

Tarek Sayjari

**Ensuring the QoS requirements for scalable
SDWSN through network slicing using IEEE
802.15.4e TSCH**

Revised Version

São Paulo

2023

Tarek Sayjari

**Ensuring the QoS requirements for scalable SDWSN
through network slicing using IEEE 802.15.4e TSCH**

Revised Version

Doctoral thesis presented to Escola Politécnica of Universidade de São Paulo (USP) to obtain the degree of Doctor of Science.

Concentration area: Computer Engineering

Advisor: Prof. Dr. Cíntia Borges Margi
Co-Advisor: Prof. Dr. Regina Melo Silveira


São Paulo


2023

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 20 de dezembro de 2023

Assinatura do autor: 

Assinatura do orientador: 

Catálogo-na-publicação

Sayjari, Tarek

Ensuring the QoS requirements for scalable SDWSN through network slicing using IEEE 802.15.4e TSCH / T. Sayjari -- versão corr. -- São Paulo, 2023.

118 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.INTERNET DAS COISAS 2.WIRELESS 3.REDES DE COMPUTADORES 4.SENSOR I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Acknowledgements

My principal and special thanks are directed to my supervisors, Prof. Dr. Cintia Borges Margi and Prof. Dr. Regina Melo Silveira for their great efforts. During these years, their orientation and advises were very essential to complete this work. I would also like to thank my colleagues at the Escola Politécnica da Universidade de São Paulo (EPUSP). We worked together as a team, and our discussion helped me a lot to overcome the difficulties and improve this work.

I would like to express my gratitude to the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 for the financial support during this research.

Finally, I would like to direct my great thanks to my family, and to everyone who contributed to the completion of this work.

Resumo

SAYJARI, Tarek. **Ensuring the QoS requirements for scalable SDWSN through network slicing using IEEE 802.15.4e TSCH**. 2023. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2023.

As redes de sensores sem fio definidas por software surgiram para superar os desafios das redes de sensores sem fio tradicionais, como roteamento, compartilhamento de recursos, gerenciamento de rede e configuração. No entanto, como em qualquer outra rede sem fio, os tráfegos de controle e dados competem pelos recursos limitados das redes de sensores sem fio definidas por software. Essa competição pode afetar o desempenho dos planos de controle e dados. O IEEE 802.15.4e Time Slotted Channel Hopping provou sua eficiência com redes de recursos limitados e melhorou a confiabilidade e o atraso fim-a-fim por meio do fatiamento da rede. A literatura indica que garantir o nível requerido de qualidade de serviço para as aplicações consideradas em redes com recursos tão limitados ainda representa um grande desafio, especialmente para redes escaláveis. Neste trabalho combinamos as vantagens do Time Slotted Channel Hopping com as características das redes de sensores sem fio definidas por software para projetar um mecanismo de escalonamento centralizado a fim de atender aos requisitos das aplicações em tempo real. Para atingir esse objetivo, apresentamos uma abordagem de escalonamento que isola os diferentes tipos de tráfego e atribui dinamicamente o escalonamento apropriado. Essa abordagem monitora o desempenho da aplicação, alocando mais ou menos recursos conforme a necessidade. Até onde sabemos, esta é a primeira abordagem para garantir os requisitos da aplicação para as redes de sensores sem fio definidas por software escaláveis sem hardware adicional. Adotando IT-SDN como a rede de sensores sem fio definida por software, nossa abordagem proposta é avaliada minuciosamente em comparação tanto com a abordagem de isolamento de tráfego quanto com os requisitos das aplicações. Para esta análise consideramos até 4 aplicações com diferentes níveis de prioridade, além de tamanhos de rede de até 225 nós. Considerando os planos de controle e de dados, os experimentos são realizados usando o Contiki OS e o simulador Cooja.

Os resultados obtidos comprovaram a eficiência da abordagem proposta, que aumentou a taxa de entrega de dados em até 28% e diminuiu o atraso de dados em até 57% em comparação com a abordagem de isolamento dos tráfegos. Além disso, a nossa abordagem foi capaz de garantir os requisitos das aplicações para redes de tamanho maior.

