyant un paquet `config` dédié à chaque flux. Les paquets suivent le nouveau chemin configuré, garantissant l'isolation des flux pendant la convergence.

(Re)-scheduling Algorithm

We propose a heuristic rescheduling algorithm to minimize updates when redirecting data flows in SDN-enabled WSNs. Instead of computing a new schedule from scratch, we focus on updating the subset of the path that differs between the old and new paths, minimizing the end-to-end delay.

The algorithm follows a greedy approach, optimizing the schedule for the specific subpath to minimize buffering time. It computes the number of cells needed for each hop, considering link quality, and allocates cells accordingly. Consecutive cells are preferentially selected to minimize forwarding delay.

In cases where the last cell of the subpath overlaps with the first cell of the remaining path, the algorithm updates the rest of the schedule hop-by-hop toward the border router. The update stops if the last cell of the current hop is scheduled before the cells for the next hop. The schedule is valid when the last cell in the subpath is scheduled before the first cell of the remaining path.

The algorithm allows for flexibility in reusing timeslots through old and new paths, differentiating them with channel offsets to avoid collisions during convergence. The example in Fig. 5.3 illustrates this process, showing consecutive cells for the new subpath (A,C,D) in blue and hop-by-hop updates for the overlapping schedule in green.

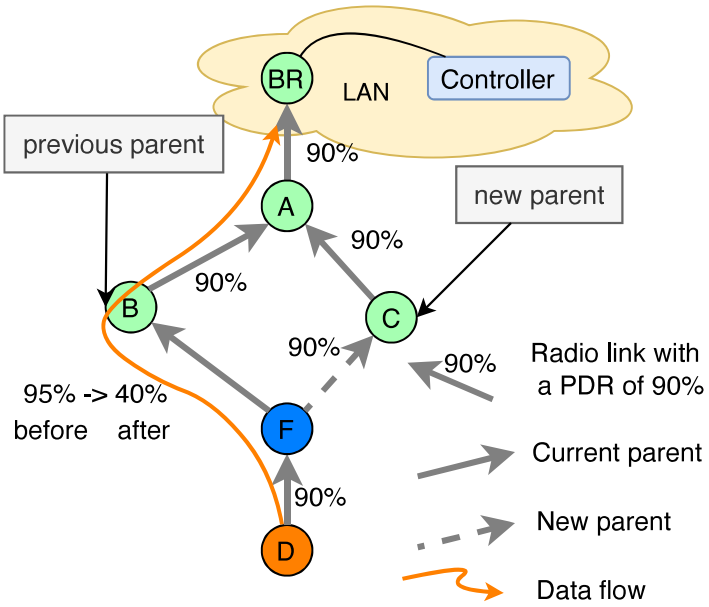

Figure 8: Un contrôleur doit reconfigurer les plans de contrôle et de données pour rediriger les flux à travers le lien (F→C) lorsque le PDR du lien (F→B) diminue de 95% à 40%

Évaluation des Performances

Dans cette évaluation, nous implémentons et comparons trois solutions : l'original SDN-TSCH (SDN-TSCH-orig), notre version étendue (SDN-TSCH-reconf) et la fonction de planification distribuée MSF pour TSCH. Le scénario de simulation implique une topologie changeante, en mettant l'accent sur un seul changement de lien, comme illustré dans la Fig. 8.

La Fig. 9 présente le PDR par flux pour $Flow_F$ et $Flow_D$ après la dégradation de la qualité du lien. Alors que SDN-TSCH-orig a du mal à maintenir la fiabilité en raison d'un nombre insuffisant de cellules, SDN-TSCH-reconf offre de manière cohérente une fiabilité ultra-élevée même pendant les changements de topologie. MSF est confronté à des défis pour une approvisionnement rapide en ressources, entraînant un PDR plus faible.

La Fig. 10 illustre le temps de reconfiguration, avec SDN-TSCH-reconf en moyenne à 14s. La surcharge de contrôle pendant la reconfiguration est représentée à la Fig. 11, où SDN-TSCH-reconf utilise efficacement 4 paquets `config` par reconfiguration, tandis que MSF en nécessite 12. De plus, MSF connaît des temps de détection plus longs (1.5 à 18 minutes).

Dans l'ensemble, SDN-TSCH-reconf démontre une reconfiguration efficace avec un délai minimal et une fiabilité élevée par rapport à SDN-TSCH-orig et MSF,

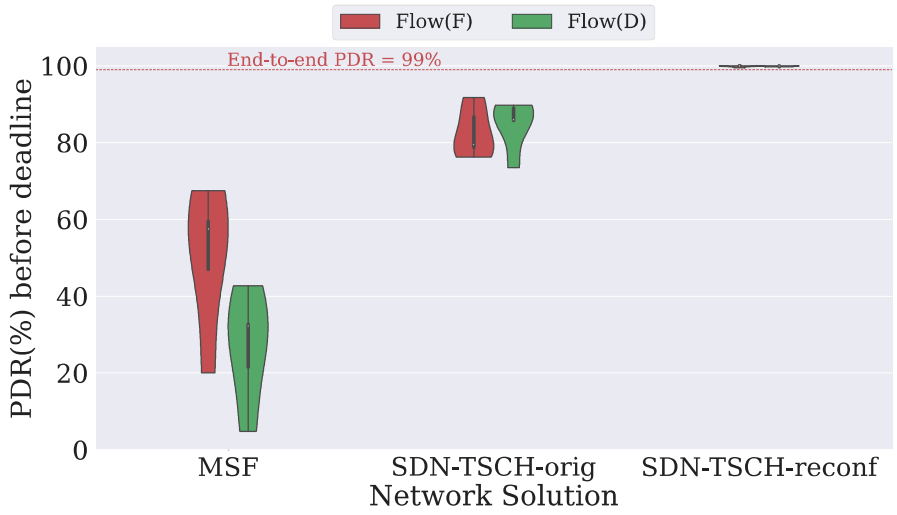

Figure 9: Taux de Livraison de Paquets par Flux

comme le montre l'évaluation des performances.

