Adaptive and Secure Link Scheduling in 6TiSCH Network for Resource Efficient Data Communication

Thesis submitted to the
Indian Institute of Technology Guwahati
for the award of the degree

of

Doctor of Philosophy

in

Computer Science and Engineering

Submitted by

Karnish Nasrin Ahmed Tapadar

Under the guidance of

Dr. Manas Khatua & Prof. Tamarapalli Venkatesh

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

November, 2024

Dr. Manas Khatua & Prof. Tamarapalli Venkatesh
Computer Science and Engineering
Indian Institute of Technology Guwahati
North Guwahati, Assam, India – 781039
Email : manaskhatua@iitg.ac.in, t.venkat@iitg.ac.in

Certificate

This is to certify that this thesis entitled, **"Adaptive and Secure Link Scheduling in 6TiSCH Network for Resource Efficient Data Communication"**, being submitted by **Karnish Nasrin Ahmed Tapadar**, to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, for partial fulfillment of the award of the degree of Doctor of Philosophy, is a bonafide work carried out by her under my supervision and guidance. The thesis, in my opinion, is worthy of consideration for award of the degree of Doctor of Philosophy in accordance with the regulation of the institute. To the best of my knowledge, it has not been submitted elsewhere for the award of the degree.

Date:
Place: Guwahati

.....
Dr. Manas Khatua & Prof. Tamarapalli Venkatesh
(Thesis Supervisor)

Declaration

I certify that:

- The work contained in this thesis is original and has been done by myself and under the general supervision of my supervisor.
- The work reported herein has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (concepts, ideas, text, expressions, data, graphs, diagrams, theoretical analysis, results, etc.) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. Elaborate sentences used verbatim from published work have been clearly identified and quoted.
- I also affirm that no part of this thesis can be considered plagiarism to the best of my knowledge and understanding and take complete responsibility if any complaint arises.

Date:
Place: Guwahati

Karnish Nasrin Ahmed Tapadar

Dedicated to

My Loving Husband and My Beloved Parents

For their love, support and sacrifices

Acknowledgements

I am deeply grateful to everyone whose support made this journey possible. I sincerely appreciate all who contributed to my personal and professional growth.

First and foremost, I would like to express my deepest gratitude to my supervisors, *Dr. Manas Khatua & Prof. Tamarapalli Venkatesh*, for their unwavering support, encouragement, patience, and invaluable guidance throughout this journey. A special thanks to *Dr. Manas Khatua*, whose persistent corrections, insightful suggestions, and engaging discussions have been instrumental in shaping my Ph.D. work. His mentorship, teaching me how to write papers and approach problems with a fresh perspective, has been invaluable. I am truly grateful to my supervisors for giving me the opportunity to work under their guidance, and I will always cherish their contributions to my growth as a researcher.

I extend my heartfelt appreciation to my Doctoral Committee members, *Prof. Diganta Goswami*, *Dr. Ashok Singh Sairam*, and *Prof. Sushanta Karmakar*, for their insightful comments and constructive suggestions. I am also grateful to the anonymous reviewers of my research in various forums for their critical feedback, which enhanced the quality of my work.

I would like to express my gratitude to the former Heads of the Department, *Prof. S. V. Rao* and *Prof. Jatindra Kumar Deka*, as well as the current Head, *Prof. Tamarapalli Venkatesh*, for their support in providing access to facilities, a peaceful research environment, and travel assistance for conferences. My sincere appreciation also goes to all the professors of the CSE Department for their compassionate teaching and to the staff and security personnel for their essential support. I also wish to express my gratitude to the Directors, Deans, and other officials of IITG, whose collective efforts make this institute a centre for world-class research and education.

This journey would not have been possible without the financial support from the esteemed *MHRD*, *Government of India*, and my institute, IIT Guwahati.

I am deeply grateful to my senior, Dr. Alakesh Kalita, for his guidance and constant support in addressing my doubts. My heartfelt appreciation goes to my fellow labmates in the IoT lab — Habila, Soumya, Surja, Aditya, Pavan, Pratik,

and Sreeparna for making the workplace a wonderful experience. I thank my PhD colleagues Meenu, Bharadwaj, Sujit, Amit, Tathagat, Gourishankar, Nidhi, Meenakshi, and Sumita. A special thanks to Priyanka Di for being there for me from my interview to the initial phases of my PhD journey.

No journey is wonderful without friends, and I'm grateful to those who made this roller-coaster ride smoother. Thank you, Komal, Manoj, and Saurav, for being my pillars of support. The parties, evening chais, gossip, and countless memories made this journey unforgettable. Komal and Manoj, I'm especially grateful for your unwavering support during my low times, listening to my frustrations and standing by me through every up and down, especially during rejections. Manoj, your valuable suggestions and constant encouragement mean a lot. Thank you for always being there. A special thanks to my friend Susma for her kindness, readiness to help, and support have been a blessing. I am incredibly fortunate to have had her by my side. I also extend my heartfelt thanks to Vanshali for her support and companionship and to Tani for being there. Lastly, the institute's facilities made my stay comfortable, and the presence of these amazing friends enriched the vibrant campus life.

Above all, I am profoundly grateful to my family, who have always cared for me and my well-being since childhood. I owe everything to my parents, A.N. Tapadar and N.I. Tapadar, whose love and sacrifices have shaped me into who I am today. My heartfelt thanks to my elder brother, Dr. Sufian Ahmed Tapadar, for his guidance and encouragement, always pushing me to strive for excellence. I am equally thankful to my brothers, Raihan Ahmed Tapadar and Dr. Rizwan Ahmed Tapadar, for their unwavering love and support. My gratitude extends to my sisters-in-law, Dr. Afrida Hussain and Nazmin Sultana, for their contributions to my life. The little joys of our family, Mehvish and Arshan, have brought immense happiness and lightened my mood over the years. Finally, I express my deepest gratitude to my soulmate, Jakaria, for his immense love and support, enriching my life in countless ways. I feel blessed to have him by my side throughout this journey.

Lastly, I am grateful to Almighty God, whose guidance and blessings have illuminated my path.

Date:
Place: Guwahati

Karnish Nasrin Ahmed Tapadar

Abstract

Internet of Things (IoT) has emerged as a transformative technology, enabling connectivity of physical devices to the Internet for data exchange and intelligent decision-making across various application domains. However, industrial environments have strict demands for highly reliable, scalable, and low-latency communication networks to support critical applications. Thus, Industrial Internet of Things (IIoT) has evolved and gained prominence in the industrial domain to meet the demands of industrial automation, predictive maintenance, and large-scale communication. The IEEE 802.15.4e Time Slotted Channel Hopping (TSCH) protocol is a key enabler for IIoT, offering deterministic communication attributes and mitigating interference. The IPv6 over the TSCH mode of IEEE 802.15.4e network (called 6TiSCH) is designed to ensure high reliability, low latency, and longer network lifetime for various IIoT applications while using the IP-based network services. Basically, the 6TiSCH network leverages higher layer protocols of traditional IPv6 network to operate over IEEE 802.15.4e TSCH Medium Access Control (MAC) protocol. TSCH uses "link scheduling" for data communication, which determines how the nodes in a network communicate with each other by allocating communication resources in terms of time and frequency. However, the TSCH standard does not specify how the schedule should be formed and maintained, although the network performance relies heavily on the effectiveness of the constructed link schedule.

To address these challenges, the 6TiSCH Working Group (WG), formed by Internet Engineering Task Force (IETF), published the Minimal Scheduling Function (MSF) (RFC 9033) and 6TiSCH Operation Sublayer (6top) Protocol (6P) (RFC 8480) as the important components in 6TiSCH protocol stack. The MSF and 6P together provide guidelines for building and managing link schedules in a distributed manner for the 6TiSCH network. This dissertation addresses several critical challenges associated with link scheduling in 6TiSCH networks, with a focus on the design, management, and security of the link scheduling function and its associated 6P protocol. The primary goal is to improve the overall performance of 6TiSCH networks, ensuring their effectiveness in IIoT applications.

This thesis identifies notable limitations in the published MSF standard, particularly its reliance on updating the number of scheduled cells by unity, irrespective of traffic demand, which leads to high signalling overhead. To address this, an *Improved Minimal Scheduling Function* (IMSF) is proposed, which aims to reduce 6P control overhead while improving end-to-end latency and reliability. The IMSF is discussed in Chapter 3 of this thesis. Further, it is observed that all existing scheduling functions, including MSF and the proposed IMSF, suffer from the disadvantages of random cell selection during cell allocation and deallocation, and are not free from schedule collisions. Additionally, these approaches are

not well-suited for delay-sensitive networks, as the random allocation of cells within the slotframe leads to increased end-to-end delay. To address the issue of random cell selection during allocation, this thesis introduces a *Receiver-based Traffic rate agnostic Distributed link Scheduling function* (RTDS) in Chapter 3, aiming to optimize end-to-end latency, resource allocation and signalling overhead. RTDS offers substantial improvements over the standard MSF and other existing schemes. However, the problem of random cell selection during deletion remains unexplored, as the RTDS only considered the issue during cell addition. Thus, a *Latency-aware Cell Deletion* (LCD) scheme is proposed in Chapter 3 to minimize average end-to-end latency and energy consumption.

Although the policy for schedule collision detection and mitigation is defined in MSF standard, the MSF as well as the other existing schemes fail to address all collision scenarios effectively and also heavily rely on inefficient sender-based cell relocation strategy for collision mitigation. To tackle the issue of schedule collision detection and mitigation, this thesis introduces an *Enhanced Collision Detection* (ECD) mechanism using the Exponentially Weighted Moving Average (EWMA) of the cell delivery ratio in Chapter 4. Further, it proposes a *receiver-based cell relocation* strategy along with ECD, called ECDR, for collision detection and mitigation. The proposed schemes, as discussed in Chapter 4, prove effective in improving the overall network performance.

Furthermore, this dissertation explores vulnerabilities present in the MSF and 6P protocol and demonstrates the feasibility of two different types of attack called *cell depletion attack* and *schedule instability attack* in Chapter 5. In both the attacks, the main objective of adversary is to consume scheduling resources and make the schedule unstable by performing unnecessary and incorrect 6P transactions. These attacks disrupt the communication schedule of victim nodes as well as victims' successors, thus severely causing schedule instability, and impacting their reliability and energy consumption. The Minimal Security Framework (RFC 9031) standardized by IETF cannot prevent such attacks on the MSF. To counter these vulnerabilities, two novel mitigation schemes, namely *Secure Cell Scheduling Function based on Parent-Child authentication* (SCSF-PC) and *Secure Cell Scheduling Function based on Weak Estimator Learning Automata* (SCSF-WELA) are proposed. In these solution schemes, as discussed in Chapter 5, rule-based and learning-based mechanisms are combined to ensure schedule stability and efficient resource utilization. Finally, it is worthwhile to mention that all the proposed solutions in this thesis are evaluated by extensive simulation experiments using the 6TiSCH simulator. The results demonstrate significant improvements in network performance, making the contributions of this thesis valuable for the advancement of IIoT networks.

As a whole, this dissertation presents substantial improvements in the performance of 6TiSCH networks, providing novel solutions to challenges in link scheduling that are essential for resource-efficient data communication in IIoT applications.

Keywords: IIoT, IEEE 802.15.4e, TSCH, 6TiSCH, Link Scheduling, RPL, Schedule Collision, MSF, 6P protocol, 6P Vulnerabilities

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

6LoWPAN IPv6 over Low power Wireless Personal Area Network.

6P 6TiSCH Operation Sublayer (6top) Protocol.

6TiSCH IPv6 over the TSCH mode of IEEE 802.15.4e.

6TiSCH-MC 6TiSCH Minimal Configuration.

ACK Acknowledgment.

AMCA Asynchronous Multi-Channel Adaptation.

ASN Absolute Slot Number.

BLINK Radio Frequency Identification Blink.

CCA Clear Channel Assessment.

CDR Cell Delivery Ratio.

CoAP Constrained Application Protocol.

CoJP Constrained Join Protocol.

CSMA/CA Carrier-Sense Multiple Access with Collision Avoidance.

DAO DODAG Advertisement Object.

DIO DODAG Information Object.

DIS DODAG Information Solicitation.

DODAG Destination Oriented Directed Acyclic Graph.

DSME Deterministic and Synchronous Multi-channel Extension.

EB Enhanced Beacon.

ETX Expected Transmission Count.

EWMA Exponentially Weighted Moving Average.

IEEE-SA Institute of Electrical and Electronics Engineers Standards Association.

IETF Internet Engineering Task Force.

IIoT Industrial Internet of Things.

IoT Internet of Things.

IPv6 Internet Protocol Version 6.

KPIs Key Performance Indicators.

LLDN Low Latency Deterministic Network.

LLN Low power and Lossy Network.

MAC Medium Access Control.

MSF Minimal Scheduling Function.

PDR Packet Delivery Ratio.

QoS Quality of Service.

RPL Routing Protocol for Low power and Lossy Networks.

RSSI Received Signal Strength Indicator.

SDN Software Defined Network.

SF Scheduling Function.

TDMA Time Division Multiple Access.

TSCH Time Slotted Channel Hopping.

WG Working Group.

"You have to dream before your dreams can come true."

∼A.P.J. Abdul Kalam (1931 - 2015)

1

Introduction

1.1 Internet of Things

Back in 1999, Kevin Ashton from MIT introduced the term Internet of Things (IoT), capturing the attention of governments and industrialists as a revolutionary technology poised to drive sustainable economic growth worldwide. IoT refers to a network of physical objects (often called "*things*" or "*nodes*") such as devices, vehicles, buildings and other items embedded with electronics, software, sensors, and network connectivity that enables these objects to collect and exchange data with manufacturers, operators and other connected devices. The core objective of IoT is to "connect the unconnected" [1–3]. The transition facilitated by IoT allows sensing and control of the physical world, effectively making ob-

jects smarter and integrating them into a connected network. Analysts predict that by 2025, there will be over 30 billion connected IoT devices. Today, IoT is employed in various application domains such as healthcare [4], smart homes [5, 6], smart cities [7, 8], smart grids [9, 10], smart transportation [11], industry [12, 13], and agriculture [14].

IoT is

The network of smart physical objects (e.g. devices, vehicles, buildings, etc.) embedded with sensors/actuators, computation unit, memory unit, power source, and network connectivity, which enables the physical object to collect and exchange data, analyze the collected data to extract new insight and respond accordingly.

IoT enables nodes to exchange information about their surroundings by establishing communication networks, allowing computational systems to interact with the physical world. Data collected from these nodes is analyzed to assess environmental conditions, and based on this analysis, IoT systems can modify the environment as needed [15, 16]. IoT devices, as shown in Figure 1.1, are equipped with sensors, actuators, and a processing unit to enable autonomous or semi-autonomous behaviour. The processing unit is responsible for acquiring data from sensors, analyzing this data, and generating control signals for actuators. It also manages other functions like communication and power management. The device typically includes on-chip flash memory for storing application data and program code. A communication unit facilitates connectivity with other devices and networks, enabling data exchange and remote control. Finally, a power source provides the necessary energy to operate all components of the smart object.

The nodes in IoT are considered *smart* [1, 17]) due to their three key capabilities: (i) sensing or monitoring their environment, (ii) communicating or exchanging information with other nodes, and (iii) altering the behaviour of their surroundings. Typically, nodes use various sensors for environmental monitoring and employ wireless communication to transmit data to a central controller. Environmental adjustments, known as *actuation*, are executed by actuator devices that respond to input from the nodes [18, 19]. These combined capabilities—sensing, communication, and actuation—enable IoT to enhance the efficiency,

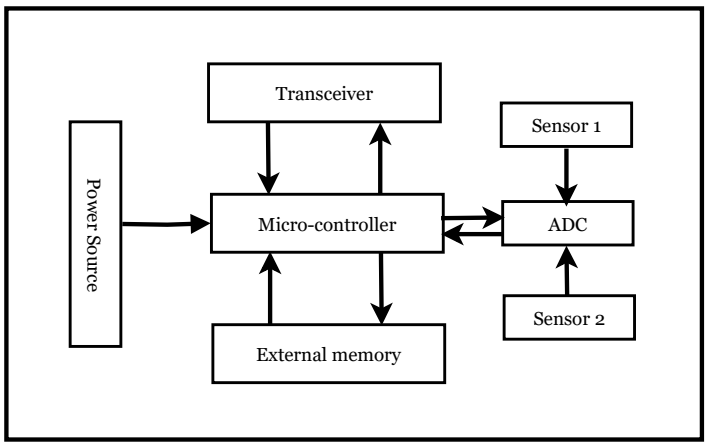


Figure 1.1: *Components of an IoT Node*

safety, and security of its physical-world applications. Most IoT devices[1] are resource-constrained with limited processing capability (typically only a few kilobytes (KBs) of RAM), less memory (also in KBs), and limited power supply (mainly battery operated) [20]. In recent years, numerous standards have been introduced to support different application domains of IoT [20–23]. These include *Bluetooth* [24], *WirelessHART* [25], *ZigBee* [26], *IEEE 802.15.4* [27], and *ISA-100.11a* [28]. These standards primarily operate in the 2.4 *Ghz* frequency band, while other standards such as IEEE 802.11.ah [29], LoRaWAN (Low Power and Long-range Wide Area Network) [30] and Narrowband Internet of Things (NB-IoT) [31] utilize the *Sub* $1GHz$ frequency band. Among these standards, *IEEE 802.15.4* is the most widely used protocol for resource-constrained Low power and Lossy Network (LLN) [20]. LLN are well-suited for IoT applications with devices having limited power, memory, and processing capacity, enabling communication among sensors and actuators in constrained conditions. Traditional IoT is not suitable for industrial applications with stringent requirements such as high reliability, deterministic latency, and scalability. Thus, Industrial Internet of Things (IIoT) has evolved and is specifically tailored for industrial applications.

[1]Note that we use the terms device, mote, and node interchangeably in this thesis.

1.2 Industrial IoT

The Industrial Internet of Things (IIoT) has emerged and gained significant attention from researchers and industrialists as it enables automated process and system monitoring, massive data collection and analytics for predictive maintenance, process control and automation. IIoT in particular aims to connect thousands of industrial things to the Internet to provide intelligent services to enhance the safety, scalability and efficiencies of numerous industrial applications. As shown in Figure 1.2, IIoT networks demand higher performance levels and have more stringent specifications due to their direct impact on productivity and safety compared to consumer IoT.

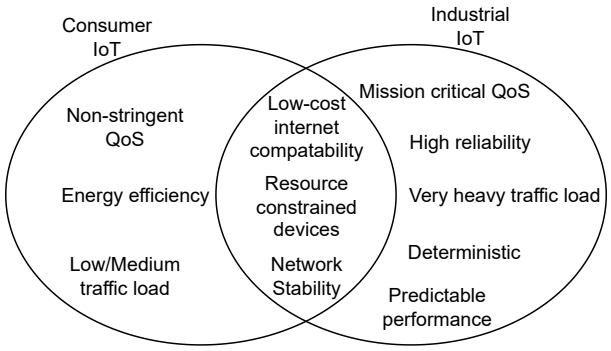


Figure 1.2: *Characteristics and requirements of consumer IoT and IIoT Networks*

IIoT applications require the underlying wireless communication protocols to provide high reliability, bounded latency, energy efficiency, and scalability. Therefore, it is very challenging to implement and fulfil various requirements of IIoT applications with resource-constrained devices. The underlying wireless communication protocols play a vital role in minimizing costs and improving the performance of industrial applications, notably in terms of latency, reliability, and power efficiency.

The IEEE 802.15.4 defines both the Physical (PHY) layer and Medium Access Control (MAC) layer specifications of the protocol stack and is considered to be suitable in various IoT applications. However, studies have shown that the IEEE 802.15.4 standard suffers from unbounded latency, limited transmission reliability, less protection against interference and multi-path fading [32], [33]. As a result, this standard fails to meet the stringent

requirements of different industrial applications. One key reason for these limitations is the usage of a single channel in the $2.4GHz$ frequency band, which does not incorporate a built-in frequency hopping mechanism to effectively mitigate interference and multipath fading. Furthermore, as it relies on the Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) algorithm, it cannot guarantee bounded delay and provides low reliability even with a lesser number of nodes [23]. On the other hand, ZigBee suffers from interoperability and scalability issues as it does not leverage existing communication infrastructure and relies on vendor-specific implementations. Similarly, WirelessHART and ISA-100.11a face scalability challenges due to their centralized approach to node communication management, which limits the reuse of resources. $Sub-1GHz$ based standards are tailored for long-range applications with low data rates, while LoRaWAN can operate at $2.4GHz$ band for higher data rates but with a reduced range.

Thus, to meet the emerging needs of industrial applications, the Institute of Electrical and Electronics Engineers Standards Association (IEEE-SA) published *IEEE 802.15.4e* [34] amendment in 2012 as an extension of the existing IEEE 802.15.4 [27], which redesigned the IEEE 802.15.4 MAC. The revised IEEE 802.15.4e [35] includes several enhancements to the MAC layer of the original protocol stack. It specifies five new MAC protocols known as MAC behaviour modes to support application-specific requirements and a few general enhancements applicable to all types of applications. These modes are Deterministic and Synchronous Multi-channel Extension (DSME), Time Slotted Channel Hopping (TSCH), Asynchronous Multi-Channel Adaptation (AMCA), Low Latency Deterministic Network (LLDN), and Radio Frequency Identification Blink (BLINK). In addition to modifications at the MAC layer, IEEE 802.15.4e enables simultaneous transmissions using multiple physical channels, with a maximum of 16 channels. It also supports channel hopping, allowing nodes to switch their communication channels after each transmission and thus helps to mitigate multi-path fading and interference on transmission channels in LLNs. Thus, IEEE 802.15.4e enhances throughput and reliability in resource-constrained LLNs.

1.3 IEEE 802.15.4 TSCH

Among the IEEE 802.15.4e MAC modes, TSCH and DSME have been incorporated in the next revision, IEEE 802.15.4-2016 [36], due to their broad applicability across various IoT domains. In particular, TSCH has emerged as a cornerstone technology for the advancement of Industry 4.0 because it provides bounded delay, high reliability and increases network capacity utilizing its Time Division Multiple Access (TDMA) based transmission channel access with multichannel and channel/frequency hopping capabilities. TSCH [37]

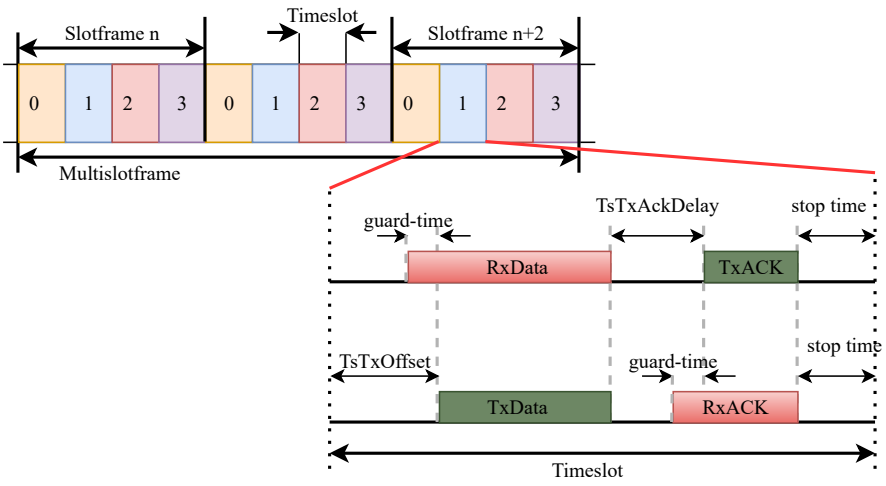


Figure 1.3: *An illustration of TSCH Slotframe structure with the sequence of events in a Timeslot*

is intended to meet the demands of industrial automation, process control and non-delay tolerant applications. Specifically, the TDMA feature of TSCH divides time into smaller sub-units of fixed duration, referred to as *timeslots*[1] which ensures deterministic delay and increases throughput by eliminating collision among nodes. Several consecutive timeslots constitute one slotframe, and it repeats cyclically over time, as shown in Figure 1.3. The length of a slotframe is determined by the number of timeslots it comprises, and every node must be synchronised to adhere to the slotframe schedule. On the other hand, the channel hopping feature mitigates the effects of interference and multipath fading and thus helps in improving network reliability. The multichannel feature allows multiple nodes to transmit

[1]Each timeslot is sufficiently long (mostly 10 ms) for a mote to transmit a packet (maximum 127 *Bytes* long) along with the reception of its Acknowledgment (ACK).

data at the same timeslot and thus increases network capacity.

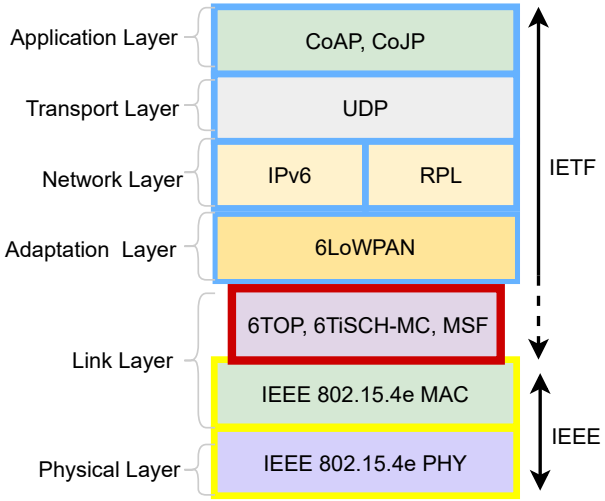


Figure 1.4: *6TiSCH protocol stack*

1.4 The 6TiSCH Network

As IEEE 802.15.4e defines only the lower layer of the protocol stack i.e. the low data rate PHY (250 *Kbps*) and MAC specifications, the Internet Engineering Task Force (IETF) created the IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) Working Group (WG) to connect the resource-constrained TSCH-enabled devices used in industry to the Internet. The main focus of this WG is to establish interoperability between TSCH and IPv6 [38] to bring IPv6 into industrial standards. It standardizes *6TiSCH network protocol stack* [20] under the open standard category, bridging IEEE 802.15.4e TSCH MAC-based multi-hop and lossy networks with existing IPv6 networks, as illustrated in Figure 1.4. This stack encompasses everything needed to operate IPv6 networks over resource-constrained TSCH networks. The protocol stack includes Routing Protocol for Low power and Lossy Networks (RPL) [39] for multihop routing, IPv6 over Low power Wireless Personal Area Network (6LoWPAN) [40] for header compression and Constrained Join Protocol (CoJP) [41] for secure joining procedures. 6TiSCH Minimal Configuration (6TiSCH-MC) [42] for 6TiSCH

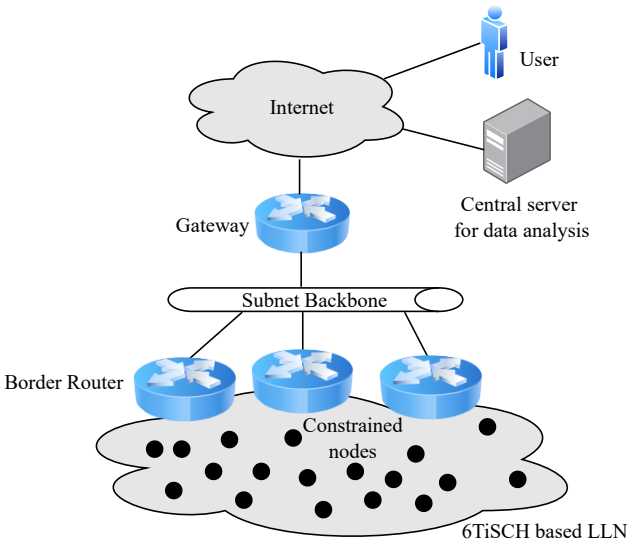


Figure 1.5: *A 6TiSCH based LLN network connected to the central server and end user and over Internet*

network bootstrapping or formation while 6TiSCH Operation Sublayer (6top) Protocol (6P) and Minimal Scheduling Function (MSF) mananges 6TiSCH scheduling, and Constrained Application Protocol (CoAP) enables low-overhead secure RESTful interaction. As such, 6TiSCH provides IPv6-based internet connectivity to the TSCH networks and is one of the key enablers of IIoT. Figure 1.5 illustrates how the 6TiSCH network gets connected to the traditional IPv6 network through a network gateway. Specifically, constrained nodes inside the LLN are attached to the Backbone through Border Routers, which then connect to the Internet via a gateway. Users can access data in real-time, or data can be stored on central servers for further processing through IPv6 connectivity.

1.5 Link Scheduling

In addition to interoperability challenges, one of the key tasks in 6TiSCH networks is *link scheduling*. TSCH relies on link scheduling for data communication within the network, allowing nodes to know exactly when to transmit or receive data. Link scheduling involves creating a schedule that determines how and when nodes interact in the network. Specifi-

cally, it informs nodes whether they should transmit, receive, or enter into sleep state during designated timeslots. A data communication schedule[1] consists of cells (represented as a

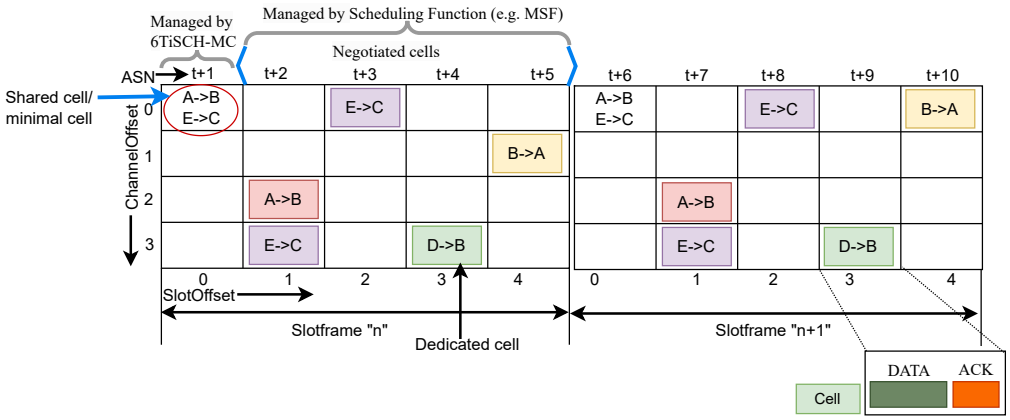


Figure 1.6: *An example link schedule considering 4 channels and a slotframe with 5 slots. The cell with (`SlotOffset`, `ChannelOffset`) as (0,0) is called the minimal cell and is used for broadcast traffic*

tuple of SlotOffset (Ts_{of}) & ChannelOffset (Ch_{of}) in a slotframe denoting the time and channel to transmit a packet). Cells can be either dedicated or shared. A dedicated cell is assigned to a single pair of nodes, making it contention-free, while a shared cell is used by more than one pair of nodes, leading to contention-based access. In shared cells, the standard Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) algorithm is employed to resolve contention. Shared cells handle the transmission of control packets for network formation and maintenance, whereas dedicated cells are designated for data packet transmission.

Figure 1.6 illustrates an example of a link schedule, where 3 channels and 5 slots are considered for demonstration purposes. In this figure, the notation B→A within a filled cell indicates that the cell is specifically designated for the transmission of a packet from node Ⓑ to node Ⓐ. The SlotOffsets and ChannelOffsets are used as cell indices (shown on the `X-axis` and `Y-axis`, respectively), and the slotframe repeats after 5 *timeslots*, with the slotframe length set to five for this example. Here, node Ⓓ transmits its data packet to node Ⓑ in the dedicated cell (3,3) of slotframe n. On the other hand, in the shared

[1]Throughout this thesis, the terms link schedule, communication schedule, and cell schedule are used interchangeably to represent link schedule.

cell of the same slotframe, node Ⓐ transmits to node Ⓑ and node Ⓔ transmits to node Ⓒ simultaneously. Efficient link scheduling is crucial for achieving reliable and timely communication in IIoT. The algorithm responsible for scheduling cells is referred to as Scheduling Function (SF). However, IEEE 802.15.4e TSCH MAC did not define how the schedule should be formed and managed, and it only executes a given schedule provided by the upper layers [38]. To address these concerns, recently, the 6TiSCH-WG released the Minimal Scheduling Function (MSF) [43] as a standardized scheduling function for 6TiSCH networks. In addition, a new sub-layer called 6TiSCH Operation Sublayer (6top) and its associated protocol 6P [44] has been introduced for managing the 6TiSCH schedule. It is important to note that transmission scheduling in *minimal cell* is done as per 6TiSCH Minimal Configuration (6TiSCH-MC) [42] and is used for broadcast traffic. According to 6TiSCH-MC standard, [`SlotOffset`=0 and `ChannelOffset`=0] are reserved for the minimal cell. The remaining cells, starting from the second timeslot to the last timeslot of the slotframe are scheduled by the SF of the network after/along with the formation of 6TiSCH network.

> *The IEEE 802.15.4e TSCH standard did not define any policy for forming and managing link schedules for IoT networks.*

Designing a link schedule is application-specific, tailored to meet particular requirements, and can be broadly classified into *Centralized*, *Distributed*, and *Autonomous* approaches based on how TSCH resources (cells) are allocated. In *Centralized* approaches, a single node, typically the network coordinator, is responsible for computing, distributing, and maintaining the schedule for all nodes based on traffic and topology information received from all other nodes. The *Distributed* approach allows each node to compute its schedule independently through negotiation with neighbouring nodes, using limited, locally exchanged information. In the *Autonomous* approach, nodes autonomously create their schedules based on network topology without any path reservation, central coordinator, or neighbour-to-neighbour negotiation.

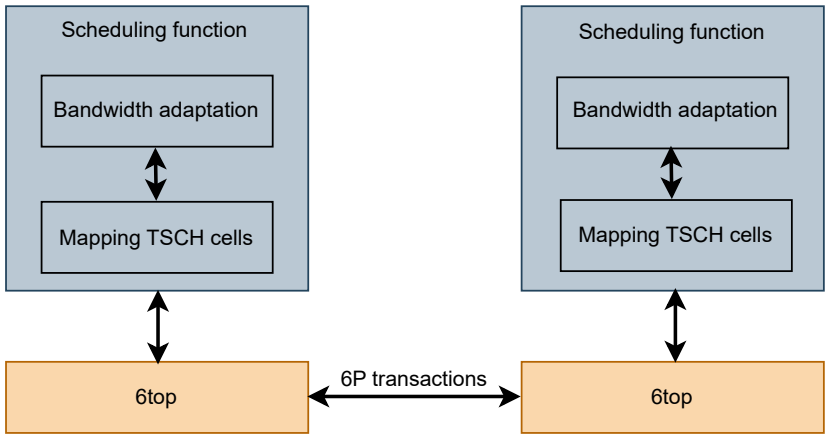


Figure 1.7: *Role of the Scheduling Function (SF) and 6top in the 6TiSCH architecture*

In 6TiSCH networks, the 6P protocol and SF work together to enable distributed dynamic scheduling. The 6P protocol facilitates local node-to-node cell negotiations, allowing nodes to update their schedules by triggering 6P transactions based on the designed SF of the network. The SF acts as the decision-making entity, determining (i) when to add or remove cells from the schedule, (ii) how many cells should be allocated for communication, and (iii) which cells should be selected for transmitting or receiving packets. Conceptually, the SF serves as an upper-level bandwidth adaptation technology that interacts with underlying services to map TSCH cells to one-hop neighbours. This interaction is visually depicted in Figure 1.7. It is noteworthy that a 6TiSCH network operates with three types of cells: *minimal*, *autonomous*, and *negotiated*. The 6TiSCH-MC [42] standard defines the allocation of the minimal cell, which is used for default resource provisioning during network bootstrap. Autonomous cells are managed autonomously without requiring negotiation and are used to perform 6P communications. On the other hand, negotiated cells are designated for data packet transmission and are established through 6P negotiations. The allocation and management of autonomous and negotiated cells are handled by the scheduling function, MSF. In addition, MSF outlines the policies for designing and managing communication schedules in a distributed manner and provides guidelines for adapting schedules in response to traffic patterns, changes in parent nodes, and schedule collisions.

1.6 Security in Link Scheduling

The 6TiSCH protocol stack integrates the Minimal Security Framework [41] as an integral part of its design. This framework defines a set of fundamental security features aimed at ensuring confidentiality, integrity, and freshness of messages exchanged within the network, including 6P messages. The standardized link scheduler MSF (RFC 9033) adheres to the 6TiSCH Minimal Security specification as outlined in [41]. In particular, MSF did not define any new security protocol for scheduling.

1.7 Motivation of the Research Work

The IEEE 802.15.4e standard leaves several research challenges open to researchers. A few of them are efficient scheduling and management of TSCH cells, the evaluation of protocols under varying network conditions, improving the network formation process, and enhancing security within 6TiSCH networks. Among these challenges, efficient design and secure management of TSCH schedules for 6TiSCH networks is the primary focus of this thesis. This is because the IEEE802.15.4e TSCH standard lacks guidelines on how to design and manage link scheduling functions, derive appropriate link schedules, or effectively coordinate between nodes. The way a link schedule is defined impacts Key Performance Indicators (KPIs) of the network, such as end-to-end latency, reliability, network capacity, and the battery lifetime of the nodes. Although transmission scheduling for Time Division Multiple Access (TDMA)-based networks has been a prominent research area in recent years, most TDMA scheduling algorithms have traditionally been designed for single-channel networks. Multi-channel TDMA scheduling solutions have only recently emerged in the literature. However, these solutions are not well suited for IoT networks because (i) they are not intended for resource-constrained nodes, (ii) they do not support packet-wise channel hopping, and (iii) they are not efficient in terms of channel utilisation as they do not consider spatial reuse of channels. Due to these limitations, new scheduling approaches have recently been proposed and studied in the literature for TSCH-based IoT networks like 6TiSCH.

Designing an efficient scheduling function for 6TiSCH networks is challenging, especially in large-scale, multi-hop networks. Optimizing link scheduling is crucial for resource-efficient communication and enhancing the performance of IIoT applications. To address these needs, researchers have proposed various link scheduling schemes for improving different Quality of Service (QoS) requirements. Centralized schedulers (for e.g. [45, 46]) rely on a central node to compute and distribute the schedule to all other nodes. However, these centralized schemes are generally not suitable for IIoT applications due to significant signalling overhead caused by information flow from each node to the central node and vice versa for schedule creation. Moreover, centralized schemes are difficult to manage in large, dynamic networks where a change in topology and traffic demand is common, and such changes demand re-computation and redistribution of the schedules. Considering the focus on topology and traffic-aware networks with low signalling overhead makes distributed scheduling advantageous. This is particularly important because 6TiSCH networks are composed of resource-constrained nodes with limited memory and computational power [47]. As a result, distributed scheduling is considered more suitable for IIoT applications. A number of works based on distributed link scheduling approach ([43, 48–54]) also exist in the literature. However, a common limitation among the existing works is their lack of adaptability to network and traffic dynamics, and are not suitable for all traffic rates. While the IETF standardized MSF [43] adapts dynamically to traffic rate changes, it lacks an efficient cell selection method, leading to random cell allocation and deallocation. Because of this, MSF cannot always achieve low end-to-end latency. Furthermore, MSF updates (adds or deletes) the number of scheduled cells by unity, irrespective of the actual traffic requirement, which incurs high signalling overhead on 6P. In addition, the random cell allocation strategy can not achieve a collision-free schedule in TSCH networks.

Since scheduling in 6TiSCH is entirely distributed, there is a nonzero probability that two pairs of nearby neighbouring nodes schedule a negotiated cell at the same [`SlotOffset`, `ChannelOffset`] position in the TSCH schedule. In such cases, the data exchanged by these two pairs may collide on that cell, and the situation is referred to as schedule collision [43].

Multiple colliding cells within a schedule lead to increased interference and packet loss, degrading both network reliability and overall performance. However, only a few studies (e.g. [43,55,56]) have addressed schedule collision detection and mitigation. None of the schemes, including the standard MSF [43], effectively detects and handles all collision scenarios. For mitigation, current approaches rely on inefficient sender-based random relocation strategy, resulting in excessive relocation requests and increased network overhead. Furthermore, the

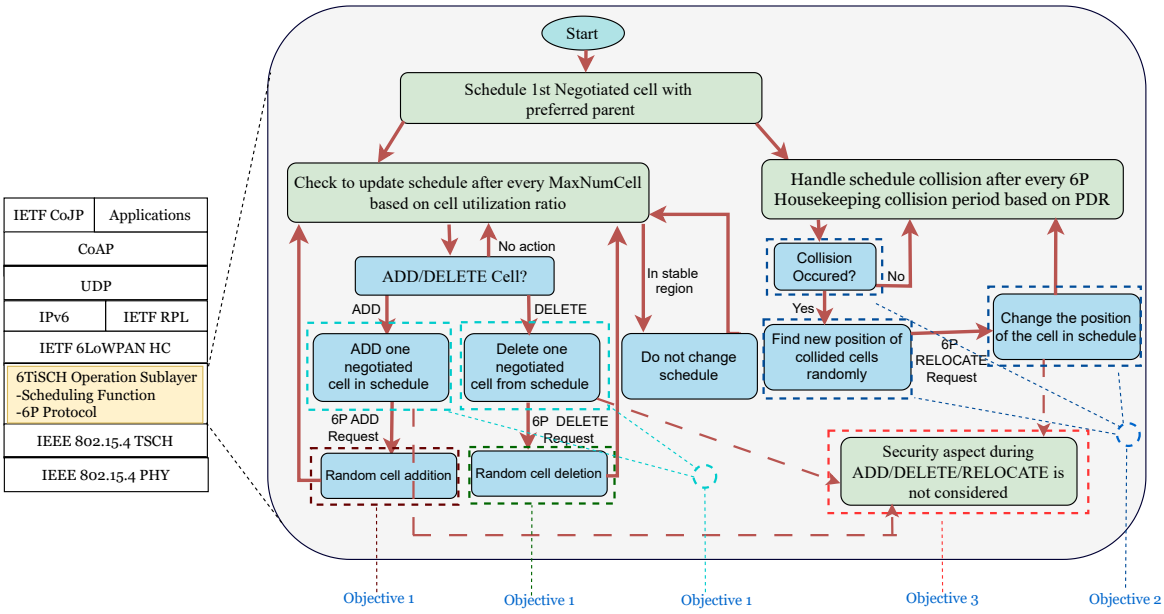


Figure 1.8: *Illustrating core functionalities of a Scheduling Function (SF) and 6P Protocol following the MSF standard (RFC 9033)*

literature reveals a gap in addressing the feasibility of attacks and their potential impacts on cell scheduling and the 6P protocol. Despite the implementation of secure communication through Minimal Security Framework [41], the link scheduling and 6P protocol exhibit several security vulnerabilities [57]. Any attack targeting the SF or 6P can compromise the desired functioning of link scheduling and can impair the normal resource allocation procedures, potentially leading to schedule instability, reduced data delivery and increased energy consumption in affected nodes. However, this issue is currently less investigated in the literature. Most existing studies [58–61] on the security of 6TiSCH focus on aspects such as time synchronization, routing, and jamming attacks, leaving the vulnerabilities of the link scheduling process less explored. The only works in [62] and [57] present an assessment

of the security vulnerabilities of 6P, which is responsible for resource negotiation during schedule formation and maintenance in 6TiSCH networks. Figure 1.8 illustrates the core functionalities of the scheduling process following the MSF standard with our contributions highlighted in different events of the scheduling process. The scheduling process begins by scheduling the first negotiated cell with its preferred parent using the 6P protocol. Following this, the schedule undergoes periodic checks after reaching a predefined maximum number of cells, assessing cell utilization to decide if cell updation is needed. Depending on utilization metrics, a single cell may be added or removed through 6P ADD or DELETE requests. To manage schedule collisions, a housekeeping process assesses the Packet Delivery Ratio (PDR) after each period, relocating collided cells to new positions as needed.

Therefore, the existing issues and unexplored research directions of 6TiSCH link scheduling motivate us to develop improved schemes which can address the issues. Thus, the main aim of this dissertation is to design, manage and secure the link scheduling function and its associated 6P protocol to improve the performance of the multi-hop 6TiSCH network. In brief, the key motivations for this thesis work are as follows:

- *Ill-suited TDMA-based Solutions*: Existing TDMA-based scheduling solutions are not well-suited for TSCH networks. Multi-channel solutions tailored to TSCH are required.

- *Need for Distributed Scheduling*: Centralized schemes are not suitable for IIoT applications due to their high signalling overhead during schedule creation. They are also challenging to manage in large, dynamic networks where topology and traffic demand frequently change, requiring the re-computation and redistribution of schedules. Given the focus on topology-aware, traffic-aware networks, distributed scheduling with minimal signalling overhead is more advantageous for IIoT applications.

- *Lack of Link Scheduling Guidelines*: The IEEE 802.15.4e TSCH standard lacks guidelines on how to design and manage link schedules. Recently, the proposed MSF guidelines provide basic functionalities for scheduling. However, it suffers from many disadvantages, such as high 6P signalling overhead, cell adaptation in unity regardless of

the demand, random cell allocation and deallocation.

- *Need for Resource-efficient Communication for Schedule Formation and Maintenance*: Optimizing link scheduling scheme is crucial for improving the performance and efficiency of 6TiSCH networks.
- *Issues of Traffic Adaptiveness in Existing Scheduling Solutions*: Except MSF, the existing solutions lack adaptability to network dynamics and are unsuitable for all traffic rates. The IETF standardized MSF faces high signalling overhead and random cell selection issues for cell allocation and deallocation.
- *Handling Schedule Collisions*: Existing schemes fail to effectively detect and mitigate schedule collisions, leading to significant degradation in network performance.
- *Security Vulnerabilities in Scheduling Functions*: The Minimal Security Framework proposed by IETF can not prevent attacks on MSF or any other SFs proposed for IIoT networks. Any attacks on the scheduling process can compromise the desired behaviour of link scheduling and thus the networks.

Thesis goal

Enhance the performance of 6TiSCH link scheduling in terms of end-to-end delay, reliability, signalling overhead, schedule stability, and energy efficiency by designing adaptive and secure scheduling functions complying with the IEEE 802.15.4 TSCH and IETF MSF standards.

1.8 Thesis Objectives

Industrial IoT applications demand high reliability, scalability, bounded latency, and energy efficiency, which all depend heavily on the effective design and management of the underlying data communication schedule. Many existing scheduling functions (SFs) fall short in adapting to dynamic network conditions such as changes in topology, traffic demand, and

channel availability. Factors like node position, density, and application behaviour further necessitate adaptive scheduling in IIoT networks. With the local view of the network, multiple pairs of neighbouring nodes may schedule the same negotiated cell for communication which may lead to schedule collisions. Mitigating schedule collisions is crucial for network performance because, when a collision occurs, it persists across slotframes, and thus, degrades network reliability and increases energy consumption. Further, vulnerabilities within the SF and its associated 6P protocol remain largely unexplored. Therefore, the objectives of this thesis are framed to improve the performance of multi-hop 6TiSCH network link scheduling in terms of reliability, end-to-end delay, energy efficiency, 6P overhead, schedule stability, and security. In brief, the research objectives of this Thesis are described as follows:

- *Adaptive Low-latency Distributed Link Scheduling Function*: Proposing schemes to reduce 6P signaling overhead, reduce end-to-end latency, and enhance transmission reliability and energy efficiency.
  - Proposing a scheme to dynamically adapt the schedule according to network dynamics to improve resource allocation and reduce the signalling overhead.
  - Designing a cell selection and allocation approach aimed at minimizing end-to-end latency while enhancing transmission reliability and energy efficiency.
  - Proposing an enhanced cell deletion scheme to minimize end-to-end latency.
- *Collision Detection and Mitigation in Link Schedules*: Developing a scheme to effectively detect and mitigate schedule collisions, improving overall network performance.
- *Securing Link Scheduling Function and 6P Protocol*: Investigating vulnerabilities in link scheduling function and the associated 6P protocol, assessing their impacts, and finally designing effective mitigation strategies for those vulnerabilities.
  - Demonstrating the feasibility of cell depletion and schedule instability attacks within the SF module of the 6TiSCH network.

– Proposing effective strategies to mitigate these threats and enhance the overall security of the system.

1.9 Challenges

This thesis addresses several challenges in the context of IEEE 802.15.4-based 6TiSCH networks, where efficient scheduling and resource management are crucial. These networks operate under tight constraints, including limited energy, bandwidth, and memory, while also needing to support dynamic topologies and varying traffic demands. Determining the optimal allocation of time slots and channels for each link to transmit data, to minimize collisions and maximize network performance. This is a crucial aspect of TSCH, as effective link scheduling directly impacts reliability, latency, and energy efficiency. 6TiSCH networks can be dynamic, with changing traffic patterns, node availability, and network topology. The link scheduling function must be adaptable to these changes. Meeting these goals is especially difficult when node positions and traffic patterns shift over time and space. Moreover, in many scenarios, the network must operate without prior knowledge of the topology, making it impossible to pre-compute optimal scheduling parameters. The scheduler must, therefore, be flexible enough to adapt on the fly. Many IIoT applications require reliability and low latency while also aiming for long network lifetimes under different traffic rates. Poor scheduling can lead to frequent collisions, increased retransmissions, and wasted energy. The dynamic nature of such networks, with nodes joining, leaving, or changing roles, demands that the scheduling algorithm be both adaptive and efficient. Although collisions are impossible to completely avoid in a distributed setting, robust strategies can significantly reduce their frequency and impact.

One of the core challenges lies in dynamically adapting the communication schedule to adapt to changing network conditions. In a 6TiSCH network, each node independently decides when to request or release communication cells based on network dynamics. This forms a distributed resource allocation problem, similar in nature to load balancing and congestion control where global coordination is infeasible and decisions must be made locally

and quickly. While such problems are known to be NP-hard in general, practical solutions often rely on heuristic or feedback-based methods that balance efficiency and responsiveness. Another key challenge is minimizing end-to-end latency while ensuring reliable and energy-efficient transmissions. This is a multi-objective combinatorial optimization problem, requiring the selection of transmission cells that minimize delay and avoid contention. Making such decisions involves evaluating factors like slotOffset positions, link quality, and current utilization. Although the bounded size of the slotframe makes the search space manageable, the task becomes non-trivial in dense deployments. Latency-aware scheduling, therefore, requires intelligent heuristics that can perform well under partial and localized information. Another issue arises in the deletion of underutilized cells. While the standard MSF applies arbitrary deletion mechanisms, they often do so without considering latency or future traffic demand. Deleting the wrong cells can degrade performance or increase end-to-end delay. This makes the cell deletion task analogous to a backward resource reallocation problem, where careful decisions must be made to undo prior allocations without disrupting active or upcoming transmissions. Schedule collisions are another significant challenge. Collisions occur when two or more nodes unintentionally schedule transmissions on the same cell, leading to packet loss. Detecting such collisions is difficult in a distributed environment, where each node only has partial knowledge of its neighbour's schedules. This problem is comparable to conflict detection in dynamic graphs and bears similarities to the hidden terminal issue in wireless networks. Further, any attacks on the scheduling process can compromise the desired behaviour of link scheduling and thus the networks.

1.10 Thesis Contributions

This thesis aims to improve the performance of multi-hop 6TiSCH link scheduling by addressing key challenges in distributed scheduling, schedule collision detection and mitigation, and security vulnerabilities in scheduling functions. We propose several schemes to address these issues and ensure efficient, secure link scheduling operations. The contributions of this thesis are organized into three parts, each corresponding to an objective mentioned above

and are presented in a separate chapter in this thesis. The following subsections briefly describe each of these contributions one by one.

1.10.1 Adaptive Low-latency Distributed Link Scheduling Function

At first, we exemplified the shortcomings of the MSF (RFC 9033) standard in the IIoT environment and observed that the negotiated cell addition and deletion are performed linearly irrespective of the traffic demands, which leads to excessive 6P signalling overhead. To overcome this, we first propose the *Improved Minimal Scheduling Function* (IMSF). The main goal of IMSF is to reduce 6P control overhead while improving end-to-end latency and transmission reliability. This is done by updating the number of schedule cell by *multiple cells together* instead of updating cells by unity. IMSF first *determines the number of required cells* to support the desired communication in the next cycle. Accordingly, the 6P negotiation is carried out so the node can judiciously update its resources to meet the expected traffic demand. Further, to address the issue of random cell selection and then cell allocation, we propose a *Receiver-based Traffic rate-agnostic Distributed link Scheduling function* (RTDS) for 6TiSCH networks. RTDS builds on the principles of IMSF for determining how many cells need to be added/or (deleted) to/(or from) the schedule to make it adaptive to present network conditions and traffic demand. However, it replaces the random cell selection method with the *receiver-based*, *delay-aware cell selection and allocation* strategy. In this approach, the receiver, in collaboration with the sender, utilizes its available free slots to judiciously select cells for allocation from the free list. Instead of randomly selecting cells for allocation, the receiver employs a delay-aware cell selection strategy that aims to minimize end-to-end latency by allocating transmission cells as closely as possible for successive nodes along the path from source to sink in multihop networks. This reduces the buffering delay at intermediate nodes, reducing packet transmission time. Additionally, we propose a mathematical model to theoretically estimate and compare the end-to-end delay for both the random cell selection strategy and the proposed receiver-based delay-aware

cell selection strategy. However, the problem of selecting cells for deletion remains unexplored. It is observed that a random cell deletion can elevate end-to-end latency and energy consumption. To address this, we propose a *Latency-aware Cell Deletion* (LCD) scheme to minimize associated average end-to-end latency and energy consumption, which in turn enhancing the efficiency of distributed link scheduling in 6TiSCH networks. Finally, we perform extensive simulation experiments to evaluate the performance of all three proposed schemes - IMSF, RTDS and LCD individually using the 6TiSCH simulator[1] [63].

1.10.2 Collision Detection and Mitigation in Link Schedules

Existing schedule collision detection and mitigation schemes based on Packet Delivery Ratio (PDR) do not address all collision scenarios comprehensively. For example, they cannot detect collisions when a single cell is scheduled between a node and its preferred parent or when all cells towards the parent experience collisions. These schemes typically rely on a sender-based relocation strategy, where the transmitting node initiates a 6P relocation request to its parent to relocate the collided cells. However, the preparation of the candidate cell list (*CCellList*) by the sender is done randomly, without considering the receiver's available slot information. Similarly, the parent's selection of cells from the provided *CCellList* is also random. Our observations indicate that many relocation attempts fail due to a mismatch between the sender's proposed cells and the receiver's available slots. As a result, even though the receiver may have available slots, the collided cells cannot be relocated. Additionally, if the number of available slots at the sender is fewer than the required length of the *CCellList*, the sender does not prepare the list or initiate a relocation request, even when slots are available for relocation. These challenges arise primarily due to the default sender-based relocation mechanism, resulting in a high rate of unsuccessful relocations. This, in turn, leads to an increase in 6P relocation transactions, and the collisions persist until a successful relocation is completed, ultimately degrading network performance. To address these issues, this thesis introduces an *Enhanced Collision Detection* (ECD) mech-

[1]https://bitbucket.org/6tisch/simulator/

anism based on the Rule-based and Exponentially Weighted Moving Average (EWMA) of *Cell Delivery Ratio* (CDR). Furthermore, we propose a *receiver-based cell relocation strategy* to address the inefficiencies of the sender-based mitigation approach. We called the proposed scheme as *ECD with cell Relocation* (ECDR). By leveraging the knowledge of its own available slots and the sender's busy slots, the receiver can make more accurate decisions about cell relocation, avoiding the random cell selection with a *delay-aware cell selection* for relocation of colliding cells. Finally, the effectiveness of the proposed schemes is evaluated through simulation experiments using the 6TiSCH network simulator and proved effective in improving the overall network performance.

1.10.3 Securing Link Scheduling Function

It is observed that the security vulnerabilities within the MSF and its associated 6P have not been fully explored in the literature including our previously proposed works. This thesis identifies few vulnerabilities present in the scheduling functions. First, we demonstrate the feasibility of *cell depletion attack* in MSF. This attack exploits vulnerabilities in parent nodes' knowledge of their legitimate child nodes and manipulates the cell scheduling process by deliberately requesting or occupying excessive time slots. It causes resource exhaustion and deprives legitimate nodes of their fair share of resources. Initially, we propose a *Secure Cell Scheduling Function based on Parent-Child Authentication* (SCSF-PC) scheme to detect and mitigate cell depletion attack. However, the potential threat of *schedule instability attack* posed by compromised child nodes remains unaddressed, where attacker frequently induces incorrect cell allocations and deallocations requests, creating significant schedule instability and thereby disrupting the network's overall functionality. Being unaware of the attack, the parent node accepts these requests as they appear to be standard 6P operations between a child and a parent, causing resource imbalances and leading to communication failures, increased latency, and energy wastage. The minimal security framework (RFC 9031) standardized by IETF cannot prevent such attacks on the MSF as the commonly shared network key does not provide strict source authentication for data messages at the

individual node level. Our simulation results validated the severity of these attacks. To alleviate these vulnerabilities, we extend the SCSF-PC scheme by introducing the *Secure Cell Scheduling Function based on Weak Estimator Learning Automata* (SCSF-WELA) scheme. This is a comprehensive mitigation strategy that combines rule-based decisions and WELA-based estimation to identify and prevent any malicious actions from adversary. This strategy effectively identifies compromised nodes and blocks their malicious actions, ensuring the network's stability and performance. Finally, we validate the effectiveness of the proposed defence schemes using extensive simulation experiments on 6TiSCH simulator.

1.11 Organization of the Thesis

The thesis consists of six chapters. Figure 1.9 illustrates the organization of this thesis, which maps the different chapters onto the area of our work and the corresponding publications. In brief, the rest of the thesis is organized as follows:

Chapter 1: Introduction

Introduction, motivation, and contributions of the thesis are presented in this chapter.

Chapter 2: Background and Literature Survey

This chapter provides detailed background on the 6TiSCH network, with a particular focus on the standardized MSF and the related 6P protocol, followed by the discussion on state-of-the-art works related to 6TiSCH link scheduling. It also describes the experimental environment used for measuring the performance of different schemes.

Chapter 3: Adaptive Low-latency Distributed Link Scheduling Function

This chapter proposes three schemes, namely IMSF, RTDS and LCD, to provide an adaptive low-latency distributed scheduling function for 6TiSCH network. Theoretical analysis and simulation experimental results are provided to validate the proposed

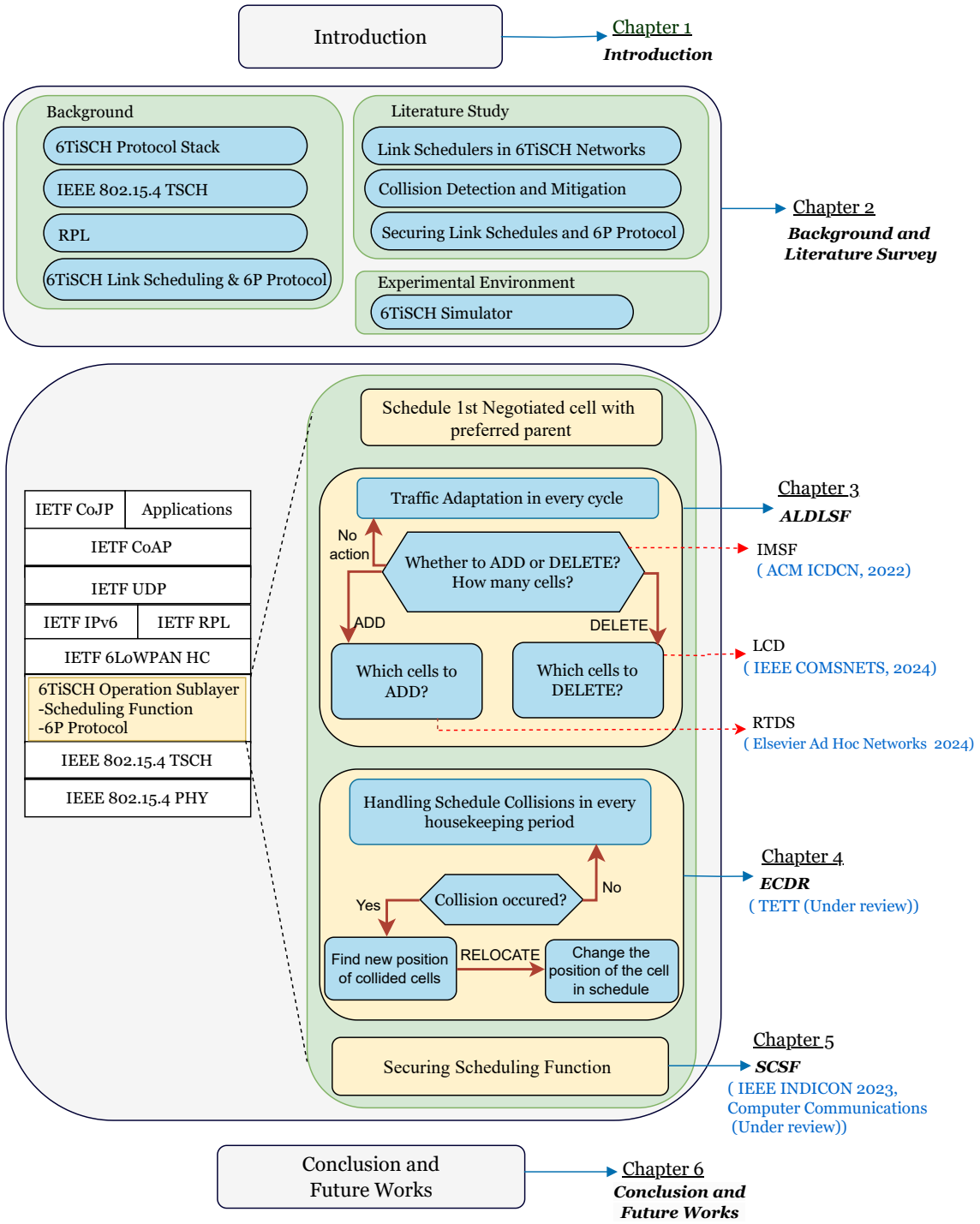


Figure 1.9: *Area of our work-wise organization of the thesis*

schemes.

Chapter 4: Collision Detection and Mitigation in Link Schedules

This chapter examines the limitations of existing schedule collision detection and mitigation mechanisms in 6TiSCH networks. It introduces two novel schemes, ECD and ECDR, aimed at addressing these collisions to enhance IIoT network performance. The chapter concludes by presenting experimental results that validate the effectiveness of the proposed solutions.

Chapter 5: Securing Link Scheduling Function

This chapter investigates vulnerabilities in the MSF scheduler and 6P protocol, highlighting the *cell depletion* and *schedule instability* attacks. It proposes two mitigation schemes, SCSF-PC and SCSF-WELA, to address these threats. Simulation experimental results are provided to validate proposed schemes.

Chapter 6: Conclusions and Future Directions

Finally, this chapter concludes the thesis by summarizing the work done and outlining research directions for possible future work on link scheduling in 6TiSCH.

"The best way to predict your future is to create it."

~Abraham Lincoln (1809 - 1865)

2

Background and Literature Survey

This chapter provides a brief overview of the key concepts necessary for understanding the work presented in this thesis. It begins by discussing the overall IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) protocol stack with a focus on specific protocols relevant to this study, including the IEEE 802.15.4e Time Slotted Channel Hopping (TSCH) MAC protocol. Since the objectives of this dissertation focus on 6TiSCH link scheduling, we discuss the standardized Minimal Scheduling Function (MSF) and the 6TiSCH Operation Sublayer (6top) Protocol (6P) in detail. Thereafter, the chapter presents a comprehensive literature review on link scheduling for 6TiSCH networks and a description of the experimental setup used to evaluate the proposed solutions in this dissertation.

2.1 Background

2.1.1 6TiSCH Protocol Stack

The 6TiSCH Working Group (WG) was established by the IETF with the goal of enabling IPv6-based Internet connectivity for IEEE 802.15.4e TSCH networks. The primary aim of 6TiSCH is to merge the high reliability and low energy consumption features of TSCH with the seamless interoperability and integration of the IP protocol. 6TiSCH has been a significant effort to bring IPv6 to industrial low-power wireless networks, effectively bridging TSCH networks with 6LoWPAN adaptation layer [64]. Currently, the WG focuses on ensuring interoperability between IEEE 802.15.4e TSCH and the IETF upper-layer protocol stack and the efficient scheduling of communication cells based on signals from the TSCH link layer and other upper layers.

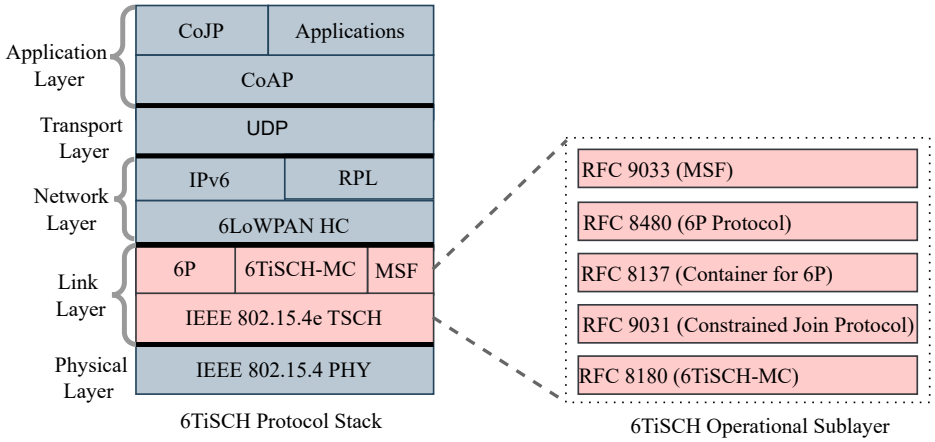


Figure 2.1: *The 6TiSCH network protocols stack*

Figure 2.1 demonstrates the complete protocol stack of the 6TiSCH network, detailing the various standards operating at each layer. The protocol stack utilizes the IEEE 802.15.4 standard at the Physical layer, a wireless technology designed for short-range, low-rate communication [20], [65], and [38]. At the MAC layer, TSCH is employed to ensure high reliability, predictable latency, guaranteed bandwidth, and low energy consumption. The WG released 6TiSCH Minimal Configuration (6TiSCH-MC) (RFC8180) [42] for bootstrapping of the network and a Minimal Scheduling Function (MSF) (RFC9033) [43].

Above the TSCH layer, the 6P operation sublayer (RFC8480) [44] facilitates distributed dynamic schedule management. At the network layer, the 6LoWPAN framework [66] integrates and compresses IPv6 datagrams into TSCH frames using stateless header compression and fragmentation as mentioned in IPv6 over IEEE 802.15.4 (RFC4944) [67] and contextual header compression techniques as mentioned in IPv6 Datagram Compression for IEEE 802.15.4 (RFC6282) [68], 6LoWPAN Paging Dispatch (RFC8025) [69] and 6LoWPAN Routing Header (RFC8138) [40]. Routing Protocol for Low power and Lossy Networks (RPL) is the routing protocol to relay IPv6 packets across multiple hops, enabling efficient routing and network representation in the form of Destination Oriented Directed Acyclic Graph (DODAG) topology following the Objective Function (OF) (RFC6552) [70]. At the application layer, the Constrained Application Protocol (CoAP) (RFC7252) [71] enables lightweight and RESTful interactions among IoT devices. Additionally, the Constrained Join Protocol (CoJP) (RFC9031) [41] supports the secure joining of IoT nodes into the network. Note that the IIoT uses this protocol stack under open standard category for data communication to meet the stringent needs of IIoT applications.

2.1.2 IEEE 802.15.4 TSCH

IEEE 802.15.4 TSCH is the MAC layer protocol to support Industrial IoT applications with low latency, guaranteed packet delivery and energy efficiency requirements. TSCH merges channel hopping features with time-slotted access. It divides time into small intervals called timeslots. These timeslots are sufficiently long, usually 10 ms, to accommodate the transmission of a packet and the reception of its Acknowledgment (ACK). These timeslots are grouped into repeating sequences known as slotframes that repeat cyclically over time to form/maintain a communication schedule for both data and control packet transmission. The length of a slotframe is determined by the number of timeslots it comprises. A node can operate in one of three radio states during a given timeslot: *receiving* (`Rx`), *transmitting* (`Tx`), or *idle* and can use only one physical channel at the given timeslot for either `Tx` or `Rx`. Each timeslot can use multiple available frequencies, forming a matrix-like schedule where

different nodes can communicate simultaneously on different channels at each timeslot. A specific *cell*[1] in the slotframe is represented by a pair (Ts_{of}, Ch_{of}), where Ts_{of} is the position of the timeslot in slotframe and Ch_{of} corresponds to an integer value between 0 to 15 that will be mapped to an actual channel at each repetition of slotframe.

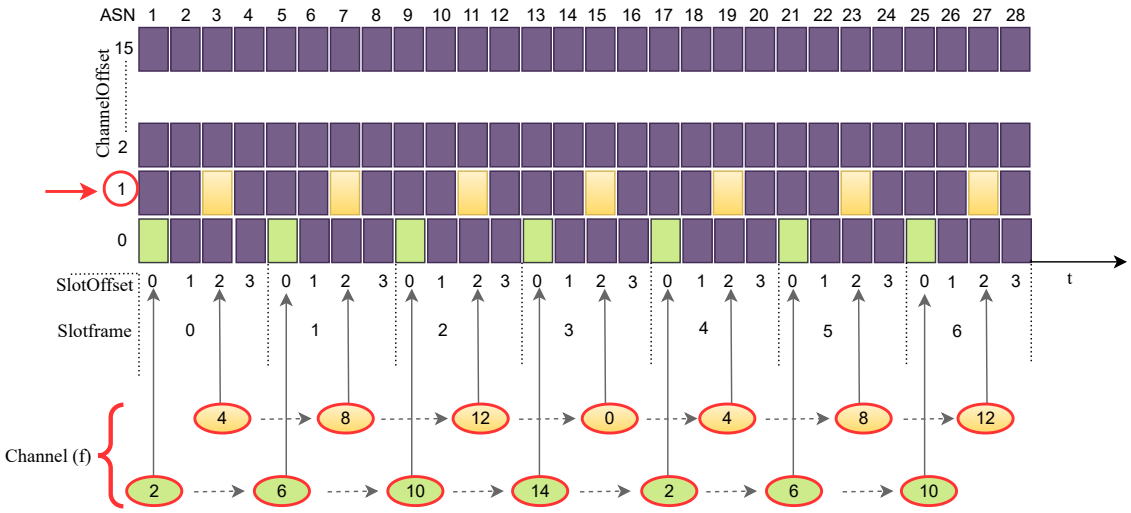


Figure 2.2: *An example illustrating channel hopping feature of TSCH*

The channel hopping feature of TSCH leverages the availability of multiple channels, enabling nodes to use different frequencies for each subsequent frame, as illustrated in Figure 2.2. This technique helps to mitigate the effects of multipath fading and external interference, enhancing communication reliability and thus useful to address the unpredictable radio link behaviours often encountered in industrial environments. Since transmitting nodes change their communication channels in every timeslot, the receiving nodes must also adjust their receiving channels accordingly, ensuring synchronization with the transmitting nodes' channel hopping sequence. Thus, the actual radio frequency used for a given cell maintaining this channel synchronization between transmitter and receiver is calculated using the following equation:

$$f = F\ (ASN\ +\ Ch_{of})\ \%\ N_c \text{ and } ASN = k * SF_{length} + Ts_{of} \tag{2.1}$$

Where Absolute Slot Number (ASN) indicates the total number of slots that have elapsed

[1]A cell is represented by the tuple [`SlotOffset` and `ChannelOffset`], which denotes the time and channel together in the slotframe to transmit a packet

since the network's initialization. It is an integer that begins at 0 and increments by one with each timeslot, and it is the same for all the nodes in a network. Ch_{of} is the logical ChannelOffset to identify a specific communication channel. N_c is the total number of available channels, k signifies the number of slotframe cycles since network inception, Ts_{of} is SlotOffset, SF_{length} corresponds to the length of the slotframe. F is a lookup table containing available channels, and f is the calculated actual radio frequency. The output of this equation provides a pseudo-random channel hopping sequence.

Figure 2.2 illustrates an example of the channel hopping mechanism. In this example, a pair of nodes is allocated the `SlotOffset=2` and `ChannelOffset=1`, resulting in the cell [2, 1] as determined by the scheduling algorithm. According to Equation (2.1), the transmitter and receiver nodes switch their physical communication channels during the assigned timeslot, transitioning, for instance, from channel 4 to 8, then from 8 to 12, and so on. Besides channel hopping, TSCH supports concurrent transmissions on multiple channels, thereby increasing network throughput and capacity. This is achieved by assigning different `ChannelOffsets` to nodes that share the same `SlotOffset`. Note that there are 16 available channels, each identified by a unique `ChannelOffset`.

2.1.3 RPL

RPL enables multi-hop data delivery in a Low power and Lossy Network (LLN) with multiple source nodes and one sink node. Let $G(N, L)$ be a LLN where N is the set of all nodes and L represents the links between these nodes. A link exists when two nodes are within each other's transmission range. RPL builds and manages a DODAG with the central node serving as the root and directing all traffic towards the root. The DODAG is built using the Objective Function (OF) considering metrics like minimum Expected Transmission Count (ETX) or hop count. A DODAG is organized based on the rank assigned to each node, which represents its logical distance from the destination. To prevent routing loops, RPL maintains the monotonic property of rank and ensures that a node does not select a parent with a higher rank than itself. Within the DODAG, each node identifies its parent set

comprising potential upstream neighbours for data delivery. From this parent set, a node selects its most suitable parent, referred to as the *preferred parent* [1] [39]. The selection is based on the OF, where the node typically chooses the parent with the most favourable routing characteristics. To determine this, the node calculates the rank of each neighbour in the parent set, and the one with the lowest rank is chosen as the preferred parent, forming the primary routing path. One commonly used OF is the Minimum Rank with Hysteresis Objective Function, as detailed by Gnawali *et al.* [72]. The rank of node n (rank_n) is calculated as follows:

$$\text{rank}_n = \begin{cases} \text{rank}_p + \text{cost}_{n,p} & \text{if } n \text{ is not the root} \\ \text{rank}_{\text{root}} & \text{if } n \text{ is the root} \end{cases} \tag{2.2}$$

where $\text{cost}_{n,p}$ represents the cost of using the link between node n and its neighbor p, and rank_p is the rank of node p. By default, $\text{cost}_{n,p}$ can either be ETX or hop count. Once a node selects its preferred parent, it schedules its first *negotiated cell* with that parent according to the SF, enabling data packet transmission. Notably, 6TiSCH ensures that the link layer topology aligns with the routing topology, allowing nodes to stay synchronized with their best-connected parent [42].

2.1.4 6TiSCH Link Scheduling and 6P Protocol

TSCH employs "link scheduling" for data communication within a LLN network. Link scheduling establishes and manages a schedule that dictates the communication patterns among nodes. The algorithm responsible for creating and managing these schedules is called a Scheduling Function (SF). The SF determines which links are allowed to transmit packets, the type of activity (transmission, reception, or sleep), the timing of transmissions, the specific channels to be used, and whether a cell is shared or dedicated. However, the TSCH MAC standard does not offer explicit guidelines for designing a SF. In this context, IETF WG added a new sublayer *6top layer* above the TSCH in the 6TiSCH protocol stack for schedule management. This sublayer distinguishes between cell negotiation managed by

[1] Throughout this thesis, "parent" and "preferred parent" are used interchangeably.

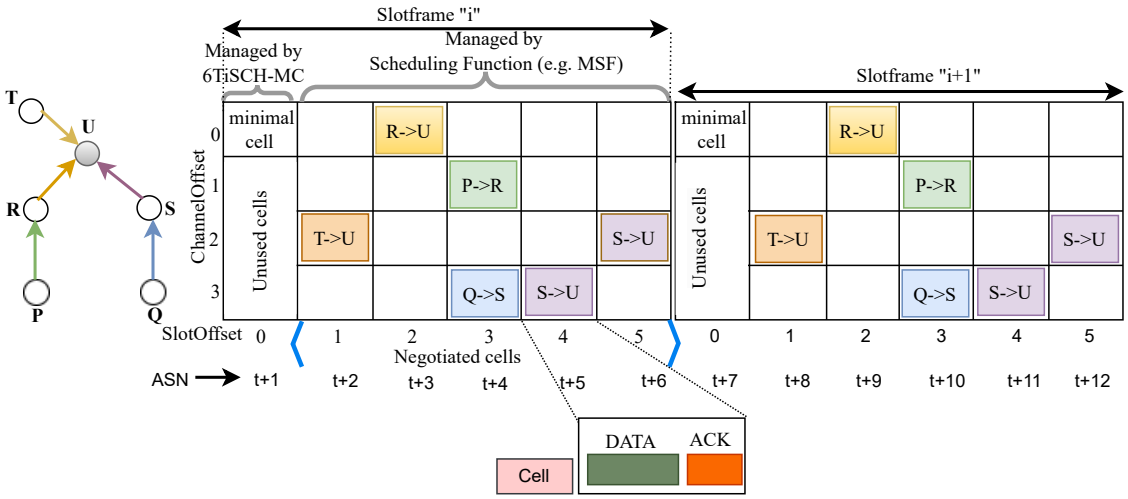


Figure 2.3: *An example of TSCH network topology and its possible link schedule considering 4 channels and a slotframe with 6 timeslots. The cell with [`SlotOffset`, `ChannelOffset`] as (0,0) is called the minimal cell*

the 6P protocol, and the SF responsible for implementing the scheduling algorithm. The 6P protocol facilitates local node-to-node cell negotiation, empowering nodes to update and manage their schedules by initiating 6P transactions based on the SF recommendations. The SF acts as the decision-making entity, determining (i) when to add, remove, or relocate cells within the schedule and adapting the schedule to match current network conditions and traffic demands, (ii) the number of cells required for communication, and (iii) which cells should be selected for packet transmission or reception. SF is responsible for allocating a unique cell to a pair of nodes for their communication, ensuring that all transmissions occur in their respective assigned cells. When a cell is being used by a pair of nodes, it is not used by any other pairs present in their communication range. As a result, TSCH enables delay-bounded data packet delivery and improves overall network throughput.

A 6TiSCH operates with three distinct types of cells: *minimal*, *autonomous*, and *negotiated*. The minimal cell is used for default resource provisioning during network bootstrap. Autonomous cells are allocated autonomously by each node without negotiation through 6P and are primarily responsible for 6P communications. Negotiated cells are dedicated to data communication and are allocated through 6P negotiations between nodes. RFC 8180 [42] outlines the scheduling guidelines for the minimal cell, while RFC 9033 [43] and

RFC 8480 [44] provide the framework to manage the scheduling of autonomous and negotiated cells, and they togetherly enable distributed scheduling in the 6TiSCH network. As shown in Figure 2.3, the scheduling of minimal cell at (`SlotOffset`, `ChannelOffset`) as (0,0) is the responsibility of 6TiSCH Minimal Configuration (6TiSCH-MC) while the scheduling of negotiated cells which starts from `SlotOffset` 1 is managed and scheduled by the SF and 6P protocol. Each parent-child pair requires a *dedicated cell* negotiated cell for their communication. The following sections provide detailed explanations of the working of 6TiSCH-MC, MSF, and the 6P protocol.

2.1.4.1 6TiSCH Minimal Configuration (6TiSCH-MC)

6TiSCH WG defines the 6TiSCH-MC standard [42] which allocates minimal bandwidth for network advertisement and bootstrapping processes, or as a fallback mechanism during network schedule failures. It defines the minimal cell which is a single static shared cell per slotframe. In Figure 2.3, the minimal cell is located at [`SlotOffset`=0 and `ChannelOffset`=0] and is used for advertising control packets such as Enhanced Beacon (EB) and DODAG Information Object (DIO), DODAG Information Solicitation (DIS), keep-alive, facilitating essential communication and coordination among network nodes during network setup and operation. As the transmission of control packets happens only in shared cells, nodes need to access the shared cell first before transmitting their control packets. Therefore, nodes perform TSCH CSMA/CA channel access mechanism to access the channel associated with the shared cell. The main difference between the traditional CSMA/CA and TSCH CSMA/CA is in setting the *backoff* timer by a node when it finds that the shared channel is busy using Clear Channel Assessment (CCA). The backoff timer is set in terms of number of shared cells instead of continuous time period in TSCH CSMA/CA.

2.1.4.2 6TiSCH Operation Sublayer (6top) Protocol (6P)

6P enables distributed scheduling in 6TiSCH networks by facilitating negotiation among neighbouring nodes. When a SF requires an update to the schedule, the 6P protocol man-

ages the transaction. However, the decision regarding when and how many cells should be added or removed is decided by the SF. A complete negotiation between two neighbouring nodes is referred to as a "6P Transaction". It begins when a node initiates a request to add/delete/relocate one or more cells with its parent and ends when the cell(s) has been added/deleted/relocated in the schedule of both nodes or if the 6P Transaction has failed. 6P negotiations can be carried out using either 2-step or 3-step transactions. A 2-step 6P transaction involves four message exchanges: (i) a 6P request message from the child node, (ii) a Link Layer ACK from the parent, (iii) a 6P response message from the parent, and (iv) a Link Layer ACK from the child.

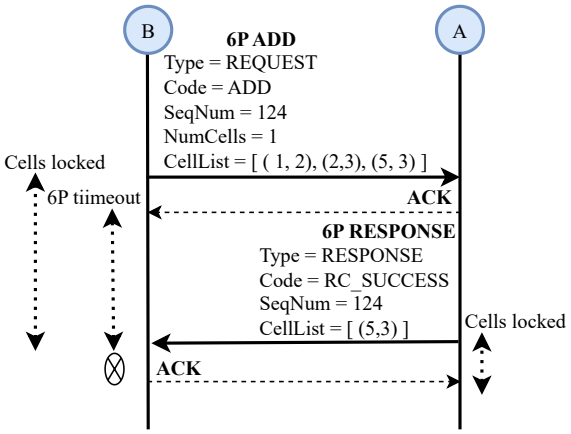


Figure 2.4: *An illustration of 6P 2-step ADD transaction. Node B requests the addition of 1 cell, providing 3 candidate cells to node A. Node A responds by selecting 1 cell from the candidate cells*

Figure 2.4 illustrates a 2-step 6P ADD transaction. Node (B) initiates a 6P ADD Request to node (A) for the addition of a cell to their schedule. Node (B) includes a list of candidate cells in the *CellList* and locks these cells, awaiting a response from Node (A). To prevent indefinite waiting, node (B) starts a 6P timer in case a response from node (A) is not received within a specified timeframe. When node (A) receives the 6P ADD Request, its SF selects a specified number of cells (*NumCells*) from the provided candidate list and responds with a 6P Response message. Node (A) locks the selected cells awaiting confirmation from node (B). Upon successful completion of the 6P ADD transaction, both nodes incorporate the selected cell into their respective schedules, thereby establishing a new communication link.

It is noteworthy that a sequence number (seq_{no}) is used during the entire 6P transaction to ensure that the Request, Response, and Confirmation belong to the same transaction.

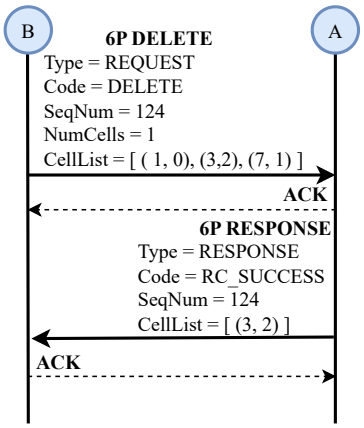


Figure 2.5: *An illustration of a cell deletion using 6P protocol, where node B requests node A to delete a cell providing the list of scheduled cells. Node A responds by randomly selecting a cell for deletion*

On the other hand, when the SF decides to delete a cell, it initiates a 6P DELETE transaction. Figure 2.5 illustrates a scenario where node (B) sends a 6P DELETE Request to node (A), requesting the deletion of a cell from their schedule. In deletion, the CODE field is set to DELETE, and *NumCells* specifies the number of cells to be deleted from the current schedule. The *CellList* contains zero or more negotiated cells which are already scheduled between nodes (B) and (A). If the *CellList* is empty, the parent node (node (A)) chooses *NumCells* cells with the sender (node (B)) and deletes them. If the *CellList* contains more number of cells than *NumCells*, then the parent node chooses exactly *NumCells* cells from the *CellList* and deletes them.

Further, the 6P protocol facilitates cell negotiation to relocate collision-prone cells. This periodic collision detection and relocation process is referred to as "6top housekeeping". Figure 2.6 illustrates a 2-step 6P relocation transaction between two nodes. Node (B) sends a 6P Relocation Request to node (A), specifying the list of cells to relocate (*RList*) along with a candidate cells list (*CCellList*) for the relocation. Upon receiving the request, node (A)'s SF first checks whether the length of *RList* is larger or equal to the number of cells to

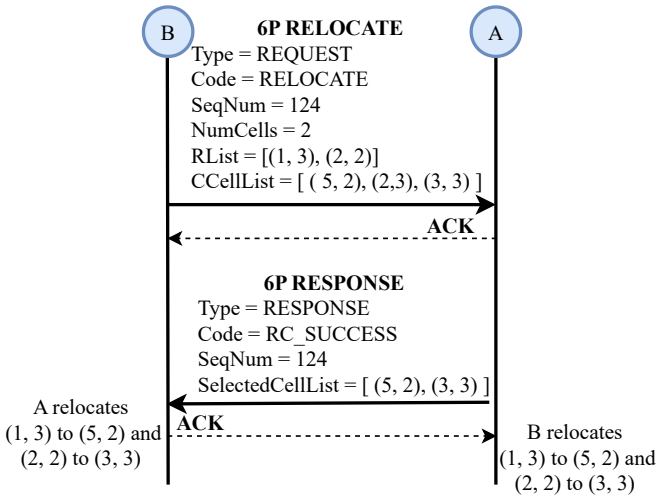


Figure 2.6: *A 6P cell relocation transaction. Node B requests node A to relocate 2 cells, providing a list of 3 candidate cells. Node A responds by selecting 2 cells from the provided candidate list*

relocate and verifies that all cells listed in the *RList* are currently scheduled with Node (B). Upon successful validation, Node (A) evaluates the *CCellList* to determine which candidate cells can be accommodated in its schedule and sends a 6P Response with the selected cells. If any of these checks fail, Node (A) replies with a 6P Response containing the error code *RC_ERR_CELLLIST*, and the relocation process is aborted. Upon receiving a successful Response message, node (B) relocates the cells from the *RList* to the newly selected cells provided in the response, following the same sequence.