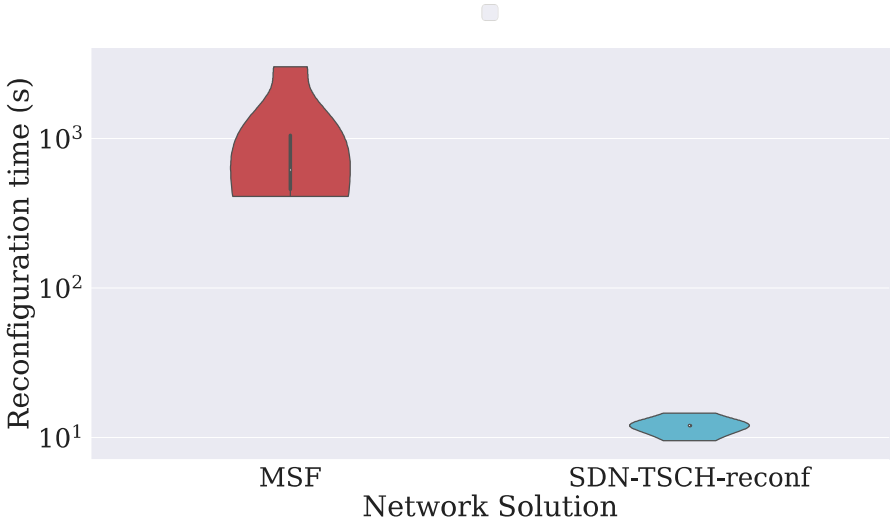

Figure 10: Temps de Reconfiguration

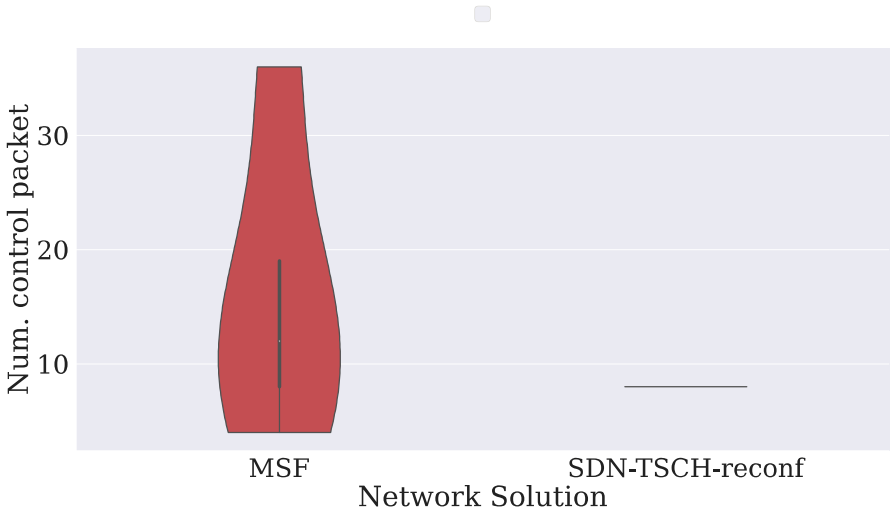

Figure 11: Surcharge de Contrôle pour la Reconfiguration

Conclusion

Cette thèse explore la gestion cruciale des flux dans les réseaux de capteurs sans fil industriels planifiés en utilisant le SDN. Les contributions clés incluent :

1. Communication du Plan de Contrôle : Développement d'une communication robuste pour les réseaux sans fil planifiés, avec la configuration de chaque nœud possédant des ressources dédiées pour éliminer les collisions. Proposition d'un système d'admission des appels pour définir des règles d'acheminement et des ressources dédiées.

2. Estimation de la Qualité de Liens (QdL) : Proposition d'une approche précise de QdL pour les réseaux SDN planifiés en utilisant une technique passive pour surveiller les balises de synchronisation. Évaluation de l'efficacité des plans de contrôle dédiés et partagés en termes de fiabilité et d'efficacité énergétique.

3. Maintenance Efficace du Réseau : Introduction d'une solution efficace pour la maintenance de réseau dans les réseaux SDN planifiés. Surveillance continue des changements de qualité des liens, avec une reconfiguration initiée lors de changements significatifs. Proposition d'un mécanisme efficace pour minimiser le volume et le temps de reconfiguration sans compromettre les SLAs des flux critiques.

Les résultats soulignent l'efficacité des solutions pour respecter les SLAs, assurer une utilisation efficace des ressources et relever les défis dynamiques des réseaux sans fil industriels.

Bibliography

[1] Bushra Rashid and Mubashir Husain Rehmani. "Applications of wireless sensor networks for urban areas: A survey". In: ***Journal of network and computer applications*** 60 (2016), pp. 192–219.

[2] Mihai T Lazarescu. "Design of a WSN platform for long-term environmental monitoring for IoT applications". In: ***IEEE Journal on emerging and selected topics in circuits and systems*** 3.1 (2013), pp. 45–54.

[3] Aditya Gaur, Bryan Scotney, Gerard Parr, and Sally McClean. "Smart city architecture and its applications based on IoT". In: ***Procedia computer science*** 52 (2015), pp. 1089–1094.

[4] JeongGil Ko, Chenyang Lu, Mani B Srivastava, John A Stankovic, Andreas Terzis, and Matt Welsh. "Wireless sensor networks for healthcare". In: ***Proceedings of the IEEE*** 98.11 (2010), pp. 1947–1960.

[5] Dragan Peraković, Marko Periša, and Petra Zorić. "Challenges and Issues of ICT in Industry 4.0". In: ***Design, simulation, manufacturing: The innovation exchange*** (2019), pp. 259–269.

[6] Aysegul Tuysuz Erman and Ozlem Durmaz Incel. "Medium access control and routing in industrial wireless sensor networks". In: ***Industrial Wireless Sensor Networks***. CRC Press, 2017, pp. 231–258.

[7] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. "Wireless sensor network survey". In: ***Computer networks*** 52.12 (2008), pp. 2292–2330.

[8] Carlos F García-Hernández, Pablo H Ibarguengoytia-Gonzalez, Joaquín García-Hernández, and Jesús A Pérez-Díaz. "Wireless sensor networks and applications: a survey". In: ***IJCSNS International Journal of Computer Science and Network Security*** 7.3 (2007), pp. 264–273.

[9] Johan Åkerberg, Mikael Gidlund, and Mats Björkman. "Future research challenges in wireless sensor and actuator networks targeting industrial automation". In: ***2011 9th IEEE International Conference on Industrial Informatics***. IEEE. 2011, pp. 410–415.

[10] Mohsin Raza, Nauman Aslam, Hoa Le-Minh, Sajjad Hussain, Yue Cao, and Noor Muhammad Khan. "A critical analysis of research potential, challenges, and future directives in industrial wireless sensor networks". In: ***IEEE Communications Surveys & Tutorials*** 20.1 (2017), pp. 39–95.

[11] Li Da Xu, Eric L Xu, and Ling Li. "Industry 4.0: state of the art and future trends". In: ***International journal of production research*** 56.8 (2018), pp. 2941–2962.

[12] Ondrej Kreibich, Jan Neuzil, and Radislav Smid. "Quality-based multiple-sensor fusion in an industrial wireless sensor network for MCM". In: ***IEEE Transactions on Industrial Electronics*** 61.9 (2013), pp. 4903–4911.