2.1.4.3 Minimal Scheduling Function (MSF)

The MSF (RFC 9033) [43] defines the bootstrapping process for a new node to join the 6TiSCH network. Once a new node/pledge becomes a part of the network, it can establish and dynamically adapt its schedule based on traffic demands, routing changes, and schedule collisions. MSF allocates Autonomous and Negotiated cells. Autonomous cells are scheduled using the hash of the node's `EUI64` address as shown in Equation 2.3. There are two types of autonomous cells: *AutoTxCell* and *AutoRxCell*. For *AutoTxCell*, the destination node's `EUI64` address is used, while for *AutoRxCell*, the node's own `EUI64` address is utilized. On the other hand, negotiated cells are scheduled through 6P negotiations. Notably, 6TiSCH

supports dynamic scheduling on top of the 6TiSCH-MC with the help of SF and 6P protocol to align the schedule with the communication requirements of various applications. Thus, after joining the network, a node relies on MSF and 6P for scheduling. Next, we briefly outline the core functionalities of MSF.

$$\begin{aligned} SlotOffset &= 1 + \text{hash}(EUI64, SF_{length} - 1) \\ ChannelOffset &= \text{hash}(EUI64, N_c) \end{aligned} \tag{2.3}$$

(I) *Network Bootstrapping:* MSF defines the behaviour of a node from the moment its radio is turned on until it successfully joins the 6TiSCH network. To join the 6TiSCH network, a node must go through a series of steps before being able to transmit messages within the network. The joining process can be divided into six stages: (i) The pledge randomly selects a channel and starts listening for EBs on that frequency. (ii) Upon receiving EBs, the pledge selects a Join Proxy (JP). After receiving the first EBs, the pledge continues to listen for additional EBs to gather information on the number of neighbours and decide which neighbour to choose as the JP for the joining process (iii) After selecting a JP, the node generates a Join Request and installs an `AutoTxCell` to the JP for sending the request over this cell. The JP receives the Join Request via its `AutoRxCell` and forwards it to the Join Registrar/Coordinator (JRC). The JRC responds through multiple hops using `AutoTxCells`. The JP then sends the Join Response back to the node over an `AutoTxCell`, allowing the node to learn the keying material and configuration settings and become a "joined node". (iv) The joined node receives DIOs, computes its own Rank, and selects a preferred parent as outlined in Section 2.1.3 (v) The node sets up its first negotiated `Tx` cell with the chosen parent using a 6P ADD Request (vi) Finally, the node starts transmitting EBs and DIOs over the minimal cell, allowing new nodes to discover and join the network.

Journey of a pledge

New node/Pledge → TSCH synchronized node → TSCH joined node → One AutoRxCell → Selects preferred parent → One negotiated Tx cell → serves as router/JP for other nodes.

(II) *Addapting to Traffic Changes:* Once the pledge has successfully joined the 6TiSCH network and scheduled its first negotiated cell with its preferred parent, it can dynamically adjust the allocation of negotiated cells according to changing traffic demands. To facilitate this, MSF utilizes *NumCellsUsed* and *NumCellElapsed* counters. These counters keep track of the node's current utilization of cells toward its preferred parent. *NumCellsElapsed* increments with each elapsed cell, while *NumCellsUsed* increments only when a cell is actively used for transmitting a packet to the neighbour. Cell utilization (C_U) for a link is periodically evaluated once *NumCellsElapsed* reaches the predefined threshold *MaxNumCell*. C_U is computed as the ratio of *NumCellsUsed* to *NumCellsElapsed*. If the cell utilization exceeds the upper threshold (U_{th}), MSF triggers the 6P protocol to add a new cell to the schedule. Conversely, if utilization drops below the lower threshold (L_{th}), MSF prompts the deletion of a randomly selected cell from the schedule. After calculating the cell utilization, both counters are reset, and the process repeats. Figure 2.7 provides an illustration of how

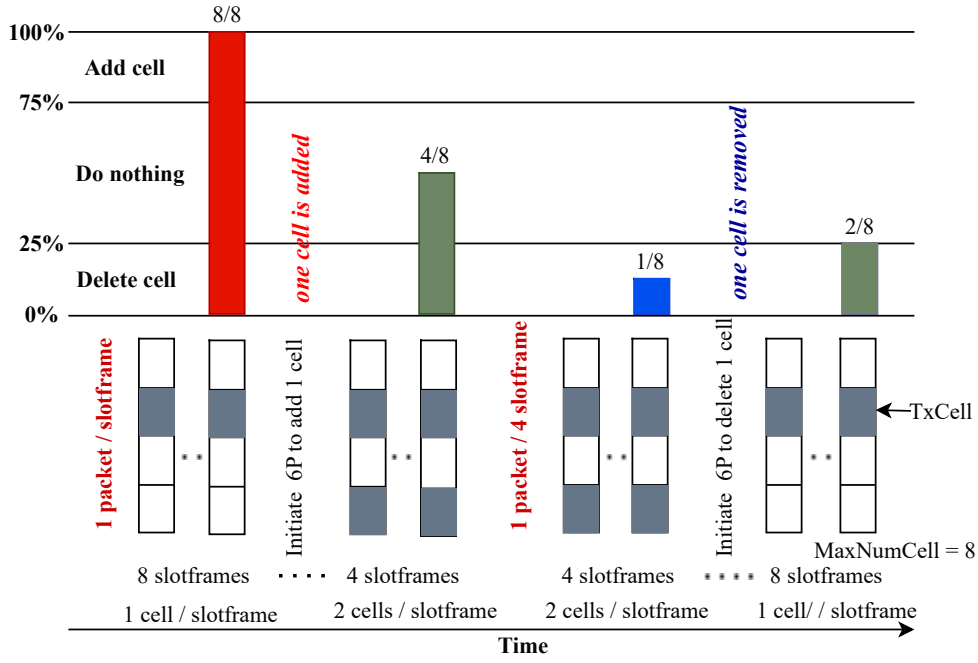


Figure 2.7: *An illustration traffic adaptation in MSF with MaxNumCell as 8. Adds a cell if C_U > 75% and deletes a cell if C_U < 25%*

schedule adaptation happens in MSF corresponding to traffic demands. For this example,

we consider $MaxNumCell$ as 8. Initially, each node has one negotiated cell scheduled with its parent. At the outset, let's say the traffic load is 1 packet/slotframe (P/SF). Thus, after 8 slotframes, the *NumCellElapsed* reaches *MaxNumCell*, and the C_U becomes 100% (8/8). Consequently, MSF decides to add one cell and triggers a 6P command to initiate the cell addition process. After this cell addition, the node operates with two negotiated cells per slotframe. Four slotframes later, when C_U is evaluated, it drops to 50% (4/8), which is in the stable zone for MSF. Thus, no action needs to be taken. However, consider a scenario in which the traffic load between the nodes drops to 1P/4SF. Hence, $NumCellElapsed$ reaches $MaxNumCell$ after 4 slotframes, and the cell usage drops to 12.5% (1/8). So, MSF issues a 6P transaction to remove one negotiated cell from its present schedule. Once this action is taken, cell utilization rises to 25%, which falls in the stable region of cell utilization.

(III) *Handling Preferred Parent Switching:* As per the RPL specifications [70], a node may switch its preferred parent when there is a variation in link quality. In such cases, the communication schedule needs to be updated accordingly. The node first employs the 6P protocol to negotiate and establish the same number of cells with the new preferred parent as were previously allocated to the old parent. Once the new cells are successfully negotiated, the node initiates a 6P CLEAR transaction with the former parent to release all previously assigned negotiated cells.

(IV) *Handling Schedule Collisions:* Due to the distributed nature of scheduling in 6TiSCH networks, it is possible for two pairs of nearby nodes to schedule a negotiated cell at the same [`SlotOffset`, `ChannelOffset`] within the slotframe. If these nodes attempt to exchange data simultaneously, it can result in a "schedule collision". MSF identifies a collision if one cell exhibits a significantly lower Packet Delivery Ratio (PDR) compared to other cells. To address this, MSF uses the 6P Relocation transaction to move the collided cell to a different position in the slotframe. Specifically, if the difference exceeds the value of $RELOCATE\_THRES$, the node relocates the cell.

2.1.5 Minimal Security Framework

MSF [43] adheres to the 6TiSCH Minimal Security specification (RFC 9031) [41]. This framework employs the Constrained Join Protocol (CoJP) protocol for secure joining of nodes into the network. The joining process involves three key entities: the pledge (the node seeking to join), the JRC, and the JP. The JP serves as an intermediary between the pledge and the JRC. CoJP assumes that both the pledge and JRC share a pre-shared key (PSK) for mutual authentication. Since CoJP messages are not protected at the link layer, the PSK functions as a Master Secret, establishing a security context for end-to-end protection of CoJP messages via the OSCORE [73] security protocol. Upon successful joining, the JRC provides the pledge with key materials to enable secure communication with the other network nodes. These key materials are shared among all network nodes and typically include two symmetric network keys: one for integrity protection of EBs and another for encrypting data messages, including 6P transactions. The joined node relies on IEEE 802.15.4 link-layer security services to protect TSCH frames.

2.2 Literature Survey

In recent years, numerous studies have explored resource allocation and link scheduling in 6TiSCH networks. This section presents an overview of the various link scheduling algorithms proposed in the literature. The aim of this review is to summarize the existing methods and identify gaps that need to be addressed. To structure the review, the literature is categorized into three parts based on the focus of each study. In the first part, we review studies that focused on the design and development of link scheduling functions tailored explicitly for 6TiSCH networks. The second part highlights research efforts that address the detection and mitigation of collisions within link schedules. Lastly, we review works that analyze the security aspects of 6TiSCH, identifying vulnerabilities and proposing mechanisms to ensure secure routing, reliable time synchronization, protection against selective jamming attacks, and the secure utilization of the SF and 6P protocol.

2.2.1 Link Schedulers in 6TiSCH Networks

Several scheduling approaches have been developed to meet specific application requirements and network conditions. This section provides a concise overview of the existing scheduling functions tailored for 6TiSCH networks.

The Traffic-Aware Scheduling Algorithm (TASA) [74] is a centralized scheme where the central coordinator applies graph theory techniques such as recursive matching and colouring to generate a conflict-free, minimal-length schedule. Similarly, Kotsiou *et al.* [46] introduced a whitelisting technique to create a collision-free, centralized schedule, and the authors in [75] proposed a low-overhead uplink scheduling method for large-scale IoT applications. However, these centralized approaches result in significant signalling overhead due to continuous information flow between nodes and the central coordinator. Additionally, such approaches are unsuitable for large, dynamic networks where frequent changes in topology and traffic patterns require constant recalculation and redistribution of the schedule. In a distributed scheduling framework, nodes autonomously compute their schedules by negotiating with neighbouring nodes based on local, partial information exchanged between them. This eliminates the need for a central coordinator and allows the network to adapt more efficiently to dynamic changes. The emphasis on topology and traffic-aware networks with minimal signalling overhead highlights the advantages of distributed scheduling. In addition, the simplicity of these methods is crucial, particularly considering the limited memory and computational capabilities of nodes within the 6TiSCH network [47], [76]. For instance, DeTAS [77] and Wave [78] are two scheduling functions explicitly designed for convergecast networks. These schemes aim to create a collision-free schedule by employing distributed information gathering with central computation. In both approaches, nodes request resources from their parent nodes, and these requests are propagated upwards through the network hierarchy until they reach the sink node. The sink then allocates cells in a top-down manner based on the aggregated resource requests from lower-level nodes. However, while information gathering in these algorithms is distributed, the actual schedule computation remains centralized. As a result, these schemes are more suitable for networks with static topologies

and stable traffic patterns, similar to other centralized scheduling approaches.

Orchestra [79] leverages RPL structures to identify neighbouring and parent nodes and autonomously allocates cells to individual nodes using a hash function based on either the node's own MAC address or the MAC address of a neighbouring node. The choice between these addresses depends on whether the cell is required for transmission (Tx) or reception (Rx). However, Orchestra fails when different nodes demand different bandwidths and lacks adaptability to traffic demand. ALICE [80] was introduced as an enhancement to Orchestra, which modifies the node-based cell allocation with a link-based approach and recomputes the schedule in each slotframe to minimize collisions caused by the hash functions. However, similar to Orchestra, ALICE lacks the ability to adapt to varying traffic demands, limiting its effectiveness in dynamic network environments. Following this, OST [81] was proposed to address the limitations of ALICE and Orchestra. OST employs a binary resource allocation tree to reduce slot collisions among directional links. However, it is important to note that OST was not developed in compliance with the 6TiSCH standards established by the 6TiSCH WG. Domingo-prieto *et al.* [82] developed a distributed scheduling policy using proportional, integral, and derivative control to minimize the number of packets in the queue. However, the 6P negotiation process in their approach does not guarantee collision-free cell allocations.

Palattela *et al.* [48] introduced an On-the-Fly Bandwidth Reservation Algorithm (OTF) that allocates cells according to the traffic to be handled. It assigns more cells if the required number of cells exceeds the number of already scheduled cells. However, the authors did not specify how traffic assessment should be performed. To improve on OTF, Righetti *et al.* [49] proposed EOTF to achieve better performance than OTF. The authors allocated 2 extra cells to avoid packet drops when the queue level exceeds a threshold due to queue overflow. However, both approaches rely on random cell allocation and deallocation. Hamza *et al.* [50] proposed the Enhanced Minimal Scheduling Function (EMSF), which estimates the traffic load of a node using a Poisson distribution model and allocates the required cells. However, similar to previous approaches, the allocation of these estimated cells is done

randomly within the slotframe, which may lead to inefficiencies such as increased collisions and suboptimal use of available resources. Stratum scheduling [83] divides the slotframe into smaller stratums with nodes selecting cells within their designated stratum. Hosni *et al.* [51] extended this by optimizing stratum sizes to reduce collisions. However, this slotframe organization reduces network capacity, and the block-based structure causes high packet drop rates, resulting in low packet delivery. Both approaches also face the singular cell update issue, similar to MSF, limiting their adaptability to varying traffic demands.

Daneels *et al.* [53] introduced the Recurrent Low-Latency Scheduling Function (ReSF) to reduce delays for recurrent traffic by reserving a low-latency path from the source to the sink, activated only when such traffic is detected. However, identifying and maintaining this low-latency path in dynamic networks is challenging due to constantly changing network conditions. Additionally, adding a new cell requires rescheduling all cells along the path to the border router, which complicates the scheduling process and reduces flexibility. The authors of LLSF [54] also aimed to minimize the end-to-end latency using a daisy-chain technique. LLSF over-provision cells compensate for poor radio links, but if link quality deteriorates, all cells along that path must be relocated. On the other hand, LDSF [52] organizes the slotframe into blocks, where each transmitter selects blocks based on its hop distance to the border router and reserves cells for retransmission opportunities. However, when retransmissions or packet drops are minimal, the reserved ghost cells remain unused, resulting in resource wastage. Moreover, LDSF is designed for very low traffic rates, limiting its applicability to networks with varying or higher traffic rates. Similarly, the authors in [84] divide the slotframe into portions, with each portion characterized by a density value. Nodes select cells from the portion with the lowest density value, i.e., the least crowded portions. However, such a cell selection method does not guarantee reduced latency.

Recently, IETF finalized MSF [43] as the standard scheduling function for 6TiSCH network, which addresses key aspects required for real-world deployments. However, MSF may not always deliver low-latency performance due to its random cell allocation and deallocation strategy. Moreover, its cell updation process leads to increased 6P overhead and higher

energy consumption. Chang et al. [85] studied the impact of the $MaxNumCell$ parameter of MSF, providing insights into its influence on network performance. Tanaka *et al.* [86] proposed a new scheduling function for 6TiSCH, called YSF, which focused on minimizing latency. However, YSF lacks traffic adaptability and allocates the same number of cells at each hop, limiting its performance in varying traffic conditions. Additionally, YSF has only been evaluated in low-traffic scenarios.

2.2.2 Collision Detection and Mitigation in Link Schedules

This section presents a brief overview of the existing collision detection and mitigation schemes in 6TiSCH scheduling.

Centralized schemes like TASA [74] can create collision-free schedules with a full view of the network. However, as previously discussed, these schemes are not well-suited for IIoT networks. Similarly, DeTAS [77] and Wave [78] can also generate collision-free schedules but at the cost of high signalling overhead and increased delays. In contrast, distributed schedulers rely on local collaboration, where each node negotiates with immediate neighbours with localized information. While this simplifies the scheduling, it introduces a trade-off between simplicity and efficiency. Localized network views increase the likelihood of collisions since nodes may not be aware of transmissions occurring in other parts of the network. Moreover, many distributed approaches utilize basic 6P cell negotiation procedures, which do not guarantee collision-free allocations. This limitation highlights the need for collision detection mechanisms to prevent performance degradation.

The standard MSF [43] provides guidelines for handling collisions in link schedules. MSF employs a PDR-based collision detection mechanism followed by the random relocation of the collided cell. However, this approach is inefficient, often resulting in repeated relocation requests and increased signalling overhead. Muraoka *et al.* [55] and Chang *et al.* [56] adopted a similar approach to MSF by detecting underperforming cells based on PDR and subsequently relocating those cells to random positions within the slotframe. While Chang *et al.* introduced a cost-aware relocation strategy to mitigate collisions, the decision de-

pends on the overall cost to the network if relocation were to happen or not. However, their approach may not significantly improve performance, as it may potentially lead to repetitive collisions if relocation is not executed. This can degrade network reliability and energy efficiency. Furthermore, since their evaluation was limited to a 5-node network, the scalability and effectiveness of this cost-aware strategy in larger and dynamic networks remain uncertain. The authors in [87] proposed a parent-initiated 6P transaction approach to mitigate 6P transaction failures due to collisions. Rather than having the child node directly initiate 6P requests to the parent, the child node instead awaits a cell offer message from the parent. In this approach, the parent node identifies the child node with the highest cell demand. Only the selected child sends the 6P Request to the parent This coordinated approach reduces the number of collisions during 6P transactions.

2.2.3 Securing Link Scheduling and the Associated 6P Protocol

This section reviews existing research on the security of link scheduling and 6P protocol, highlighting vulnerabilities and the strategies proposed to address these threats. Numerous studies have explored different facets of the 6TiSCH architecture, such as the design and management of link scheduling ([43], [86], [74]), network formation dynamics ([42], [88], [89]), and routing protocol performance ([90], [39]). However, only a limited number of studies have focused on security aspects of the 6TiSCH protocol stack or identified its potential vulnerabilities.

In a 6TiSCH network, the 6P protocol and SF are crucial components for network communication. Any attacks against these protocols can cause communication disruptions and severely impact overall network performance. In a recent study, Righetti *et al.* [57] identified vulnerabilities in the 6P protocol and demonstrated how an attacker manipulates victim node pairs into executing inconsistent 6P transactions. As a result, each node in the pair maintains different schedules without being aware of the discrepancy. The study demonstrated the feasibility of two distinct attacks on the 6P protocol, namely the *Traffic Dispersion Attack* and the *Overloading Attack*, both carried out by compromised nodes

under adversarial control. In the traffic dispersion attack, the attacker creates a schedule mismatch between two negotiating nodes Ⓐ and Ⓑ, where Ⓐ is the child node and Ⓑ is its preferred parent. By impersonating node Ⓐ and sending false 6P DELETE or RELOCATE requests to its preferred parent Ⓑ, the attacker convinces node B to adopt a new schedule S'. As a result, nodes Ⓐ and Ⓑ unknowingly end up with different schedules S and S', where S is the actual schedule for communication between nodes Ⓐ and Ⓑ. On the other hand, the overloading attack creates a mismatch between transmission (Tx) and reception (Rx) cells, forcing node Ⓑ to add Rx cells without corresponding Tx cells with node Ⓐ. The authors proposed a mitigation strategy to counteract the overloading attack.

Some studies in the literature have focused on routing and time synchronization attacks. However, the mitigation schemes proposed in those studies can not prevent attacks targeting cell scheduling functions and the 6P protocol. For example, DIS attacks have been extensively explored in the literature ([59], [60], [91]). The authors of [59] presented the spam DIS attack, where a malicious node floods the network with DIS messages using fabricated identities and forces legitimate nodes to restart the Trickle algorithm repeatedly and broadcast excessive DIO messages, leading to energy depletion and a denial of service. The impact of DIS flooding on network performance, especially the increase in control packet overhead, is emphasized in [91]. To address these threats, lightweight mitigation schemes were developed in [91, 92]. In [60], an anomaly-based lightweight Intrusion Detection System (IDS) was proposed, leveraging threshold values to detect and mitigate RPL attacks. Similarly, Althubaity *et al.* [93] explored how malicious nodes manipulate RPL rank values to disrupt the routing process and destabilize the network. To combat this, they proposed the Forged Rank and Routing Metric Detector (FORCE) scheme. In [94], the effects of DIS attacks on 6TiSCH network formation were examined. Extending their work in [95], the authors proposed a non-cooperative gaming model to determine response probability to DIS packets. They also introduced a trust-based mechanism to detect and mitigate malicious DIS transmissions.

Yang *et al.* [58] highlighted vulnerabilities in time synchronization within TSCH networks

and identified two specific attacks: the Absolute Slot Number (ASN) and time synchronization tree attack. In the ASN attack, adversaries manipulate control messages to provide incorrect ASN values to legitimate nodes, preventing them from synchronizing with the network. In the time synchronization tree attack, falsified DIO packets are used to damage the structure of the synchronization tree, thereby disrupting the network's topology and synchronization process. The authors proposed countermeasures such as intrusion detection algorithms, encryption, and authentication to defend against these attacks. Similarly, Tiloca *et al.* [61] explored how attackers can exploit the predictable and periodic nature of channel hopping sequences in TSCH to uncover the victim's communication schedule. To counter this vulnerability, they introduced the Dynamic and Irregular Slot Hopping (DISH) solution, which allows nodes to change their communication patterns in a pseudo-random and unpredictable manner during each slotframe. This approach makes it more challenging for adversaries to predict and disrupt node transmissions.

Table 2.1: *Existing works on link scheduling in 6TiSCH*

Schemes	Year	TA	RCCC	CRS	CDAM	CDS	AV		CAS		Performance Under	
							6P	SF	SBR	RB	LTR	HTR
Stratum [51]	2017	✗	✗	random	✗	✗	✗	✗	✓		poor	poor
Orchestra [79]	2015	✗	✗	✗	✗	✗	✗	✗	✓		average	poor
ALICE [80]	2019	✗	✗	✗	✗	✗	✗	✗	✓		average	poor
OST [81]	2020	✓	✗	✗	✗	✗	✗	✗		✓	average	average
LDSF [52]	2020	✗	✗	random	✗	✗	✗	✗	✓		good	poor
LLSF [54]	2016	✗	✗	random	✗	✗	✗	✗	✓		poor	poor
Pid-based [82]	2018	✓	✓	random	✗	random	✗	✗	✓		average	poor
EOTF [49]	2018	✓	✓	random	✗	random	✗	✗	✓		poor	average
MSF [43]	2021	✓	✗	random	✗	random	✗	✗	✓		poor	poor
EMSF [50]	2019	✓	✓	random	✗	random	✗	✗	✓		poor	average
ReSF [53]	2018	✗	✗	✗	✗	✗	✗	✗	✓		poor	good
Muraoka [55]	2016	✗	✗	random	✗	✗	✗	✗	✗	✗	poor	poor
CCR [56]	2018	✗	✗	cost-aware	✗	✗	✗	✗	✗	✗	poor	poor
YSF [86]	2022	✗	✗	random	✗	✗	✗	✗	✓		good	poor
V6P [57]	2020	✗	✗	✗	✗	✗	✓	✗	✗	✗	poor	poor
V6PE [62]	2022	✗	✗	✗	✗	✗	✓	✗	✗	✗	poor	poor
This Thesis		✓	✓	latency-aware	✓	✓	✓	✓	latency-aware	✓	good	good

TA - Traffic Adaptive. RCCC - Required Cell Count Computation. CRS - Cell Relocation Strategy. CDAM - Cell Depletion Attack and Mitigation. CDS - Cell Deletion Strategy. AV - Addressed Vulnerability. CAS - Cell Allocation Strategy. SBR - Sender Based and Random. RB - Receiver Based. LTR - Low Traffic Rate. HTR - High Traffic Rate.

2.2.4 Limitations in the Existing Works

The scheduling functions discussed above aim to improve different Quality of Service (QoS) requirements for 6TiSCH networks. However, a common limitation is their lack of adapt-

ability to network and traffic dynamics, making them unsuitable for varying traffic rates. While MSF standard allows for dynamic traffic management, it performs negotiated cell addition and deletion in a linear manner irrespective of actual traffic demand, which results in significant 6P signalling overhead. Additionally, none of the existing approaches adequately consider pending packets or link quality when updating cells for the next slotframe cycle. Furthermore, these methods rely on random cell selection for both allocation and deallocation, which makes them unsuitable for delay-sensitive networks. Therefore, there is a clear need for a dynamic scheduling function capable of adapting itself to the changes in the network and traffic dynamics with an emphasis on cell count computation and cell selection strategies to enhance the overall performance of the 6TiSCH networks. Moreover, only a few studies have addressed schedule collision detection and mitigation in link scheduling. These studies have primarily relied on PDR for detecting cell collisions. However, none of the schemes effectively detects and handles all collision scenarios and uses an inefficient sender-based random relocation strategy, resulting in excessive relocation requests and increased network overhead. Furthermore, the literature highlights a gap in assessing the feasibility and impact of attacks on cell scheduling and the 6P protocol.

Finally, Table 2.1 summarizes the key features of existing works on link scheduling functions in 6TiSCH networks and compares them with the contributions made in this dissertation. Specifically, Objective 1 addresses the aspects highlighted in columns 3, 4, 7, 9, and 10 of Table 2.1, Objective 2 corresponds to column 5, and Objective 3 focuses on columns 6 and 8 of the table.

2.3 Experimental Environment

Numerous open-source network simulators, such as NS-3[1], OMNet++[2], and the TinyOS[3] simulator TOSSIM [96] are available for simulating low-power wireless networks. However, these simulators do not fully adhere to the 6TiSCH standard and lack the required imple-

[1]https://www.nsnam.org/
[2]https://omnetpp.org/
[3]http://www.tinyos.net/

mentations, scalability, and simplicity required for 6TiSCH networks. Another open-source implementation, Contiki[1], is specifically designed for resource-constrained IoT devices with limited memory and processing power. While TSCH was introduced in Contiki in 2015, it initially lacked support for designing scheduling functions using 6P signalling, and the Cooja mote was unable to operate TSCH due to the absence of the 6P protocol of the 6TiSCH stack in Cooja. Recently, the 6P protocol has been integrated into Contiki, but the standardized MSF (RFC 9033) is still unavailable.

The 6TiSCH WG, responsible for defining the complete protocol stack for 6TiSCH network, developed the 6TiSCH simulator. This simulator implements the 6TiSCH protocol stack accurately as standardized, allowing the evaluation of 6TiSCH network performance under diverse conditions. It serves as a comprehensive tool for testing and evaluating network behaviour. In this dissertation, we use the latest version (Version 1.2.0) of the 6TiSCH Simulator[2] [63] to implement and evaluate our proposed contributions. Table 2.2 provides a comparative analysis of the available network simulators. The following subsection provides an overview of the simulator and its key capabilities.

Table 2.2: *Comparison of the 6TiSCH simulator and the other network simulation tools*

Simulator	Learning Curve	Scalability	6TiSCH Implementation	Standard-Compliant
NS-3	High	Medium	None	NA
OMNet++	High	Medium	None	NA
TOSSIM	Medium	High	None	NA
Cooja (emulator)	High	Low	Partial	Partially
OpenSim (emulator)	High	Low	Yes	Yes (byte-accurate)
6TiSCH simulator	Low	High	Yes	Yes (behavioral)

2.3.1 6TiSCH Simulator

The 6TiSCH simulator is an open-source, Python-based tool that supports all standardized protocols for the 6TiSCH network. Its source code[3] is available on GitHub with a large community of contributors and forks. The core developers include Yasuyuki Tanaka, Keoma Brun-Laguna, Mališa Vučinić, and Thomas Watteyne, along with many contributors from

[1]https://www.contiki-os.org/
[2]https://bitbucket.org/6tisch/simulator/
[3]https://bitbucket.org/6tisch/simulator/src/master/

Figure 2.8: *Screenshot of the 6TiSCH simulator graphical user interface*

research institutions and industry. This discrete-event simulator accurately models the behaviour of a complete 6TiSCH stack, including network formation, routing, and scheduling. It supports protocols standardized by the IETF that ensure low-power IPv6 connectivity for IoT environments. The simulator offers a comprehensive platform for accurately simulating various aspects of network behaviour, such as interference effects, scheduling function performance, 6LoWPAN header compression, RPL routing, IEEE 802.15.4 TSCH MAC layer frame drops, 6P transactions, and application responses. It allows detailed analysis of different 6TiSCH sublayers under a variety of conditions while ensuring accuracy in standard-compliant hardware implementations. Researchers can easily evaluate custom enhancements within the context of a fully standards-compliant network stack. The simulator also supports automated execution on multiple cores, making it ideal for extensive performance testing and evaluation. The simulator includes a graphical user interface (GUI), as shown in Figure 2.8, allowing users to control the simulation and view real-time statistics for various cells, links, and nodes.

All our contributions are thoroughly evaluated using various network topologies such as linear, tree, fully meshed and random, along with different configurable network parameters.

"The secret to getting ahead is getting started ."

~Mark Twain (1835-1910)

3

Adaptive Low-latency Distributed Link Scheduling Function

In this chapter, we discuss the solution corresponding to the first objective of the thesis. We first highlight few limitations of the existing scheduling functions for 6TiSCH networks, and then propose few countermeasures to improve the performance of 6TiSCH networks by designing an adaptive, low-latency distributed link scheduling function. In this solution, we propose three schemes: *Improved Minimal Scheduling Function* (IMSF), *Receiver-based Traffic rate agnostic Distributed link scheduling function* (RTDS), and *Latency-aware Cell Deletion* (LCD). All the proposed schemes are mainly tailored for data-gathering applications. IMSF focuses on making scheduling responsive to traffic and network conditions i.e.

making it adaptive, RTDS addresses the limitations of random cell allocation, and LCD optimizes the cell deletion or deallocation strategy. The proposed schemes are evaluated through theoretical analysis and extensive simulation experiments.

3.1 Introduction

As discussed in Chapter 2, the IEEE 802.15.4 standard does not explicitly specify methodologies for constructing and implementing link scheduling. To address this, 6TiSCH Working Group (WG) introduced the Minimal Scheduling Function (MSF) [43] as a reference scheduling function for 6TiSCH network and an intermediary 6top sub-layer, as documented in [44], has been introduced above the TSCH layer to enable dynamic scheduling. Many other link scheduling schemes have been proposed in the literature, each aimed at accommodating distinct application-specific needs. Centralized schedulers (e.g. [45], [46]), in which a central node is responsible for computing the schedule for all nodes, are not suitable for IIoT applications as they require huge signalling overhead to quickly adapt with varying traffic and network conditions. So, distributed scheduling is more suitable in IIoT applications.

Designing an efficient link schedule (an example of a link schedule is illustrated in Figure 2.3, Chapter 2) is crucial for achieving optimal performance of a 6TiSCH network in terms of latency, energy consumption, and reliability during data transmissions. Many IIoT applications experience varying traffic rates, from low to high. For instance, in smart energy management, traffic may be minimal during off-peak hours but can surge during peak times when energy demand is high. Most of the existing scheduling schemes are designed for a particular traffic rate but not for both high and low rates. Therefore, there is a pressing need for a scheduling mechanism that adapts dynamically to current network conditions and traffic rates to maintain high transmission reliability, and low end-to-end latency while consuming less energy under any traffic rate condition.

3.1.1 Motivation

Even though few distributed scheduling functions have been defined to improve different QoS requirements, most of the approaches are not flexible in terms of topology changes and traffic adaptation. Although the MSF scheme can dynamically manage traffic changes, it updates the number of scheduled cells in unity even when there are substantial changes in traffic requirements and, thus, incurs high signalling overhead on the 6P protocol. One of the two crucial components of a link scheduling design policy is cell selection and allocation strategy, which determines the position (i.e. `SlotOffset` and `ChannelOffset`) of a cell going to be scheduled in a slotframe for a link to achieve the desired performance. MSF does not specify any suitable cell selection method and, thus, allocates cells randomly for a link schedule. Because of this, MSF cannot always achieve low end-to-end latency. The approaches in [51], [52] designed their strategies to minimize the end-to-end latency by partitioning the slotframe into blocks or portions (called stratums). Cells are then selected randomly within the designated stratums or in the worst case, across the entire slotframe. However, such division of a slotframe into stratums reduces the network capacity and impacts reliability badly due to huge packet losses. Also, these strategies were designed to minimize the end-to-end latency for very low-traffic rate scenarios, and they perform poorly under high-traffic rate scenarios. The authors in ReSF [53] and LLSF [54] chained cells along the path from source to destination to reduce the end-to-end delay. However, these approaches are not flexible to changes in traffic rate. As per our knowledge, almost all the existing distributed scheduling functions consider only one of the components to design the scheduling function. It is worth mentioning that the cell deletion procedure employed by existing link scheduling schemes, including the standard MSF, lacks a strategic approach in selecting which cell to delete and thus relies on random deletion of cells. This degrades network performance, particularly in terms of end-to-end delay and energy consumption. Thus, to improve the performance of 6TiSCH link scheduling, it is essential to address schedule adaptation based on network dynamics, as well as optimize the randomized cell allocation and deallocation processes.

3.1.2 Contributions

To overcome the discussed limitations, we first propose the *Improved Minimal Scheduling Function (IMSF)*. The proposed IMSF can dynamically adapt the link schedule based on present network conditions and traffic demand. It decides how many cells need to be added/deleted to/from the schedule to make it adaptive to current network conditions. The main goal is to reduce 6P control overhead while improving end-to-end latency and transmission reliability. Further, to address the limitations of random cell selection and allocation, we propose the *Receiver-based Traffic rate-agnostic Distributed link Scheduling function (RTDS)* for 6TiSCH networks. RTDS builds upon IMSF principles to determine the number of cells that need to be added or removed, adapting to current network conditions and traffic demands. Further, RTDS replaces the random cell selection method with a receiver-based, delay-aware cell selection strategy. In this approach, the receiver collaborates with the sender to select available free slots for cell allocation, aiming to minimize end-to-end latency by allocating transmission cells as close as possible for successive nodes along the path from source to sink. By using receiver-based cell selection instead of sender-based approach, RTDS reduces packet waiting time in the receiving node's buffer before forwarding to the final destination. Further, both the IMSF and RTDS did not address how to decide which cells should be deleted in case of underutilization of cells, and thus they follow random cell deletion approach. It is observed that a random cell deletion can elevate end-to-end latency and energy consumption. To address this, we propose a *Latency-aware Cell Deletion (LCD)* mechanism for distributed link scheduling for 6TiSCH networks. It is important to note that before deleting cells from the TSCH schedule, the transmitting node must determine how many cells should be removed to reach the stable region of cell utilization (C_U) as defined in MSF. LCD provides a delay-aware cell selection approach to delete cells from the present schedule, further improving the efficiency of distributed link scheduling. Additionally, this chapter provides a theoretical estimate of the end-to-end delay under both the random cell selection strategy and the proposed receiver-based delay-aware strategy. Finally, the proposed schemes are validated by simulation on the 6TiSCH simulator

and prove effective in improving the overall network performance at all traffic rates.

We summarize the major contributions of this chapter as follows,

- An Improved Minimal Scheduling Function (IMSF) is proposed to adapt the TSCH schedule according to traffic and network conditions dynamically.
- Building upon IMSF, a Receiver-based Traffic rate-agnostic Distributed link Scheduling function (RTDS) is designed to optimize end-to-end latency, resource allocation, and signalling overhead.
- An enhanced cell deletion strategy, Latency-aware Cell Deletion (LCD), is proposed to further minimize the end-to-end latency in IIoT applications. LCD builds upon the cell deletion framework of IMSF.
- The default 6P ADD and DELETE cell negotiation procedures are modified to directly incorporate the proposed RTDS and LCD schemes.
- Provided an estimation of the end-to-end delay under the random cell selection strategy and in the proposed receiver-based delay-aware strategy.
- Finally, simulations are conducted on the 6TiSCH simulator and the performance comparisons against existing benchmark schemes are performed to validate the efficacy of our proposed solutions.

The rest of the chapter is organized as follows. Section 3.2 present our proposed IMSF scheme. Subsequently, Sections 3.3 and 3.5 present RTDS and LCD schemes, respectively. Analysis of Delay and 6P overhead is discussed in Section 3.4. We provide performance evaluation of the proposed schemes using simulation experiments in Section 3.7. Finally, we conclude this chapter in Section 3.8. At the outset, the frequently used symbols and their corresponding meanings are presented in Table 3.1.

Table 3.1: *Important notations*

Symbol	Meaning
$slot_{length}$	Duration of the timeslot
SF_{length}	Length of the slotframe. In this work, it is 101
Ts_{of}	Position of SlotOffset in slotframe
Ch_{of}	Logical ChannelOffset
N_c	Total number of physical channels used
NumCellsElapsed	Number of scheduled negotiated cells that have elapsed in the present evaluation
NumCellsUsed	Number of scheduled negotiated cells that have been used for transmission in the cycle
C_U	Cell utilization
MaxNumCell	Maximum cells a node can elapse before calculating cell utilization. In this work, it is 16
NCS	Number of negotiated cells per slotframe scheduled for a link between the nodes
n_i	Negotiating node n_i
n_j	Preferred parent of negotiating node n_i
ETX(n_i, n_j)	Average transmissions required between node n_i and node n_j to deliver a packet successfully
x	Number of negotiated cells that need to be added to the schedule to reach the stable zone
y	Number of negotiated cells needs to be deleted from the current schedule to reach stable zone
U_{th}	Upper threshold of the stable zone of cell utilization. It is 75%
L_{th}	Lower threshold of the stable zone of cell utilization. It is 25%
a_{slots}	A set of all slots in the negotiated slotframe
BS_i	Contains already allocated slots (busy slots) of n_i
BS_j	Contains already allocated slots of n_j
selected_slots	Selected slots to be scheduled between the negotiating nodes
Req_delay	6P ADD request delay
Res_delay	6P ADD response delay
ohd	One hop buffering delay between the nodes

3.2 IMSF

The two crucial components of a link scheduling design policy are (*i*) *Required cell count*: which indicates the number of cells to be scheduled between two nodes, i.e. for a link, and (*ii*) *Cell selection and allocation strategy*: this determines the position (i.e. *SlotOffset* and *ChannelOffset*) of a cell to be scheduled in a slotframe for a link in order to achieve the desired performance. IMSF is designed to address the first part of this objective. It is introduced to dynamically meet traffic requirements while minimizing 6P signalling overhead. Specifically, IMSF estimates how many cells need to be added/deleted to/from the schedule to adapt to present network conditions and traffic demand. In 6TiSCH, negotiated cells are used for data communication, and the number of these cells varies dynamically to accommodate changing traffic demands. We observed that in MSF (RFC 9033), the negotiated cell addition and deletion are performed linearly, which results in high 6P signalling overhead. IMSF improves upon MSF by first determining the number of cells required to support the communication in the next cycle. Accordingly, the 6P negotiation is carried out so the node can judiciously update its resources to meet the expected traffic demand,

thereby improving end-to-end latency and reliability. In the next two Subsections, we detail how IMSF estimates the number of cells to adapt the schedule according to varying traffic demands and manages the dynamic addition and deletion of cells.

3.2.1 Estimation of Required Cell Count to Add/Delete

IMSF determines the appropriate number of cells to add or remove based on the current cell utilization (C_U) status and network conditions. It computes the number of cells required for communication between two nodes, considering the pending packets in the sender's buffer, link quality between them, traffic conditions and cell usage of the already scheduled cells. Each node calculates the number of requisite cells to be added (denoted as x) or deleted (denoted as y) to or from the existing schedule to make it traffic adaptive dynamically using local information only. IMSF considers two key factors to estimate x for the next cycle. First, the Expected Transmission Count (ETX) for a link, which represents the average number of transmissions required for successful packet delivery. So, ETX captures the present quality of the link for carrying data traffic. A link with poorer quality requires more number of retransmissions, and thus, more cells need to be scheduled to achieve the desired performance. Second, IMSF considers the number of pending packets in each node's buffer, along with newly generated and incoming packets, to ensure that queued transmissions are scheduled as soon as possible.

$$L_{th} < \frac{T(B,A) \times ETX}{MaxNumCell \times (len(NCS) + x)} <= U_{th} \tag{3.1}$$

For a link Ⓑ → Ⓐ, node Ⓑ measures the average traffic load to node Ⓐ for each slotframe cycle which is denoted by *T(B, A)*. *NCS* is the number of negotiated transmission cells currently allocated for the link Ⓑ → Ⓐ and *MaxNumCell* is the measure of cell utilization calculation cycle. It is set to 16 for our consideration. Note that L_{th} and U_{th} denote the lower and upper thresholds of the stable zone, respectively, as mentioned in Section 2.7 in Chapter 2. Assuming that the node Ⓑ knows its incoming traffic rate, it increments $T(B, A)$ by the traffic rate destined for node Ⓐ. Finally, the node Ⓑ adds the value of $T(B, A)$ with

the pending packets waiting in the queue to get the final traffic load for the link Ⓑ → Ⓐ. Please note that x is an integer, and our objective is to determine the maximum value of x such that, after adding x cells to the currently scheduled NCS cells, the resulting cell utilization must fall within L_{th} and U_{th}. On average, to be at 50% cell utilization with the value of $MaxNumCell$ as 16, x can be calculated as $x = \frac{(T(B,A) \times ETX) - NCS}{6}$.

Furthermore, instead of adopting the one-by-one cell deletion approach, IMSF proposes an efficient estimation of the number of cells required to be removed from a node's existing schedule to release the unwanted cells through a single 6P transaction. The estimation of the number of cells to delete (y) between nodes Ⓑ and Ⓐ is presented in Equation 3.3. Where U_{th} is the value of the upper threshold of cell utilization (C_U) of the link. C_U monitors the current usage of the negotiated cells towards the parent node, and is calculated as the ratio of *NumCellsUsed* and *NumCellsElapsed* as shown in Equation 3.2. The *NumCellsElapsed* counter tracks the number of negotiated transmission cells that have elapsed since its initialization, regardless of whether the cell is used for transmission. On the other hand, *NumCellsUsed* counts only those cells which are used for transmission. Additionally, to handle changes in the RPL preferred parent, IMSF schedules the same number of cells to the new parent as it had scheduled with the previous parent, and releases the cells previously allocated to the old parent.

$$C_U = \frac{NumCellsUsed}{NumCellsElapsed} \tag{3.2}$$

$$y = \left\lfloor len(NCS)\left(\frac{U_{th} - C_U}{U_{th}}\right) \right\rfloor \tag{3.3}$$

3.2.2 Decision on Add/Delete Cell

Based on the estimated value x or y, IMSF decides how many cells to add or delete to bring the cell utilization into the stable zone, i.e., within the range of L_{th} to U_{th} while minimizing over-provisioning of resources. In brief, IMSF works as follows as per the Algorithm 1:

- If the estimated C_U lies between L_{th} and U_{th}, IMSF takes no action and continues

Algorithm 1: Improved MSF (IMSF)

Input: C_U, U_{th}, L_{th}, a_{slots}
Output: Add or Delete or No Action

```
if U_th < C_U then
    compute x from Equation 3.1
    if x > length(a_slots) then
        x ← a_slots
    initiate 6P ADD REQUEST to add x cells
else if C_U < L_th then
    compute y from Equation 3.3
    if (len(NCS) − y) < 1 then
        y ← (len(NCS) − 1)
    initiate 6P DELETE REQUEST to delete y cells
else
    No Action
```

with the present schedule.