[13] Mamoona Majid, Shaista Habib, Abdul Rehman Javed, Muhammad Rizwan, Gautam Srivastava, Thippa Reddy Gadekallu, and Jerry Chun-Wei Lin. "Applications of wireless sensor networks and internet of things frameworks in the industry revolution 4.0: A systematic literature review". In: ***Sensors*** 22.6 (2022), p. 2087.

[14] Vehbi C Gungor and Gerhard P Hancke. "Industrial wireless sensor networks: Challenges, design principles, and technical approaches". In: ***IEEE Transactions on industrial electronics*** 56.10 (2009), pp. 4258–4265.

[15] Quan Wang and Jin Jiang. "Comparative examination on architecture and protocol of industrial wireless sensor network standards". In: ***IEEE Communications Surveys & Tutorials*** 18.3 (2016), pp. 2197–2219.

[16] A Salleh, MK Ismail, NR Mohamad, MZ An Abd Aziz, MA Othman, and MH Misran. "Development of greenhouse monitoring using wireless sensor network through ZigBee technology". In: ***International Journal of Engineering Science Invention*** 2.7 (2013), pp. 6–12.

[17] Alparslan Sari, Alexios Lekidis, and Ismail Butun. "Industrial networks and IIoT: Now and future trends". In: ***Industrial IoT: Challenges, Design Principles, Applications, and Security*** (2020), pp. 3–55.

[18] Marco Ehrlich, Lukasz Wisniewski, and Jürgen Jasperneite. "State of the art and future applications of industrial wireless sensor networks". In: ***Kommunikation und Bildverarbeitung in der Automation: Ausgewählte Beiträge der Jahreskolloquien KommA und BVAu 2016 zum 10jähri-***

gen Jubiläum des inIT-Institut für industrielle Informationstechnik (2018), pp. 28–39.

[19] Andreas Ramstad Urke, Øivind Kure, and Knut Øvsthus. "A survey of 802.15. 4 TSCH schedulers for a standardized industrial Internet of Things". In: ***Sensors*** 22.1 (2021), p. 15.

[20] Seong-eun Yoo, Poh Kit Chong, Daeyoung Kim, Yoonmee Doh, Minh-Long Pham, Eunchang Choi, and Jaedoo Huh. "Guaranteeing real-time services for industrial wireless sensor networks with IEEE 802.15. 4". In: ***IEEE Transactions on Industrial Electronics*** 57.11 (2010), pp. 3868–3876.

[21] Fatma H El-Fouly and Rabie A Ramadan. "Real-time energy-efficient reliable traffic aware routing for industrial wireless sensor networks". In: ***IEEE Access*** 8 (2020), pp. 58130–58145.

[22] M Aykut Yigitel, Ozlem Durmaz Incel, and Cem Ersoy. "QoS-aware MAC protocols for wireless sensor networks: A survey". In: ***Computer Networks*** 55.8 (2011), pp. 1982–2004.

[23] Rodrigo Teles Hermeto, Antoine Gallais, and Fabrice Theoleyre. "Scheduling for IEEE802. 15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey". In: ***Computer Communications*** 114 (2017), pp. 84–105.

[24] Nikos Bizanis et al. "SDN and virtualization solutions for the Internet of Things: A survey". In: ***IEEE Access*** 4 (2016), pp. 5591–5606.

[25] Dominik Henneke et al. "Analysis of realizing a future industrial network by means of Software-Defined Networking (SDN)". In: ***IEEE WFCS***. 2016.

[26] ***IEEE Standard for Low-Rate Wireless Networks***. IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015). 2020.

[27] Humaira Abdus Salam and Bilal Muhammad Khan. "IWSN-standards, challenges and future". In: ***IEEE Potentials*** 35.2 (2016), pp. 9–16.

[28] Pei Huang, Li Xiao, Soroor Soltani, Matt W Mutka, and Ning Xi. "The evolution of MAC protocols in wireless sensor networks: A survey". In: ***IEEE communications surveys & tutorials*** 15.1 (2012), pp. 101–120.

[29] Asis Nasipuri, Jun Zhuang, and Samir R Das. "A multichannel CSMA MAC protocol for multihop wireless networks". In: ***WCNC. 1999 IEEE Wireless Communications and Networking Conference (Cat. No. 99TH8466)***. Vol. 3. IEEE. 1999, pp. 1402–1406.

[30] Shao-Cheng Wang and Ahmed Helmy. "Performance limits and analysis of contention-based IEEE 802.11 MAC". In: ***Proceedings. 2006 31st IEEE Conference on Local Computer Networks***. IEEE. 2006, pp. 418–425.

[31] Jian Ma, Hongchao Wang, Dong Yang, and Yujun Cheng. "Challenges: from standards to implementation for industrial wireless sensor networks". In: ***International Journal of Distributed Sensor Networks*** 12.2 (2016), p. 3898535.

[32] Khaled Abid, Hicham Lakhlef, and Abdelmadjid Bouabdallah. "A survey on recent contention-free MAC protocols for static and mobile wireless decentralized networks in IoT". In: ***Computer Networks*** 201 (2021), p. 108583.

[33] Ricardo C Carrano, Diego Passos, Luiz CS Magalhaes, and Celio VN Albuquerque. "Survey and taxonomy of duty cycling mechanisms in wireless sensor networks". In: ***IEEE Communications Surveys & Tutorials*** 16.1 (2013), pp. 181–194.

[34] Fayez Alfayez, Mohammad Hammoudeh, and Abdelrahman Abuarqoub. "A survey on MAC protocols for duty-cycled wireless sensor networks". In: ***Procedia Computer Science*** 73 (2015), pp. 482–489.

[35] Mohammed Zaki Hasan, Fadi Al-Turjman, and Hussain Al-Rizzo. "Evaluation of a duty-cycled protocol for TDMA-based wireless sensor networks". In: ***2016 International Wireless Communications and Mobile Computing Conference (IWCMC)***. IEEE. 2016, pp. 964–969.

[36] Adam Dunkels. ***The contikimac radio duty cycling protocol***. 2011.

[37] Gustavo Künzel, Gustavo Cainelli, Ivan Müller, Carlos Eduardo Pereira, and Leandro Soares Indrusiak. "A reliable and low-latency graph-routing approach for iwsn using q-routing". In: ***2020 X Brazilian Symposium on Computing Systems Engineering (SBESC)***. IEEE. 2020, pp. 1–8.

[38] Thomas Clausen, Jiazi Yi, and Axel Colin De Verdiere. "Loadng: Towards aodv version 2". In: ***2012 IEEE Vehicular Technology Conference (VTC Fall)***. IEEE. 2012, pp. 1–5.

[39] Guoyou He. "Destination-sequenced distance vector (DSDV) protocol". In: ***Networking Laboratory, Helsinki University of Technology*** 135 (2002), pp. 1–9.