- If the estimated C_U is more than U_{th}, IMSF decides to add x more cells. Before that, it checks if x is greater than the number of available slots in the slotframe. If so, IMSF resets x by a_{slots}. Finally, it initiates a 6P transaction to add those x many cells.

- If the estimated C_U is below L_{th}, then IMSF decides to delete cells from the current schedule. It first checks if $(len(NCS)-y) < 1$); if so, it decides to delete $(len(NCS)-1)$ cells from its schedule to ensure at least one cell remains scheduled between the node and its parent. Otherwise, it deletes the y number of cells. Finally, it initiates a 6P delete transaction to remove those cells from the schedule.

IMSF addresses the adaptive scheduling aspect, however, it does not specify the positions where cells should be added or deleted. Thus, a cell selection method is required to determine which specific cell positions within the slotframe should be allocated or deallocated for a link to minimize end-to-end latency. To address this, we propose RTDS.

3.3 RTDS

Ensuring low end-to-end delay and reduced energy consumption while maintaining a high Packet Delivery Ratio (PDR) under any traffic rate is crucial for many industrial applications. To address these needs, this chapter introduces a receiver-based mechanism for cell selection and allocation that aims to minimize end-to-end latency. It is worth noting that existing link scheduling approaches, including the standard MSF and our previously proposed IMSF, rely on a sender-based random cell selection strategy for link scheduling. RTDS builds upon IMSF and incorporates a cell selection and allocation mechanism to determine the most suitable cell positions for minimizing end-to-end latency.

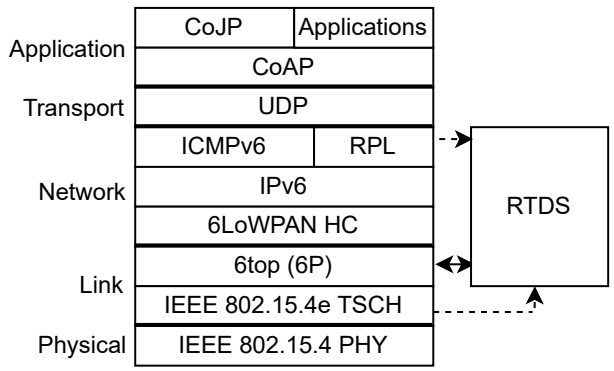


Figure 3.1: *The 6TiSCH protocol stack with RTDS.*

As illustrated in Figure 3.1, the proposed RTDS is integrated into the 6TiSCH protocol stack. It operates above the TSCH layer along with the 6top layer, interacting with 6top [44], TSCH MAC layer, IPv6 and RPL [39]. This module operates in conjunction with the default 6TiSCH-MC [42], and it makes use of the minimal cell for broadcasting EBs and DIOs. On the other hand, the cells scheduled by RTDS are exclusively designated for transmitting data packets. RTDS closely interacts with the routing layer running RPL, an IPv6 standard routing protocol for LLNs. RTDS ensures that when a node changes its routing parent, it can swiftly allocate or deallocate timeslots for communication with the new or old parent. RTDS utilizes the 6P protocol for the allocation, deallocation, and relocation of cells. Additionally, RTDS interacts with TSCH to receive information such

as Absolute Slot Number (ASN), slotframe information, ETX etc. RTDS considers both the crucial components of adaptive cell scheduling - *required cell count computation* and *cell selection and allocation* mechanisms. The first component is covered in the previous Section 3.2. In the following subsection, we present the proposed cell selection strategy. It is important to note that the proposed cell selection approach operates independently of the required cell count computation policy, making it compatible with any strategy for determining x and y

3.3.1 Receiver-based Cell Selection Strategy

This section proposes a receiver-based cell selection strategy aimed at minimizing end-to-end delay. In the sender-based strategy, the sender provides a list of candidate cells to the receiver and thus restricts the receiver to select and allocate cells from that communicated candidate cell list only. The sender's candidate cell selection process does not consider the receiver's information and thus selects the candidate cells from its own available slots only, and the selection is done in a random fashion. So, the sender limits the number of available slots for the receiver to choose for allocation. The receiver also selects cells from the candidate list randomly. On the other hand, the proposed receiver-based cell selection strategy in RTDS allows the receiver to utilize all its free slots while collaborating with the sender and can judiciously select cells for allocation from that free list. Moreover, instead of randomly selecting cells for allocation, the receiver employs a delay-aware selection approach that minimizes end-to-end latency. Basically, it aims to allocate transmission (`Tx`) cells as close as possible for any two successive nodes along a path from source to sink so that the buffering delay of a packet in intermediate nodes becomes minimal. How this is achieved is explained in the *SlotOffset* selection sub-section. A node initiates the selection and allocation process by sending a cell allocation request to its parent. Note that it is required to do the cell selection before performing the cell allocation in the schedule. The proposed cell selection strategy is discussed in two parts —*SlotOffset* selection and *ChannelOffset* selection, as each cell is represented by a tuple of *SlotOffset* and *ChannelOffset*.

Algorithm 2: Negotiated Cell Slot Selection

Input: x : required cells, n_i : negotiating node, n_j : preferred parent of n_i, BS_i : busy slots of n_i
Output: *selected_slots*

$a_{slots} \leftarrow$ all slots of a slotframe // (Initialize all slots as available slots)
a_{slots}.**remove** (*slotlOffset* 0)
$BS_j \leftarrow$ **find** already allocated slots of node n_j
$a_{slots} \leftarrow a_{slots}$.**remove**$\{BS_i, BS_j\}$ // (Updates a_{slots} list)
if a_{slots} is **null then**
    return null
if n_j is *DODAGRoot* **then**
    $selected\_slots \leftarrow max\_slots(a_{slots}, x)$ // (Finds the greatest x slots from a_{slots})
else
    // (n_j is not a root node)
    $Tx\_slots\_n_j \leftarrow$ **find** Negotiated Tx slots of n_j
    $min\_slot\_parent \leftarrow min\_slots(Tx\_slots\_n_j, 1)$
        // (Finds minimum slot number in $Tx\_slots\_n_j$)
    if $min(a_{slots}) < min\_slot\_parent$ **then**
        $a_{slots} \leftarrow \{s \in a_{slots} \mid s < min\_slot\_parent\}$
        if $len(a_{slots}) > x$ **then**
            $selected\_slots \leftarrow max\_slots(a_{slots}, x)$ // (Selects greatest x slots from a_{slots})
        else
            $selected\_slots \leftarrow a_{slots}$
    else
        if $len(a_{slots}) > x$ **then**
            $selected\_slots \leftarrow min\_slots(a_{slots}, x)$ // (Selects the smallest x slots from a_{slots})
        else
            $selected\_slots \leftarrow a_{slots}$
return *selected_slots*
    // (Selected slots for allocation)

3.3.1.1 *SlotOffset* (Ts_{of}) Selection

The proposed *SlotOffset* selection algorithm (Algorithm 2) works as follows. The algorithm takes four inputs x, n_i, n_j and BS_i. Here, x is the required number of cells to be scheduled for a link (x is calculated as discussed in Section 3.2.1), and n_i and n_j correspond to the negotiating node and its preferred parent, respectively. BS_i contains the set of occupied slots i.e. already scheduled slots of node n_i. The function $max\_slots(a_{slots}, x)$ selects the largest x slots from the a_{slots} while $min\_slots(a_{slots}, x)$ picks the smallest x slots from the a_{slots}. The algorithm begins by excluding slot 0 from the selection, as it is reserved for the minimal cell [42]. Next, the algorithm identifies the slots which are already scheduled for the parent node n_j, which is BS_j. Then, the slots present in both the lists BS_i and BS_j are also excluded from the list of available slots to prevent scheduling conflicts.

The algorithm first verifies whether the node n_j is the Destination Oriented Directed Acyclic Graph (DODAG) root. If yes, the algorithm finds the maximum x slots from the available slots using the $max\_slots()$ function. Subsequently, the node n_j directly allocates the selected slots to the requesting node n_i. This approach guarantees that the remaining nodes in the subtree should get a slot before the scheduled slot of the parent node n_j, enabling it to efficiently forward data packets for the entire path of the subtree.

Otherwise, the algorithm finds min_slot_parent as the smallest slot number from the list $Tx\_slots\_n_j$, indicating the minimum slot the parent node has used to forward its data packets. It further checks whether the minimum slot number from the a_{slots} is less than min_slot_parent value. If the condition is $TRUE$, the algorithm finds slots in the a_{slots} which are less than min_slot_parent. Further, the algorithm checks whether the number of slots in updated a_{slots} is more than the number of cells requested (i.e. x) by the node n_i. If so, finally, it selects the largest x slots from the a_{slots} using the $max\_slots()$ function. By doing this, the algorithm ensures that in a path from source to sink, the predecessor link should get a cell before a cell for the successor link. Further, it targets that two adjacent links along the path should get cells as close as possible so that buffering time in the queue of intermediate nodes becomes minimal. For example, for a path Ⓖ→Ⓑ→Ⓐ, the link Ⓖ→Ⓑ should get

a cell before the cell for Ⓑ→Ⓐ and as much close as possible so that the packet buffering time in the queue of B becomes minimum. On the other hand, if the algorithm does not find any slot before the transmission slot of the parent node, the algorithm chooses the smallest x slots from the a_{slots} using the $min\_slots()$ function. This ensures that the node gets a transmission opportunity as early as possible in the slotframe in order to minimize the end-to-end latency. In the worst-case scenario, if there exist insufficient available slots to fulfil the node's request, the algorithm selects all slots from a_{slots} and allocates all of them to the requesting node. This approach is adopted to prevent any negative impact on PDR. Note that the algorithm allocates nothing to the requesting node n_i if there is no available slot.

It is important to note that all computations are performed at the receiver's end. Specifically, the cell selection algorithm is executed by the parent node n_j, which is the preferred parent of n_i. This allows the receiver to judiciously allocate cells to all its children, thereby reducing buffering delays and avoiding schedule collisions. Additionally, the receiver-based allocation completes the negotiation process in two steps instead of a 3-step 6P transaction (detailed in Section 3.3.2). As a result, resource consumption due to 6P message exchange will also be less than the default approach used by existing scheduling functions.

Algorithm 3: *ChannelOffset* Selection

Input: $selected\_slots$, N_c: number of channels
Output: $Ch_{of}[]$

```
for each slot in selected_slots do
    Ch_of[] = randomsample(0, N_c − 1)
return Ch_of[]
```

3.3.1.2 *ChannelOffset* (Ch_{of}) Selection

The *ChannelOffsets* corresponding to the *SlotOffsets* selected by Algorithm 2 are chosen according to the Algorithm 3. The algorithm takes two inputs: $selected\_slots$ and N_c, in which $selected\_slots$ are the selected slots returned by Algorithm 2, and N_c is the number of channels to be considered for *ChannelOffset* selection. Each *SlotOffset* in the $selected\_slots$

picks one *channelOffset* from the available N_c channels randomly.

At the end of both the *SlotOffset* and *ChannelOffset* selection phases, RTDS invokes the cell allocation procedure as explained in Section 3.3.2. Once the required cell allocation is successfully completed, the child node gets x number of additional negotiated cells scheduled with its parent node facilitating traffic between them. Figure 3.2 illustrates a network topology and the corresponding link schedule designed in line with our proposed scheme.

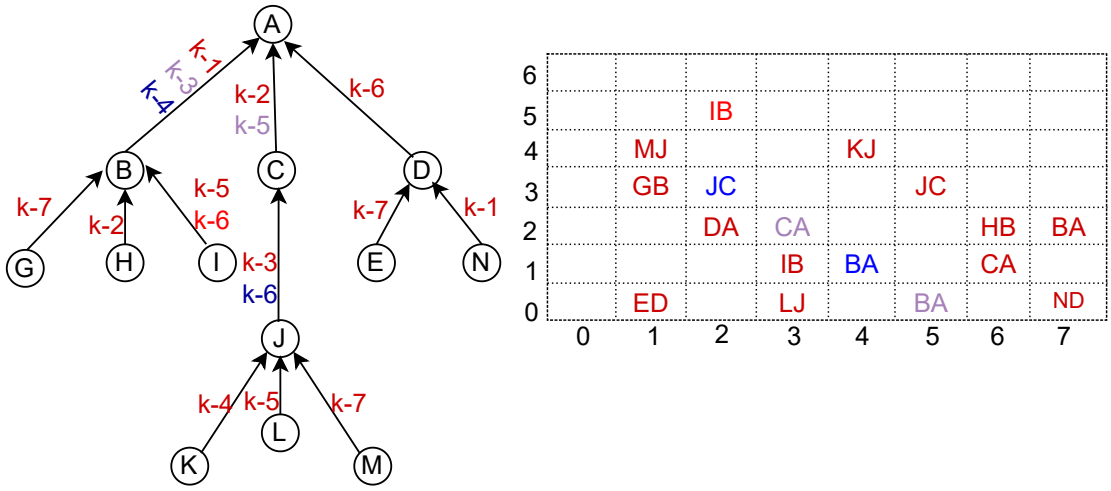


Figure 3.2: *An example of a network topology and its corresponding schedule designed by RTDS considering 7 channels and a slotframe with 8 slots, i.e. $N_c = 7$, $k = 8$. The values shown with the links are selected SlotOffset only.*

3.3.1.3 Few Examples of Cell Selection Process

For a more comprehensive understanding of the proposed cell selection process, we discuss a few specific instances of cell selection for the network topology shown in Figure 3.2.

*Case*1) Consider that the schedule is initially empty for node Ⓒ. Suppose node Ⓒ computes x as 1 for the link Ⓒ→Ⓐ. Node Ⓒ sends a 6P ADD request to node Ⓐ for adding one more cell. Node Ⓐ calculates the slot to be allocated to node Ⓒ as follows. First, it sets a_{slots} as $\{1, 2, 3, 4, 5, 6, 7\}$. The BS_C list is empty as Ⓒ currently has no scheduled slot. Let BS_A be $\{7\}$, which means Ⓐ has an already allocated slot with Ⓑ at slot 7. The updated a_{slots} according to line 5 in the algorithm 2 is $\{1, 2, 3, 4, 5, 6\}$. Here, the parent node Ⓐ is the DoDAG root. Therefore, the algorithm selects the greatest slot from the a_{slots}, which is 6. This methodology ensures that the subtree of Ⓒ gets cells before

the cell for link Ⓒ→ Ⓐ. Thus, the link Ⓒ→ Ⓐ gets the slot in $k-2$, where k is the number of slots in the slotframe. Now for the selected slot 6, *ChannelOffset* will be selected according to the Algorithm 3. In this example, the selected *ChannelOffset* is 1.

*Case*2) Assume link Ⓙ→ Ⓒ requires a cell (i.e. x is 1), and the list BS_J is empty. Let the list BS_C is $\{6\}$, as Ⓒ has already allocated a slot at position 6. So, the updated a_{slots} is $\{1,2,3,4,5,7\}$. The values of $min\_slots(a_{slots},1)$ and min_cell_parent are 1 and 6, respectively. Therefore, the condition in line 12 is true; thus, the algorithm finds slots in the a_{slots} less than 6. So, the updated a_{slots} becomes $\{1,2,3,4,5\}$. Finally, it selects the greatest slot from the a_{slots}, which is 5. Thus, the link Ⓙ→ Ⓒ gets the slot $k-3$. Thus, the algorithm is able to ensure that for the path Ⓙ→Ⓒ→Ⓐ, Ⓙ→Ⓒ gets a cell before Ⓒ→Ⓐ with a minimum slot gap in between.

*Case*3) Suppose node Ⓘ calculated x as 2 for the link Ⓘ→Ⓑ and the list BS_I empty. Let the list BS_B is $\{1,4\}$. The updated a_{slots} becomes $\{2,3,5,6,7\}$. The value of smallest slot in a_{slots} is 2 and value of min_cell_parent is 1. Therefore, the condition in line 12 is false. Thus, the algorithm selects the smallest two slots from the a_{slots}, which are 2 and 3. Thus, the link Ⓘ→Ⓑ gets the other two slots in $k-6$ and $k-5$ before the cells for link Ⓑ→Ⓐ and with a minimum possible slot gap between the adjacent links Ⓘ→Ⓑ and Ⓑ→Ⓐ.

3.3.2 Modified 6P Protocol for Cell Allocation

As we have proposed a receiver-based cell selection mechanism, the parent node n_j is responsible for all the computations and allocating cells to the requesting node n_i. Thus, the parent node n_j needs information about the slots already scheduled by node n_i to avoid any schedule conflicts. Therefore, it is required to update the message fields in 6P control messages used for cell allocation procedure. In the updated 6P protocol, the requesting node n_i informs node n_j with its already scheduled slots instead of sending candidate cells as *CellList* as mentioned in default 6P protocol operations shown in Section 2.1.4.2. Figure 3.3 illustrates our modified 6P protocol. In this depiction, node Ⓑ sends a 6P ADD

request to its parent node (A), specifying the number of cells ($NumCells$) to be added for the link (B)→(A) and its active slot list ($BusySlotListChild$), indicating the slots already scheduled by node (B). The $BusySlotListChild$ may be empty if the requesting node has no scheduled slots. Upon receiving the ADD request, node (A)'s scheduling function uses Algorithms 2 and 3 to determine which cells should be selected for scheduling between the nodes. Node (A) then sends a 6P Response to node (B), conveying the list of chosen cells ($SelectedCellList$). Note that in Algorithm 2, $NumCells$ and $BusySlotListChild$ correspond to x and BS_i, respectively. Upon receiving a successful 6P Response message, nodes (A) and (B) add the selected cells to their respective schedules. Notably, the modified 6P negotiation process uses receiver-based cell selection in a 2-step 6P transaction, thereby reducing the overall 6P overhead.

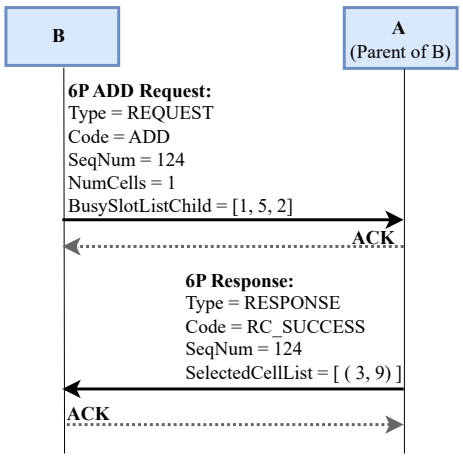


Figure 3.3: *Modified 6P with 2P ADD transaction*

3.4 Analysis of Delay and 6P Overhead

This section presents an analysis of the average end-to-end delay and 6P overhead of the random cell scheduling function MSF and the proposed RTDS scheduling scheme.

3.4.1 End-to-End Delay Analysis of MSF

We calculate the average waiting time for a source node to deliver a packet to the destination node following a specific routing path r.

Assuming a node needs x cells to enter the stable boundary of cell utilization ratio once it becomes unstable. In MSF, one cell is added or deleted at a time in each evaluation cycle. Therefore, a node must wait for multiple evaluation cycles to allocate those x cells required to reach the stable region. So, installation of those x cells requires a resource allocation period of x evaluation cycles. It shows that the resource allocation period depends on the number of cells per slotframe already scheduled between the two nodes, say NCS. By default, one evaluation cycle is completed once the NCS reaches maximum limit $MaxNumCell$. We can also say that one evaluation cycle lasts for $\frac{MaxNumCell}{NCS}$ slotframes.

Let $T(NCS, x)$ be the time required to add x new cells to a link having NCS number of already allocated cells. This time can be estimated by adding the duration of x consecutive evaluation cycles as follows. The initial segment of the Equation 3.4 represents the same.

Note that the evaluation window for each cycle varies with the current number of scheduled cells, and in every cycle, the number of cells elapsed ($NumCellsElapsed$) corresponds to the total number of cells scheduled between the nodes. Furthermore, cell addition is managed by the 6P protocol, which requires an additional 6P negotiation time. Specifically, Req_delay and Res_delay represent the time slots spent by the 6P ADD Request message and 6P ADD Response message in the transmission queue before getting a transmission cell. Assuming cells are distributed uniformly over a slotframe, the average 6P request and response delay can be computed as $\frac{SF_{length}}{2 \times NCS}$. The latter portion of Equation 3.4 includes these additional times.

$$T(NCS, x) = SF_{length} \times \left(\sum_{c=0}^{x-1} \frac{MaxNumCell}{NCS + c} + \frac{1}{2 \times NCS} + \frac{1}{2 \times NCS} \right) \tag{3.4}$$

Now, we have the total time by which the node will acquire all its required cells to communicate with its preferred parent.

To obtain the end-to-end delay, it is required to calculate the waiting time in the buffer for

one hop communication (ohd). The negotiated cells are scheduled randomly, and we assume negotiated cells are uniformly distributed in the slotfarme. A packet remains enqueued until the availability of the required negotiated transmission cells and the average waiting time corresponds to the required cell allocation period. Thus, ohd is the number of slots a node needs to wait to acquire the required cells needed to transmit the data, divided by the number of negotiated cells scheduled between the two nodes. Note that the transmission queue fills up as long as not enough cells are allocated for the traffic load; some packets may experience a long wait time to reach their destination, ultimately affecting end-to-end latency. We calculate the normalized delay in terms of the number of timeslots, which is then multiplied by the timeslot duration ($slot_{length}$) to get the actual delay. Thus, we have the ohd as follows:

$$ohd = \frac{T(NCS, x)}{2 \times NCS} \tag{3.5}$$

If the link between nodes n_i and n_j is poor, a packet may require, on average, ETX retransmissions for successful delivery. So, the Equation (3.5) can be modified as:

$$ohd(n_i, n_j) = \frac{T(NCS(n_i, n_j), x(n_i, n_j)) \times ETX(n_i, n_j)}{2 \times NCS(n_i, n_j)} \tag{3.6}$$

Thus, the average end-to-end delay for transmitting a packet along the routing path r can be computed as:

$$e2e\_delay(r) = \sum_{(i,j) \in r}^{|r|} ohd(n_i, n_j) \tag{3.7}$$

where, $|r|$ represents the number of links in the path r, and (i, j) denotes the link between nodes n_i and its parent node n_j. Finally, the actual end-to-end delay in terms of seconds is:

$$e2e\_delay(r) = slot_{length} \times \sum_{(i,j) \in r}^{|r|} ohd(n_i, n_j) \tag{3.8}$$

In summary, minimizing the end-to-end delay requires determining the necessary cell count (x) to reach the stable zone, followed by the time needed to allocate these x cells using a single 6P ADD transaction.

3.4.2 End-to-End Delay Analysis of RTDS

RTDS allocates the estimated number of required cells (x) using a single 6P transaction, limiting the allocation period to just one slotframe length. Therefore, the time required to allocate the x cells considering the 6P request and response delays is calculated as shown in Equation (3.4):

$$T(NCS, x) = SF_{length} \times \left(1 + \frac{1}{2 \times NCS(n_i, n_j)} + \frac{1}{2 \times NCS(n_i, n_j)}\right) \tag{3.9}$$

Now, to get the end-to-end delay, we first compute *ohd* considering the retransmission as explained in Equations (3.6) and (3.9) for *T(NCS, x)*.

Finally, the end-to-end delay along the routing path r is computed using Equations (3.8) and (3.9) for getting the value of *T(NCS, x)*. Figure 3.4 shows the end-to-end latency comparison of numerical and simulation results to validate the proposed RTDS under a scenario where each node generates two packets/slotframe (P/SF). It is observed that the simulation results show slightly higher values compared to the numerical results due to the inclusion of processing and propagation delays in the simulations.

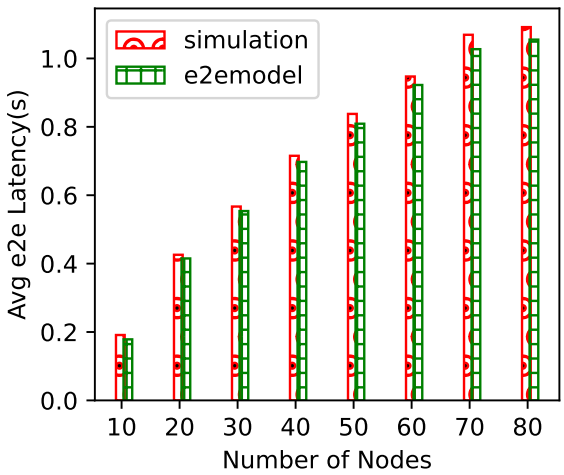


Figure 3.4: *Numerical and simulation results comparison of end-to-end latency under varying network size*

3.4.3 Discussion on 6P Overhead

This section presents a comparison of the 6P overhead, which represents the number of 6P control message exchanges needed to negotiate cell allocations between two nodes.

On average, a node needs x additional cells to reach a stable region after becoming unstable. Since MSF installs one cell at a time, it requires x evaluation cycles to check and update its schedule. In every cycle, one 6P transaction is required to schedule one cell. Thus, the 6P overhead in MSF can be expressed as: $x \times$ (Overhead of one 6P transaction)

A 6P transaction is a complete negotiation between two neighbouring nodes. It ends when the cell has been added/deleted or when the 6P transaction fails. A single 6P transaction involves four message exchanges between the nodes as explained in Section 2.1.4.2. Moreover, in a lossy communication channel, the failure probability of a 6P transaction can be significant [97]. Let ETX be the average number of retransmissions required for successful delivery of a 6P packet. With the transmission cost of each 6P packet being unity, the 6P overhead required for a node to reach the stable region from the unstable region is then given by:

$$6P\_Overhead_{MSF} = x \times (4 \times ETX(n_i, n_j)) \tag{3.10}$$

Where n_j is the parent of n_i who is the negotiating node. The 6P overhead of Stratum and LLSF schemes is similar to that of MSF, as they also follow a one-by-one approach for the addition and deletion of cells. Consequently, a node requires x 6P transactions to schedule x cells.

In RTDS, multiple cells can be added or deleted using a single modified 6P transaction, as discussed in Section 3.3.2. Therefore, once a node becomes unstable, the 6P overhead to reach the stable boundary corresponds to one modified 6P transaction. If the link quality is poor, on average, ETX transmissions are needed before a successful 6P transaction. Thus, the 6P overhead is:

$$6P\_Overhead_{RTDS} = (4 \times ETX(n_i, n_j)) \tag{3.11}$$

Please note that the proposed cell selection strategy is not applicable for cell deletion;

therefore, cell deletion follows the default random procedure, as described in Section 2.1.4.2, similar to existing link scheduling schemes. To address this, we propose the LCD mechanism, which is discussed in the following section.

3.5 LCD

This section introduces an enhanced cell deletion mechanism aimed at reducing end-to-end latency. Illustratively, consider Figure 3.5, for the link Ⓐ→Ⓑ, when MSF decides to delete a cell from the three scheduled cells between nodes Ⓐ and Ⓑ, it typically involves selecting a cell at random for deletion. On the other hand, the proposed LCD mechanism is designed to minimize end-to-end latency by taking into account the temporal relationship of the scheduled cells with the transmission cells of the parent node. Specifically, out of the scheduled cells, LCD preserves the cells having the parent node's reception and transmission cells closest.

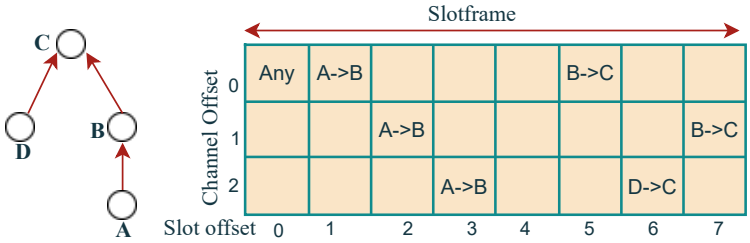


Figure 3.5: *TSCH schedule with 7 slots and 3 channels*

LCD utilizes IMSF to determine the number of cells to delete when the schedule experiences underutilization of scheduled cells while introducing an additional mechanism to decide which specific cells should be removed. Specifically, in LCD, instead of adopting the one-by-one cell deletion approach, the estimation of the number of cells required to be removed from a node's existing schedule is decided by our proposed IMSF to release the unwanted cells by a single 6P transaction, as discussed in Section 3.2.1. Then, the selection of estimated y cells for deletion from the presently scheduled cells between nodes Ⓐ and Ⓑ is performed using Algorithm 4. Notably, the value of y is going to be the value of *NumCells* during the 6P transaction, as shown in Figure 2.5.

Algorithm 4: Cell Deletion Mechanism

Input: y : number of cells to be deleted, n_i : negotiating node, n_j : parent of node n_i
Output: *selected_slots*

```
1  CellList ← find the scheduled negotiated cells between n_i and n_j
2  if y ≤ len(CellList) then
3      tx_cells_parent_n_j ← find the scheduled transmission cells of parent node n_j
4      if len(tx_cells_parent_n_j) > 0 then
5          min_slot_parent ← find the smallest SlotOffsets from tx_cells_parent_n_j
6          max_slot_parent ← find the largest SlotOffsets from tx_cells_parent_n_j
7          min_slot_Celllist ← find the smallest SlotOffsets from CellList
8          max_slot_Celllist ← find the largest SlotOffsets from CellList
9          for each element y in CellList do
10             gap ← y.SlotOffset − min_slot_parent
11             Add y.SlotOffset and the corresponding gap to gap_List
12         if max_slot_Celllist < min_slot_parent then
13             selected_slots ← pick y SlotOffsets from gap_List having y lowest gap
               value
14         else
15             if min_slot_Celllist > max_slot_parent then
16                 selected_slots ← pick y SlotOffsets from gap_List having y lowest gap
                   value
17             else
18                 selected_slots ← pick y SlotOffsets from gap_List having y highest
                   gap value
19     else
           // (n_j is the root node) selected_slots ← pick the greatest y number of
           SlotOffsets from CellList
20     return selected_slots
21 else
22     return RC_ERR_CELLLIST
```

Depending upon the cell utilization, a node n_i initiates a cell deletion request to its parent node n_j, specifying the number of cells (y) to be removed from their schedule. Then, node n_j employs the Algorithm 4 to decide which specific cells to be deleted from the current schedule. If the number of cells requested for deletion exceeds those scheduled with node n_i, node n_j responds with an error message. Otherwise, n_j retrieves its negotiated transmission cells and finds the smallest and largest time slots used for transmitting packets and also the smallest and largest slots among its scheduled slots with node n_i. Then, n_j prioritizes the selection of *SlotOffsets* for deletion with the larger temporal distance to its

first transmission cell aiming to reduce end-to-end latency of data transmissions from source to sink node. There can be three possible cases.

- **Case 1**: All the scheduled cells between node n_i and n_j are positioned before the transmission cells of parent node n_j.
- **Case 2**: All the transmission cells of the parent node n_j are scheduled before the cells for the link $n_i \rightarrow n_j$. In both cases, node n_j will select y slots with the smallest *gap* values.
- **Case 3**: If there is a mixed occurrence of scheduled cells for link n_i and n_j and transmission cells n_j. Then, node n_j selects the slots with the largest *gap* values.

The rationale behind the proposed methodology is to optimize scheduling arrangement by ensuring that the scheduled cells between n_i and n_j remain closer to the transmission cells of the parent node n_j. This minimizes the buffering time in the queue of the parent node n_j, facilitating the early forwarding of packets to reduce end-to-end delay. The proposed strategy ensures that it gets a chance to be forwarded with minimum delay upon receiving a packet. However, if no transmission cell exists for the parent node, LCD opts to delete the farthest scheduled slots in the slotframe. This is to ensure minimum delay with the remained schedule after performing the deletion. This scenario is more likely when node n_j is the root node in a DODAG topology. For example, in Figure 3.5, there are three scheduled cells between nodes (A) and (B) at slots 1, 2, and 3, with node B's transmission cells at slots 5 and 7. The proposed LCD selects the cell at slot 1 for deletion since (A)→(B) positioned at slots 2 and 3 are closer to node B's first transmission cell. If the link (B)→(C) wants to delete a cell from its scheduled cells positioned at slots 5 and 7, with node (C) is the DODAG root meaning no further forwarding is required. It will select the cell appearing later in the slotframe for deletion. Therefore, cell (B)→(C) at slot 7 is chosen for deletion.

3.6 Complexity Analysis

The time complexity of Algorithm 1 is $O(1)$. This is because the algorithm consists of a sequence of conditional statements and operations where each step takes a constant amount of time to execute. Condition checks for $U_{th} < C_U$ and $C_U < L_{th}$ involve simple comparisons, each taking $O(1)$ time. The calculation of x and y from Equations 3.1 and 3.3, respectively, depends on arithmetic operations, which also have $O(1)$ complexity. Checking available slots and ensuring constraints, such as $x > a_{slots}$ and $(len(NCS) - y) < 1$, require basic comparisons, which again run in $O(1)$ time. The assignment of values to x and y, as well as the initiation of 6P ADD / DELETE requests, are operations in constant time. While the actual execution of 6P transactions on the network may vary, the algorithm itself only initiates these operations in $O(1)$ time. Regardless of the input values for C_U, U_{th}, and L_{th}, the algorithm will always perform a limited number of these constant-time operations, and there are no loops or recursive calls that would make the execution time dependent on the input size. Therefore, the execution time of the algorithm remains constant regardless of the input.

The execution of Algorithms 2 and 3 takes place independently on each node within the network. The time complexity of Algorithm 2 is primarily determined by the operations performed on the slotframe and slot selection functions. The initialization of the slot set takes $O(S)$ time, where S represents the total number of slots in the slotframe. While operations such as filtering based on conditions, and list size checks have an average $O(1)$ complexity. The key computational steps involves min_slots and max_slots operations for finding minimum or maximum slot value, which requires scanning the slot set, each taking at most $O(S)$ time. In the worst case, selecting x slots requires $O(S*x)$ complexity. Since x is a small constant, the overall complexity of Algorithm 2 is $O(S)$. S is a constant value, and its maximum value is the slotframe length. As every node independently runs the algorithms, the complexity remains the same, irrespective of network size. Similarly, Algorithm 3 has a complexity of $O(selected\_slots)$, where $selected\_slots$ is a constant value, and its maximum value is the slotframe length. The space complexity of Algorithm 2 is $O(n + m)$, where

n and m are the sizes of a_{slots} and busy slots BS_i, respectively. The space complexity of Algorithm 3 is $O(s)$, where s is the size of the *selected_slots*.

The time complexity of Algorithm 4 can be analyzed as follows: first retrieving the scheduled negotiated cells and the scheduled transmission cells of the parent node takes constant time, $O(1)$, the subsequent steps involve finding the minimum and maximum SlotOffsets which takes linear time, $O(m)$ and $O(p)$ using Python's $min()$ and $max()$ functions that performs linear scans, and their time complexity is proportional to the number of elements in the list, where m is the size of $CellList$ and p is the size of $tx\_cells\_parent\_n_j$. The algorithm then iterates through $CellList$ in a loop that runs m times. The dominant factor in determining the time complexity arises from the need to pick y *SlotOffsets* from the *gap_List* based on their values, which typically involves sorting the *gap_List* (containing m elements). This sorting operation takes $O(m * logm)$ time. Therefore, the overall time complexity of Algorithm 4 is $O(m * logm)$, where m represents the number of scheduled negotiated cells between the negotiating node n_i and its parent n_j. However, it has been observed that in practical runs of this algorithm, the value of m is at most 30. Given this constraint, even though the asymptotic complexity is $O(m*logm)$, for the observed range of m, the algorithm effectively performs a bounded number of operations, leading to a practical time complexity that can be considered constant, $O(1)$, where the number of negotiated scheduled cells remains small.

Compared to the benchmark algorithm MSF, defined in the standard (RFC 9033) [43], which has a time complexity of $O(1)$, our proposed solution consists of four algorithms that are independently executed based on scenarios. These algorithms have time complexities of $O(1)$, $O(S)$, $O(selected\_slots)$, and $O(m * logm)$. However, the values of S, *selected_slots* and m are constant values with their maximum values bounded by the length of the slotframe (101 slots). Due to this constant upper bound, in practice, these algorithms exhibit a time complexity of $O(1)$.

3.7 Performance Evaluation

This section begins by detailing the simulation settings. Next, we present benchmark scheduling algorithms for comparison and describe performance metrics for evaluation. Finally, we present and discuss the simulation results.

Table 3.2: *Experimental parameters*

Parameter	Value/Type
Slotframe Length	101 timeslots
Timeslot duration	10 ms
Tx queue size	10 packets
Number of nodes	10-80
TSCH maximum payload length	102 Bytes
MAC max retransmission	5
Number of channels (N_c)	16
Each iteration cycle	3600 slotframes
Number of iterations	30
Low Traffic Rate	1 packet/20slotframes
High Traffic Rate	2 packet/slotframe
Traffic generation	All non-root nodes
L_{th}	25%
U_{th}	75%

3.7.1 Simulation Settings

This section analyzes the operational performance of the proposed schemes using the 6TiSCH simulator. The chosen simulation parameters are based on real-world deployment scenarios and follow the recommended values from IEEE 802.15.4e TSCH and IETF 6TiSCH standards. For our experiments, we employ random deployment of the nodes in a region of 2 km $\times$ 2 km for all simulation runs, which is a common setup for wireless sensor networks and industrial IoT applications. We employ the RPL MRHOF-ETX objective function [72] to enable multi-hop communication as it is the standard choice for multi-hop communication in constrained IoT networks. Node 0 serves as the DODAG root and so the application's sink node. All non-root nodes complete the join process and finally join the RPL routing structure using the standard CoJP [41] scheme. Each node maintains up to three RPL parents, and it selects one preferred parent among these parents as its default routing parent.

It aligns with real-world RPL-based deployments where multiple parent choices enhance reliability. We consider all transmitted application packets throughout the simulation duration for latency and reliability calculations. We conduct each simulation 30 times with each iteration cycle spanning 3600 slotframes for all the scheduling algorithms, and all results are reported as an average value with a 95% confidence level. Traffic conditions are tested under both low-traffic and high-traffic scenarios to reflect both typical and congested network conditions. We set the slotframe length to 101 slots, as recommended by the IEEE 802.15.4e TSCH standard. Similarly, we use 16 frequency channels, following IEEE 802.15.4e specifications to ensure channel diversity and interference mitigation. The MAC retransmission limit is set to 5, as per the standard's reliability constraints. The threshold values L_{th} and U_{th} are taken from RFC 9033, which recommends 25% and 75% as the optimal values ensuring efficient bandwidth utilization and preventing excessive slot allocations. Table 3.2 summarizes the key simulation parameters. By following the recommended values from the IEEE 802.15.4e TSCH, IETF 6TiSCH and RFC9033 standards, our evaluation closely mirrors real-world IoT deployments.

3.7.2 Benchmark Schemes and Performance Metrics

The following benchmark link scheduling algorithms are considered for comparison, all of which are distributed in nature:

- **MSF** [43] is the first full-fledged algorithm standardized for 6TiSCH networks, adaptive to changes in network conditions.
- **Stratum** [51] targets to reduce latency by dividing slotframe into different stratums where each node selects a cell within a stratum based on its hop distance.
- **LLSF** [54] aimed to minimize latency through daisy-chaining timeslots.
- **EOTF** [49] aimed to adapt the bandwidth reservation to network requirements.

The MSF implementation is already integrated into the simulator. The code and scripts for

Stratum, LLSF and EOTF scheduling functions are available online [1], which we have used for simulation purposes.

The following performance metrics were used for comparison:

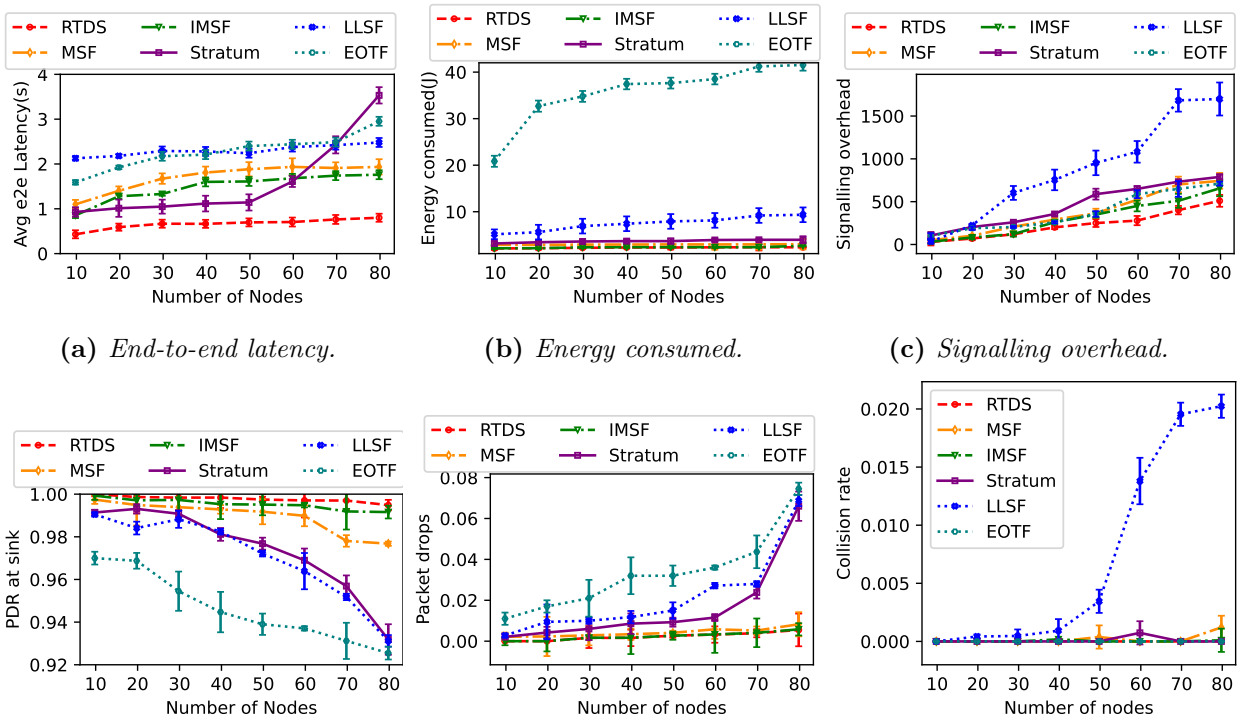
- *End-to-end latency* - The total time required for a packet to travel from the source node's application layer to the destination node's application layer.
- *Energy consumption* - We use the energy consumption model proposed by Vilajosana *et al.* [98] for computing energy consumption.
- *Signalling overhead* - Total number of 6P message exchanges required for cell negotiation between nodes in the network.
- *Packet Delivery Ratio (PDR)* - Ratio of the number of application packets received at the sink node to the total number of packets generated and sent by all the nodes in the network.
- *Packet drop rate* - The ratio of packet drops due to the lack of transmission cells and transmission queue overflow to the total number of transmitted packets.
- *Collision rate* - The ratio of collisions that occur during data transmission to the total number of transmitted packets within the network.