[40] Sunil Kumar Singh, Prabhat Kumar, and Jyoti Prakash Singh. "A survey on successors of LEACH protocol". In: ***Ieee Access*** 5 (2017), pp. 4298–4328.

[41] T. Winter. ***Routing Protocol for Low-Power and Lossy Networks***. RFC 6550,6551,6552. IETF, 2012.

[42] Christos Nakas, Dionisis Kandris, and Georgios Visvardis. "Energy efficient routing in wireless sensor networks: A comprehensive survey". In: ***Algorithms*** 13.3 (2020), p. 72.

[43] George N Rouskas and Ilia Baldine. "Multicast routing with end-to-end delay and delay variation constraints". In: ***Proceedings of IEEE INFOCOM'96. Conference on Computer Communications***. Vol. 1. IEEE. 1996, pp. 353–360.

[44] Chékra El Fehri, Mohamed Kassab, Slim Abdellatif, Pascal Berthou, and Abdelfettah Belghith. "LoRa technology MAC layer operations and Research issues". In: ***Procedia computer science*** 130 (2018), pp. 1096–1101.

[45] Andri Rahmadhani and Fernando Kuipers. "When lorawan frames collide". In: ***Proceedings of the 12th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization***. 2018, pp. 89–97.

[46] Bluetooth Special Interest Group (SIG). "Bluetooth Low Energy". In: (2010).

[47] Seyed Mahdi Darroudi and Carles Gomez. "Bluetooth low energy mesh networks: A survey". In: ***Sensors*** 17.7 (2017), p. 1467.

[48] Mathias Baert, Jen Rossey, Adnan Shahid, and Jeroen Hoebeke. "The Bluetooth mesh standard: An overview and experimental evaluation". In: ***Sensors*** 18.8 (2018), p. 2409.

[49] Stefan Aust, R Venkatesha Prasad, and Ignas GMM Niemegeers. "Outdoor long-range WLANs: A lesson for IEEE 802.11 ah". In: ***IEEE Communications Surveys & Tutorials*** 17.3 (2015), pp. 1761–1775.

[50] Minyoung Park. "IEEE 802.11 ah: sub-1-GHz license-exempt operation for the internet of things". In: ***IEEE Communications Magazine*** 53.9 (2015), pp. 145–151.

[51] Toni Adame, Albert Bel, Boris Bellalta, Jaume Barceló, Javier Gonzalez, and Miquel Oliver. "Capacity analysis of IEEE 802.11 ah WLANs for M2M communications". In: ***Multiple Access Communcations: 6th International Workshop, MACOM 2013, Vilnius, Lithuania, December 16-17, 2013. Proceedings 6***. Springer. 2013, pp. 139–155.

[52] Erfan Mozaffariahrar, Fabrice Theoleyre, and Michael Menth. "A survey of Wi-Fi 6: Technologies, advances, and challenges". In: ***Future Internet*** 14.10 (2022), p. 293.

[53] Rapeepat Ratasuk, Nitin Mangalvedhe, Yanji Zhang, Michel Robert, and Jussi-Pekka Koskinen. "Overview of narrowband IoT in LTE Rel-13". In: ***2016 IEEE conference on standards for communications and networking (CSCN)***. IEEE. 2016, pp. 1–7.

[54] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. "A survey on LPWA technology: LoRa and NB-IoT". In: ***Ict Express*** 3.1 (2017), pp. 14–21.

[55] Jianping Song, Song Han, Al Mok, Deji Chen, Mike Lucas, Mark Nixon, and Wally Pratt. "WirelessHART: Applying wireless technology in real-time industrial process control". In: ***2008 IEEE Real-Time and Embedded Technology and Applications Symposium***. IEEE. 2008, pp. 377–386.

[56] Venkata Prashant Modekurthy, Abusayeed Saifullah, and Sanjay Madria. "DistributedHART: A distributed real-time scheduling system for WirelessHART networks". In: ***2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)***. IEEE. 2019, pp. 216–227.

[57] Xi Jin, Fanxin Kong, Linghe Kong, Wei Liu, and Peng Zeng. "Reliability and temporality optimization for multiple coexisting WirelessHART networks in industrial environments". In: ***IEEE Transactions on Industrial Electronics*** 64.8 (2017), pp. 6591–6602.

[58] ISA Standard. "Wireless systems for industrial automation: process control and related applications". In: ***ISA-100.11 a-2009*** (2009), p. 30.

[59] Stig Petersen and Simon Carlsen. "WirelessHART versus ISA100. 11a: The format war hits the factory floor". In: ***IEEE Industrial Electronics Magazine*** 5.4 (2011), pp. 23–34.

[60] Fadillah Purnama Rezha and Soo Young Shin. "Performance evaluation of ISA100. 11A industrial wireless network". In: ***IET International Conference on Information and Communications Technologies (IETICT 2013)***. IET. 2013, pp. 587–592.

[61] Xavier Vilajosana, Thomas Watteyne, Tengfei Chang, Mališa Vučinić, Simon Duquennoy, and Pascal Thubert. "Ietf 6tisch: A tutorial". In: ***IEEE Communications Surveys & Tutorials*** 22.1 (2019), pp. 595–615.

[62] Jin-Shyan Lee, Chun-Chieh Chuang, and Chung-Chou Shen. "Applications of short-range wireless technologies to industrial automation: A ZigBee approach". In: ***2009 Fifth Advanced International Conference on Telecommunications***. IEEE. 2009, pp. 15–20.

[63] "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)". In: ***IEEE Std 802.15.4-2003*** (2003), pp. 1–680. DOI: 10.1109/IEEESTD.2003.94389.

[64] Farzad Veisi, Majid Nabi, and Hossein Saidi. "An Empirical Study of the Performance of IEEE 802.15.4e TSCH for Wireless Body Area Networks". In: ***2019 IEEE Wireless Communications and Networking Conference (WCNC)***. 2019, pp. 1–6. DOI: 10.1109/WCNC.2019.8885743.

[65] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)". In: ***IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*** (2006), pp. 1–320. DOI: 10.1109/IEEESTD.2006.232110.

[66] Naveed Salman, Imtiaz Rasool, and Andrew H Kemp. "Overview of the IEEE 802.15. 4 standards family for low rate wireless personal area networks". In: ***2010 7th international symposium on wireless communication systems***. IEEE. 2010, pp. 701–705.

[67] Petcharat Suriyachai, Utz Roedig, and Andrew Scott. "A survey of MAC protocols for mission-critical applications in wireless sensor networks". In: ***IEEE communications surveys & tutorials*** 14.2 (2011), pp. 240–264.

[68] "IEEE802.15.4-2015 - IEEE Standard for Low-Rate Wireless Networks". In: ***IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011) (April 2016)*** (2016), pp. 1–709.

[69] Domenico De Guglielmo, Simone Brienza, and Giuseppe Anastasi. "IEEE 802.15. 4e: A survey". In: ***Computer Communications*** 88 (2016), pp. 1–24.

[70] Fabrice Theoleyre et al. "Experimental validation of a distributed self-configured 6TiSCH with traffic isolation in low power lossy networks". In: ***ACM MSWiM***. 2016.

[71] Tengfei Chang, Thomas Watteyne, Kris Pister, and Qin Wang. "Adaptive synchronization in multi-hop TSCH networks". In: ***Computer Networks*** 76 (2015), pp. 165–176.

[72] Ines Khoufi, Pascale Minet, and Badr Rmili. "Beacon advertising in an IEEE 802.15. 4e TSCH network for space launch vehicles". In: ***Acta Astronautica*** 158 (2019), pp. 76–88.

[73] Thang Phan Duy and YoungHan Kim. "An efficient joining scheme in IEEE 802.15. 4e". In: ***2015 International Conference on Information and Communication Technology Convergence (ICTC)***. IEEE. 2015, pp. 226–229.

[74] David Stanislowski, Xavier Vilajosana, Qin Wang, Thomas Watteyne, and Kristofer SJ Pister. "Adaptive synchronization in IEEE802. 15.4 e networks". In: ***IEEE Transactions on Industrial Informatics*** 10.1 (2013), pp. 795–802.

[75] Thomas Watteyne, Ankur Mehta, and Kris Pister. "Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense". In: ***Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks (PE-WASUN)***. ACM. Tenerife, Spain, 2009, pp. 116–123. DOI: 10.1145/1641876.1641898.

[76] Atis Elsts, Xenofon Fafoutis, Robert Piechocki, and Ian Craddock. "Adaptive channel selection in IEEE 802.15. 4 TSCH networks". In: ***2017 Global Internet of Things Summit (GIoTS)***. IEEE. 2017, pp. 1–6.

[77] Pedro Henrique Gomes, Thomas Watteyne, and Bhaskar Krishnamachari. "MABO-TSCH: multihop and blacklist-based optimized time synchronized channel hopping". In: ***Transactions on Emerging Telecommunications Technologies*** 29.7 (2018), e3223.

[78] Gaurav Jolly and Mohamed Younis. "An energy-efficient, scalable and collision-free MAC layer protocol for wireless sensor networks". In: ***Wireless Communications and Mobile Computing*** 5.3 (2005), pp. 285–304.

[79] Maria Rita Palattella, Nicola Accettura, Luigi Alfredo Grieco, Gennaro Boggia, Mischa Dohler, and Thomas Engel. "On optimal scheduling in duty-cycled industrial IoT applications using IEEE802. 15.4 e TSCH". In: ***IEEE Sensors Journal*** 13.10 (2013), pp. 3655–3666.

[80] Mohamed Osman and Frederic Nabki. "OSCAR: An optimized scheduling cell allocation algorithm for convergecast in IEEE 802.15. 4e TSCH networks". In: ***Sensors*** 21.7 (2021), p. 2493.

[81] Vasileios Kotsiou, Georgios Z Papadopoulos, Periklis Chatzimisios, and Fabrice Theoleyre. "LDSF: Low-latency distributed scheduling function for industrial Internet of Things". In: ***IEEE internet of things journal*** 7.9 (2020), pp. 8688–8699.

[82] Rasool Tavakoli, Majid Nabi, Twan Basten, and Kees Goossens. "Topology management and TSCH scheduling for low-latency convergecast in in-vehicle WSNs". In: ***IEEE Transactions on Industrial Informatics*** 15.2 (2018), pp. 1082–1093.

[83] Masafumi Hashimoto, Naoki Wakamiya, Masayuki Murata, Yasutaka Kawamoto, and Kiyoshi Fukui. "End-to-end reliability-and delay-aware scheduling with slot sharing for wireless sensor networks". In: ***2016 8th International Conference on Communication Systems and Networks (COMSNETS)***. IEEE. 2016, pp. 1–8.

[84] IETF. ***IPv6 over the TSCH mode of IEEE 802.15.4e (6tisch)***. https://datatracker.ietf.org/wg/6tisch. 2018.

[85] Q. Wang et al. ***6top Protocol (6P)***. draft. draft-ietf-6tisch-6top-protocol-11. IETF, 2018.

[86] Rodrigo Teles Hermeto et al. "Scheduling for IEEE802.15.4-TSCH and Slow Channel Hopping MAC in Low Power Industrial Wireless Networks". In: ***Comput. Commun.*** 114.C (Dec. 2017), pp. 84–105. DOI: 10.1016/j.comcom.2017.10.004.

[87] Seungbeom Jeong, Hyung-Sin Kim, Jeongyeup Paek, and Saewoong Bahk. "OST: On-demand TSCH scheduling with traffic-awareness". In: ***IEEE INFOCOM 2020-IEEE Conference on Computer Communications***. IEEE. 2020, pp. 69–78.

[88] Nicola Accettura, Elvis Vogli, Maria Rita Palattella, Luigi Alfredo Grieco, Gennaro Boggia, and Mischa Dohler. "Decentralized traffic aware scheduling in 6TiSCH networks: Design and experimental evaluation". In: ***IEEE Internet of Things Journal*** 2.6 (2015), pp. 455–470.

[89] Ridha Soua, Pascale Minet, and Erwan Livolant. "Wave: a distributed scheduling algorithm for convergecast in IEEE 802.15. 4e TSCH networks". In: ***Transactions on Emerging Telecommunications Technologies*** 27.4 (2016), pp. 557–575.

[90] Simon Duquennoy et al. "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH". In: ***SenSys***. ACM. Seoul, South Korea, 2015, pp. 337–350. ISBN: 978-1-4503-3631-4. DOI: 10.1145/2809695.2809714.

[91] Tengfei Chang, Mališa Vučinić, Xavier Vilajosana, Simon Duquennoy, and Diego Roberto Dujovne. ***6TiSCH Minimal Scheduling Function (MSF)***. RFC 9033. IETF, 2021.

[92] Adrian Farrel et al. ***A path computation element (PCE)-based architecture***. RFC 4655. IETF, 2006.

[93] Sana Rekik, Nouha Baccour, Mohamed Jmaiel, Khalil Drira, and Luigi Alfredo Grieco. "Autonomous and traffic-aware scheduling for TSCH networks". In: ***Computer Networks*** 135 (2018), pp. 201–212.

[94] Seohyang Kim, Hyung-Sin Kim, and Chong-kwon Kim. "A3: Adaptive autonomous allocation of TSCH slots". In: ***Proceedings of the 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021)***. 2021, pp. 299–314.

[95] Yichao Jin et al. "A centralized scheduling algorithm for IEEE 802.15. 4e TSCH based industrial low power wireless networks". In: ***2016 IEEE WCNC***. 2016.