3.7.3 Simulation Results

In this chapter, first, we evaluate the performance of the proposed schemes, IMSF and RTDS, under two different traffic rates, representing low and high traffic conditions in an application. Specifically, the simulations are conducted using one packet per twenty slotframes (1P/20SF) to represent low traffic and two packets per slotframe (2P/SF) to represent high traffic. However, the proposed schemes are not confined to these traffic rates only. It is important to note that the primary focus of the proposed IMSF is on the cell

[1] http://github.com/vkotsiou/Scheduling

computation process. Building on this, RTDS extends IMSF by adopting the cell computation method while introducing a delay-aware, receiver-based cell selection and allocation approach. Furthermore, LCD extends the cell deletion framework of IMSF and incorporates a latency-aware cell deletion mechanism to determine which specific cells should be deleted. The following first two subsections present a detailed analysis of the simulation results for the above-mentioned two traffic rates. The subsequent subsection provides a comparative analysis of the LCD scheme against the standard MSF and our proposed IMSF, emphasizing how the random cell deletion strategy employed in MSF and IMSF can lead to increased end-to-end latency. In MSF, cell deletion is executed by randomly selecting one of the scheduled cells between two communicating nodes. Although IMSF estimates the number of cells to delete, it still removes them randomly. The results highlight the importance of a latency-aware cell deletion, as demonstrated by the LCD in minimizing end-to-end latency.



(a) *End-to-end latency.* **(b)** *Energy consumed.* **(c)** *Signalling overhead.*

(d) *PDR at the sink.* **(e)** *Packet drop rate.* **(f)** *Collisions rate.*

Figure 3.6: *Comparison of results under varying network sizes with low traffic rate (e.g.: 1P/20SF)*

3.7.3.1 In case of Low Traffic Rate (1P/20SF)

Figure 3.6 illustrates various performance metrics at a traffic rate of 1P/20SF. We conducted experiments with network sizes ranging from 10 to 80 nodes to gather these results. At first, we calculated the average latency across different network sizes, as shown in Figure 3.6a. Our proposed scheme RTDS stands out as the most effective approach for reducing end-to-end latency. For example, in an 80-node network, it reduces the end-to-end latency by 58%, 77%, 67%, 72% and 54% compared to MSF, Stratum, LLSF, EOTF and IMSF, respectively. RTDS schedules cells for a path in a cascading manner, allowing data packets to be forwarded with the minimum possible delay and thus able to minimize the average end-to-end delay. On the other hand, MSF schedules cells at random slot positions, and it has also been observed that it requires more retransmissions due to cell collisions, ultimately resulting in higher end-to-end delay. While the Stratum scheme initially demonstrates latency improvements compared to MSF, it is worth noting that its performance with respect to end-to-end latency deteriorates as the network size grows. This degradation is attributed to the increased contention for cells within a smaller stratum, leading to increased retransmissions caused by packet drops.

For energy consumption, we consider the following charge consumption associated with each cell: 1. Sleep: when a node's radio is off. 2. Idle_listen: when a node listens for data but receives nothing. 3. tx_data: when a node sends some data without expecting ACK. 4. rx_data: when a node receives some data without exchanging an ACK 5. tx_data_rx_ack: when a node transmits data and anticipates receiving an ACK. 6. rx_data_tx_ack: when a node receives data and sends back an ACK. Throughout the simulation, we tally the occurrences of ACK and data transmissions and subsequently converted this activity to the energy consumption of the nodes. It can be observed that the proposed RTDS exhibits lower energy consumption compared to all other schemes. For example, in an 80-node network, RTDS lowers the energy consumption by 22%, 40%, 74%, 94% and 11% compared to MSF, Stratum, LLSF, EOTF and IMSF, respectively. This improvement is attributed to the reduced need for packet retransmissions, lower 6P overhead and a lower occurrence of packet

drops. As shown in Figure 3.6c, RTDS exhibits lower 6P signalling overhead compared to all the benchmark schemes. IMSF and RTDS execute a single 6P transaction to add or delete cells rather than employing a one-by-one addition or deletion process that entails multiple evaluation cycles and triggers numerous 6P transactions. Increased 6P transactions lead to a higher volume of 6P message exchanges, resulting in elevated 6P overhead. It is important to note that the number of 6P transactions performed between two nodes depends on the calculation of x and y using Equations 3.1 and 3.3, which, in turn, rely on the network's current statistics. Thus, due to RTDS scheduling nature, it reduces 6P overhead compared to IMSF. For instance, RTDS reduces the signalling overhead by 31%, 35%, 69%, 28% and 28% compared to MSF, Stratum, LLSF, EOTF and IMSF, respectively.

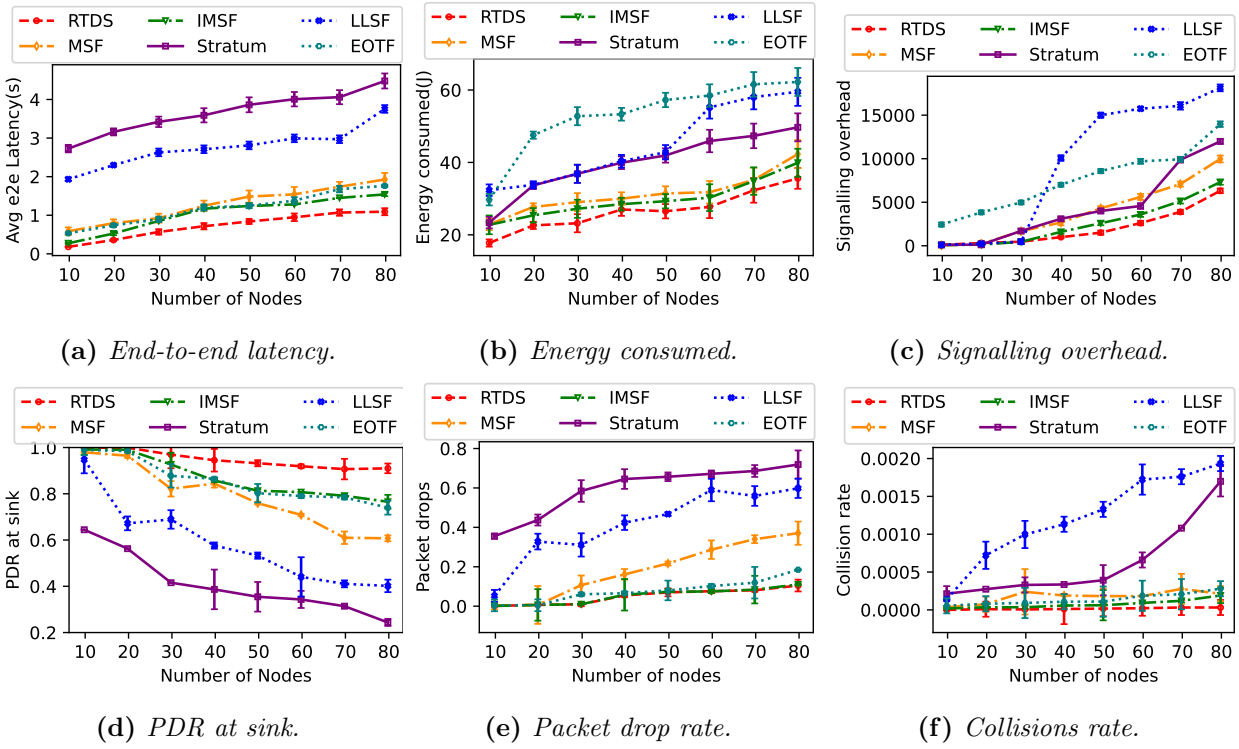
Figure 3.6d shows that the proposed RTDS achieves higher PDR and thus maintains higher transmission reliability by delivering more than 99% of the packets to the sink node, even in a network with 80 nodes. It is evident that RTDS effectively minimizes end-to-end latency while maintaining high PDR. In contrast, the Stratum scheme encounters a significant drop in PDR when it aims to minimize latency. We use packet drop rate and collision rate metrics to provide a more detailed insight. MSF has a higher packet drop rate than RTDS as MSF updates cells in unity, and thus, many packets are dropped due to the unavailability of negotiated transmission cells. The transmission queue fills up as long as not enough resources are allocated for the traffic load; some packets may experience a long wait time to reach their destination. Further, most of the losses occur during this cell allocation period. When the transmission queue is full, the node drops all new packets. When resources are sufficient, the average empty rate of the queue equals its fill rate, and packets are not dropped anymore. Further, it schedules cells randomly using simple 6P negotiation, increasing the chances of schedule collisions, as shown in Figure 3.6f. The better performance of the proposed RTDS scheme is due to the faster resource allocation process, which makes the transmission cells available to the nodes as and when required with a minimum waiting time. Further, based on the sender and receiver's information, the receiver node allocates cells judiciously among all its children. Figure 3.6e illustrates the

rate of packet drops due to the non-availability of transmission cells and transmission queue overflow. We can note that Stratum has a dominant packet drop rate, as its block-based design cannot accommodate the needed transmission cells for the nodes inside a small block. On the other hand, the proposed RTDS minimizes packet dropping significantly compared to all the benchmark schemes. In addition, RTDS has no collisions with any network sizes with this low traffic rate, as shown in Figure 3.6f.

3.7.3.2 In case of High Traffic Rate (2P/SF)

The results for traffic rates of 2P/SF are presented in Figure 4.3. Figure 3.7a shows the end-to-end latency comparison. Under this traffic rate, RTDS achieves an average end-to-end latency of 1.09 seconds with 80 nodes, significantly lower than the MSF, even when considering the dynamic topological changes in real-time. Stratum performs the worst under high traffic rates, crossing 4.48 seconds in an 80-node network. The proposed RTDS, on the other hand, consistently achieves low latency with minimum variance regardless of traffic rates. This improvement is primarily achieved by efficiently forwarding data packets with minimum buffering delay, which eventually leads to lower end-to-end delays. In addition, RTDS makes the required transmission cells available to the nodes faster, reducing the resource allocation period. Thus, RTDS can effectively limit end-to-end latency to nearly the length of a slotframe. In contrast, this is not the case with MSF, which schedules cells at random positions in a slotframe without any coordination of transmission and reception cells. RTDS is robust to high traffic rates and provides lower delays. However, with the Stratum scheme, there is a noticeable increase in collisions as traffic rates increase. Additionally, as the network size expands, more nodes compete for the same cells within a small block or stratum to transmit their data packets. This heightened contention among nodes leads to an increased frequency of retransmissions, ultimately resulting in more significant delays in the network.

Figure 3.7b presents the energy consumption of all the schemes. Even at this traffic rate, the proposed RTDS maintains lower energy consumption than all benchmark schemes. To



(a) *End-to-end latency.* **(b)** *Energy consumed.* **(c)** *Signalling overhead.*

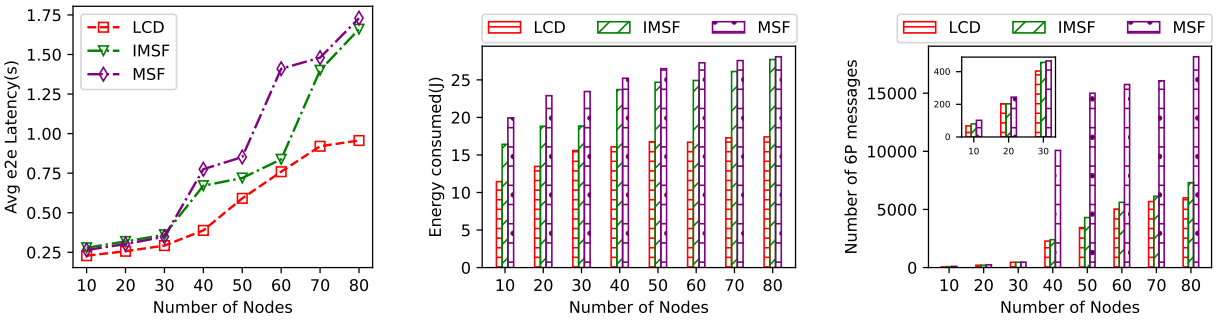
(d) *PDR at sink.* **(e)** *Packet drop rate.* **(f)** *Collisions rate.*

Figure 3.7: *Comparison of results under varying network sizes with high traffic rate (e.g.: 2P/SF)*

illustrate, RTDS lowers energy consumption by 15% compared to the standard MSF and 10% compared to IMSF. Furthermore, unlike Stratum, LLSF, and EOTF, RTDS showcases significantly lower energy consumption, with reductions of 28%, 40%, and 42%, respectively. Note that the energy consumption considering a traffic rate of 2P/SF is more than that with a 1P/20SF traffic rate. This increase in energy consumption is primarily attributed to the higher volume of generated packets, resulting in more frequent transmission and reception activities within the network. However, it is worth highlighting that the RTDS scheme consistently exhibits the lowest energy consumption in both high and low-traffic rates scenarios. This improvement is due to its ability to reduce 6P control packets and also minimise the need for retransmissions. Figure 3.7c presents the signalling overhead of all the schemes.

As shown in Figure 3.7d, the proposed RTDS achieves higher PDR even at high traffic rates compared to the benchmark schemes. It maintains higher PDR by delivering more than 90% of the packets to the sink node in an 80-node network. As shown in Figure 3.7e, RTDS reduces the drop rate resulting from the unavailability of negotiated transmission

cells and full queues. It achieves this by promptly making the *Tx* cells available to the nodes that require transmission. In addition, by efficiently allocating cells and coordinating transmissions among the subtree, the proposed receiver-based cell allocation minimises collisions to a large extent compared to random cell selection mechanisms of all benchmark schemes, as shown in Figure 3.7f.



(a) *End-to-end latency* **(b)** *Energy consumed* **(c)** *6P message exchanges*

Figure 3.8: *Comparison of results under varying network sizes where each node generates 1 to 2 packets/slotframe*

3.7.3.3 Impact of Cell Deletion Strategy on Performance

Figure 3.8 demonstrates the impact of random cell deletion on performance and provides a performance comparison between LCD, MSF [43], and IMSF. This performance comparison focuses on LCD, MSF, and IMSF because LCD builds upon the IMSF framework while introducing a latency-aware cell deletion mechanism. Our goal is to highlight the impact of strategic cell deletion on end-to-end latency, which is not addressed in other approaches. MSF employs a fully random deletion strategy and removes one scheduled cell at a time without determining the number of cells to be deleted from the schedule, leading to inefficiencies and an increased end-to-end delay. IMSF improves upon MSF by estimating the number of cells to be removed but still relies on random selection for deletion. LCD further refines this process by not only determining the number of cells to be deleted but also selecting specific cells in a way that minimizes latency. Figure 3.8 emphasizes this distinction by comparing LCD against MSF and IMSF, demonstrating how random cell deletion in the latter schemes results in performance degradation. Since LCD is an enhancement of

IMSF and directly addresses the limitations of MSF's deletion mechanism, comparing it with these schemes provides the most relevant and insightful performance analysis. This comparison allows us to clearly demonstrate the benefits of latency-aware cell selection for deletion over a random approach. It is worth noting that for this evaluation and to highlight the importance of cell selection for deletion, we focus solely on the cell deletion aspect of IMSF to maintain consistency.

Figure 3.8a shows end-to-end latency comparison. IMSF improves latency over MSF. However, LCD further minimizes latency by strategically removing cells from the schedule, ensuring that the post-deletion schedule has cells closer to the parent transmission cells, minimizing packet forwarding delays. Furthermore, both IMSF and LCD execute a single 6P transaction to delete unwanted cells from the schedule instead of using a one-by-one deletion process that requires multiple evaluation cycles and numerous 6P transactions. More 6P transactions result in more 6P message exchanges and thus create more 6P overhead. The proposed LCD and IMSF reduce the 6P communication overhead compared to MSF, as demonstrated in Figure 3.8c. It is worth noting that, in the case of cell deletion, the number of 6P transactions performed between two nodes depends on the calculation of y using Equation 3.3, which again depends on the network's statistics present at that time. LCD, with its deletion approach, incurs less 6P overhead than IMSF and due to the reduced 6P message exchanges and the designed scheduling, LCD boasts lower energy consumption, followed by IMSF, as demonstrated in Figure 3.8b.

3.8 Summary

This chapter introduces *Improved Minimal Scheduling Function* (IMSF) to optimize 6TiSCH network performance. IMSF efficiently adapts to dynamic network conditions and adds or removes cells with a single 6P transaction, minimizing convergence time. It estimates the required number of cells based on the current network state and traffic demands. To further improve performance, we propose the *Receiver-based Traffic rate-agnostic Distributed link Scheduling function* (RTDS). RTDS improves end-to-end latency, resource allocation,

and signalling overhead by employing a delay-aware receiver-based cell selection and allocation strategy on top of IMSF. Finally, to address random cell deletion, we propose a *Latency-aware Cell Deletion* (LCD) mechanism that selects cells for deletion based on delay considerations, further optimizing network performance. The chapter also provides an analytical discussion of end-to-end delay and 6P overhead. Extensive simulations using the 6TiSCH simulator demonstrate the superior performance of the proposed schemes. Although IMSF outperforms benchmark schemes, RTDS further achieves lower end-to-end latency while maintaining a high PDR regardless of traffic load. For example, in a 70-node network operating at 2P/SF, RTDS achieves a 38% and 73% reduction in end-to-end latency compared to MSF and Stratum, respectively. In the same scenario, it improves the PDR by 48% and 65%, and reduces the energy consumption by 15% and 28% compared to MSF and Stratum. Furthermore, LCD reduces end-to-end latency by 46% compared to MSF.

While a well-designed scheduling function can reduce schedule collisions, it cannot be avoided fully. The proposed cell selection and allocation scheme, in this chapter, can not fully eradicate collisions and does not address collision mitigation in link schedules. In Chapter 4, we delve into collision detection and mitigation approach for 6TiSCH networks.

"In the middle of every difficulty lies opportunity."

~Albert Einstein (1879-1955)

4

Collision Detection and Mitigation in Link Schedules

This chapter introduces an enhanced collision detection mechanism based on Cell Delivery Ratio (CDR). To mitigate collisions, we propose a receiver-based latency-aware cell relocation strategy. The effectiveness of the proposed schemes is evaluated through simulations on the 6TiSCH network simulator.

4.1 Introduction

Despite advancements in scheduling algorithms, random cell negotiation locally can lead to schedule collisions. With the local view of the network, neighbouring pairs of nodes may

select the same negotiated slots for communication. In such case, schedule collision occurs if these nodes are within the interference range of each other and transmit simultaneously on the same channel. As a consequence, if a schedule has multiple colliding cells, the network suffers from high packet losses which yields to degrading network reliability and overall performance. Although in the previous chapter (i.e., Chapter 3), we have proposed an adaptive receiver-based link scheduler, but we did not address how to handle collision in link schedules of the 6TiSCH network. Therefore, effective collision detection and mitigation mechanisms are essential for ensuring efficient network performance and data reliability. Specifically,

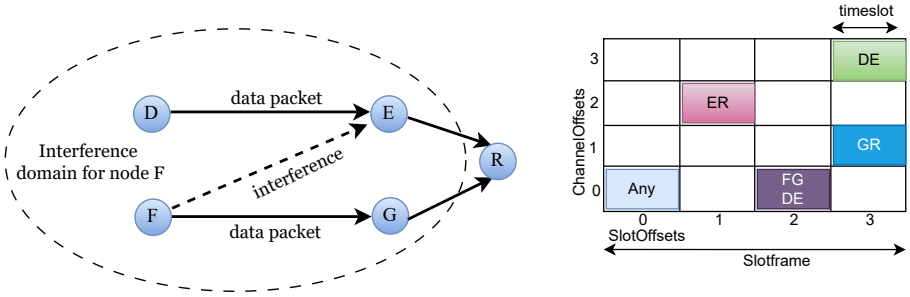


Figure 4.1: *An illustration of schedule collision*

due to the distributed nature of 6TiSCH scheduling, there is a non-zero probability of two neighbouring node pairs allocating the same cell (Ts_{of}, Ch_{of}) in the TSCH schedule. In such a scenario, data exchanged by the two pairs of nodes may collide on that cell, which we call schedule collision. Figure 4.1 depicts a scenario of a schedule collision. Link (D)→(E) has the same cell position as for the link (F)→(G). In this case, if nodes (D) and (F) attempt to transmit to nodes (E) and (G) at the same time, since (F) and (E) share the same interference domain, their concurrent packet transmissions can collide, leading to reliability degradation, packet drops, and wasted energy due to retransmissions. Thus, detecting schedule collisions and relocating those collision-prone cells to a different cell position in the schedule is crucial. This process of relocating a cell to a new position in the slotframe is referred to as cell relocation. The 6top protocol facilitates cell negotiation to relocate the collision-prone cells. This periodic collision detection and relocation process is called "6top housekeeping" and involves a handshake mechanism for pairwise negotiation. The decision

to initiate the relocation request is made by the SF running on each node, and cell relocation is then carried out using the 6P protocol as discussed in Section 2.1.4.2 of Chapter 2.

4.1.1 Motivation

Existing approaches ([55] and [56]) including the standardized Scheduling Function (SF) Minimal Scheduling Function (MSF) [43], often fall short in detecting and handling all collision scenarios effectively. The key aspect of their collision detection method relies on the assumption that if a node has multiple cells scheduled to its parent, all cells should exhibit the same Packet Delivery Ratio (PDR). This is typically achieved by identifying the cell with the highest PDR and comparing the PDR of other cells against that highest PDR. However, this approach does not comprehensively address all collision scenarios. For instance, it cannot detect collisions when only a single cell is scheduled between a node and its preferred parent, nor does it handle the case effectively where all cells toward the parent experience collisions. These limitations emphasize the need for more robust collision detection mechanisms in 6TiSCH networks. Notably, each node selects a preferred routing parent (following the RPL [39] protocol) to forward its data traffic toward the final destination. The node then uses the 6P protocol to negotiate cell scheduling with this selected parent.

Mitigating cell allocation conflict is crucial for network performance. When a collision occurs, it persists across slotframes and thus severely degrades network reliability and wastes energy. It is important to note that the mitigation approach employed in existing scheduling functions ([55], [43], and [56]) typically relies on a sender-based relocation strategy. In this approach, the transmitting node initiates a 6P relocation request to its parent to relocate the collided cells. More specifically, the sender provides a roster of candidate cells, allowing the parent node to select the cells for relocation from this provided list (as we discussed in Section 2.1.4.2). The authors in [56] proposed a cost-aware relocation strategy, and the decision to relocate depends on the overall cost of the network.

It is worth mentioning that in all existing schemes, the preparation of the candidate cell list (*CCellList*) by the sender is done randomly, and it lacks consideration of the receiver's

side information. Similarly, the selection of cells by the parent node from the provided *CCellList* is also random. This often leads to failed relocation attempts due to a mismatch between the sender's proposed cells and the receiver's available slots. Consequently, even though the receiver has available slots, it fails to relocate the collided cells. Furthermore, when the number of available slots at the sender is less than the fixed length of the candidate cell list (*CellListLength*), the transmitting node does not prepare the candidate cell list or initiate a 6P relocation request, even if some slots are available for relocation. We have observed that these challenges arise due to the default sender-based cell relocation approach, leading to a high rate of unsuccessful relocation attempts. This, in turn, significantly increases 6P relocation transactions, causing collisions persist until a successful relocation is accomplished, which ultimately degrades the network performance.

4.1.2 Contribution

These challenges motivate the development of a more efficient and comprehensive strategy for collision detection and mitigation. Our first scheme aims to detect all potential collision scenarios effectively, thereby preventing network performance degradation. We named the proposed scheme as Enhanced Collision Detection (ECD). We leverage the Exponentially Weighted Moving Average (EWMA) of the Cell Delivery Ratio (CDR) as a dynamic link quality estimator. Note that the CDR is nothing but the PDR per cell for a node towards its preferred parent. EWMA offers an assessment of the link's current state by considering the past history of packet transmission status within a sliding window. Further, the proposed ECD can detect collisions in all three mentioned scenarios. In our second scheme, we introduce a receiver-based cell relocation strategy called Enhanced Collision Detection with cell Relocation (ECDR) to address the limitations of the sender-based relocation approach. The receiver, possessing knowledge of its own available slots and the sender's busy slots, makes better decisions regarding cell relocation, eliminating the inefficiencies associated with the default relocation approach. Additionally, while selecting new cells for relocation, the receiver employs a delay-aware cell relocation mechanism to minimize the end-to-end

latency. We integrate the proposed ECDR strategy into the standard 6P message exchange protocol. Finally, we conduct comprehensive simulations using the 6TiSCH simulator to evaluate the effectiveness of our proposed schemes and prove effective in improving the overall network performance.

We summarize the major contributions of this chapter as follows:

- An Enhanced Collision Detection (ECD) mechanism is proposed to capture all collision scenarios effectively.
- A receiver-based latency-aware collision mitigation scheme called Enhanced Collision Detection with cell Relocation (ECDR) is designed to reduce relocation overhead and end-to-end latency.
- The default 6P cell relocation method is modified to facilitate seamless integration of the receiver-based cell relocation strategy.
- Finally, comparison-based performance evaluation of the proposed solutions is conducted using 6TiSCH simulator.

The remainder of this chapter is organized as follows. Section 4.2.1 discusses schedule collisions and the role of cell relocation in mitigating them. Section 4.2 presents the proposed collision detection and mitigation mechanisms. Section 4.3 evaluates the performance of the proposed schemes and compares them with existing benchmarks. We summarized this chapter in Section 4.4. The frequently used symbols and their corresponding meaning are tabulated in Table 4.1.

4.2 Proposed Methodology

This section begins by examining the potential performance degradation caused by cell allocation conflicts in 6TiSCH link schedules. We then discuss the proposed cell collision detection (ECD) mechanism. Finally, we describe the receiver-based latency-aware cell

Table 4.1: *Important notations*

Symbol	Meaning
N	Number of nodes in the network
Ts_{of}	Position of SlotOffset in slotframe
Ch_{of}	Logical ChannelOffset
N_c	Total number of physical channels used
SF_{length}	Length of the slotframe
n_i	Negotiating node n_i
n_j	Preferred parent of negotiating node n_i
$numcells$	Number of cells identified as collided
$RList$	List of cells identified as collided
$CCellList$	A list of candidate cells for the parent node
$selected\_slots$	Selected slots for performing the relocation
$SelectedCellList$	Selected cells for performing the relocation
num_tx	Counts the number of transmission attempts
num_tx_ack	Counts the number of successful transmission attempts
α	Smoothing constant
$RELOCATE\_THRES$	The threshold value for considering a cell for relocation
length(tx_cell_list)	Number of transmission cells scheduled between nodes
$CellListLength$	Default number of cells to be free for preparing $CCellList$
BS_i	Contains already allocated slots (busy slots) of n_i
BS_j	Contains already allocated slots of n_j
LT_{n_i}	Lifetime of node n_i
NL	Network's lifetime
P/SF	Packets per slotframe

relocation approach (ECDR), which is designed to mitigate collisions and improve network performance.

4.2.1 Cell Allocation Conflict in Link Schedules

A cell can experience a low CDR due to various factors, including poor link quality or cell conflict. Packet losses caused by environmental factors, such as fading, would typically affect all cells corresponding to a link equally. Thus, if there is a consistent difference in packet losses among the scheduled cells with the same neighbour, we can say those packet losses stem from cell allocation conflict. 6TiSCH networks utilize the TSCH channel hopping technique by which a cell switches frequencies across slotframes. While a cell may experience poor performance in some slotframes due to channel interference, the channel hopping sequence helps to mitigate the issue over time. So, a cell may perform poorly in some slotframes but recover in others. However, the performance degradation corresponding

to a specific cell is consistent throughout all slotframes if it is due to cell allocation conflict. The probability of such conflicts is related to the product of the neighbouring nodes and the packet rate divided by the total available cells. The probability of cell conflict (PC) increases with more nodes, packet rate, and shorter slotframe lengths. With more nodes competing for fewer cells within a shorter slotframe, the probability of conflict and resulting collisions rises. Due to the use of dedicated cells and the lack of back-off mechanisms in 6TiSCH networks, schedule collisions pose a significant threat to performance degradation.

4.2.2 Enhanced Collision Detection (ECD)

The primary goal of this work is to efficiently detect cell collisions and mitigate their impact on network performance. Our proposed approach leverages the link metric CDR to identify cells experiencing collisions. Given that the success rate of packet transmissions on a cell is inherently stochastic, we compute the Exponentially Weighted Moving Average (EWMA) of CDR. This estimator estimates the link quality between two nodes by averaging the success rate of packet transmissions on a cell over a time window. We assign higher weights to the recent value of CDR than the historical values of CDR. This helps to filter and smooth out fluctuations in the estimation. Specifically, each node monitors the ratio of packets correctly received on its cells scheduled with the neighbouring node during every housekeeping time window. At the end of *t-th* time window, each node calculates the estimated CDR for all its negotiated cells towards its preferred parent using Equation 4.1.

$$CDR_{EWMA,c}(t) = (1 - \alpha) \times CDR_c(t) + \alpha \times CDR_{EWMA,c}(t - 1) \tag{4.1}$$

where $CDR_c(t)$ is determined by Equation 4.2. In Equation 4.2, num_tx denotes the number of packets transmitted by the node towards its parent on cell c and num_tx_ack represents the number of packets for which an ACK is received from the parent on that cell within the *t-th* housekeeping time window. The parameter $\alpha \in [0, 1]$ is the smoothing factor, controlling the influence of the past estimate on the current one.

$$CDR_c(t) = \frac{num\_tx\_ack}{num\_tx} \tag{4.2}$$

Algorithm 5: Cell Collision Detection Algorithm

Input: *tx_cell_list* : List of all cells between the executing node and its parent in a slotframe for which enough number of transmission attempts are made, $RELOCATE\_THRES$: Threshold to decide schedule collision, $CDR_{EWMA}(t-1)$ for each scheduled cell of the executing node

Output: *relocation_cell_list*: List of detected collided cells

Do the following at the end of *t-th* housekeeping time window
if *length(tx_cell_list)* is 1 **then** `// (There is only one scheduled cell with the parent)`
  $c \leftarrow$ **get** cell from *tx_cell_list*
  get $RSSI(t)$
  $expectedCDR(t) \leftarrow$ mapped from $RSSI(t)$
  compute $CDR_{EWMA,c}(t)$ using **Equations** 4.1 and 4.2
  if $RELOCATE\_THRES < (expectedCDR(t)$ - $CDR_{EWMA,c}(t))$ **then**
    relocation_cell_list $\leftarrow$ *relocation_cell_list*.**append** (c)
else `// There is multiple scheduled cells with the parent`
  for each cell c in *tx_cell_list* **do**
    compute $CDR_{EWMA,c}(t)$ using **Equations** 4.1 and 4.2
    $CDR\_list \leftarrow CDR\_list$.**append** $(c, CDR_{EWMA,c}(t))$ `// Saves the CDR`
    $TotalCDR \leftarrow TotalCDR + CDR_{EWMA,c}(t)$
  $AverageCDR \leftarrow \frac{TotalCDR}{length(tx\_cell\_list)}$
  for each cell c in CDR_list **do**
    if $RELOCATE\_THRES < (AverageCDR$ - $CDR_{EWMA,c}(t))$ **then**
      relocation_cell_list $\leftarrow$ *relocation_cell_list*.**append** (c)
  if *length(relocation_cell_list)* is 0 **then**
    get $RSSI(t)$
    $expectedCDR_{bundle}(t) \leftarrow$ mapped from $RSSI(t)$
    if $RELOCATE\_THRES < (AverageCDR$ - $expectedCDR_{bundle}(t))$ **then**
      relocation_cell_list $\leftarrow$ **append** (all cell in CDR_list) `// (Relocate the whole bundle)`
return *relocation_cell_list* `// (List of colliding cells)`

Algorithm 5 is designed to identify schedule collisions within the network. It is executed by a transmitting node n_i and takes two major inputs: (i) *tx_cell_list* is the list of transmission (Tx) cells between node n_i and its parent node n_j, which identifies the number of negotiated cells scheduled between a node and its routing parent. We exclusively consider only those negotiated Tx cells in which a sufficient number of transmission attempts are made. This decision ensures that cells are not relocated unnecessarily when the number of transmission attempts and, thus, successful transmissions on a cell is insignificant. (ii)

$RELOCATE\_THRES$ is a threshold value defining the minimum difference required to classify a cell as experiencing a collision and thus need of relocation. The algorithm will return the list of identified collided cells if any such cells are found. Briefly, the algorithm works as follows:

Step 1: The node n_i maintains a record of the number of packets transmitted and acknowledged on each cell scheduled with its parent node n_j.

Step 2: During every 6top housekeeping period, node n_i calculates the CDR for each negotiated transmission cell with its parent node n_j using EWMA according to Equation 4.1.

Step 3: Three possible cases need to be addressed:

(a) Only one cell is scheduled between node n_i and n_j, and a collision occurs in that cell. Here, collision detection involves comparing the computed CDR of the cell with the expected CDR associated with the Received Signal Strength Indicator (RSSI) of the data packets received over that cell. If a significant difference is observed, the cell is flagged as a collided cell (lines 2-8 in Algorithm 5).

(b) Multiple cells are scheduled between node n_i and n_j, and few of them but not all experience collision. Ideally, all cells scheduled towards the same neighbour should exhibit the same CDR. To identify collision-prone cells, the algorithm computes the average CDR of all cells towards the preferred parent n_j (lines 10-14 in Algorithm 5). Next, it compares each individual cell's CDR with all cells' average CDR. If a cell exhibits significantly lower CDR than the average CDR, it is considered to have experienced a collision (lines 15-17 in Algorithm 5).

(c) Multiple cells are scheduled between nodes n_i and n_j, and all cells in the bundle experience collision. The algorithm evaluates the average CDR of the cell bundle against the expected CDR of the bundle, considering the RSSI of the received packets over those cells. If a significant difference is observed, it indicates that the entire bundle of cells towards the parent n_j exhibits collision (lines 18-22 in Algorithm 5).

Please note that we used the Pister-Hack propagation model [99] to estimate the initial

RSSI values for each pair of nodes in the network. These RSSI values are then converted to PDR values using a conversion table that is established from real-world deployments [1]. This accurately captures the relationship between RSSI and PDR observed in large indoor industrial environments operating at the 2.4 GHz band. Algorithm 5 operates in two main scenarios based on the number of cells in the *tx_cell_list*. If the list contains only one cell, the algorithm performs a series of constant-time operations, including getting the cell, retrieving RSSI, mapping to $expectedCDR$, computing CDREWMA, and a comparison, potentially appending the cell to the relocation list. In this case, the time complexity is O(1). However, if the *tx_cell_list* contains multiple cells (let's say n cells), the algorithm iterates through this list twice. The first loop computes CDR_{EWMA} for each cell, appends to *CDR_list*, and calculates the total CDR, taking $O(n)$ time. After calculating the average CDR, the algorithm iterates through the *CDR_list* again, performing constant-time operations for each cell, resulting in another $O(n)$ time complexity. Finally, there is a conditional check that could lead to the appending of all cells from *CDR_list* to the relocation list, which also takes $O(n)$ time in the worst case. Therefore, in the scenario with multiple cells, the dominant factor is the linear traversal of the cell lists, making the overall time complexity of Algorithm 5 $O(n)$, where n is the number of cells in the *tx_cell_list*. However, it's important to note that the length of the *tx_cell_list* is bounded by the length of the slotframe, which in the worst case is a constant value (100 slots). Due to this constant upper bound on n, in practice, the algorithm exhibits a time complexity that can be considered $O(1)$.

4.2.3 Enhanced Collision Detection with cell Relocation (ECDR)

This section introduces a receiver-based latency-aware cell relocation strategy to address the limitations associated with the default sender-based relocation approach. While selecting new cells for relocation, the receiver cooperates with the sender and employs a delay-aware cell relocation strategy to minimize end-to-end latency. Building upon our previous work in Chapter 3, where we propose a receiver-based approach for new cell allocations, here we

[1]The connectivity traces were sourced from the http://wsn.eecs.berkeley.edu/connectivity/project at the University of California, lead by Thomas Watteyne and Prof Kris Pister. The data set used is "soda".

have proposed an enhanced approach to our previous idea for cell relocation. Specifically, we have used a different approach when the receiving node n_j is a Destination Oriented Directed Acyclic Graph (DODAG) root. In the context of cell relocation, simply selecting the largest available slot (as proposed in Chapter 3) may not always lead to optimal latency reduction.

4.2.3.1 Candidate *SlotOffset* Selection for Cell Relocation

Upon detecting collisions using the ECD scheme, the sender node n_i initiates a modified 6P relocation request to its parent node n_j. Node n_j then executes Algorithm 6 to identify suitable new slots for relocation. Note that the standard 6P relocation request mechanism has been modified to accommodate our receiver-based cell allocation and relocation scheme. This modification is discussed in the next subsection. The relocation algorithm requires the following inputs: *numcells*, n_i, n_j and BS_i. Here, *numcells* denotes the number of cells required to be relocated, as determined by Algorithm 5, which equals the number of cells in *relocation_cell_list*. n_i, n_j represent the source node and its preferred parent, respectively. BS_i is the list of already occupied slots by node n_i. Initially, the algorithm excludes `SlotOffset` zero from consideration as it is reserved for the minimal cell [42]. Next, the parent node n_j identifies its already scheduled slots (BS_j). Subsequently, all slots present in the lists BS_i and BS_j are removed from the available slot list to avoid schedule conflict. After completing this process, if no available slots are found, node n_j sends a response message indicating the same. However, suppose the number of available slots exceeds the number of cells to be relocated. In that case, two scenarios are considered: (i) If n_j is a DODAG root node, it proceeds to identify the reception slots of node n_i (lines 21 and 22 of the algorithm). Then, n_j determines the minimum slot value used by the requesting child node n_i for packet reception and also identifies the available slots following that minimum reception slot. Finally, n_j selects the *numcells* number of such minimum slots from the available slots for relocation using the *min_slots* function. This ensures that the collided cell between nodes n_i and n_j is relocated to a slot closer to and after the receiving

Algorithm 6: Latency-aware Candidate SlotOffset Selection for Efficient Cell Relocation

Input: $numcells$: number of cells required to be relocated, n_i : requesting node, n_j : preferred parent of n_i, BS_i : busy slots of n_i

Output: $selected\_slots$

$a_{slots} \leftarrow$ all slots in the negotiated slotframe
a_{slots}.**remove** ($slotlOffset$ 0)
$BS_j \leftarrow$ **find** already used slots of parent n_j
$a_{slots} \leftarrow a_{slots}$.**remove**$\{BS_i, BS_j\}$
if a_{slots} is **empty then**
    return no available slots to make relocation
else if number of slots in $a_{slots} > numcells$ **then**
    if n_j is **not** a DODAG root node **then**
        $tx\_slots\_n_j \leftarrow$ **find** transmission cells of n_j
        $min\_tx\_parent \leftarrow min\_slots(tx\_slots\_n_j, 1)$
        $min\_slot\_a_{slots} \leftarrow min\_slots(a_{slots}, 1)$
        if $min\_slot\_a_{slots} < min\_tx\_parent$ **then** `// (There are free slots before parent slot)`
            $temp\_a_{slots} \leftarrow \{s \in a_{slots} \mid s < min\_tx\_parent\}$
            if $|temp\_a_{slots}| > numcells$ **then**
                $selected\_slots \leftarrow max\_slots(temp\_a_{slots}, numcells)$ `// (closer to parent tx slot)`
            else
                $selected\_slots \leftarrow$ **random** $(a_{slots}, numcells)$
        else
            $selected\_slots \leftarrow min\_slots(a_{slots}, numcells)$
    else `//` n_j `is root node`
        $Rx_{ji} \leftarrow$ **find** Rx slots of n_j with n_i
        $Rx\_slots\_n_i \leftarrow BS_i$ - Rx_{ji}
        $min\_rx\_n_i \leftarrow min\_slots(Rx\_slots\_n_i, 1)$
        $temp\_a_{slots} \leftarrow \{s \in a_{slots} \mid s > min\_rx\_n_i\}$
        if $|temp\_a_{slots}| > numcells$ **then**
            $selected\_slots \leftarrow min\_slots(temp\_a_{slots}, numcells)$ `// (after and closest to sender)`
        else
            $selected\_slots \leftarrow$ **random** $(a_{slots}, numcells)$
else
    $selected\_slots \leftarrow a_{slots}$
return $selected\_slots$

slot of the sender n_i whenever possible. This enables node n_i to forward data packets as soon as they arrive, thereby minimizing latency and enhancing network performance. (*ii*) When n_j is not a DODAG root node, the receiving node n_j finds the slots it has used to forward its data packets to its preferred parent node. From these slots, the smallest slot value used for packet transmission is found out. It then checks if the smallest available slot value is less than the smallest transmission slot value of node n_j. If so, it indicates that there are free slots before the parent transmission slot and thus proceeds to find the available slots preceding the smallest transmission slot of n_j. Subsequently, the algorithm verifies if the number of updated free slots exceeds the number of cells to be relocated. If the condition holds true, n_j proceeds to select the largest slots from the chosen free slots for relocation using the *max_slots* function. This ensures that the newly scheduled cell after relocation is positioned before the transmission cell of the parent node and as close as possible. As a result, the buffering time in the queue of node n_i is minimized. On the other hand, if the algorithm does not find any free slot before the transmission slot of the parent node, it proceeds to select the smallest slots from the available slots for relocation. In the worst-case scenario, where there are insufficient available slots to fulfil the relocation request, the algorithm relocates the collided cells to whatever free slots are available.

Note that functions $min\_slots(tx\_slots\_n_j, 1)$ and $max\_slots(a_{slots}, numcells)$ are used to determine the minimum slot value from $tx\_slots\_n_j$ and to select the largest *numcells* slot values from a_{slots}, respectively. For the slots selected by Algorithm 6, node n_j randomly chooses one *ChannelOffset* from the available N_c channels to compile the final selected cell list (*SelectedCellList*) for relocation.

The time complexity of Algorithm 6 is primarily determined by operations on the set of slots. Initializing the set a_{slots} takes $O(S)$ time, where S denotes the total number of slots in the slotframe. Operations like removing elements from the set have an average time complexity of $O(1)$ with hash-based sets, but worst-case scenarios may reach $O(S)$. Finding used slots of a parent, retrieving transmission cells, and performing set difference operations also have a worst-case complexity of $O(S)$. The dominant operations involve

finding minimum/maximum slotOffsets and searching for slots based on specific conditions, which can take up to $O(S \cdot numcells)$ time. Where $numcells$ is the number of cells to be relocated and its value is 5. Since $numcells$ is a small constant, the overall complexity of Algorithm 6 is $O(S)$. Given that the maximum value of S is less than equal to 101 (slotframe length), the computational complexity remains low. Moreover, every node executes the algorithm independently, and the complexity remains constant regardless of the size of the network.