[96] M. R. Palattella et al. "On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH". In: ***IEEE Sensors Journal*** 13.10 (2013), pp. 3655–3666. DOI: 10.1109/JSEN.2013.2266417.

[97] Guillaume Gaillard, Dominique Barthel, Fabrice Theoleyre, and Fabrice Valois. "Kausa: KPI-aware scheduling algorithm for multi-flow in multi-hop IoT networks". In: ***Ad-hoc, Mobile, and Wireless Networks: 15th International Conference, ADHOC-NOW 2016, Lille, France, July 4-6, 2016, Proceedings 15***. Springer. 2016, pp. 47–61.

[98] Nikumani Choudhury, Moustafa M Nasralla, Prakhar Gupta, and Ikram Ur Rehman. "Centralized graph based TSCH scheduling for IoT network applications". In: ***2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)***. IEEE. 2021, pp. 1639–1644.

[99] Yosr Jarraya et al. "A survey and a layered taxonomy of software-defined networking". In: ***IEEE communications surveys & tutorials*** 16.4 (2014), pp. 1955–1980.

[100] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. "OpenFlow: enabling innovation in campus networks". In: ***ACM SIGCOMM computer communication review*** 38.2 (2008), pp. 69–74.

[101] Fei Hu, Qi Hao, and Ke Bao. "A survey on software-defined network and openflow: From concept to implementation". In: ***IEEE Communications Surveys & Tutorials*** 16.4 (2014), pp. 2181–2206.

[102] Isaias Martinez-Yelmo, Joaquin Alvarez-Horcajo, Juan Antonio Carral, and Diego Lopez-Pajares. "eHDDP: Enhanced Hybrid Domain Discovery Protocol for network topologies with both wired/wireless and SDN/non-SDN devices". In: ***Computer Networks*** 191 (2021), p. 107983.

[103] Wei Zhou, Li Li, Min Luo, and Wu Chou. "REST API design patterns for SDN northbound API". In: ***2014 28th international conference on advanced information networking and applications workshops***. IEEE. 2014, pp. 358–365.

[104] Musa Ndiaye, Gerhard P Hancke, and Adnan M Abu-Mahfouz. "Software defined networking for improved wireless sensor network management: A survey". In: ***Sensors*** 17.5 (2017), p. 1031.

[105] Muhammad Ali Hassan, Quoc-Tuan Vien, and Mahdi Aiash. "Software defined networking for wireless sensor networks: a survey". In: ***Advances in Wireless Communications and Networks*** 3.2 (2017), pp. 10–22.

[106] Ngoc-Tu Nguyen, Bing-Hong Liu, Shao-I Chu, and Hao-Zhe Weng. "Challenges, designs, and performances of a distributed algorithm for minimum-latency of data-aggregation in multi-channel WSNs". In: ***IEEE Transactions on Network and Service Management*** 16.1 (2018), pp. 192–205.

[107] Michael Baddeley, Reza Nejabati, George Oikonomou, Mahesh Sooriyabandara, and Dimitra Simeonidou. "Evolving SDN for low-power IoT networks". In: ***2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)***. IEEE. 2018, pp. 71–79.

[108] Jetmir Haxhibeqiri, Pedro Heleno Isolani, Johann M Marquez-Barja, Ingrid Moerman, and Jeroen Hoebeke. "In-band network monitoring technique to support SDN-based wireless networks". In: ***IEEE Transactions on Network and Service Management*** 18.1 (2020), pp. 627–641.

[109] Tie Luo, Hwee-Pink Tan, and Tony QS Quek. "Sensor OpenFlow: Enabling software-defined wireless sensor networks". In: ***IEEE Communications letters*** 16.11 (2012), pp. 1896–1899.

[110] Salvatore Costanzo, Laura Galluccio, Giacomo Morabito, and Sergio Palazzo. "Software defined wireless networks: Unbridling SDNs". In: ***2012 European Workshop on Software Defined Networking***. IEEE. 2012, pp. 1–6.

[111] Laura Galluccio et al. "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for Wireless Sensor networks". In: ***INFOCOM***. 2015. DOI: 10.1109/INFOCOM.2015.7218418.

[112] de Oliveira et al. "TinySDN: Enabling tinyOS to software-defined wireless sensor networks". In: ***XXXIV Simpósio Brasileiro de Redes de Computadores. Bahia*** (2016), pp. 1229–1237.

[113] Philip Levis, Samuel Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, et al. "TinyOS: An operating system for sensor networks". In: ***Ambient intelligence*** (2005), pp. 115–148.

[114] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. "Collection tree protocol". In: ***Proceedings of the 7th ACM conference on embedded networked sensor systems***. 2009, pp. 1–14.

[115] Renan CA Alves et al. "It-sdn: Improved architecture for sdwsn". In: ***SBRC***. 2017.

[116] Andrew R Curtis, Jeffrey C Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. "DevoFlow: Scaling flow management for high-performance networks". In: ***Proceedings of the ACM SIGCOMM 2011 Conference***. 2011, pp. 254–265.

[117] Marcio LF Miguel et al. "SDN architecture for 6LoWPAN wireless sensor networks". In: ***Sensors*** 18.11 (2018), p. 3738.

[118] Ahmad Shabani Baghani and Majid Khabbazian. "RPL Point-to-point communication paths: analysis and enhancement". In: ***IEEE Internet of Things Journal*** 10.1 (2022), pp. 166–179.

[119] Remous-Aris Koutsiamanis, Georgios Z Papadopoulos, Xenofon Fafoutis, Julián Martín Del Fiore, Pascal Thubert, and Nicolas Montavont. "From best effort to deterministic packet delivery for wireless industrial IoT networks". In: ***IEEE Transactions on Industrial Informatics*** 14.10 (2018), pp. 4468–4480.

[120] Remous-Aris Koutsiamanis, Georgios Z Papadopoulos, Tomas Lagos Jenschke, Pascal Thubert, and Nicolas Montavont. "Meet the PAREO functions: Towards reliable and available wireless networks". In: ***ICC 2020-2020 IEEE International Conference on Communications (ICC)***. IEEE. 2020, pp. 1–7.

[121] Pascal Thubert, Maria Rita Palattella, and Thomas Engel. "6TiSCH centralized scheduling: When SDN meet IoT". In: ***2015 IEEE conference on standards for communications and networking (CSCN)***. IEEE. 2015, pp. 42–47.

[122] Michael Baddeley et al. "Isolating SDN control traffic with layer-2 slicing in 6TiSCH industrial IoT networks". In: ***IEEE NFV-SDN***. 2017.

[123] O. Iova et al. "Stability and efficiency of RPL under realistic conditions in Wireless Sensor Networks". In: ***IEEE PIMRC***. 2013. DOI: 10.1109/PIMRC.2013.6666490.