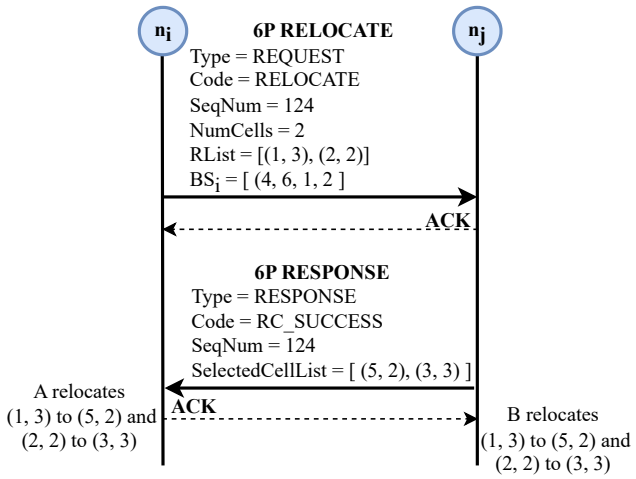


Figure 4.2: *Modified 6P RELOCATION transaction*

4.2.3.2 Modified Steps for 6P Relocation Request

As we have introduced a receiver-based cell relocation mechanism, the parent node n_j takes responsibility for all computations and determines new cells to perform relocation. As a result, node n_j requires information about the slots already scheduled by node n_i to prevent any potential scheduling conflicts. This necessitates updating the message fields in 6P control messages used in the cell relocation process. In the modified 6P relocation protocol, the requesting node n_i informs node n_j about its already scheduled slots as BS_i instead of sending candidate cells as a *CCellList*. Figure 4.2 illustrates this modified 6P relocation process, where node n_i sends a 6P Relocation request to its parent node n_j, specifying the number of collided cells ($numcells$) along with the list of these cells in *RList*.

Upon receiving the RELOCATION request, node n_j uses Algorithm 6 to determine new slots for the collided cells. It then sends a 6P Response to node n_i, conveying the list of chosen cells in *SelectedCellList* for relocation. Both nodes n_i and n_j update their schedules accordingly, adding the newly selected cells and removing the collided one listed in *RList*. Here, *RList* corresponds to *relocation_cell_list* in Algorithm 5. It is worth noting that the modified 6P relocation procedure facilitates receiver-based cell relocation through a 2-step 6P transaction without incurring any additional control overhead. This modification significantly reduces the 6P control overhead associated with relocation transactions by minimizing the frequency of repeated 6P Relocation requests. Moreover, it eliminates the need for preparing a candidate cell list at node n_i, overcoming the constraint of relocation only occurring when the number of available slots exceeds a predefined cell list length. Additionally, it resolves the issue of node n_j being restricted to selecting cells solely from the candidate list provided by node n_i, which sometimes prevented relocation despite available free slots at n_j. By coordinating with the sender node n_i and determining slot availability based on actual conditions rather than relying on a predefined list generated randomly, this approach optimizes the relocation process and improves overall network performance.

4.3 Performance Evaluation

We evaluate the performance of our proposed schemes under realistic conditions where packets may be dropped due to low link quality and collisions. The assessment begins with a description of the simulation settings, followed by an overview of the benchmark schemes used for comparison and the performance metrics used for evaluation. Finally, we present and analyze the simulation results.

4.3.1 Simulation Settings

We assess the operational performance of our proposed scheme using the Python-based 6TiSCH simulator v.1.2.0 [100]. The simulation parameters are chosen based on real-world

Table 4.2: *Simulation parameters*

Parameter	Value
Slotframe Length	101 timeslots
Timeslot duration	10 ms
Number of channels	16
Deployment strategy	Random
RPL Objective Function	MRHOF - ETX
RPL parents	1
Tx queue size	10 packets
TSCH maximum payload size	102 Bytes
Number of nodes	10-80
MAC max retransmission	5
Application traffic pattern	2, 4 (packets/slotframe)
α	0.2
$RELOCATE\_THRES$	0.5
$Battery_{cappacity}$	2821.5 mAH
Each iteration cycle	3600 slotframes
Number of iterations	30 per scheme
Traffic generated by	All non-root nodes

IoT and industrial wireless networking scenarios while adhering to the recommended values from IETF 6TiSCH, IEEE 802.15.4e TSCH, and RFC 9033 standards. To enable multi-hop communication, our experiments use the RPL MRHOF objective function [72], which is widely used in low-power and lossy networks (LLNs) due to its reliability in dynamic environments. Node 0 is designated as the DODAG root and functions as the application's sink node. After completing the join process, all non-root nodes join the RPL routing structure using the standard CoJP [41] scheme. Adhering to the RPL specification, each node selects a preferred routing parent, with a maximum of 3 RPL parents allowed. To accurately model wireless channel behaviour, we employ the Pister-Hack radio propagation model [99], which effectively simulates fading effects in low-power wireless networks. The random node deployment within a 2 km $\times$ 2 km area reflects typical smart city, industrial automation, and agricultural IoT deployments, where nodes are often scattered over a large geographical area. The proposed schemes are integrated into the scheduling function at layer 2 (TSCH layer). When the ECD scheme identifies a cell requiring relocation, it triggers the 6P protocol above TSCH and initiates a 6P relocation request to the preferred parent. The cell is then relocated to a new location, as per the ECDR scheme, following a successful relocation

transaction. Specifically, during each 6top housekeeping period, Algorithm 5 is executed to identify cells experiencing schedule collisions. Upon detecting a collision, Algorithm 6 is invoked to resolve the collision. The simulations are conducted 30 times for both the proposed and benchmark schemes, and the results are reported as average values at a 95% confidence level, ensuring statistically reliable results. Traffic conditions are tested under two different rates (2 packets/slotframe and 4 packets/slotframe) to analyze performance under moderate and high network loads, simulating IoT applications with periodic and event-driven traffic. Furthermore, the slotframe length, the number of channels, and MAC retransmission limit are set based on IEEE 802.15.4e TSCH standard to maintain compatibility with real-world IoT networks. The threshold $RELOCATE\_THRES$ is taken from the standard MSF (RFC 9033) and $\alpha = 0.2$ was initially tested through simulations with varying values, and it was found that this value provided better performance . The battery capacity (2821.5 mAh) is chosen to reflect low-power IoT devices operating in constrained environments. Table 4.2 provides a summary of the simulation parameters.

4.3.2 Benchmark Schemes and Performance Metrics

- **MSF** [43] is the comprehensive scheduling algorithm devolved by the 6TiSCH Working Group (WG) and is the standardized scheduling function for 6TiSCH networks. It defines node joining, schedule design, and management mechanisms. An implementation of MSF is available in the simulator.

- On the other hand, **CCR** [56] is a relocation scheme that emphasizes cost-effectiveness. We have implemented CCR on top of MSF to evaluate its performance.

To assess the performance of the proposed and benchmark schemes, we use the following metrics:

- *End-to-end latency* - the total time from when a packet is generated at the source node's application layer until it reaches the destination node's application layer.

- *PDR* - the ratio of the number of application packets successfully received by the sink node to the total number of packets transmitted by all other nodes in the network. It is calculated at the sink node to evaluate the effectiveness of packet delivery across the network.

- *6P relocation overhead* - the total number of 6P transactions executed in the network for relocating collided cells.

- *Energy consumption* - we evaluated it using the model proposed by Vilajosana *et al.* [98], which leverages real measurements to estimate the total energy consumed accurately.

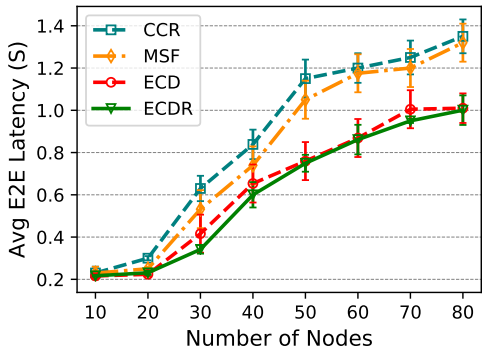
- *Collision rate* - represents the proportion of collisions during data transmission relative to the total number of packets transmitted within a network.

- *Network lifetime* - duration until the first node depletes its energy and becomes inoperative. The lifetime of an individual node n_i, denoted as LT_{n_i} is calculated as $\frac{Battery_{capacity} \times S_{duration}}{Energy_{slotframe} \times 365 \times 24 \times 3600}$. Where $S_{duration}$ represents the duration of the slotframe in seconds, $Battery_{capacity}$ denotes the battery capacity of a node in μC, and $Energy_{slotframe}$ indicates the average energy consumption in μC during a slotframe. To compute $Energy_{slotframe}$, the simulator [100] relies on the energy consumption model defined by [98]. $(365 \times 24 \times 3600)$ converts the seconds into years. Finally, we have to identify the node with the minimum individual lifetime. Thus, the network lifetime can be expressed as:

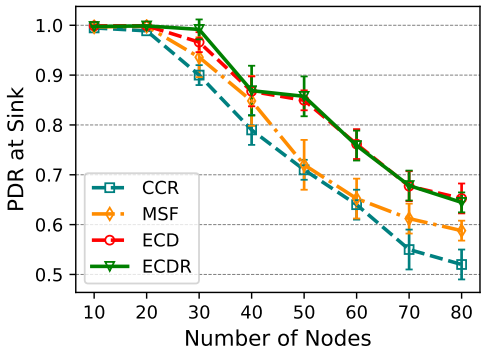
$$NL = \min\{LT_{n_i} \mid i \in [1, N-1]\} \tag{4.3}$$

Table 4.3: *Charge consumed by each state of the node*

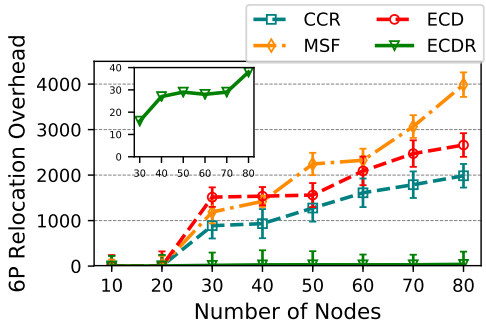
Nodes State	Charge (μC)	Nodes State	Charge (μC)
Sleep	0.0	Idle_listen	6.4
tx_data	22.6	rx_data	32.6
tx_data_rx_ack	49.5	rx_data_tx_ack	54.5



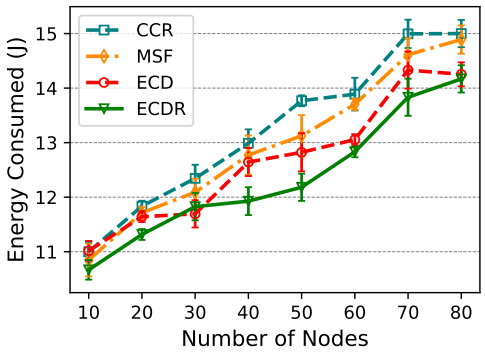
(a) *End-to-end latency.*



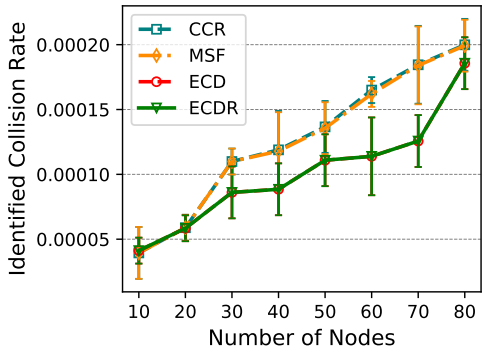
(b) *PDR at sink.*



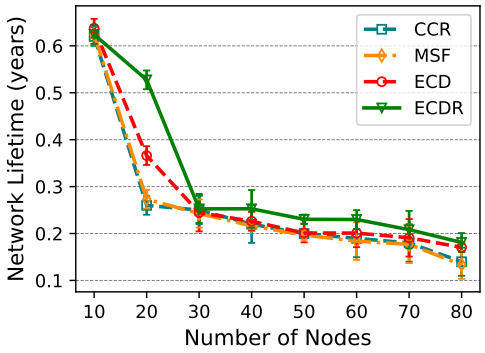
(c) *6P relocation overhead.*



(d) *Energy consumed.*



(e) *Collision rate.*



(f) *Network lifetime.*

Figure 4.3: *Comparison of results under varying network sizes with traffic rate 2P/SF*

4.3.3 Experimental Results

This section evaluates the effectiveness of the proposed collision detection and mitigation schemes by analyzing their performance across different network sizes. We employ a periodic traffic model where each node generates traffic at a rate of 2 packets per slotframe (P/SF). The evaluation assesses the impact of the proposed schemes on the key performance metrics outlined in Section 4.3.2. Note that all other scheduling parameters remain consistent

between the proposed and benchmark schemes to isolate and analyze the effectiveness of the collision detection and mitigation methods. We present results for two versions of our proposed solution: Enhanced Collision Detection (ECD), which focuses solely on collision detection and ECD with cell Relocation (ECDR), which integrates collision detection with the proposed relocation mechanism.

We analyze the average end-to-end delay, as shown in Figure 4.3a. The results demonstrate that the proposed collision detection and mitigation schemes together effectively reduce end-to-end latency compared to the benchmark schemes. This improvement is attributed to two key factors. First, the proposed collision detection mechanism identifies collisions more effectively, reducing the frequency of retransmissions of the packet being transmitted on the collided cell, resulting in quicker packet delivery to the destination. Second, the receiver-based relocation mechanism reduces the likelihood of relocation failures, freeing the collided cells while a new cell is being allocated to transmit the packet. Additionally, prioritizing cells closer to the forwarding node's transmission cell during relocation minimizes queuing delays, facilitating quicker packet transmission. Our approach outperforms both CCR and MSF. In particular, ECDR reduces end-to-end latency by 24% and 26% compared to MSF and CCR, respectively, in an 80-node network.

Figure 4.3b illustrates the PDR at the sink node over time for the network under consideration. Collisions significantly degrade PDR, as the nodes that suffered schedule collisions keep retransmitting at the next opportunity on the same collided cell. Collisions are repetitive until resolved, leading to a cycle of repeated collisions and packet drops. Therefore, accurate collision detection is crucial for maintaining high PDR. The results clearly demonstrate that the proposed ECD achieves higher PDR than both MSF and CCR. CCR performs worst, as it does not always relocate the collided cells. For instance, in a 60-node network, the proposed scheme increases the PDR by 19% and 17% compared to CCR and MSF, respectively. The reasons for the improved PDR can be attributed to the proposed collision detection scheme that analyzes the collective performance of all cells between two nodes over a specific timeframe before making decisions. ECD is more effective than MSF

and CCR, which may not account for all potential collision scenarios. Further, MSF and CCR lack mechanisms to handle collisions involving single or entire cell bundles, leading to packet drops. We have observed that the relocation of collided cells is often not possible in MSF due to relocation failures, and until the relocation is successful, packets get dropped after reaching maximum retransmission counts.

Figure 4.3c demonstrates a significant reduction in relocation overhead achieved by our ECDR scheme compared to MSF and CCR. This improvement stems from the difference in relocation strategies. Unlike MSF and CCR, our receiver-based relocation strategy in ECDR makes a substantial reduction in the number of repeated 6P relocation requests required to relocate collided cells. Table 4.4 quantifies this benefit by detailing the average number of repeated relocation requests for both schemes. The proposed approach significantly reduces the frequency of repeated relocation requests compared to the benchmark schemes and, thus, lesser relocation overhead. For example, in an 80-node network, the ECDR reduces relocation overhead by 99% compared to MSF and 98% compared to CCR. This translates to lower energy consumption for the proposed scheme. The increased 6P relocation transactions in MSF result in higher energy consumption, as depicted in Figure 4.3d. CCR and MSF consume up to 12% and 7% more energy than the proposed scheme in a 50-node network. We adopt the energy consumption model proposed by Vilajosana *et al.* [98] to estimate energy consumption. Each cell in the slotframe is associated with various node states i.e. sleep, idle listening, data transmission, and reception. During the simulation, we monitor the occurrences of Acknowledgment (ACK) and data transmissions. Using energy consumption values for each state (as specified in Table 4.3), we calculate the charge consumption of each node. To determine the average energy consumption across all nodes, we convert the charge consumed into energy by multiplying the total charge used by each device by its operating voltage (i.e. 3 Volts) across simulation iterations. Figure 4.3d presents the resulting average energy consumption for each scheme. This energy efficiency is particularly important in resource-constrained environments where frequent transmission and reception activities can quickly deplete node batteries [101].

Figure 4.3e shows the collision rate observed in the proposed and benchmark schemes under varying network sizes. As expected, the collision rate increases with network size. This is due to the larger number of scheduled cells in larger networks, which raises the likelihood of multiple nodes selecting the same cell in the schedule and resulting in a higher probability of schedule collisions. We have observed that in a 60-node network, on average, 83 instances of single negotiated transmission cells were scheduled between nodes. However, neither MSF nor CCR possesses a mechanism to detect collisions within these scheduled cells. On the other hand, our proposed ECD successfully identifies collisions in 3 out of 58 instances within this network size. Additionally, in a network size of 80, ECD detects collisions in a bundle comprising two cells. However, with MSF and CCR, there is no way to check such scenarios. It is further observed that MSF and CCR occasionally register false positives for collisions, and also, the relocation attempts fail many times, and the same cell gets repeatedly counted as collided. Figure 4.3f illustrates the network lifetime for each scheme, calculated using Equation 4.3. MSF generates a large number of 6P relocation requests, many of which are repeated, negatively impacting the network lifetime. In contrast, ECDR addresses this issue, resulting in an extended network lifetime compared to MSF and CCR.

Table 4.4: *Comparison of 6P relocation transactions between sender-based (e.g. MSF) and the proposed receiver-based relocation approach (ECDR)*

Number of Nodes	Repeated Relocation Requests		Relocation Failure - NCC & NMR		Relocation Failure - NASR	
	MSF	ECDR	MSF	ECDR	MSF	ECDR
10	0	0	0	0	0	0
20	1	0	0	0	1	0
30	1413	6	0	0	1416	0
40	1176	3	37	0	1178	0
50	3977	10	0	0	3981	0
60	2230	9	104	0	2232	0
70	3047	5	35	0	2810	5
80	2306	6	7	0	1092	6

NCC: No candidate cells, NMR: No match of cells at receiver from provided candidate cell list, NASR: No available slots at receiver

Table 4.4 presents a comparative analysis of 6P relocation transactions in MSF and our proposed ECDR. The table first highlights the repeated relocation requests (columns 2 and 3), indicating the average number of repeated relocation attempts required by each scheme to relocate collided cells. ECDR reduces the average number of attempts to relocate

collided cells (as shown in column 3), owing to its receiver-based relocation mechanism that minimizes relocation failures while relocating the collided cell. Additionally, the table details the number of relocation requests that failed in MSF due to non-preparation of the candidate list by the sender, despite having free slots, as well as instances where relocation failed because the receiver did not have free slots matching those provided in the candidate cell list provided by the sender (as shown in column 4). ECDR effectively avoids these failures (as shown in column 5) by employing a receiver-based relocation strategy that fosters collaboration between sender and receiver during cell selection. This approach eliminates the need for the transmitting node to prepare a candidate cell list, thus overcoming the limitation that relocation can occur only when available slots exceed a predefined cell list length. Moreover, ECDR resolves situations where the receiving node is restricted to selecting cells solely from the candidate list provided by the sender, which can otherwise prevent relocation despite the availability of free slots at the receiver. Finally, we detail relocation failures due to the non-availability of cells at the receiver for both schemes (columns 6 and 7), providing further insights into the performance differences between MSF and our proposed approach.

4.4 Summary

Schedule collisions degrade network performance and reliability. Therefore, in this chapter, we proposed two novel schemes to address this challenge to enhance IIoT network performance. First, we introduced an *Enhanced Collision Detection* (ECD) approach utilizing the EWMA of the CDR. This method effectively detects all collision scenarios, providing a more accurate and dynamic assessment of cell collisions. Second, we proposed a receiver-based mitigation scheme called *ECD with cell Relocation* (ECDR), which enables the receiver to actively participate in the collision mitigation process and collaborate with the sender to make cell relocation decisions. By leveraging the knowledge of its available slots, the receiver makes better cell relocation decisions, significantly reducing 6P relocation overhead. Additionally, ECDR incorporates a delay-aware relocation approach to minimize end-to-end latency. The effectiveness of the proposed solutions is evaluated using the 6TiSCH simula-

tor. The results demonstrate significant performance improvements over existing collision handling schemes, MSF and CCR. For example, in a 70-node network, ECDR reduces end-to-end latency by 25% and 21%, compared to CCR and MSF, respectively. It also lowers 6P relocation overhead by 98% and 99%, respectively. Furthermore, ECDR improves PDR by 23% and 11% compared to CCR and MSF, respectively.

As discussed in previous chapters, the design and operation of link scheduling play a critical role in maintaining network performance, such as end-to-end latency, reliability, network capacity, and the battery lifetime of individual nodes. Thus, the proper functioning of link scheduling is crucial for ensuring overall network performance. However, despite the security measures provided by the Minimal Security Framework [41], there exist vulnerabilities within the link scheduling and 6P protocol that could compromise their desired behaviour and severely impact the network performance. Therefore, in Chapter 5, we delve deeper into these vulnerabilities associated with link scheduling and the 6top protocol in 6TiSCH networks, exploring their implications and potential solutions.

"The future depends on what we do in the present."

~Mahatma Gandhi (1869–1948)

5

Securing Link Scheduling Function

As discussed in previous chapters, IETF introduced MSF and 6P protocol to dynamically manage link scheduling in 6TiSCH networks. However, the security of the 6TiSCH architecture, particularly vulnerabilities within MSF and its associated 6P protocol, remains under-explored. This chapter identifies two key vulnerabilities: *cell depletion attack* and *schedule instability attack*. To counter these threats, we propose two mitigation schemes, *Secure Cell Scheduling Function based on Parent-Child Authentication (SCSF-PC)* and *Secure Cell Scheduling Function based on Weak Estimator Learning Automata (SCSF-WELA)*. The attacks and the proposed solutions are evaluated using the 6TiSCH network simulator.

5.1 Introduction

The way the link schedule is defined and managed has a significant impact on networking metrics such as end-to-end latency, reliability, network capacity, and the battery life of individual nodes. As a result, the correct operation of link scheduling is crucial for optimizing the network's overall performance. It is worth emphasizing that the 6P protocol and the Scheduling Function (SF) are essential components within the 6TiSCH stack, handling resource allocation and link schedule maintenance to support seamless network communication. Disruptions or attacks targeting these protocols can compromise the proper functioning of link scheduling and can impair the normal resource allocation procedures, potentially leading to schedule instability, reduced data delivery and increased energy consumption in affected nodes.

5.1.1 Motivation

However, vulnerabilities in 6P and link scheduling functions within the 6TiSCH network have not been explored in the literature except for the works in [57, 62]. Most existing studies on the security of 6TiSCH focus on aspects such as time synchronization, routing, and jamming attacks, leaving the vulnerabilities of the link scheduling process less explored. In particular, the recent study in [57] identified vulnerabilities in the 6P protocol, revealing that attackers can manipulate nodes into executing inconsistent 6P transactions, resulting in mismatched schedules between negotiating nodes without their awareness. This chapter demonstrates that the link scheduling and 6P protocol exhibit vulnerabilities that can compromise the correct behaviour of cell scheduling and severely impact network performance. Specifically, there is a scope for *cell depletion attack* and the *schedule instability attack* in the existing link scheduling function (both the attack models are discussed in detail in Section 5.2.1). *Cell depletion attack* leverages vulnerabilities in parent nodes' knowledge of their legitimate child nodes, manipulating the cell scheduling process by exploiting time slot allocations aiming to exhaust or disrupt the available resources. The attack involves malicious nodes deliberately requesting or occupying excessive time slots, causing resource exhaustion

and depriving legitimate nodes of their fair share of resources. The consequences of a successful cell depletion attack can be significant and may lead to operational disruptions or even complete denial-of-service to the legitimate nodes, causing network performance degradation. The *Schedule instability attack*, on the other hand, involves an attacker initiating irregular scheduling operations with its preferred parent. By mimicking legitimate 6P operations, the attacker induces frequent, incorrect cell allocations and deallocations, creating significant schedule instability and thereby disrupting the network's overall functionality. Being unaware of the attack, the parent node accepts these requests as they appear to be standard 6P operations between a child and a parent, causing resource imbalances and leading to communication failures, increased latency, and energy wastage. Specifically, two types of attacks are identified: the *cell depletion attack* (when the attacker is not a child node) and the *schedule instability attack* (when the attacker acts as a child node). Table 5.1 summarizes the existing works on the security of 6TiSCH architecture, highlighting vulnerabilities and the strategies proposed for mitigation.

Table 5.1: *Existing important works on security of 6TiSCH architecture*

Schemes	Focus of Attack	Defense Mechanism	Mitigation of Vulnerabilities in SF & 6P	Year
[102]	RPL DIS	No defense mechanism proposed	No	2019
[60]	RPL DIS	Anomaly-based IDS	No	2019
[91]	RPL DIS	Prevents trickle timer resets	No	2020
[92]	RPL DIS	DIS attack prevention	No	2021
[93]	RPL rank	Specification-based IDS	No	2020
[94]	RPL DIS	No defense mechanism proposed	No	2022
[95]	RPL DIS	trust-based mechanism	No	2023
[58]	TSCH Time Synchronisation	intrusion detection algorithms, encryption	No	2016
[61]	TSCH Time Synchronisation	Irregular Slot Hopping (DISH) solution	No	2018
[62]	6P	No defense mechanism proposed	No	2020
[57]	6P	local monitoring for Overloading attack	Partially	2022
This thesis	SF & 6P	SCSF-PC and SCSF-WELA	Yes	

5.1.2 Contribution

To address these vulnerabilities, this chapter proposes *SCSF-PC* and *SCSF-WELA* schemes to counter the cell depletion and schedule instability attacks, respectively. *SCSF-PC* miti-

gates the impact of cell depletion attack by incorporating parent-child authentication and validation of the nodes. The proposed mitigation scheme ensures that only legitimate child nodes of a parent node can participate in the scheduling process, thereby preventing malicious nodes from depleting the network resources. For the schedule instability attack, *SCSF-WELA* employs a comprehensive detection and mitigation strategy by closely monitoring cell scheduling activities for irregular patterns. This scheme combines rule-based techniques with stochastic learning to capture subtle attack behaviours, including stealthy ADD and DELETE operations with altered parameters. Given the non-stationary nature of the estimating parameter (RX_{util}), Weak Estimation Learning Automata (WELA) is used to accurately capture all actions of the attacker, ensuring robust detection and mitigation. Thus, by employing a combination of the rule-based method and stochastic learning method WELA, the proposed mechanism can accurately identify compromised nodes through their abnormal cell utilization patterns. Once a node is flagged as compromised, the SF blocks any further malicious 6P requests from that node, thereby preventing any further disruptive actions by the attacker. Finally, this chapter provides an extended performance evaluation to assess the impact of the attacks and the effectiveness of the proposed mitigation approaches using 6TiSCH simulator.

In summary, this study addresses the unexplored attack vector posed by the compromised child and non-child nodes and offers a comprehensive mitigation strategy against any malicious actions by the attacker, thereby enhancing the security and resilience of 6TiSCH scheduling. In brief, the contributions of this chapter are as follows:

- Demonstrated the existence of new vulnerabilities in MSF and their exploitation to consume scheduling resources as well as to make the link schedule unstable.
- Initially, a scheme called Secure Cell Scheduling Function, based on Parent-Child Authentication (SCSF-PC), has been proposed to detect and mitigate *cell depletion attack*.
- We extend the SCSF-PC scheme by proposing a Secure Cell Scheduling Function based

on Weak Estimator Learning Automata (SCSF-WELA) as a comprehensive mitigation scheme that addresses both *cell depletion attack* and *schedule instability attack*. The proposed mitigation schemes can be easily incorporated into the standard MSF.

- Finally, we implement and conduct extensive simulations to evaluate the impact of attacks and the effectiveness of our proposed mitigation schemes in reducing network disruptions, energy consumption, and schedule instability caused by the attacks.

The rest of the chapter is organized as follows. Section 5.2 presents the proposed attack model and countermeasure strategies. Section 5.4 presents the detailed results and analysis, highlighting the effectiveness of our proposed solutions. Finally, we summarized this chapter in Section 5.5.

Table 5.2: *Important notations*

Symbol	Meaning
RX_{util}	Utilization of `Rx` cells
$\eta_{\text{RxCellUsed}}$	The number of Rx cells that have been used in the cycle
$\eta_{\text{RxCellElapsed}}$	The number of Rx cells that have elapsed in the cycle
U_{th}	Upper threshold of stable zone (75%).
L_{th}	Lower threshold stable zone (25%).
$E_{request}$	Expected request type from the child node
p_{stable}	The probability estimate that the cell utilization is stable
$p_{unstable}$	The probability estimate that the cell utilization is unstable
θ_1	The threshold for significant probability difference
θ	The threshold for checking convergence
τ_{sus}	The threshold for suspicion score to flag a node as compromised
seq_{no}	Sequence number used in every 6P message
SI	Schedule instability

5.2 Proposed Work

This section is divided into four subsections. Subsection 5.2.1 details the threat model. Subsection 5.2.2 provides a detailed discussion on the cell depletion attack and its mitigation. Subsection 5.2.4 explains the execution of the schedule instability attack. Finally, Subsection 5.2.5.2 presents the proposed mechanism for identifying compromised nodes and strategies

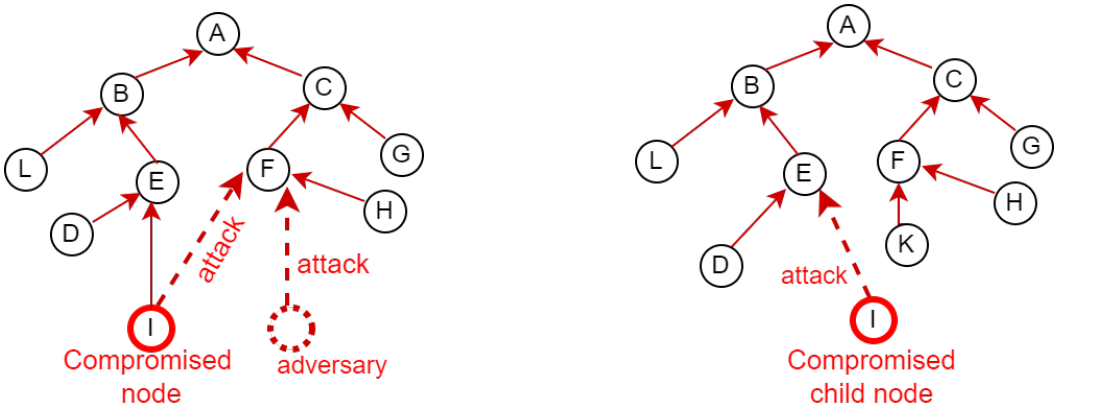
to prevent the attack. The frequently used symbols and their corresponding meanings are tabulated in 5.2.

5.2.1 Threat Model

The attacker's main objective is to disrupt the link scheduling process by manipulating time slot allocation to deplete available resources and destabilize the network. By deliberately requesting and occupying numerous additional time slots, the attacker exhausts available resources, overwhelming the system with excessive and manipulative demands. This tactic disrupts the network's normal operation, impairing the communication capabilities of legitimate nodes and maintaining a persistent state of resource utilization instability.

The remaining sections of this chapter consider a threat model where the attacker is an internal entity within the network with control over multiple nodes within the 6TiSCH network, which can launch attacks targeting specific victim nodes. This adversary can exploit these compromised nodes to launch attacks targeting multiple victim pairs simultaneously. The adversary, being an internal node, has access to the shared network key. However, it is important to note that the commonly shared network keys do not provide strict source authentication for data messages at the individual node level making it difficult to verify the origin of messages. While recipients can verify that the sender belongs to the same network, they cannot conclusively identify the sender solely based on the source address. This vulnerability exposes the network to risks such as message eavesdropping, forgery, injection, and identity impersonation [57]. These vulnerabilities are particularly concerning because security breaches, such as forgery and misattribution, become more damaging when a single key is shared among multiple participants [44]. A compromised node within the network can use this shared key to inject or modify valid 6P messages exchanged between other nodes or with its parent node, thereby manipulating the link schedule. Despite 6P negotiations being intended for specific node pairs, a compromised node within the network can passively monitor or actively tamper with these communications. This enables them to manipulate the 6TiSCH scheduling process and disrupt communication between nodes. Further, by

leveraging the default standard protocols, the adversary can execute malicious actions with the aim of disrupting the network's normal operations by targeting link scheduling to disrupt network communications. Multiple nodes can be exploited simultaneously to amplify the attack's impact. This enables the attacker to target several victim nodes concurrently, leading to increased energy consumption, packet drops, schedule instability, and potentially causing service disruption.



(a) *Cell depletion attack* **(b)** *Schedule instability attack*

Figure 5.1: *Example of an attack scenario*

Building on the above-discussed threat model, we have demonstrated the feasibility of two specific attacks: *cell depletion attack* and *schedule instability attack*. In the cell depletion attack, the adversary acts as a non-child node, intentionally requesting and occupying excessive time slots with its neighbouring nodes to exhaust the communication resources. In the schedule instability attack, the adversary operates as a child node, deliberately creating fluctuations in the scheduling process to destabilize the network's communication schedules. A detailed discussion of the attacks is given in Section 5.2.2 and 5.2.4, respectively. As mentioned in Section 2.1.3 in Chapter 2, each node selects a preferred routing parent to forward its data traffic towards the final destination as specified by RPL [39]. The node then negotiates with the chosen preferred parent using the 6P protocol to add or delete cells to manage the network dynamics based on the scheduling policy. However, the preferred parent is unaware of which nodes are its legitimate children, as it lacks knowledge of the nodes that have selected it as their preferred parent. Consequently, the preferred parent

cannot ascertain whether a 6P request is coming from its own child or not. This lack of information about child nodes at the parent level is a vulnerability exploited by the cell depletion attack. An attacker can issue fake or excessive 6P transactions to neighbouring nodes, aiming to allocate cells and exhaust or disrupt available resources. This results in starving for resources in the legitimate child nodes of the targeted parent node. In addition, by exploiting the behaviour of the cell scheduling process discussed in Section 2.7, the attacker can execute irregular operations with its preferred parent, resulting in significant schedule instability.

Figure 5.1 shows the two types of attack scenarios. Figure 5.1a illustrates a *cell depletion attack*, while Figure 5.1b showcases a *schedule instability attack* means attack by a compromised child node. In this example, node I is the compromised node, with node E as its preferred parent. However, node I attempts to establish a schedule with node F, which is the preferred parent of nodes H and K. In fact, any adversary can establish a schedule with node F through impersonation, as shown in Figure 5.1a. Since node F lacks information about its legitimate children, it unwittingly accepts the scheduling request from node I or an adversary. This allows the adversary to successfully establish a schedule with node F, thereby depleting F's resources. Additionally, in Figure 5.1b, node E, being unaware of node I's compromised status, accepts all additional 6P requests from node I, considering node I as its child. This exacerbates resource exhaustion or incorrect resource allocation, significantly disrupting normal network communications.

5.2.2 Cell Depletion Attack Execution

The primary objective of a cell depletion attack is to manipulate the allocation and utilization of time slots in the cell scheduling process to deplete the available resources. This attack involves malicious nodes intentionally requesting and occupying excessive time slots, resulting in resource exhaustion. Thus, by overwhelming the system with unnecessary or excessive demands for time slots, the attackers aim to disrupt the network's normal operation and hinder the communication capabilities of legitimate nodes. This attack exploits

parent nodes' lack of information about their legitimate child nodes. As illustrated in Figure 5.1a, node I or an adversary can execute this attack through a 6P ADD transaction with node F. Algorithm 7 outlines the execution of a cell depletion attack model by an attacking node. The function $RPLgetRank()$ determines the RPL rank of a node, while $RPLgetNeighbours()$ retrieves the list of its neighbouring nodes. When a node gets synchronized and has a parent node, it indicates that the node has successfully received EB frames and DIO messages from its radio neighbours. This makes the attacker aware of its neighbouring nodes and can select these neighbours as potential targets for the attack. Specifically, the attacker targets neighbouring nodes with a lower RPL rank than itself. For each selected target node, the attacker sends 6P Add Requests to add $CellListLength$ number of cells using the function $Send6pAddRequest(macAddr, CellListLength)$, aiming to deplete the resources of the targeted node as much as possible and thereby disrupt their normal operations. Specifically, the attack involves a compromised node issuing excessive or fake 6P ADD requests to the selected targets, leading to resource exhaustion at the parent and disrupted communication for legitimate child nodes.

Algorithm 7: Cell Depletion Attack Model

Input: $CellListLength$: maximum requesting cells
Output: Invokes a malicious 6P Add Request

```
1 parentNode ← find preferred parent of node
2 if nodeIsSynced() and parentNode exist then
3     neighboursList ← RPLgetNeighbours()
4     myrank ← RPLgetRank()
5     for all neighbour in neighboursList do
6         if neighbour.rank ≤ myrank and neighbour.macAddr ≠ parentNode.macAddr
          then
7             attackParentList.append (neighbour.macAddr)
8     for  all macAddr in attackParentList do
9         Send6pAddRequest(macAddr, CellListLength)
```

5.2.3 Cell Depletion Attack Mitigation

To counteract the execution of cell depletion attack outlined in Section 5.2.2, we propose an *SCSF-PC* scheme incorporating parent-child authentication and validation measures within a scheduling function. The vulnerability lies in a node's lack of information regarding its children, allowing the node to accept requests for negotiated cells coming from the attacker node. To address this vulnerability, our proposed solution involves storing the children's information at each parent node. By leveraging this stored information, the parent node can prevent allocating cells to malicious nodes once it receives a 6P ADD request from such a node.

Algorithm 8: Finding Children of a Node

Input: *packet*, *neighbourList*
Output: *6P_ChildrenList*

```
1 if packet.TYPE == packet.TYPE_DAO then
2     DAO_SrcIP ← packet.net.SrcIP
3     DAO_SrcMAC ← packet.mac.SrcMAC
4     for all node in neighbourList do
5         if node.macAddr == DAO_SrcMAC and node.IPAddr == DAO_SrcIP then
6             if DAO_SrcMAC not in 6P_ChildrenList then
7                 6P_ChildrenList.append (DAO_SrcMAC)
```

The challenge is how the parent node will come to know which neighbours are its children nodes. To achieve this, we propose Algorithm 8, which a parent node executes to identify its legitimate children within a network. Upon receiving DIO packets, a node decides its parent node and then communicates this information to the root node via a DAO message. The DAO packet is then relayed through the default routing parent node, which enables the parent node to record the child node's information before forwarding the packet to the root node. So, whenever a parent node receives a DAO packet, it verifies whether the MAC and IP addresses of its any neighbour node match the source MAC and IP addresses present in the received DAO packet. If both addresses match, it signifies that the packet originates from one of its child nodes. Then the parent node adds the MAC address of that node to the list of its children if the node is not already present in the list.

Algorithm 9: Defense Mechanism Incorporated in MSF for Handling Malicious 6P ADD Requests

Input: $6P\_ChildrenList$, 6P ADD Request
Output: Accept or Reject the 6P ADD Request

```
1 if RequestingNode.macAddr not in 6P_ChildrenList then
2     allocated_neg_cells ← Negotiated_Rx_Cells(macAddr)
3     if len(allocated_neg_cells) > NUM_INITIAL_NEGOTIATED_CELL or
      Request.NumCells > NUM_INITIAL_NEGOTIATED_CELL then
4         Block this 6P ADD Request
5         return
6     else
7         Accept the ADD request to allocate
          NUM_INITIAL_NEGOTIATED_CELL many cells
8 else
9     Continue the 6P cell addition procedure
```

To mitigate the cell depletion attack, the Algorithm 9 is proposed and can be incorporated into MSF as a mechanism for handling malicious 6P ADD requests. Note that the Algorithm 9 is executed by the scheduling function running at the parent node whenever it receives any 6P ADD request. Upon receiving a 6P ADD request, the parent node must check if a node in its children's list has initiated the request. If the request has come from a node not present in the children list, the parent node further examines whether any `Rx` cells have been allocated with that requesting node or not. For doing this verification, the parent node executes the function $Negotiated\_Rx\_Cells(macAddr)$ where $macAddr$ is the MAC address of the requesting node. If the `Rx` cells scheduled with the node exceeds the default negotiated `Tx` cell count (represented by $NUM\_INITIAL\_NEGOTIATED\_CELL$, and its default value is 1 in the standard) or if the number of requested cells to add is greater than $NUM\_INITIAL\_NEGOTIATED\_CELL$, the 6P request to ADD cells is rejected, and subsequently the requesting node is blocked. This condition, as specified in line 3 of the algorithm, serves two purposes. First, it ensures that only one cell is allocated in the initial 6P ADD request, which is required as minimal configuration. Secondly, it handles the scenario where a requesting node already has one scheduled cell. In this case, any subsequent request will be denied unless the requesting node is included in the children's list.

However, if there is no scheduled cell with the requesting node, the request is considered to come from a new child and so is accepted. To avoid any rejection of legitimate ADD requests, we allow the allocation of one scheduled cell, which in turn helps to prevent cell requests from being declined before knowing the child.

While the *SCSF-PC* scheme addressed the vulnerabilities posed by compromised non-child nodes, the potential threat from compromised child nodes remains unaddressed. The next section extends our analysis to include this threat and proposes an enhanced mechanism to detect and prevent malicious behaviour originating from both child and non-child nodes. Our goal is to provide a comprehensive defence strategy to secure the link scheduling function, ensuring the schedule stability and overall performance of the 6TiSCH network. The following subsection provides a detailed explanation of the schedule instability attack.

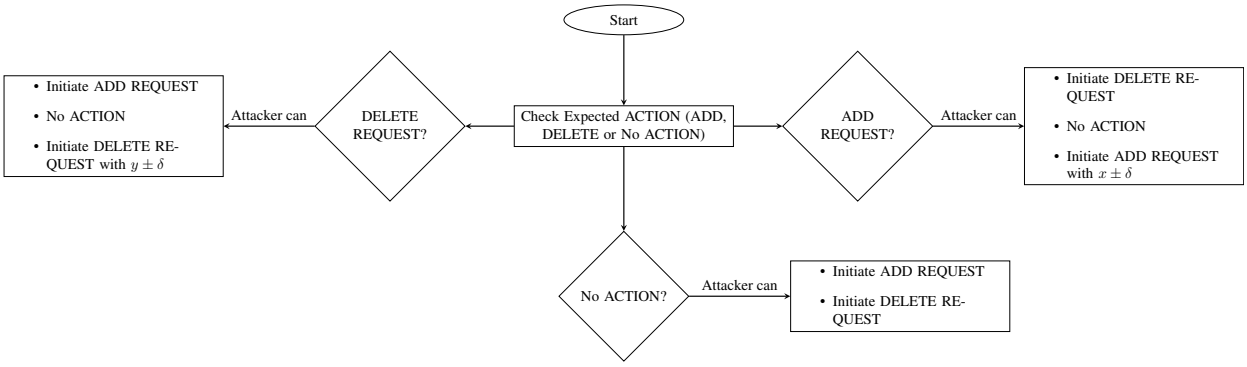


Figure 5.2: *An illustration of attack execution by the compromised child node*

5.2.4 Schedule Instability Attack Execution

This attack aims to perform abnormal yet legitimate 6P operations with its preferred parent to make the 6TiSCH schedule unstable. For example, this attack can be executed through a 6P ADD transaction or a 6P DELETE transaction by node I with node E, as shown in Figure 5.1b. As discussed in Section 3.2, cell utilisation (C_U) is a key metric in the scheduling protocol that dictates how child nodes interact with their parent nodes to manage present network traffic. When the C_U between a child and its parent exceeds 75%, the child node triggers a 6P ADD request to its parent to add more number of cells (say, x) to accommodate the increased traffic demand. Conversely, if the C_U falls below 25%, the child initiates a 6P

DELETE request to remove some cells (say, y) from their schedule. If the cell utilization remains between 25% and 75%, no action is required to adjust the cell allocation. Note that in MSF, the values of x and y are declared as unity. However, in IMSF 3.2, the values of x and y can be more than unity.