[124] Esteban Municio et al. "Whisper: Programmable and flexible control on industrial IoT networks". In: ***Sensors*** 18.11 (2018), p. 4048.

[125] Lucia Lo Bello, Alfio Lombardo, Sebastiano Milardo, Gaetano Patti, and Marco Reno. "Experimental assessments and analysis of an SDN framework to integrate mobility management in industrial wireless sensor networks". In: ***IEEE Transactions on Industrial Informatics*** 16.8 (2020), pp. 5586–5595.

[126] Noa Zilberman, Philip M Watts, Charalampos Rotsos, and Andrew W Moore. "Reconfigurable network systems and software-defined networking". In: ***Proceedings of the IEEE*** 103.7 (2015), pp. 1102–1124.

[127] Stefano Paris, Georgios S Paschos, and Jérémie Leguay. "Dynamic control for failure recovery and flow reconfiguration in SDN". In: ***2016 12th International Conference on the Design of Reliable Communication Networks (DRCN)***. IEEE. 2016, pp. 152–159.

[128] Nurefşan Sertbaş Bülbül, Doğanalp Ergenç, and Mathias Fischer. "Towards sdn-based dynamic path reconfiguration for time sensitive networking". In: ***NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium***. IEEE. 2022, pp. 1–9.

[129] Dolvara Gunatilaka et al. "REACT: An agile control plane for industrial wireless sensor-actuator networks". In: ***IoTDI***. IEEE. 2020, pp. 53–65.

[130] Israat Haque, Mohammad Nurujjaman, Janelle Harms, and Nael Abu-Ghazaleh. "SDSense: An agile and flexible SDN-based framework for wireless sensor networks". In: ***IEEE Transactions on Vehicular Technology*** 68.2 (2018), pp. 1866–1876.

[131] Federico Orozco-Santos, Víctor Sempere-Payá, Teresa Albero-Albero, and Javier Silvestre-Blanes. "Enhancing SDN WISE with Slicing Over TSCH". In: ***Sensors*** 21.4 (2021). ISSN: 1424-8220. DOI: 10.3390/s21041075.

[132] Federico Orozco-Santos, Víctor Sempere-Payá, Javier Silvestre-Blanes, and Teresa Albero-Albero. "Multicast Scheduling in SDN WISE to Support Mobile Nodes in Industrial Wireless Sensor Networks". In: ***IEEE Access*** 9 (2021), pp. 141651–141666. DOI: 10.1109/ACCESS.2021.3120917.

[133] Valerie Galluzzi and Ted Herman. "Survey: discovery in wireless sensor networks". In: ***International Journal of Distributed Sensor Networks*** 8.1 (2012), p. 271860.

[134] Wu Chenghai, Zhang Jianjun, Fan Xiquan, Qin Kaiyu, and Liu Xiangping. "Research on dynamic routing algorithm of the combat collaboration communication network". In: ***The 27th Chinese Control and Decision Conference (2015 CCDC)***. IEEE. 2015, pp. 4440–4445.

[135] Hyung-Sin Kim et al. "Challenging the IPv6 routing protocol for low-power and lossy networks (RPL): A survey". In: ***IEEE Communications Surveys & Tutorials*** 19.4 (2017), pp. 2502–2525.

[136] Henry-Joseph Audéoud et al. "Single Reception Estimation of Wireless Link Quality". In: ***IEEE PIMRC***. 2020.

[137] G Kirubasri and N Uma Maheswari. "A study on hardware and software link quality metrics for wireless multimedia sensor networks". In: ***International Journal of Advanced Networking and Applications*** 8.3 (2016), p. 3103.

[138] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. "An empirical study of low-power wireless". In: ***ACM Transactions on Sensor Networks (TOSN)*** 6.2 (2010), pp. 1–49.

[139] Fatima tu Zahra, Yavuz S Bostanci, and Mujdat Soyturk. "Real-Time Jamming Detection in Wireless IoT Networks". In: ***IEEE Access*** (2023).

[140] Mariusz Kaczmarek, Jacek Ruminski, and Adam Bujnowski. "Accuracy analysis of the RSSI BLE SensorTag signal for indoor localization purposes". In: ***2016 Federated Conference on Computer Science and Information Systems (FedCSIS)***. IEEE. 2016, pp. 1413–1416.

[141] Nouha Baccour et al. "Radio link quality estimation in wireless sensor networks: A survey". In: ***In ACM TOSN*** 8.4 (2012), pp. 1–33.

[142] Carles Gomez, Antoni Boix, and Josep Paradells. "Impact of LQI-based routing metrics on the performance of a one-to-one routing protocol for IEEE 802.15. 4 multihop networks". In: ***EURASIP Journal on Wireless Communications and Networking*** 2010 (2010), pp. 1–20.

[143] Nouha Baccour, Anis Koubâa, Maissa Ben Jamâa, Habib Youssef, Marco Zuniga, and Mário Alves. "A comparative simulation study of link quality estimators in wireless sensor networks". In: ***2009 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems***. IEEE. 2009, pp. 1–10.

[144] Nouha Baccour, Anis Koubâa, Habib Youssef, and Mário Alves. "Reliable link quality estimation in low-power wireless networks and its impact on tree-routing". In: ***Ad Hoc Networks*** 27 (2015), pp. 1–25.

[145] Douglas SJ De Couto, Daniel Aguayo, John Bicket, and Robert Morris. "A high-throughput path metric for multi-hop wireless routing". In: ***Proceedings of the 9th annual international conference on Mobile computing and networking***. 2003, pp. 134–146.

[146] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, and Philip Alexander Levis. "Four-bit wireless link estimation." In: ***HotNets***. 2007.

[147] Yong Wang, Margaret Martonosi, and Li-Shiuan Peh. "Predicting link quality using supervised learning in wireless sensor networks". In: ***ACM SIGMOBILE Mobile Computing and Communications Review*** 11.3 (2007), pp. 71–83.

[148] Nouha Baccour, Anis Koubâa, Habib Youssef, Maissa Ben Jamâa, Denis Do Rosario, Mário Alves, and Leandro B Becker. "F-lqe: A fuzzy link quality estimator for wireless sensor networks". In: ***Wireless Sensor Networks: 7th European Conference, EWSN 2010, Coimbra, Portugal, February 17-19, 2010. Proceedings 7***. Springer. 2010, pp. 240–255.

[149] Michele Rondinone et al. "Designing a reliable and stable link quality metric for wireless sensor networks". In: ***Proceedings of the workshop on Real-world wireless sensor networks***. 2008, pp. 6–10.

[150] Xiaomin Li et al. "A review of industrial wireless networks in the context of Industry 4.0". In: ***Wireless networks*** 23 (2017), pp. 23–41.