However, this mechanism becomes vulnerable when a child node gets compromised. A malicious child node can manipulate the scheduling process by sending requests based on falsified C_U values while still adhering to the protocol's legitimate operations. For example, if the C_U exceeds 75%, the compromised node can initiate a DELETE transaction or no request instead of initiating an ADD request. Alternatively, it can send an ADD request with an altered number of cells ($x \pm \delta$), where δ is a deviation from the actual required number of cells (x). Note that the child node computes cell utilization, and the decision to add or delete cells is based on this computation. This forms the crux of the attack: if the child node manipulates the cell utilization value, it can deceive the parent node into allocating or deallocating resources based on the manipulated requests, causing instability and inefficiency in the network. This vulnerability allows a malicious child node to manipulate the scheduling operations, disrupting normal network operations and resource management. The parent node being unaware of the legitimacy of its child's requests, may accept these requests since ADD and DELETE transactions are standard 6P operations between a child and a parent.

Figure 5.2 illustrates the possible actions that can be taken by the compromised child node and manipulates the link scheduling actions. The flowchart outlines the attacker's actions depending on the expected legitimate action (ADD, DELETE, or No ACTION). As per the adaptability rules of standard MSF, the following three cases can occur in which the attacker finds vulnerability to make/keep the schedule unstable.

- *Case 1:* If the expected action is an ADD request based on the cell utilisation value, the attacker can perform one of the following malicious actions:
  - (i) Initiate DELETE REQUEST: The attacker can send a DELETE request instead of an ADD request, which can lead to the removal of necessary cells, causing more instability in scheduled communication.

  - (*ii*) No ACTION: The attacker can choose to do nothing, which prevents the necessary addition of cells, leading to congestion and potential packet loss.
  - (*iii*) Initiate ADD REQUEST with $x \pm \delta$: The attacker can send an ADD request as expected. However, by stealthily manipulating the number of cells, either more or less than required, to overburden or underutilize the resources.

- *Case 2:* If the expected action is a DELETE request, the attacker can perform one of the following malicious actions:
  - (*i*) Initiate ADD REQUEST: The attacker can send an ADD request instead of a DELETE request, leading to unnecessary allocation of cells and thus resource exhaustion.
  - (*ii*) No ACTION: This prevents the necessary deletion of cells, potentially causing underutilization.
  - (*iii*) Initiate DELETE REQUEST with $y \pm \delta$: The attacker can send a DELETE request with a manipulated number of cells.

- *Case 3:* When the cell utilization is in the normal range, the attacker can still perform malicious activities:
  - (*i*) Initiate ADD REQUEST or
  - (*ii*)Initiate DELETE REQUEST to bring the schedule in unstable region.

In summary, by sending false ADD or DELETE requests or taking No ACTION when it was necessary, the attacker can create resource imbalances, causing communication failures, increased latency, and energy wastage. This exploitation of the cell scheduling mechanism underscores the network's vulnerability to such internal threats and, thus, the need for robust security measures to counteract these attacks.

5.2.5 Schedule Instability Attack Detection and Mitigation

This section introduces a mechanism to identify compromised nodes and discusses strategies to prevent the schedule instability attack. The ability of a preferred parent to promptly detect anomalies in cell scheduling activities allows for the execution of specific actions to mitigate such attacks. For the purposes of this discussion, it is assumed that the parent node in the network is not compromised. In other words, the detection mechanism is executed by the parent node, which is not compromised.

To effectively detect any compromised behaviour, monitoring must be conducted over several consecutive cycles, with observations carried forward from one cycle to the next to identify anomalies. Using both rule-based and stochastic learning methods, each parent node will assess whether any of its children are compromised based on their cell utilization values over time. The parent node calculates the cell utilization of its `Rx` cells scheduled with each child node. We utilize Algorithm 8 to identify the child nodes of a parent. Then, the proposed attack detection and mitigation mechanism is divided into three subsections. Subsection 5.2.5.1 describes how the parent node decides about the expected request from a child node. In Subsection 5.2.5.2, we explain how the WELA method is used to estimate the probabilities of being in stable and unstable regions of cell utilization. Finally, Subsection 5.2.5.3 discusses the proposed mechanism for identifying compromised nodes and handling malicious 6P transactions initiated by these nodes.

The detection and mitigation mechanisms are designed to integrate seamlessly with existing protocols in 6TiSCH networks, and the proposed security measures do not cause additional signalling overhead.

5.2.5.1 Decision on Expected Request

This section describes how the parent node determines the type of expected request from its child nodes based on cell utilization of `Rx` cells between them. At each evaluation cycle of the Rx cell utilization window, the parent node calculates the utilization of `Rx` cells scheduled with each child using Equation 5.1. Here, $\eta_{\text{RxCellElapsed}}$ is the total number of

`Rx` cells scheduled between them from the last evaluation window, while $\eta_{\text{RxCellUsed}}$ is the number of scheduled `Rx` cells that have been used for reception of packets in the cycle. Based on this calculated utilization, the parent node uses Algorithm 10 to determine the expected request it should receive from its child node within the next window. In brief, the Algorithm 10 helps the parent node to predict whether the child node should request to add, delete, or have no action on the currently scheduled cells. If the calculated RX_{util} exceeds U_{th}, the expected request is an ADD request. Conversely, if the RX_{util} is below L_{th}, the expected request is a DELETE request. When RX_{util} falls between L_{th} and U_{th}, the parent node anticipates No ACTION from the child. Here, the values of L_{th} and U_{th} are set at 0.75 and 0.25, respectively, as per the standard MSF. If an advisory violates the mentioned scheduling rules, it can be easily detected using the rule-based scheduling approach as mentioned in Algorithm 10. However, if the adversary subtly modifies the values of x and y without violating the scheduling rules, detection becomes challenging with a rule-based approach alone. The estimation of x and y are already discussed in Chapter 3.

$$RX_{\text{util}} = \frac{\eta_{\text{RxCellUsed}}}{\eta_{\text{RxCellElapsed}}} \tag{5.1}$$

Algorithm 10: Determine Expected Request for i^{th} Cycle

Input: $RX_{util}(i-1)$
Output: $E_{request}$

```
if RX_util(i − 1) > 0.75 then
    E_request ← "ADD REQUEST"
else if RX_util(i − 1) < 0.25 then
    E_request ← "DELETE REQUEST"
else
    E_request ← "No ACTION"          // (Expected Request by parent node)
```

5.2.5.2 Stochastic Learning-Based Weak Estimator

As discussed in Section 5.2.4 about the attack patterns, a rule-based detection mechanism alone cannot capture all malicious operations by the attacker due to the varied attack patterns and behaviours. Specifically, the scenarios when the attacker issues the ADD

and DELETE operations as expected by its parent node but with a modified value of x and y. To effectively detect such behaviours, a stochastic learning approach is needed to estimate the probabilities of the schedule being in stable and unstable states in conjunction with rule-based methods. Given that our estimating parameter (RX_{util}) follows a non-stationary distribution, we require a non-stationary estimation scheme to accurately capture all actions of the attacker. Thus, we employ the Weak Estimation Learning Automata (WELA), an estimation technique rooted in stochastic learning principles [103]. WELA incorporates randomness in learning from data and estimates the parameters of a probability distribution, such as the probabilities of each outcome in a binomial distribution. The binomial distribution has two parameters: the number of Bernoulli trials and the parameter of each trial. This chapter assumes that the number of observations equals the number of trials. Thus, the aim is to estimate the Bernoulli parameter for each trial using stochastic learning methods. The estimate is updated at each time instant based on the value of the current observation.

To model this problem, we consider RX_{util}, the utilization of `Rx` cells as a binomially distributed random variable that can be either "stable" or "unstable" as defined in Equation 5.2.

$$\begin{aligned} RX_{\text{util}} &= \text{stable} && \text{if } 0.25 < RX_{\text{util}} < 0.75 \\ &= \text{unstable} && \text{if } RX_{\text{util}} \leq 0.25 \text{ or } RX_{\text{util}} \geq 0.75 \end{aligned} \tag{5.2}$$

As per the standard MSF, RX_{util} is defined as stable when it lies between 0.25 and 0.75, and unstable otherwise. This chapter interchangeably uses the following index: "stable" as "1" and "unstable" as "2". Assume RX_{util} follows a distribution $S = [s1, s2]^T$, where $s1$ and $s2$ are the true probabilities of the stable and unstable outcomes, respectively. So,

$$\begin{aligned} RX_{\text{util}} &= \text{“stable”} && \text{with probability } s1 \\ &= \text{“unstable”} && \text{with probability } s2 \end{aligned}$$

Here, $s1+s2 = 1$, and $s1$, $s2$ representing the true underlying probabilities. Let $RX_{\text{util}}(i)$

be the value of RX_{util} at i^{th} evaluation cycle. The objective is to estimate S, i.e., $s1$ and $s2$. We maintain a running estimate of S, updated at each cycle i. Let $P(i) = [p_{stable}(i), p_{unstable}(i)]^T$ be the running estimate of S, where $p_{stable}(i)$ is the estimate of $s1$ at the evaluation cycle i.

The estimates $p_{stable}(i)$ and $p_{unstable}(i)$ are updated based on the observed outcome $RX_{\text{util}}(i)$ as follows:

$$p_{stable}(i+1) \leftarrow \lambda \cdot p_{stable}(i) \qquad \text{if } RX_{\text{util}}(i) = \text{“2”}, \tag{5.3}$$

$$\leftarrow 1 - \lambda \cdot p_{unstable}(i) \qquad \text{if } RX_{\text{util}}(i) = \text{“1”}, \tag{5.4}$$

and then,

$$p_{unstable}(i+1) \leftarrow 1 - p_{stable}(i+1) \tag{5.5}$$

where λ is a user-defined tuning parameter with its value in the range $0 < \lambda < 1$, controlling the influence of new observations on the estimates. p_{stable} and $p_{unstable}$ are the estimates of the probabilities for the stable and unstable outcomes in the binomial distribution. Over time, $p_{stable}(i)$ and $p_{unstable}(i)$ should converge to the true probabilities $s1$ and $s2$. Convergence is achieved when changes in p_{stable} and $p_{unstable}$ become negligible with new observations, indicating that the estimates are stable and closely represent the true probabilities. We check for convergence when changes in probability fall below a certain threshold, as shown in Equation 5.6.

$$\begin{aligned} \left|p_{\text{stable}}(i) - p_{\text{stable}}(i-1)\right| < \theta \quad \text{and} \\ \left|p_{\text{unstable}}(i) - p_{\text{unstable}}(i-1)\right| < \theta \end{aligned} \tag{5.6}$$

WELA-based estimation, combined with a rule-based mechanism, captures the behaviour of compromised nodes in all scenarios, as the attacker may execute legitimate 6P operations as discussed in Section 5.2.4. The next section discusses the proposed method for detecting compromised nodes using the method discussed above and rule-based decisions.

Algorithm 11: Find Compromised Node when Expected Request is ADD REQUEST

Input: $E_{request}$, p_{stable}, $p_{unstable}$, θ_1, τ_{sus}
Output: *CompromisedList*

```
if E_request == "ADD REQUEST" then
    update probabilities for p_stable and p_unstable using Equations 5.3, 5.4, 5.5
    if received request is ADD then
        check for convergence using the conditions defined in Equation 5.6
        if Converged then
            if p_unstable - p_stable > θ1 then
                add the node to the CompromisedList
    else if received request is DELETE then
        update suspicion score by unity
        if suspicion score > τ_sus then
            flag the node as compromised and add it to CompromisedList
    else if no request received then
        update suspicion score by unity
        if suspicion score > τ_sus then
            flag the node as compromised and add it to CompromisedList
Return CompromisedList                    // (List of compromised nodes)
```

5.2.5.3 Compromised Node Detection and Defense Mechanism

This section presents the proposed solution to identify the compromised child node and discusses how the parent node handles malicious 6P transactions once it receives a request from such nodes. The defense mechanism combines rule-based and stochastic learning-based decision-making to effectively detect anomalies in cell scheduling activities.

To capture the behaviour of compromised nodes, each parent node needs to monitor the behaviour of each child node over several consecutive cycles. By tracking the time duration a child node spends in the unstable region, the parent node can detect irregularities.

Based on the expected request ($E_{request}$) calculated by the parent node using Algorithm 10, the parent node executes one of the Algorithms 11, 12, or 13 as a defense mechanism to identify compromised nodes. The input parameters for these algorithms include $E_{request}$, p_{stable}, $p_{unstable}$, θ_1, and τ_{sus}, where $E_{request}$ is the expected request by the parent node from a child in the next cell utilization window. Fresh requests are tracked using the sequence number (seq_{no}) present in 6P frames. The seq_{no} is used to prevent replay attacks. p_{stable}

and $p_{unstable}$ are the probabilities of being in stable and unstable regions, respectively. θ_1 and τ_{sus} are the predefined decision and suspicion threshold values used by the parent node to make decisions. The output of these algorithms is the *CompromisedList*, which contains the identified compromised nodes.

Algorithm 12: Find Compromised Node when Expected Request is DELETE REQUEST

Input: $E_{request}$, p_{stable}, $p_{unstable}$, θ_1, τ_{sus}
Output: *CompromisedList*

```
if E_request == "DELETE REQUEST" then
    update probabilities for p_stable and p_unstable using Equations 5.3, 5.4, 5.5
    if received request is DELETE then
        or convergence
        if Converged then
            if p_unstable - p_stable > θ1 then
                add the node to the CompromisedList
    else if received request is ADD then
        update suspicion score by unity
        if suspicion score > τ_sus then
            flag the node as compromised and add it to CompromisedList
    else if no request received then
        update suspicion score by unity
        if suspicion score > τ_sus then
            flag the node as compromised and add it to CompromisedList
return CompromisedList                    // (List of compromised nodes)
```

To address the attack patterns of *Case (1)* discussed in Section 5.2.4, Algorithm 11 is proposed. This algorithm is executed by the parent node when the $E_{request}$ is an ADD request. In this case, if the child node initiates a DELETE REQUEST or does not initiate any request (No ACTION), the parent node updates the suspicion score for that child by unity. Note that the suspicion score is a counter value that increments by one whenever there is a discrepancy between the expected request and the actual request from the child node. If this behaviour persists for more than τ_{sus} monitoring periods, the node is added to the list of compromised nodes. This waiting period is considered to avoid including any legitimate action during the initial scheduling phases. Specifically, we have configured the

parent node to wait for τ_{sus} monitoring periods before adding any node to the compromised list to avoid adding any legitimate nodes by mistake. However, if the child node initiates an ADD REQUEST as expected but with a modified value of x, only a rule-based mechanism can not detect it. Thus, to capture any stealthy behaviour of the attacker, the algorithm checks whether the stable/unstable probabilities have converged. If so, it evaluates whether $p_{unstable} - p_{stable} > \theta_1$. If this condition is met, the node is added to the list of compromised nodes. This scenario cannot be handled solely by a rule-based mechanism, so we utilize WELA to capture this behaviour. The main philosophy behind this design is that if a child node spends more time in the unstable region of the cell utilization, it is considered compromised. Specifically, if a node consistently shows a higher probability of being in the unstable region over a series of observations, this will be reflected by a higher value of p_{unstable}.

To address the attack patterns of *Case (2)* as discussed in Section 5.2.4, Algorithm 12 is proposed. When the $E_{request}$ is a DELETE request, but the child node initiates an ADD REQUEST or does not initiate any request (No ACTION), the parent node updates the suspicion score for that child. If this behaviour persists for more than τ_{sus} monitoring periods, the node is added to the list of compromised nodes. However, if the child node stealthily initiates a DELETE REQUEST as expected but with a modified value of y, the algorithm checks whether the stable/unstable probabilities have converged. If so, it evaluates whether $p_{unstable} - p_{stable} > \theta_1$. If this condition is met, the node is added to the list of compromised nodes.

Algorithm 13: Find Compromised Node when Expected Request is No ACTION

Input: $E_{request}$, p_{stable}, $p_{unstable}$, θ_1, τ_{sus}
Output: *CompromisedList*

```
if E_request == "No ACTION" then
    update probabilities for p_stable and p_unstable using Equations 5.3, 5.4, 5.5
    if request received is ADD or request received is DELETE then
        update suspicion score by unity
        if suspicion score > τ_sus then
            flag the node as compromised and add it to CompromisedList
return CompromisedList                          // (List of compromised nodes)
```

To handle the attack patterns of *Case (3)*, if the parent node receives any action from the child node in terms of ADD or DELETE REQUEST when the RX_{util} is in stable range, the parent node updates the suspicion score for that child node. The child node is added to the compromised list if this behaviour persists for more than τ_{sus} monitoring periods.

Algorithm 14: Defence Mechanism for Handling Malicious 6P Requests (ADD/DELETE)

Input: 6P ADD/DELETE Request, *CompromisedList*
Output: Accept or Reject the 6P Request

```
1 if RequestingNode in CompromisedList then
2     Block this 6P Request
3 else
4     Accept the ADD/DELETE requests and allocate/delete required cells,
      respectively
```

Finally, to mitigate the schedule instability attack caused by a compromised child node, Algorithm 14 is proposed. This algorithm can be incorporated into the standardized MSF or any other SF as a mechanism for handling malicious 6P requests. Note that the SF runs the algorithm at the parent node whenever it receives any 6P request from its child node. Upon receiving a 6P request, the parent node first checks if the requesting node is listed in the *CompromisedList*. The parent node blocks the request if the requesting node is in the *CompromisedList*. Otherwise, the parent continues with the cell allocation or deallocation as per the request made by the child node. In the observation period, the parent accepts whatever request comes from that node until the node comes under the compromised list. Once a node is detected as compromised, any further requests from that node get discarded.

5.3 Discussion on Complexity and Scalability

The execution of Algorithms takes place independently on each node within the network. The worst-case time complexity of Algorithm 7 is $O(n)$, as it iterates over the list of neighbours. Here, n is the number of neighbours a node has. Every node independently runs the algorithms, thus the complexity remains the same irrespective of network size. Similarly,

Algorithm 8 has a complexity of $O(n)$. Algorithm 9 operates in constant time as there are only simple operations with if-else statements without any loop, i.e. the time complexity of Algorithm 3 is $O(1)$. We have used Weak Estimation Learning Automata (WELA), which is an estimation-based approach grounded in stochastic learning principles. Unlike traditional learning algorithms, WELA has minimal computational complexity. All algorithms (Algorithms 10-14) execute independently at each parent node, and each operation in the algorithm is executed in constant time as only simple operations with if-else statements without any loop. So, the complexity of these algorithms is $O(1)$. Since each new value depends only on the previous value and the current observation rather than the entire dataset, as a result scalability is not an issue.

Table 5.3: *Simulation parameters*

Parameter	Value/Type	Parameter	Value/Type
Slotframe Length	101 timeslots	RPL parents	1
Timeslot duration	10 ms	Application traffic pattern	2 packets/slotframe
Number of channels	16	Traffic generated by	All non-root nodes
Tx queue size	10 packets	Number of nodes	10-80
MAC max retransmission	5	θ	0.4
6TiSCH SF	MSF	λ	0.9
RPL Objective Function	MRHOF - ETX	θ_1	0.10
Deployment strategy	Random	τ_{sus}	3
Each iteration cycle	3600 slotframes	$Battery_{cappacity}$	2821.5 mAH
Number of iterations	20 per scheme	Attacker node	1

5.4 Experimental Evaluation

This section presents a comprehensive performance evaluation to assess the impact of the attacks on network performance and the effectiveness of the mitigation strategies. We begin by detailing the simulation setup, followed by a discussion of the performance metrics. Finally, we present and analyze the simulation results under various scenarios to gain insights into the attacks' repercussions and the benefits of the proposed mitigation approaches.

5.4.1 Experimental Setup

The execution of both the attacks and mitigation strategies are implemented using the 6TiSCH Simulator, which emulates a realistic environment for evaluating the proposed schemes in low-power, multi-hop wireless networks. We simulated different networks composed of 10 to 80 nodes, with a random node deployment in a region measuring 2 km × 2 km for all simulation runs. This deployment strategy models practical scenarios where IoT devices are distributed over a geographic region. Node 0 is designated as the Destination Oriented Directed Acyclic Graph (DODAG) root and serves as the sink node for the application. Non-root nodes join the network once they synchronize using the standard CoJP scheme [41] and integrate into the RPL routing structure. Each node can have up to three RPL parents, with one selected as the preferred routing parent according to RPL specifications. To enable multi-hop communication, the experiments use the RPL MRHOF objective function [72]. The links between nodes are established based on the Pister-Hack propagation model [99] that accurately simulates real-world wireless signal behaviour in low-power networks.

All experiments consider a convergecast data-gathering IoT application with periodic traffic, where non-root nodes report their data to the DODAG root node, behaving as an application sink node. We select a traffic pattern of 2 packets per slotframe (P/SF) to represent moderate IoT traffic loads commonly found in monitoring and sensing applications. TSCH is configured with a slotframe length of 101 slots and 16 channels as per the TSCH standard. We use 6TiSCH-MC [42] to provide minimal bandwidth dedicated to network advertisement and bootstrapping processes, facilitating essential communication and coordination among network nodes during setup and operation. Additionally, we use MSF as the baseline 6TiSCH scheduling function, which interacts with the 6P protocol for the allocation of negotiated cells. The execution of the attacks and mitigation strategies is integrated into the SF at layer 2 (TSCH layer) and the 6top sub-layer above TSCH. The implemented attack model involves a single attacker executing both the cell depletion attack and the schedule instability attack. The attacker performs the cell depletion attack at

random intervals during the network run and the schedule instability attack during each cell utilization evaluation cycle. The values for θ, λ, and θ_1 are determined through extensive simulations, where different configurations were tested, and these chosen values provided optimal defence performance with minimal overhead. Simulations are conducted 20 times with different random seeds for all considered scenarios to ensure statistical validity. Results are reported as average values with a 95% confidence level. Table 5.3 outlines the other relevant simulation settings and parameters with their corresponding values.
For analysis, three distinct scenarios are examined:

- Normal operation: no alteration in the standard MSF
- Under attack: A malicious node executes both cell depletion and schedule instability attacks to disrupt the network.
- Defence mechanisms: SCSF-PC defence mechanism against cell depletion attack, followed by SCSF-WELA defence mechanism against the schedule instability attack. Note that SCSF-WELA encompasses the countermeasures for both attack types.

5.4.2 Performance Metrics

The attacks affect the communication reliability, energy consumption, and resource utilization of both the victim and their child nodes. The validation study focuses on a network comprising 10–80 nodes and one node acting as the attacker. To evaluate network performance, the following performance metrics are defined:

- *Schedule Instability (SI)*: Frequency of changes in a cell schedule due to cell additions and deletions. It is measured as the ratio of the number of times the network was unstable to the total number of observations, as defined below.

$$SI = \left(\sum_{i=1}^{K} U_i\right) \Big/ \left(\sum_{i=1}^{K} (S_i + U_i)\right) \tag{5.7}$$

  Here, SI represents the schedule instability, U_i and S_i represent the i^{th} instance of an

unstable and stable observation, respectively, and K is the total number of observations.

- *Average Energy Consumption*: The total energy consumed in transmission, reception, and idle states by the nodes in the network. This is evaluated using the model proposed by Vilajosana *et al.* [98], which utilizes real-world measurements to provide accurate estimates of total energy consumption.

- *Rx Energy Consumption*: The total energy consumed by a node in an `Rx` state, i.e., when its radio is in `Rx` mode during the overall experiment. This metric measures the total energy consumed by `Rx` cells.

- *6P Overhead*: The number of 6P ADD/DELETE transactions executed in the network for cell negotiation.

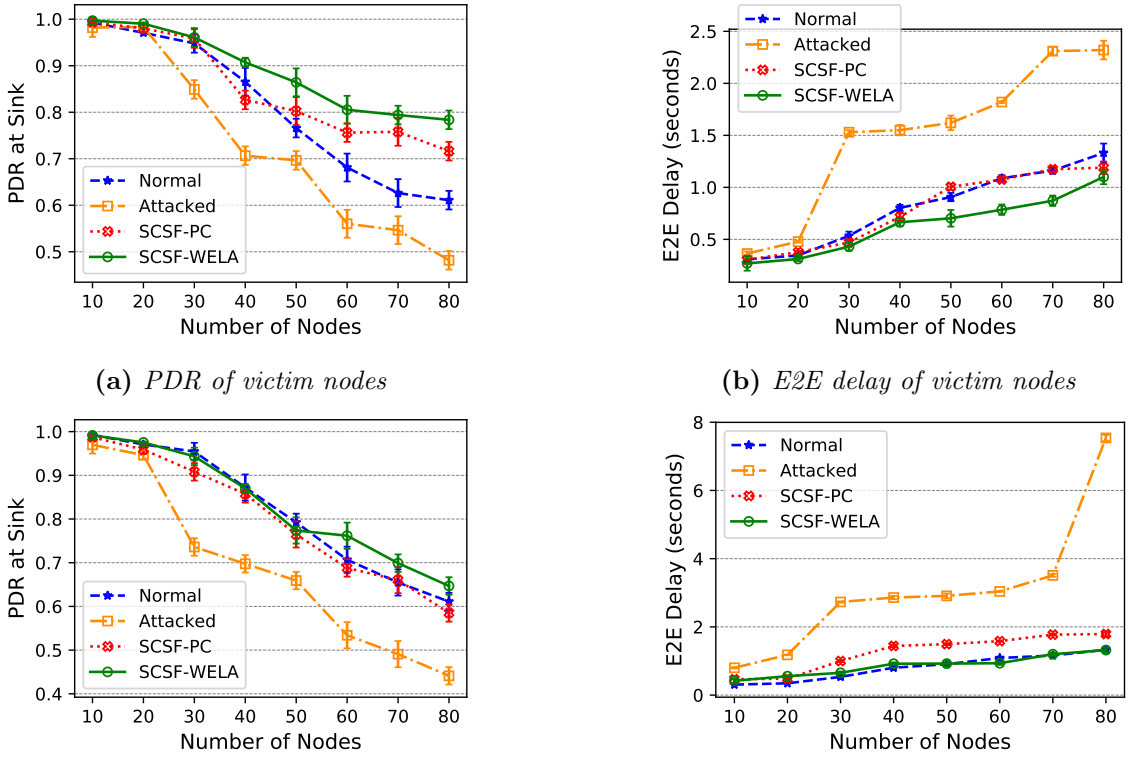
- *End-to-End (E2E) Delay*: The average time a packet travels from the source node's application layer to the destination node's application layer.

- *Packet Delivery Ratio (PDR)*: The ratio of successfully delivered packets at the root node to the total number of packets sent by the nodes in the network.

5.4.3 Results Analysis and Discussions

This section presents the simulation results for three scenarios: normal operation (i.e. no attack), with attacks but no solution in place, and with the proposed mitigation mechanisms in place. This chapter evaluates two versions of the defense mechanism: SCSF-PC, which only considers a rule-based scheme only for cell depletion attack and SCSF-WELA, which integrates the rule-based defense along with WELA-based measures to address both the attacks comprehensively. The evaluation focuses on the metrics outlined above, examining how the attack and mitigation schemes influence these metrics. All other scheduling aspects are kept consistent across the schemes to ensure that the analysis accurately reflects the effectiveness of the proposed approaches. This section is divided into three subsections:

Section 5.4.3.1 analyzes the localized impact of the attack at a more granular level, and Section 5.4.3.2 discusses the global network results. Finally, Section 5.4.3.3 explores the impact of the victim's position within the network.



(a) *PDR of victim nodes*

(b) *E2E delay of victim nodes*

(c) *PDR of victim's child nodes*

(d) *E2E delay of victim's child nodes*

Figure 5.3: *PDR and E2E delay of victim and victim's child nodes under the attack*

5.4.3.1 Localized Impact on Victim and Victim's Child Nodes

This subsection examines the impact at a more granular level, highlighting the localized impact of the attacks on the victims and evaluating the efficacy of the proposed defence schemes. Figure 5.3 illustrates the impact of the proposed attacks on PDR and E2E delay for both the victim nodes and their child nodes. This analysis highlights the differences in performance metrics under normal operation compared to that during attack and under the defense schemes, providing a clear view of how the attack degrades network performance and demonstrates the performance improvements achieved with the defense mechanisms in place.

Figure 5.3a shows the average PDR of the victim nodes which are targeted by the attacker. The PDR of these victim nodes is significantly impacted compared to the normal scenario. In case of a schedule instability attack, the impact on victim nodes is more pronounced, as the attacker performs bogus yet legitimate scheduling operations with the victim, thereby affecting the PDR of the victim node. Victim nodes experience a significant drain on their resources as they allocate a substantial portion of their Rx cells to the attacker. This resource allocation adversely affects the victim node's ability to efficiently forward packets, resulting in degraded PDR and increased E2E delay, as shown in Figure 5.3a and 5.3b, respectively. Figure 5.3c shows the impact on the PDR of victim's child nodes. The child nodes of the victim node also suffer during the attack, primarily due to the attacker creating cell scarcity for legitimate child nodes of the victims. As the attacker schedules unnecessary Tx cells with the victim, it monopolizes the resources that should be allocated to the victim's legitimate child nodes. Consequently, the PDR and E2E delay of the child nodes of the victim are significantly affected during the attack as shown in Figures 5.3c and 5.3d, respectively.

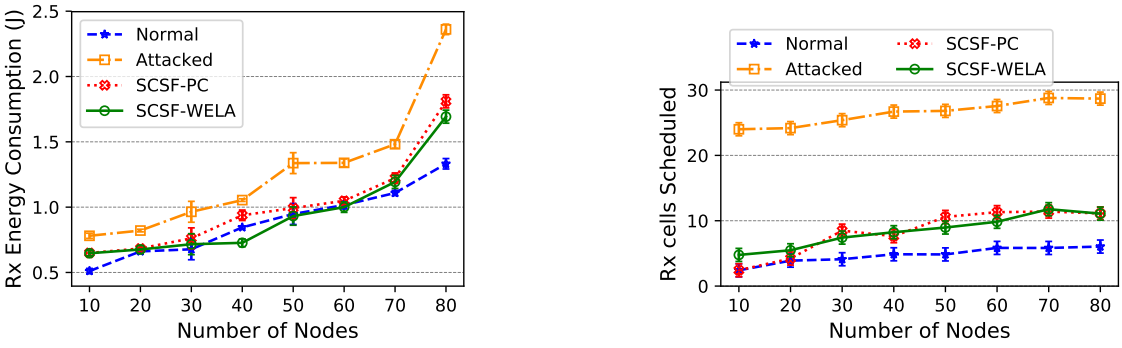
The E2E latency increases under attack conditions due to the increased number of cells scheduled with the attacker. This results in delays as legitimate traffic has to wait longer for transmission opportunities, and the impact is more evident on the victim's child nodes than on the victim itself. In Figures 5.3a and 5.3c, the difference in the impact on PDR for the victim and the victim's child nodes varies, as the cell depletion attack primarily affects the victim's child. During this attack, concurrent attacks occur on multiple nodes, thus amplifying the impact. Figures 5.3b and 5.3d show the impact on E2E delay of the victim and the victim's child nodes, respectively. In a 60-node network, the PDR of victims and victim's child is affected by 18% and 32%, respectively, while the E2E delay increased by 69% for the victim nodes and nearly tripled for the victim's child nodes. However, the proposed defense schemes significantly improve performance. Specifically, SCSF-PC increases the PDR of victim nodes by 35% and victim's child nodes by 29%. However, SCSF-WELA further enhances the PDR of victim nodes by 44% and the victim's child nodes by 43%

during attacks. Similarly, E2E delay gets reduced by 57% and 69% on victims and victim's child nodes with the SCSF-WELA defense scheme.

In the following analysis, the additional energy consumption caused by the attack is modelled. Let P be a victim node with n child nodes, represented as $C = \{c_0, c_1, c_2, c_{n-1}\}$ and $a_i \in C$ is a attacker from the set of attackers. Each child node $c_i \in C$ has selected node P as its preferred parent. Consequently, P has a number of legitimate Rx cells allocated with each child (denoted as N_{Rxc_i}) to receive packets from them. Due to the attack, node P also allocates additional Rx cells with the attacker node a_i (denoted as T_{Rxa_i}), where $N_{Rxc_i} < T_{Rxa_i}$. These extra Rx cells result in additional energy consumption at P and it is computed as follows:

$$Energy_{extra} = \sum_{a_i = c_i \in C} \left(T_{Rxa_i} - N_{Rxc_i}\right) \times E_{RxCell} \tag{5.8}$$

where E_{RxCell} is the energy consumption per `Rx` cell at node P while waiting for packets before turning its radio off. E_{RxCell} is calculated by multiplying the device operating voltage with the power consumed at the Rx state (32.6 mA, as per the model [98]).



(a) *Rx Energy consumption at victim nodes*

(b) *Rx cells scheduled at victim nodes*

Figure 5.4: *Rx energy consumption at victim nodes under varying network sizes*

Figure 5.4b shows the number of `Rx` cells scheduled at the victim node as a consequence of the attack. The attack results in the allocation of additional `Rx` cells to the malicious node, leading to increased energy consumption, as shown in Figure 5.4a. Specifically, the energy consumed by Rx cells at the victim nodes increases by 33% due to the attack. The proposed SCSF-PC defence mechanism reduces this increase by 22%, and when the SCSF-

WELA mechanism is incorporated, energy consumption is further reduced by 25% from the attack scenario in a 60-node network.

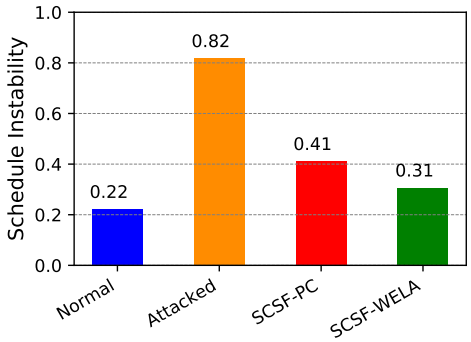


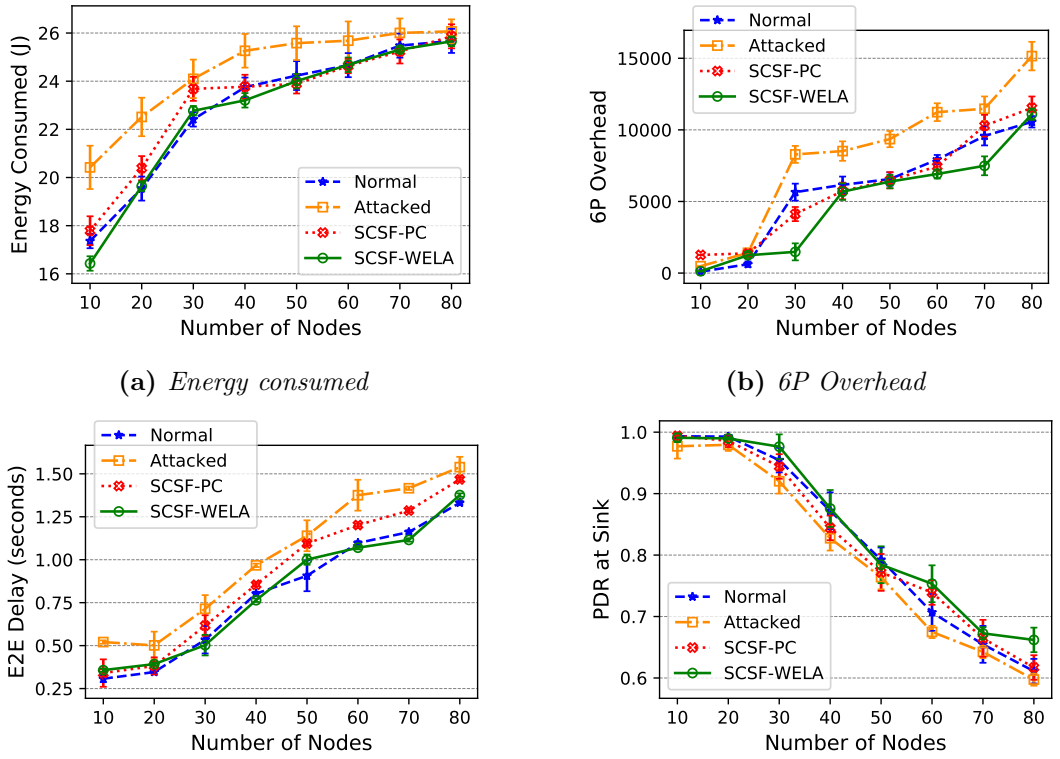
Figure 5.5: *Schedule Instability in a 50-node network*

5.4.3.2 Global Network Impact

This subsection evaluates the overall impact of the attacks on network performance and assesses the effectiveness of the mitigation mechanisms. We begin by analyzing the impact on schedule stability.

Figure 5.5 illustrates the impact of the attacks and mitigation mechanisms on schedule instability in a 50-node network. We recorded the instances when the network added or deleted cells. Under the attack, the network exhibited a higher instability score compared to the normal operations with MSF. This increase in instability is attributed to the attacker's attempts to keep the 6TiSCH schedule within the unstable boundaries of the cell utilization window, continuously triggering the ADDITION/DELETION module of the SF. Both defense mechanisms effectively reduce schedule instability compared to the attack scenario. SCSF-PC shows a notable 50% reduction in instability, demonstrating its effectiveness. However, SCSF-WELA further improves schedule stability by 62% through enhanced detection and mitigation strategies, resulting in the lowest instability score among all scenarios. This analysis confirms that the proposed mitigation approaches successfully counteract the effects of the attacks, restoring the network's stability to levels close to those observed during normal operation.

Figure 5.6a illustrates the average energy consumption of the whole network across var-



(a) *Energy consumed*

(b) *6P Overhead*

(c) *End-to-End delay*

(d) *PDR at sink*

Figure 5.6: *Comparison of results under varying network sizes*

ious network sizes. The energy consumption is calculated by summing the energy used in each radio state: transmitting, receiving, idle, and sleep mode. In each cell, the node's radio can be in one of the following states: 1. *Sleep*: The node is in a low-power state to conserve energy when no communication is required. 2. *Idle_listen*: The node is powered on and ready to communicate but is neither transmitting nor receiving. 3. *Tx_state*: The node is transmitting data. 4. *Rx_state*: The node is receiving data. The power consumption for each state is based on realistic measurements as defined in [98]. Devices are assumed to operate at a voltage of 3 Volts and consume 54.5 mA, 32.6 mA, and 6.4 mA during transmission (*Tx_state*), reception (*Rx_state*), and idle (*Idle_listen*) states, respectively. The total charge consumption P_c is calculated by multiplying the time spent in each state by the corresponding current consumption and then summing these values. The energy consumption is then computed by multiplying the total charge by the device's operating voltage, as shown in the equations below. Here, T_{Tx}, T_{Rx} and T_{Idle} represent the number of times the radio is in the

Tx_state, Rx_state and $Idle\_listen$ states, respectively, while t_{slot} denotes the slot duration in a TSCH slotframe.

$$P_c = (T_{Tx} \times 54.5 + T_{Rx} \times 32.6 + T_{Idle} \times 6.4) \tag{5.9}$$

$$E_c = \frac{P_c \times t_{slot}}{1000} \times 3\text{Volts} \tag{5.10}$$

The results indicate an increase in network energy consumption during attacks. For instance, in the case of a *cell depletion attack*, the attacker attempts to build a fake schedule with other neighbouring nodes in the network. Similarly, in *schedule* instability attack, the attacker schedules unnecessary transmission cells with its preferred parent, resulting in higher energy consumption. However, our proposed SCSF-WELA scheme effectively reduces energy consumption during attack. This is achieved by preventing the attacker from creating schedules with neighbouring nodes other than its preferred parent and proceeding by identifying and stopping resource allocation to compromised nodes. Moreover, it is observed that as the network size increases, the global impact of the attack decreases, given that only one attacker is considered in the scenario, which dilutes the attack's overall effect in larger networks.

Figure 5.6b illustrates the 6P communication overhead in the network, measured as the number of 6P transactions initiated by the SF for cell addition and deletion to adapt schedules according to traffic requirements. A 6P transaction refers to the entire negotiation process between two negotiating nodes, concluding when a cell is successfully added or deleted or when the transaction fails. One 6P transaction involves the exchange of four messages between the two nodes. The figure specifically shows the number of successful 6P transactions for each scenario. As observed, the 6P overhead increases with the number of nodes due to the higher demand for data transmission to the destination, necessitating more cell allocations and deallocations. In a 60-node network, the overhead increases by 42% under attack compared to normal operation. This is because the attacker triggers additional 6P transactions to create and maintain fake schedules with neighbouring nodes. Furthermore, the attacker aims to create schedule instability, resulting in more frequent 6P

transactions with its preferred parent. However, both the proposed mitigation approaches SCSF-PC and SCSF-WELA, effectively reduce this overhead by approximately 34% and 38%, respectively. These methods achieve this reduction by identifying and blocking malicious 6P requests from the attacker. The proposed prevention mechanism prevents the creation of schedules with the attacker when the attacker is not a legitimate child of the parent node, thereby significantly reducing unnecessary 6P transactions.

Figure 5.6c shows the average E2E delay experienced by the entire network. During the attack, the network's E2E delay rises by 25% in a 60-node network. This increase in delay is primarily due to the attacker creating cell scarcity for the legitimate children of the victim nodes, which leads to an increased delay at the victim's child nodes. This happens because the parent node allocates its share of cells to the malicious node, causing a shortage of available transmission cells for its legitimate children. Additionally, the attacker removes necessary cells, even though these cells are required to forward packets, further exacerbating the latency. As a result, the nodes under effect also experience a lower PDR, consequently impacting the total PDR, as shown in Figure 5.6d. The affected nodes experience a lower PDR due to transmission cell scarcity for the legitimate nodes, which subsequently impacts the overall network PDR. This degradation is depicted in Figure 5.6d. However, the proposed defense mechanism, when incorporated into the MSF running at each node, significantly improves network performance. In the presence of the attack, the SCSF-WELA defense mechanism enhances the end-to-end delay by 23% and the PDR by 12%, effectively mitigating the adverse impacts of the attack.