[151] Antoni Morell, Xavier Vilajosana, José López Vicario, and Thomas Watteyne. "Label switching over IEEE802. 15.4 e networks". In: ***Transactions on Emerging Telecommunications Technologies*** 24.5 (2013), pp. 458–475.

[152] Guillaume Gaillard et al. "Kausa: KPI-aware Scheduling Algorithm for Multiflow in Multi-hop IoT Networks". In: ***ADHOC-NOW***. 2016.

[153] Adnan Aijaz and Mahesh Sooriyabandara. "The tactile internet for industries: A review". In: ***Proceedings of the IEEE*** 107.2 (2018), pp. 414–435.

[154] Emilio Ancillotti, Carlo Vallati, Raffaele Bruno, and Enzo Mingozzi. "A reinforcement learning-based link quality estimation strategy for RPL and its impact on topology management". In: ***Computer Communications*** 112 (2017), pp. 1–13. ISSN: 0140-3664.

[155] Rodrigo Teles Hermeto, Antoine Gallais, and Fabrice Theoleyre. "Experimental in-depth study of the dynamics of an indoor industrial low power lossy network". In: ***Ad Hoc Networks*** 93 (2019), p. 101914.

[156] Rodrigo Teles Hermeto, Antoine Gallais, Kristof Van Laerhoven, and Fabrice Theoleyre. "Passive Link Quality Estimation for Accurate and Stable Parent Selection in Dense 6TiSCH Networks". In: ***EWSN***. Madrid, Spain: Junction Publishing, 2018, pp. 114–125. ISBN: 978-0-9949886-2-1.

[157] Fabrice Theoleyre et al. "Experimental Validation of a Distributed Self-Configured 6TiSCH with Traffic Isolation in Low Power Lossy Networks". In: ***ACM MSWiM***. Malta, Malta, 2016. ISBN: 9781450345026.

[158] Xavier Vilajosana et al. "A Realistic Energy Consumption Model for TSCH Networks". In: ***IEEE Sensors Journal*** 14.2 (2014), pp. 482–489. DOI: 10.1109/JSEN.2013.2285411.

[159] Baver Özceylan, Berk Ünlü, and Buyurman Baykal. "An energy efficient optimum shared cell scheduling for TSCH networks". In: ***2017 IEEE WiMob***. IEEE. 2017, pp. 1–8.

[160] Kihoon Jeon and Sanghwa Chung. "Adaptive channel quality estimation method for enhanced time slotted channel hopping on wireless sensor networks". In: ***2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)***. IEEE. 2017, pp. 438–443.

[161] Dimitrios Zorbas, Georgios Z Papadopoulos, and Christos Douligeris. "Local or global radio channel blacklisting for ieee 802.15. 4-tsch networks?" In: ***2018 IEEE International Conference on Communications (ICC)***. IEEE. 2018, pp. 1–6.

[162] Junfeng Xie, F Richard Yu, Tao Huang, Renchao Xie, Jiang Liu, Chenmeng Wang, and Yunjie Liu. "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges". In: ***IEEE Communications Surveys & Tutorials*** 21.1 (2018), pp. 393–430.

[163] Karen Scarfone, Peter Mell, et al. "Guide to intrusion detection and prevention systems (idps)". In: ***NIST special publication*** 800.2007 (2007), p. 94.

[164] Nasrin Sultana, Naveen Chilamkurti, Wei Peng, and Rabei Alhadad. "Survey on SDN based network intrusion detection system using machine learning approaches". In: ***Peer-to-Peer Networking and Applications*** 12 (2019), pp. 493–501.

[165] Mutaz HH Khairi, Sharifah HS Ariffin, NM Latiff, AS Abdullah, and MK Hassan. "A Review of Anomaly Detection Techniques and Distributed Denial of Service (DDoS) on Software Defined Network (SDN)." In: ***Engineering, Technology & Applied Science Research*** 8.2 (2018).

Farzad VEISI GOSHTASB

Enhancing Industrial Internet of Things through Software-Defined Networking: from building the network to flow management

Résumé

Cette thèse explore le rôle des réseaux de capteurs sans fil industriels (IWSNs) dans le cadre de l'Industrie 4.0, mettant en avant leur importance dans l'amélioration de l'efficacité et de la réactivité des opérations industrielles. Les IWSNs sont cruciaux pour l'interconnexion des capteurs et actionneurs, appliquant des accords de niveau de service (SLAs) pour une qualité de service (QoS) élevée en termes de fiabilité et de latence. Principalement, les IWSNs s'appuient sur des protocoles déterministes au niveau du contrôle d'accès au support (MAC) pour minimiser les incertitudes. Cependant, les approches de planification centralisée, essentielles pour des performances optimales, rencontrent des défis pratiques. Cette étude propose une solution de réseau défini par logiciel (SDN) pour les IWSNs, assurant une communication fiable au niveau du plan de contrôle et introduisant un système d'admission des appels pour une allocation centralisée efficace des ressources en fonction des exigences de QoS des appareils. La thèse présente également une méthode de découverte précise et écoénergétique de la topologie et d'estimation de la qualité de liaison, cruciale pour éviter la surestimation ou la sous-estimation des ressources. De plus, une solution efficace pour maintenir les réseaux SDN planifiés sur des liaisons sans fil variables dans le temps est présentée, minimisant le trafic de contrôle et le temps de reconfiguration tout en préservant les SLAs pour les flux critiques. Cette recherche offre des perspectives précieuses pour faire progresser les IWSNs dans le contexte de l'Industrie 4.0, en abordant les défis clés pour une performance et une adaptabilité accrues.

Résumé en anglaise

This thesis explores the role of Industrial Wireless Sensor Networks (IWSNs) within the framework of Industry 4.0, emphasizing their significance in enhancing efficiency and responsiveness in industrial operations. IWSNs are crucial for interconnecting sensors and actuators, enforcing Service Level Agreements (SLAs) for high Quality of Service (QoS) in terms of reliability and latency. Mainly, IWSNs rely on deterministic protocols at the Medium Access Control (MAC) layer to minimize uncertainties. However, centralized scheduling approaches, crucial for optimal performance, face practical challenges. This study proposes a Software Defined Network (SDN) solution for IWSNs, ensuring reliable control plane communication and introducing a call admission system for effective centralized resource allocation based on devices' QoS requirements. The thesis also presents an accurate and energy-efficient topology discovery and link quality estimation method, critical for preventing resource overestimation or underestimation. Additionally, an efficient solution for maintaining scheduled SDN networks over time-varying wireless links is introduced, minimizing control traffic and reconfiguration time while preserving SLAs for critical flows. This research provides valuable insights into advancing IWSNs in the context of Industry 4.0, addressing key challenges for enhanced performance and adaptability.