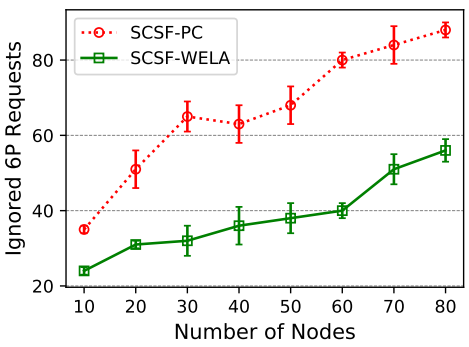


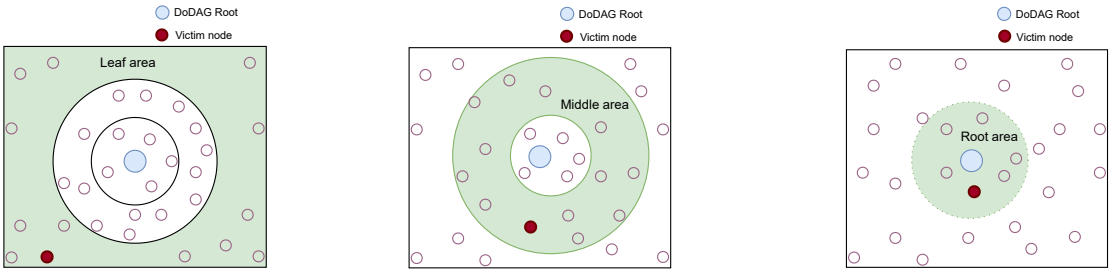
Figure 5.7: *Number of 6P Requests ignored by victim nodes from the attacker*

Figure 5.7 illustrates the effectiveness of the proposed defense mechanisms in blocking malicious 6P requests from attackers. Both the defense mechanisms for cell depletion attacks and schedule instability attacks successfully detect and prevent 6P requests initiated by compromised nodes. The SCSF-PC defense mechanism is executed by the parent node whenever it receives any 6P ADD requests and focuses on preventing the establishment of any schedule with non-child nodes. The mechanism effectively discards illegitimate requests by monitoring and validating 6P requests against the actual children list. This prevents the attacker from depleting resources and ensures that legitimate children of the victim node retain their required cell allocations. As observed, the higher number of ignored requests by the SCSF-PC scheme occurs because the attacker, when acting as a non-child, targets multiple neighbouring nodes, sending numerous 6P requests to deplete their resources. These fake requests are caught by the proposed SCSF-PC scheme and are thus ignored. On the other hand, the defense mechanism against the schedule instability attack uses stochastic learning and rule-based validation to detect abnormal cell utilization patterns. Once a node is identified as compromised, SCSF-WELA ignores malicious requests, preventing the allocation or deletion of cells and thereby maintaining schedule stability and network performance. In SCSF-WELA, the number of ignored requests is lower because the attacker, acting as a child, targets its preferred parent and sends malicious requests only to this parent.

5.4.3.3 Impact of Victim Node's Position in the Network

We analyze how the position of the victim nodes within the network affects performance. To evaluate the impact of the victim's position within the network, we collected simulation results with victims placed in three distinct areas: leaf area, middle area, and root area. Victims in the root area have a direct link to the root node. Those in the middle area are at least two hops away from the root node, while victims in the leaf area do not root any subtrees and only need to forward their own generated traffic. Figure 5.8 illustrates the attacker targeting nodes positioned in these different network areas. The received results, as shown in Figure 5.9, reveal significant differences in network performance based on the

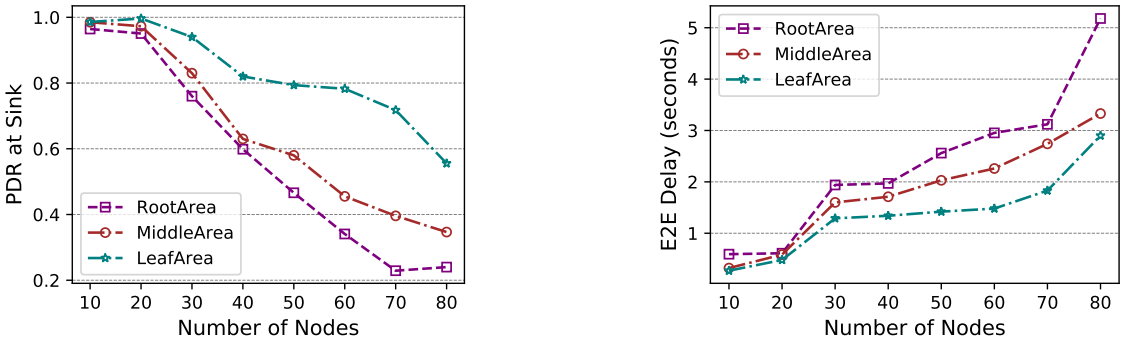
victim's location. When the victim is situated in the leaf area, the PDR experiences a moderate decline due to the limited influence of leaf nodes on overall traffic flow, as shown in Figure 5.9a. Leaf nodes are located at the network's edge and have no child nodes. Attacks in this area primarily impact the immediate victim nodes since the victim does not have a subtree to forward its traffic or establish a schedule, resulting in a lesser overall impact. In contrast, when an attacker targets a node in the middle area, it causes a substantial decrease in PDR and a notable increase in latency, as shown in Figure 5.9b. In this scenario, the victim nodes are typically more than two hops away from the root node and have multiple child nodes connected to them. This leads to more packet drops and increased latency compared to attacks in the leaf area. The most severe impact is observed when the attacker targets nodes near the root area, where victims have many neighbouring nodes. In this scenario, the PDR drops drastically, and latency increases substantially. Attacks in the root area can significantly disrupt the entire network due to their central role in data aggregation and routing.



(a) *Leaf area* **(b)** *Middle Area* **(c)** *Root area*

Figure 5.8: *Victim nodes positioned at different areas within the network*

In both the middle and root areas, the number of children connected to each victim node varies, influencing the extent of the attack's impact. Conversely, when the victim is in the leaf area, such nodes typically have no child nodes, meaning the attack primarily affects the performance of the victim node itself, with minimal broader network disruption. Specifically, in a 60-node network, when the victim is located in the root area, the victim's child nodes experienced a PDR degradation of 25% and 56% compared to the middle and



(a) *PDR at victim's child nodes*

(b) *E2E delay at victim's child nodes*

Figure 5.9: *Impact of victim's position on PDR and E2E latency at victim's child nodes*

leaf areas, respectively. Additionally, the end-to-end delay for victim's child nodes having victims near the root area increased by 17% from the middle area and nearly doubled from the leaf area victims. Thus, the impact of the attack is influenced by the position of the victim nodes within the network, which is highly dependent on RPL dynamics. Please note that all the results presented in Sections 5.4.3.1 and 5.4.3.2 are based on the random placement of the victim nodes.

5.5 Summary

This chapter explored the vulnerabilities within the standardized 6TiSCH link scheduler MSF and 6P protocol. Our research identifies the scope of potential attacks on these protocols, highlighting them as *cell depletion attack* and *schedule instability attack*. These attacks exploit weaknesses in cell scheduling, causing significant disruption in network performance, including schedule instability, increased 6P overhead, degraded reliability and increased energy consumption of the victim and victim's child nodes. The simulation results validated the severity of the attacks, demonstrating that schedule instability increases by a factor of four compared to the normal scenario in a 50-node network. The attacks lead to a 33% increase in Rx energy consumption, an 18% reduction in PDR, and a 69% increase in E2E latency for victim nodes in a 60-node network.

To alleviate these vulnerabilities, this chapter further proposed two mitigation strategies,

SCSF-PC and SCSF-WELA, to detect and prevent malicious behaviour in link schedules. These strategies effectively identify compromised nodes and block their malicious actions, ensuring the network's stability and performance. The proposed comprehensive scheme SCSF-WELA enhances schedule stability by 62% even under attack. Additionally, the scheme reduces the victims' Rx energy consumption by 25% under attack, increases the overall PDR by 12%, and improves latency by 23% compared to the attack scenario.

"Baby steps count, as long as you are going forward. You add them all up, and one day you look back and you'll be surprised."

~The Pursuit Of Happyness (An English movie)

6

Conclusions and Future Directions

This research work is motivated towards improving the performance of link scheduling in 6TiSCH networks to achieve resource-efficient data communication. The IEEE 802.15.4 TSCH MAC layer uses link scheduling for data communication in a 6TiSCH network. However, it does not define how the schedule needs to be formed and maintained. The design and management of link schedules significantly impact network performance in terms of reliability, end-to-end latency, network capacity and node battery life. The existing scheduling functions for 6TiSCH networks primarily aim to enhance specific quality of service (QoS) requirements. However, they suffer from several limitations. A key limitation is their lack of adaptability to dynamic network conditions and varying traffic demands, making them unsuitable for handling diverse traffic rates. While Minimal Scheduling Function (MSF) [43]

supports dynamic traffic management, its linear approach to cell addition and deletion results in excessive 6P signalling overhead. Additionally, most scheduling approaches neglect important factors such as pending packets and link quality during cell updates. Nevertheless, the existing works rely on random cell allocation and deallocation, further limiting their suitability for delay-sensitive networks. Another significant issue is the problem of schedule collisions inherent in the distributed nature of 6TiSCH scheduling. However, only a limited number of studies have addressed schedule collision detection and mitigation. Existing studies, including MSF, inadequately address collision detection and mitigation. Their reliance on inefficient sender-based random relocation strategies further exacerbates the problem by increasing relocation requests and relocation overhead. Furthermore, the security vulnerabilities of the link scheduling process and the 6P protocol remain largely unexplored. Despite the implementation of secure communication through Minimal Security Framework [41], the link scheduling and 6P protocol exhibit several security vulnerabilities [57]. Any attack targeting the SF or 6P can impair the normal resource allocation procedures, potentially leading to schedule instability, reduced data delivery and increased energy consumption in affected nodes. However, this issue is currently less investigated in the literature. While existing studies have focused on the security aspects of 6TiSCH such as time synchronization, routing, and jamming attacks, leaving the vulnerabilities in link scheduling process less explored.

Therefore, to improve the performance of 6TiSCH link scheduling, we worked in the following research directions: (i) developing adaptive and low-latency distributed scheduling function, (ii) addressing collision detection and mitigation in link schedules, and (iii) securing link schedules and the associated 6P protocol. Towards our first research direction, we provided three schemes: Improved Minimal Scheduling Function (IMSF), Receiver-based Traffic rate agnostic Distributed link scheduling function (RTDS), and Latency-aware Cell Deletion (LCD). For handling schedule collisions, we proposed two schemes - Enhanced Collision Detection (ECD) and ECD with cell Relocation (ECDR) for collision detection and mitigation purpose. Whereas, in our last research direction, we demonstrated the feasibility

of cell depletion attacks and schedule instability attacks in link schedules and introduced two mitigation schemes: the Secure Cell Scheduling Function based on Parent-Child Authentication (SCSF-PC) and Secure Cell Scheduling Function based on Weak Estimator Learning Automata (SCSF-WELA), to mitigate the risk of these attacks. Extensive simulations were conducted to validate each proposed solution. This chapter sums up all the proposed solutions of this dissertation along with the future research directions for 6TiSCH link scheduling.

6.1 Summary of Contributions

In this thesis, at first, we exemplified the shortcomings of the Minimal Scheduling Function (MSF) (RFC9033) in Industrial Internet of Things (IIoT) context. We observed that the linear approach used in MSF for negotiated cell addition and deletion works results in high 6P signalling overhead. We understood that the required cell count computation and cell selection strategy for allocation and deallocation are crucial for designing an efficient scheduling function. Further, they are also important to make the schedule traffic adaptive, and to minimize end-to-end latency and other resource consumption. Therefore, there is a pressing need for a scheduling mechanism to achieve low end-to-end latency while consuming less energy under any traffic rate condition. Further, it must adapt the link schedule based on present network conditions and traffic rates to ensure high transmission reliability. To achieve the above requirements, we proposed three schemes - IMSF, RTDS, and LCD. The first scheme IMSF is based on the traffic and network conditions aware link scheduling. At the same time, the second scheme RTDS deals with the random cell allocation strategy and the third scheme LCD deals with the random cell deallocation strategy in the existing link scheduling functions, including MSF. The effectiveness of the proposed solutions was verified through theoretical analysis and extensive simulation experiments. Although IMSF outperforms benchmark schemes, RTDS further achieves lower end-to-end latency while maintaining a high PDR regardless of traffic load. For example, in a 70-node network operating at 2P/SF, RTDS achieves a 38% and 73% reduction in end-to-end

latencycompared to MSF and Stratum, respectively. In the same scenario, it improves the PDR by 48% and 65%, and reduces the energy consumption by 15% and 28% compared to MSF and Stratum. Furthermore, LCD reduces end-to-end latency by 46% compared to MSF.

Further, we observed that a proper design of the scheduling function reduces schedule collisions. However, it can not be eradicated fully. With the local view of the network, neighbouring pairs of nodes may select the same negotiated slots for communication. A schedule collision occurs if these nodes are within the interference range of each other and transmit simultaneously on the same channel. As a consequence, if a schedule has multiple colliding cells, the network suffers from high packet losses, which degrades network reliability and overall performance. Therefore, effective collision detection and mitigation mechanisms are crucial for ensuring robust network performance and data reliability. Once any collision is identified, mitigating cell allocation conflict is crucial because a collision may persist across slotframes, and thus severely degrades network reliability and wastes energy. The existing schedule collision detection schemes often fail to address all collision scenarios effectively. Further, they rely on inefficient sender-based mitigation strategy. The high rate of unsuccessful relocation attempts translates to a significant increase in 6P relocation transactions, and collisions persist until successful relocation is accomplished, which ultimately degrades the network performance. To address these problems, we designed two schemes - ECD and ECDR. The first scheme ECD is a collision detection scheme that captures all collision scenarios effectively, leveraging the Exponentially Weighted Moving Average (EWMA) of the Cell Delivery Ratio (CDR) as a dynamic link quality estimator. Additionally, ECDR addresses the limitations of the existing sender-based mitigation approach. We evaluated our proposed schemes through simulation experiments. The results demonstrated significant performance improvements over existing collision handling schemes, MSF and CCR. For example, in a 70-node network, ECDR reduces end-to-end latency by 25% and 21%, compared to CCR and MSF, respectively. It also reduces the relocation overhead of 6P by 98% and 99%, respectively. Furthermore, ECDR improves PDR by 23% and 11% compared

to CCR and MSF, respectively.

The correct operation of link scheduling is essential for maintaining the performance and reliability of the network. The final part of our study identified vulnerabilities within the MSF and demonstrated the feasibility of two distinct attacks: the cell depletion attack and the schedule instability attack. These attacks can compromise the desired behaviour of link scheduling, leading to disrupted communication schedules for victim nodes and their successors, causing schedule instability, increased 6P overhead, degraded reliability and higher energy consumption. The minimal security framework standardized by IETF is unable to prevent such attacks on the MSF. Our simulation results validated the severity of these attacks. The attacks lead to a 33% increase in Rx energy consumption, 18% reduction in PDR, and a 69% increase in E2E latency for victim nodes in a 60-node network. To alleviate these vulnerabilities, we first proposed the Secure Cell Scheduling Function based on Parent-Child Authentication (SCSF-PC) scheme, which prevents cell depletion attacks initiated by compromised non-child nodes. However, the potential threat posed by compromised child nodes remains unaddressed. Therefore, we further introduced Secure Cell Scheduling Function based on Weak Estimator Learning Automata (SCSF-WELA) scheme. SCSF-WELA offers a comprehensive mitigation strategy against both attacks, enabling the identification of compromised nodes and blocking their malicious activities to ensure network stability and performance. Finally, these attacks, along with the efficacy of our proposed defense schemes, were evaluated using the 6TiSCH network simulator. The proposed comprehensive scheme SCSF-WELA enhances schedule stability by 62% even under attack. Furthermore,the scheme reduces the victims' Rx energy consumption by 25% under attack, increases the overall PDR by 12%, and improves latency by 23% compared to the attack scenario.

The major contributions of this thesis can be summarized as follows:

- An Improved Minimal Scheduling Function (IMSF) is proposed to adapt the TSCH schedule according to traffic and network conditions dynamically.
- A Receiver-based Traffic rate-agnostic Distributed link Scheduling function (RTDS) is

designed to optimize end-to-end latency, resource allocation, and signalling overhead.

- An enhanced cell deletion strategy, Latency-aware Cell Deletion (LCD), is proposed to further minimize the end-to-end latency in IIoT applications.
- The default 6P ADD and DELETE cell negotiation procedures are modified to directly incorporate the proposed RTDS and LCD schemes.
- Provided an estimation of the end-to-end delay under the random cell selection strategy and in the proposed receiver-based delay-aware strategy.
- An Enhanced Collision Detection (ECD) mechanism is proposed to capture all collision scenarios effectively.
- A receiver-based collision mitigation scheme called Enhanced Collision Detection with cell Relocation (ECDR) is designed to reduce relocation overhead and end-to-end latency.
- The default 6P cell relocation method is modified to facilitate seamless integration of the receiver-based cell relocation strategy.
- Demonstrated the existence of new vulnerabilities in MSF and their exploitation to consume scheduling resources as well as to make the 6TiSCH schedule unstable.
- Initially, Secure Cell Scheduling Function based on Parent-Child Authentication (SCSF-PC) scheme has been proposed to detect and mitigate *cell depletion attack*.
- We extended the SCSF-PC scheme by proposing a Secure Cell Scheduling Function based on Weak Estimator Learning Automata (SCSF-WELA) as a comprehensive mitigation scheme that addresses both *cell depletion attack* and *schedule instability attack*. The proposed mitigation schemes can be easily incorporated into the standard MSF.
- Finally, the comparison-based performance evaluation of the proposed solutions is conducted using 6TiSCH simulator.

6.2 Scope for Future Work

While we have made significant improvements in link scheduling within 6TiSCH network by working on several research directions, some of the future research directions still can be explored to optimize data communication in 6TiSCH networks. We describe these research scopes as follows:

- **Traffic-Class Aware Scheduling for Prioritized Delivery:** Traditional queuing disciplines, such as those used in Differentiated Services (DiffServ), enable traffic class differentiation based on the Type of Service (ToS) field in packet headers. However, DiffServ typically operates at the network layer, where traffic is treated differently based on predefined policies, often using queuing disciplines and traffic policing mechanisms. Common approaches include FIFO (First-In-First-Out), priority queuing, and weighted fair queuing, each of which determines how queued packets are selected for transmission. However, in IIoT environments, ensuring low latency and reliability for critical data demands link scheduling strategies that are effective at the TSCH MAC layer, where real-time scheduling and resource allocation occur. Future research could explore link scheduling mechanisms that can natively support differentiated services at the MAC layer, offering prioritized traffic handling and deterministic latency guarantees directly within the link layer. Unlike traditional DiffServ, which cannot ensure deterministic latency (as lower priority traffic is transmitted only when no higher priority packets are queued), a differentiated scheduling approach at the link layer could guarantee specific latency thresholds, ensuring that different traffic classes are transmitted at predictable times. In the current literature, some works have explored aspects of traffic differentiation and priority-aware scheduling. For example, the study in [104] uses a multi-queue model to prioritize critical data over non-critical data, though it does not guarantee deterministic latency. Another work, [105], proposes a fuzzy-based priority-aware TSCH scheduling technique that supports service differentiation by determining node priority using fuzzy logic and allocating slots accordingly. However, this approach does not fully integrate traffic priority into the scheduling

process. Therefore, it is another research scope where researchers can think about the development of differentiated link scheduling mechanisms within the TSCH MAC layer, capable of supporting both service differentiation and deterministic latency.

- **Ensuring Correct Link Schedules in Presence of Malicious Parent:** Communication security is a critical aspect of 6TiSCH networks that has not been extensively explored in the literature. Recent work, such as [57], provides an in-depth assessment of the security vulnerabilities associated with 6P, the protocol responsible for resource negotiation within the 6TiSCH architecture. This study is among the first to consider the implications of 6P message exchanges in the presence of malicious nodes. One of the primary concerns is that the commonly shared network keys used in 6TiSCH do not provide strict source authentication for data messages at the individual node level. This limitation makes the network susceptible to various attacks, including passive eavesdropping, active message forgery, injection, and identity impersonation [57]. These vulnerabilities are particularly alarming because security breaches such as forgery and misattribution can become more severe when a single key is shared among multiple participants [44]. A compromised node within the network could exploit this shared key to inject or alter valid 6P messages exchanged between other nodes, potentially manipulating the link schedule. This scenario allows compromised nodes to passively monitor and actively tamper with 6P negotiations, even though these negotiations are intended for specific node pairs. While our current research has addressed some vulnerabilities related to link scheduling, it assumes that the RPL preferred parent is not malicious. Therefore, it is essential to investigate scenarios where a malicious parent node is present.

- **Link Scheduling in SDN based 6TiSCH Network:** Nowadays, the integration of Software Defined Network (SDN) concepts into LLNs is an emerging area of research. However, there have been only a few studies focusing on the incorporation of SDN within 6TiSCH networks for scheduling communication cells [106, 107]. Implementing a centralized SDN architecture in LLNs presents significant challenges, particularly due

to the control traffic generated by SDN. This traffic is susceptible to jitter caused by unreliable wireless links and contention in communication cells, which can degrade the overall performance of an SDN-based network. Given these challenges, it is essential to explore effective strategies for managing resource allocation that balances both SDN control traffic and 6TiSCH network traffic after the formation of 6TiSCH network.

- **Adaptive Industrial IoT Testbed for Industrial Monitoring Applications:**

  While our simulations provide valuable insights, real-world validation through a testbed implementation would further strengthen our findings. A practical deployment would help analyze performance under real hardware constraints, environmental factors, and real-time network conditions. Existing testbeds like the FIT-IoT Lab Testbed support IoT experiments using Contiki-NG. At present, there exist certain challenges due to resource constraints and compatibility issues among the frameworks. Contiki-NG presently supports 6TiSCH Minimal configuration (RFC 8480), which provides minimal bandwidth for broadcasting EB and DIO packets only. However, the entire module for the standardized scheduling function MSF (RFC 9033) for the 6TiSCH network, which is our base framework, is not yet present on Contiki-NG. While some efforts have been made, such as integrating the 6P protocol into Contiki-NG, key components are still missing. Due to this, we conducted our evaluations using tacrshort6TiSCH including 6P and MSF. This allowed us to present a thorough and detailed analysis of our proposed frameworks. Additionally, FIT-IoT Lab is not yet compatible with the Python-based 6TiSCH simulator, limiting our ability to test our proposed schemes and benchmark algorithms in a real-world testbed. However, we plan to address this in future work by extending Contiki-NG's capabilities to support MSF fully and then implementing our solutions in a physical testbed environment. We intend to do the implementation of the entire MSF module, our proposed solutions on top of MSF and other benchmark schemes on the testbed in the near future.

  Building a testbed for 6TiSCH-based networks comes with challenges, such as hardware availability, synchronization accuracy, external interference, and scalability is-

sues. The implementation would require resource-constrained IoT devices (e.g., OpenMote, Zolertia platforms) running Contiki-NG or RIOT-OS with TSCH support. Additionally, a performance monitoring system would be essential to track network behavior. For practical applications, this testbed could be deployed in industrial IoT, smart cities, or environmental monitoring, where reliable and efficient communication is crucial.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[3] J. A. Stankovic, "Research Directions for the Internet of Things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[4] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of Things for Smart Healthcare: Technologies, Challenges, and Opportunities," *IEEE Access*, vol. 5, pp. 26 521–26 544, 2017.

[5] M. Alaa, A. Zaidan, B. Zaidan, M. Talal, and M. Kiah, "A review of smart home applications based on Internet of Things," *Journal of Network and Computer Applications*, vol. 97, pp. 48–65, 2017.

[6] E. Park, Y. Cho, J. Han, and S. J. Kwon, "Comprehensive Approaches to User Acceptance of Internet of Things in a Smart Home Environment," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2342–2350, 2017.

[7] S. P. Mohanty, U. Choppali, and E. Kougianos, "Everything you wanted to know about smart cities: The Internet of things is the backbone," *IEEE Consumer Electronics Magazine*, vol. 5, no. 3, pp. 60–70, 2016.

[8] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[9] Y. Li, X. Cheng, Y. Cao, D. Wang, and L. Yang, "Smart Choice for the Smart Grid: Narrowband Internet of Things (NB-IoT)," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1505–1515, 2018.

[10] M. Yun and B. Yuxin, "Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid," in *Proceedings of the International Conference on Advances in Energy Engineering*, 2010, pp. 69–72.

[11] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 349–359, 2014.

[12] T. Watteyne, V. Handziski, X. Vilajosana, S. Duquennoy, O. Hahm, E. Baccelli, and A. Wolisz, "Industrial Wireless IP-Based Cyber –Physical Systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1025–1038, 2016.

[13] K. Wang, Y. Wang, Y. Sun, S. Guo, and J. Wu, "Green Industrial Internet of Things Architecture: An Energy-Efficient Perspective," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 48–54, 2016.

[14] N. Ahmed, D. De, and I. Hussain, "Internet of Things (IoT) for Smart Precision Agriculture and Farming in Rural Areas," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4890–4899, 2018.

[15] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi, "From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view," *IEEE Wireless Communications*, vol. 17, no. 6, pp. 44–51, 2010.

[16] L. Mainetti, L. Patrono, and A. Vilei, "Evolution of wireless sensor networks towards the Internet of Things: A survey," in *Proceedings of the International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2011, pp. 1–6.

[17] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, "Smart objects as building blocks for the Internet of Things," *IEEE Internet Computing*, vol. 14, no. 1, pp. 44–51, 2010.

[18] M. Sha, D. Gunatilaka, C. Wu, and C. Lu, "Empirical Study and Enhancements of Industrial Wireless Sensor–Actuator Network Protocols," *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 696–704, 2017.

[19] I. Stojmenovic, "Machine-to-Machine Communications With In-Network Data Aggregation, Processing, and Actuation for Large-Scale Cyber-Physical Systems," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 122–128, 2014.

[20] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized Protocol Stack for the Internet of (Important) Things," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1389–1406, 2013.

[21] A. Aijaz and A. H. Aghvami, "Cognitive Machine-to-Machine Communications for Internet-of-Things: A Protocol Stack Perspective," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 103–112, 2015.

[22] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[23] D. De Guglielmo, S. Brienza, and G. Anastasi, "IEEE 802.15.4e: A survey," *Computer Communications*, vol. 88, pp. 1–24, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366416301980

[24] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPAN)," *IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002)*, pp. 1–700, 2005.

[25] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, "WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control," in *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*, 2008, pp. 377–386.

[26] S. Safaric and K. Malaric, "ZigBee wireless standard," in *Proceedings of the International Symposium on Electronics in Marine (ELMAR)*, 2006, pp. 259–262.

[27] "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)," *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, pp. 1–320, 2006.

[28] "Wireless Systems for Industry Automation: Process Control and Related Applications," *ISA-100.11a-2011 Standard*, pp. 1–793, 2011.

[29] "IEEE Standard for Information technology–Telecommunications and information exchange between systems - Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1 GHz License Exempt Operation," *IEEE Std 802.11ah-2016 (Amendment to IEEE Std 802.11-2016, as amended by IEEE Std 802.11ai-2016)*, pp. 1–594, 2017.

[30] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A Study of LoRa: Long Range and Low Power Networks for the Internet of Things," *Sensors*, vol. 16, no. 9, 2016.

[31] A. D. Zayas and P. Merino, "The 3GPP NB-IoT system architecture for the Internet of Things," in *Proceedings of the IEEE International Conference on Communications Workshops (ICC Workshops)*, 2017, pp. 277–282.

[32] H. Kurunathan, R. Severino, A. Koubaa, and E. Tovar, "IEEE 802.15.4e in a Nutshell: Survey and Performance Evaluation," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 3, pp. 1989–2010, 2018.

[33] S. Pollin, M. Ergen, S. C. Ergen, B. Bougard, L. V. Der Perre, I. Moerman, A. Bahai, P. Varaiya, and F. Catthoor, "Performance Analysis of Slotted Carrier Sense IEEE 802.15.4 Medium Access Layer," *IEEE Transactions on Wireless Communications*, vol. 7, no. 9, pp. 3359–3371, 2008.

[34] "IEEE Standard for Local and metropolitan area networks-Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer," *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, pp. 1–225, April 2012.

[35] IEEE, "IEEE Standard for Low-Rate Wireless Networks," *IEEE standards association*, 2015.

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

[37] T. Watteyne, M. R. Palattella, and L. A. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement," IETF RFC 7554, May 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7554

[38] X. Vilajosana, T. Watteyne, T. Chang, M. Vučinić, S. Duquennoy, and P. Thubert, "IETF 6TiSCH: A Tutorial," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 1, pp. 595–615, 2020.

[39] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, and T. Winter, "RPL: IPv6 Routing Protocol for Low-

Power and Lossy Networks," IETF RFC 6550, Mar. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6550

[40] P. Thubert, C. Bormann, L. Toutain, and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header," IETF RFC 8138, Apr. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8138

[41] M. Vučinić, J. Simon, K. Pister, and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH," IETF RFC 9031, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9031

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

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

[44] Q. Wang, X. Vilajosana, and T. Watteyne, "6TiSCH Operation Sublayer (6top) Protocol (6P)," IETF RFC 8480, Nov. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8480

[45] Y. Jin, P. Kulkarni, J. Wilcox, and M. Sooriyabandara, "A centralized scheduling algorithm for IEEE 802.15. 4e TSCH based industrial low power wireless networks," in *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, 2016, pp. 1–6.

[46] V. Kotsiou, G. Z. Papadopoulos, P. Chatzimisios, and F. Theoleyre, "Whitelisting Without Collisions for Centralized Scheduling in Wireless Industrial Networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5713–5721, 2019.

[47] A. R. Urke, O. Kure, and K. Ovsthus, "A Survey of 802.15.4 TSCH Schedulers for a Standardized Industrial Internet of Things," *Sensors*, vol. 22, no. 1, p. 15, 2022.

[48] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel, "On-the-fly bandwidth reservation for 6TiSCH wireless industrial networks," *IEEE Sensors Journal*, vol. 16, no. 2, pp. 550–560, 2015.

[49] F. Righetti, C. Vallati, G. Anastasi, and S. K. Das, "Analysis and Improvement of the On-The-Fly Bandwidth Reservation Algorithm for 6TiSCH," in *Proceedings of the IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2018, pp. 1–9.

[50] T. Hamza and G. Kaddoum, "Enhanced Minimal Scheduling Function for IEEE 802.15.4e TSCH Networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–6.

[51] I. Hosni and F. Théoleyre, "Self-healing distributed scheduling for end-to-end delay optimization in multihop wireless networks with 6TiSCH," *Computer Communications*, vol. 110, pp. 103–119, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366416306156

[52] V. Kotsiou, G. Z. Papadopoulos, P. Chatzimisios, and F. Theoleyre, "LDSF: Low-Latency Distributed Scheduling Function for Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8688–8699, 2020.

[53] G. Daneels, B. Spinnewyn, S. Latré, and J. Famaey, "ReSF: Recurrent low-latency scheduling in IEEE 802.15. 4e TSCH networks," *Ad Hoc Networks*, vol. 69, pp. 100–114, 2018.

[54] T. Chang, T. Watteyne, Q. Wang, and X. Vilajosana, "LLSF: Low Latency Scheduling Function for 6TiSCH Networks," in *Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2016, pp. 93–95.

[55] K. Muraoka, T. Watteyne, N. Accettura, X. Vilajosana, and K. S. J. Pister, "Simple Distributed Scheduling With Collision Detection in TSCH Networks," *IEEE Sensors Journal*, vol. 16, no. 15, pp. 5848–5849, 2016.

[56] T. Chang, T. Watteyne, X. Vilajosana, and Q. Wang, "CCR: cost-aware cell relocation in 6TiSCH networks," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 1, p. e3211, 2018.

[57] F. Righetti, C. Vallati, M. Tiloca, and G. Anastasi, "Vulnerabilities of the 6P protocol for the Industrial Internet of Things: Impact analysis and mitigation," *Computer Communications*, vol. 194, pp. 411–432, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422003085

[58] W. Yang, Q. Wang, Y. Wan, and J. He, "Security Vulnerabilities and Countermeasures for Time Synchronization in IEEE802.15.4e Networks," in *Proceedings of the IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2016, pp. 102–107.

[59] C. Pu, "Spam DIS Attack Against Routing Protocol in the Internet of Things," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, 2019, pp. 73–77.

[60] B. Farzaneh, M. A. Montazeri, and S. Jamali, "An Anomaly-Based IDS for Detecting Attacks in RPL-Based Internet of Things," in *Proceedings of the 5th International Conference on Web Research (ICWR)*, 2019, pp. 61–66.

[61] M. Tiloca, D. D. Guglielmo, G. Dini, G. Anastasi, and S. K. Das, "DISH: DIstributed SHuffling Against Selective Jamming Attack in IEEE 802.15.4e TSCH Networks," *ACM Transactions on Sensor Networks*, vol. 15, no. 1, dec 2018. [Online]. Available: https://doi.org/10.1145/3241052

[62] G. Carignani, F. Righetti, C. Vallati, M. Tiloca, and G. Anastasi, "Evaluation of Feasibility and Impact of Attacks Against the 6top Protocol in 6TiSCH Networks," in *Proceedings of the IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2020, pp. 68–77.

[63] E. Municio, G. Daneels, M. Vučinić, S. Latré, J. Famaey, Y. Tanaka, K. Brun, K. Muraoka, X. Vilajosana, and T. Watteyne, "Simulating 6TiSCH networks," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 3, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3494

[64] G. Mulligan, "The 6LoWPAN architecture," in *Proceedings of the 4th Workshop on Embedded Networked Sensors*, NY, USA, 2007, p. 78–82. [Online]. Available: https://doi.org/10.1145/1278972.1278992

[65] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6TiSCH: deterministic IP-enabled industrial internet (of things)," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, 2014.

[66] T. Watteyne, P. Thubert, and C. Bormann, "On Forwarding 6LoWPAN Fragments over a Multi-Hop IPv6 Network," IETF RFC 8930, Nov. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8930

[67] G. Montenegro, J. Hui, D. Culler, and N. Kushalnagar, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," IETF RFC 4944, Sep. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc4944

[68] P. Thubert and J. Hui, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," IETF RFC 6282, Sep. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6282

[69] P. Thubert and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch," IETF RFC 8025, Nov. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc8025

[70] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)," IETF RFC 6552, Mar. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6552

[71] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," IETF RFC 7252, Jun. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7252

[72] O. Gnawali and P. Levis, "The Minimum Rank with Hysteresis Objective Function," IETF RFC 6719, Sep. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6719

[73] G. Selander, J. P. Mattsson, F. Palombini, and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)," IETF RFC 8613, Jul. 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8613

[74] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic aware scheduling algorithm for reliable low-power multi-hop IEEE 802.15.4e networks," in *Proceedings of the 23rd International Symposium on Personal, Indoor and Mobile Radio Communications-(PIMRC)*, 2012, pp. 327–332.

[75] B. Li, J. Liu, and B. Ji, "Low-Overhead Wireless Uplink Scheduling for Large-Scale Internet-of-Things," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 577–587, 2021.

[76] M. Kherbache, O. Sobirov, M. Maimour, E. Rondeau, and A. Benyahia, "Decentralized TSCH scheduling protocols and heterogeneous traffic: Overview and performance evaluation," *Internet of Things*, vol. 22, p. 100696, 2023.

[77] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia, and M. Dohler, "Decentralized traffic aware scheduling in 6TiSCH networks: Design and experimental evaluation," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 455–470, 2015.

[78] R. Soua, P. Minet, and E. Livolant, "Wave: a distributed scheduling algorithm for convergecast in IEEE 802.15.4e TSCH networks," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 4, pp. 557–575, 2016.

[79] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, NY, USA, 2015, p. 337–350. [Online]. Available: https://doi.org/10.1145/2809695.2809714

[80] S. Kim, H.-S. Kim, and C. Kim, "ALICE: Autonomous Link-based Cell Scheduling for TSCH," in *Proceedings of the 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2019, pp. 121–132.

[81] S. Jeong, H.-S. Kim, J. Paek, and S. Bahk, "OST: On-Demand TSCH Scheduling with Traffic-Awareness," in *Proceedings of IEEE Conference on Computer Communications*, 2020, pp. 69–78.

[82] M. Domingo-Prieto, T. Chang, X. Vilajosana, and T. Watteyne, "Distributed PID-Based Scheduling for 6TiSCH Networks," *IEEE Communications Letters*, vol. 20, no. 5, pp. 1006–1009, 2016.

[83] I. Hosni, F. Théoleyre, and N. Hamdi, "Localized scheduling for end-to-end delay constrained Low Power Lossy networks with 6TiSCH," in *Proceedings of the IEEE Symposium on Computers and Communication (ISCC)*, 2016, pp. 507–512.

[84] T. P. Duy, T. Dinh, and Y. Kim, "Distributed cell selection for scheduling function in 6TiSCH networks," *Computer Standards and Interfaces*, vol. 53, pp. 80–88, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092054891730106X

[85] T. Chang, M. Vučinić, X. V. Guillén, D. Dujovne, and T. Watteyne, "6TiSCH minimal scheduling function: Performance evaluation," *Internet Technology Letters*, vol. 3, no. 4, p. e170, 2020.

[86] Y. Tanaka, P. Minet, M. Vučinić, X. Vilajosana, and T. Watteyne, "YSF: A 6TiSCH Scheduling Function Minimizing Latency of Data Gathering in IIoT," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8607–8615, 2022.

[87] Y. Ha and S.-H. Chung, "Enhanced 6P Transaction Methods for Industrial 6TiSCH Wireless Networks," *IEEE Access*, vol. 8, pp. 174 115–174 131, 2020.

[88] A. Kalita and M. Khatua, "Channel Condition Based Dynamic Beacon Interval for Faster Formation of 6TiSCH Network," *IEEE Transactions on Mobile Computing*, vol. 20, no. 7, pp. 2326–2337, 2021.

[89] A. Kalita and M. Khatua, "6TiSCH – IPv6 Enabled Open Stack IoT Network Formation: A Review," *ACM Transactions on Internet of Things*, vol. 3, no. 3, jul 2022. [Online]. Available: https://doi.org/10.1145/3536166

[90] D. Fanucchi, F. Righetti, C. Vallati, B. Staehle, and G. Anastasi, "Improving Link Quality Estimation Accuracy in 6TiSCH Networks," in *Proceedings of the Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, 2019, pp. 243–250.

[91] A. Verma and V. Ranga, "Mitigation of DIS flooding attacks in RPL-based 6LoWPAN networks," *Transactions on emerging telecommunications technologies*, vol. 31, no. 2, p. e3802, 2020.

[92] G. Guo, "A Lightweight Countermeasure to DIS Attack in RPL Routing Protocol," in *Proceedings of the IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, 2021, pp. 0753–0758.

[93] A. Althubaity, T. Gong, K.-K. Raymond, M. Nixon, R. Ammar, and S. Han, "Specification-based Distributed Detection of Rank-related Attacks in RPL-based Resource-Constrained Real-Time Wireless Networks," in *Proceedings of the IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, vol. 1, 2020, pp. 168–175.

[94] A. Kalita, A. Brighente, M. Khatua, and M. Conti, "Effect of DIS Attack on 6TiSCH Network Formation," *IEEE Communications Letters*, vol. 26, no. 5, pp. 1190–1193, 2022.

[95] A. Kalita, M. Gurusamy, and M. Khatua, "A Gaming and Trust-Model-Based Countermeasure for DIS Attack on 6TiSCH IoT Networks," *IEEE Internet of Things Journal*, vol. 10, no. 11, pp. 9727–9737, 2023.

[96] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, NY, USA, 2003, p. 126–137. [Online]. Available: https://doi.org/10.1145/958491.958506

[97] F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, "An Evaluation of the 6TiSCH Distributed Resource Management Mode," *ACM Transactions on Internet of Things*, vol. 1, no. 4, jul 2020. [Online]. Available: https://doi.org/10.1145/3395927

[98] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, and K. S. J. Pister, "A Realistic Energy Consumption Model for TSCH Networks," *IEEE Sensors Journal*, vol. 14, no. 2, pp. 482–489, 2014.

[99] Le, Hanh-Phuc and John, Mervin and Pister, Kris, "Energy-aware routing in wireless sensor networks with adaptive energy-slope control," *EE290Q-2 Spring*, 2009.

[100] E. Municio, G. Daneels, M. Vučinić, S. Latré, J. Famaey, Y. Tanaka, K. Brun, K. Muraoka, X. Vilajosana, and T. Watteyne, "Simulating 6TiSCH networks," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 3, p. e3494, 2019.

[101] P. Boccadoro, G. Piro, D. Striccoli, and L. A. Grieco, "Experimental Comparison of Industrial Internet of Things Protocol Stacks in Time Slotted Channel Hopping Scenarios," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[102] C. Pu, "Spam DIS attack against routing protocol in the Internet of Things," in *in Proc. of the International Conference on Computing, Networking and Communications (ICNC)*, 2019, pp. 73–77.

[103] B. J. Oommen and L. Rueda, "Stochastic learning-based weak estimation of multinomial random variables and its applications to pattern recognition in non-stationary environments," *Pattern Recognition*, vol. 39, no. 3, pp. 328–341, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320305003493

[104] H. Farag, P. Österberg, and M. Gidlund, "Congestion Control and Traffic Differentiation for Heterogeneous 6TiSCH Networks in IIoT," *Sensors*, vol. 20, no. 12, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/12/3508

[105] S. Kharb and A. Singhrova, "Fuzzy based priority aware scheduling technique for dense industrial IoT networks," *Journal of Network and Computer Applications*, vol. 125, pp. 17–27, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804518303138

[106] M. Baddeley, R. Nejabati, G. Oikonomou, S. Gormus, M. Sooriyabandara, and D. Simeonidou, "Isolating SDN control traffic with layer-2 slicing in 6TiSCH industrial IoT networks," in *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017, pp. 247–251.

[107] M. Baddeley, U. Raza, A. Stanoev, G. Oikonomou, R. Nejabati, M. Sooriyabandara, and D. Simeonidou, "Atomic-SDN: Is Synchronous Flooding the Solution to Software-Defined Networking in IoT?" *IEEE Access*, vol. 7, pp. 96 019–96 034, 2019.

Publications from the Thesis Work

Journals

1. **Karnish N. A. Tapadar**, M.Khatua and T. Venkatesh, "Traffic rate agnostic end-to-end delay optimization using receiver-based adaptive link scheduling in 6TiSCH networks", ***Ad Hoc Networks***, vol. 155, pages 103397, March, 2024, [Chapter 3]

2. **Karnish N. A. Tapadar**, M.Khatua and T. Venkatesh, "Securing Cell Scheduling Function in TSCH-based Industrial IoT Networks", ***Ad Hoc Networks***, vol. 175, pages 103864, April, 2025, [Chapter 5]

Conferences

1. **Karnish N. A. Tapadar**, M.Khatua and T. Venkatesh, "IMSF: Improved Minimal Scheduling Function for Link Scheduling in 6TiSCH Networks", ***Proceedings of the 23rd International Conference on Distributed Computing and Networking (ICDCN)***, Jan 2022, pp. 124-127, [Chapter 3]

2. **Karnish N. A. Tapadar**, M.Khatua and T. Venkatesh, "Latency-Aware Cell Deletion in Link Scheduling for 6TiSCH Networks", ***Proceedings of the 16th International Conference on COMmunication Systems & NETworkS (COMSNETS)***, January 2024, pp. 613-617, [Chapter 3]

3. **Karnish N. A. Tapadar**, A.Sakre, M.Khatua and T. Venkatesh, "Vulnerability in Cell Scheduling Function for 6TiSCH Networks: Impact and its

Mitigation", ***Proceedings of the 20th India Council International Conference (INDICON)***, December 2023, pp. 1185-1192, [Chapter 5]

Under Review Papers

Journals

1. **Karnish N. A. Tapadar**, M.Khatua and T. Venkatesh, "Holistic Design for Collision Detection and Mitigation in Link Schedules of 6TiSCH Networks", ***Transactions on Emerging Telecommunications Technologies, WILEY***, (Under review), [Chapter 4]

Publications Outside Thesis

---

1. **Karnish N. A. Tapadar**, P. Singh, and M. Khatua, "LMSF: Lightweight Minimal Scheduling Function for 6TiSCH Networks", ***Proceedings of the International Conference on Science, Technology and Engineering (ICSTE)***, February 2023, pp. 463-472.

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Guwahati 781039, India