Palavras-chaves: Redes de sensores sem fio definidas por software. Time Slotted Channel Hopping. Fatiamento da rede. Qualidade de Serviço. Isolamento de tráfego.

Abstract

SAYJARI, Tarek. **Ensuring the QoS requirements for scalable SDWSN through network slicing using IEEE 802.15.4e TSCH**. 2023. Thesis (Doctorate) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2023.

Software Defined Wireless Sensor Networks (SDWSNs) emerged to overcome the challenges of traditional Wireless Sensor Networks (WSN) such as routing, resource sharing, network management, and configuration. However, as with any other wireless network, control and data traffics compete for the limited resources over SDWSNs. This competition could affect the performance of both control and data planes. IEEE 802.15.4e Time Slotted Channel Hopping (TSCH) proved its efficiency with limited resources networks and improved both reliability and end-to-end delay through network slicing. Literature indicates that ensuring the required Quality of Service (QoS) level for the considered applications over such limited resources networks still represents a big challenge, especially for scalable networks. We combine TSCH's advantages with SDWSN's characteristics to design a centralized application-aware scheduling mechanism, which meets the application's requirements in the real-time. To achieve this goal, we present the Application-Aware (AA) scheduling approach, which isolates the different traffic types and dynamically assigns adequate scheduling. The AA approach monitors the application's performance and assigns more or less resources according to the application's necessity. To the best of our knowledge, this is the first approach to ensure the application's requirements for scalable SDWSNs without additional hardware. Adopting IT-SDN as the SDWSN framework, our proposed approach is thoroughly evaluated in comparison to both the Application Traffic Isolation (ATI) approach and the application's requirements. In our analysis, we consider up to 4 applications with different priority levels, in addition to network sizes of up to 225 nodes. Considering both control and data planes, the experiments are carried out using Contiki OS and Cooja simulator.

The obtained results proved the efficiency of our proposed approach, which increased the data delivery rate by up to 28% and decreased the data delay by up to 57% in comparison to the ATI approach. Moreover, the AA approach was capable of ensuring the application's requirements for increasing network sizes.

Keywords: Software-Defined Wireless Sensor Networks. Time Slotted Channel Hopping. Network Slicing. Quality of Service. Traffic Isolation.

List of Figures

Figure 1 – SDN architecture (HALEPLIDIS et al., 2015)	27
Figure 2 – SDWSN architecture	29
Figure 3 – Packet handling procedure by the IT-SDN node	31
Figure 4 – TSCH components (KHARB; SINGHROVA, 2019)	31
Figure 5 – Timeslot’s numbering in TSCH	32
Figure 6 – Timeslot’s types in TSCH	32
Figure 7 – Channel hopping feature	33
Figure 8 – 6TiSCH minimal scheduling	33
Figure 9 – Scheduling approach’s classification	34
Figure 10 – The considered scenarios to obtain the convergence time considering: (a) One active slot, (b) Two active slots, and (c) Three active slots	45
Figure 11 – Convergence time for IT-SDN based TSCH	46
Figure 12 – Grid topology with 36 nodes and 4 applications (sinks)	47
Figure 13 – Comparison between 6TiSCH minimal scheduling and ContikiMAC	49
Figure 14 – CDTI approach with two scenarios: (a) 2S and (b) 3S	50
Figure 15 – CDTI approach’s evaluation in the data plane	51
Figure 16 – CDTI approach’s evaluation in the control plane	52
Figure 17 – ATI approach	52
Figure 18 – ATI’s search feature	53
Figure 19 – ATI approach’s evaluation: data delivery rate	54
Figure 20 – ATI approach’s evaluation: data delay	55
Figure 21 – ATI approach’s evaluation: control overhead	56
Figure 22 – ATI approach’s evaluation: energy consumption	56
Figure 23 – ATI approach’s evaluation: control plane	58
Figure 24 – Application-Aware (AA) IT-SDN system	60
Figure 25 – Sequence of the system operations	61
Figure 26 – Sch 0 scheduling	62
Figure 27 – Scheduling calculation procedure	64
Figure 28 – The new calculated scheduling (three applications case)	64
Figure 29 – Simulation scenarios for ATI and AA approaches	69
Figure 30 – Comparison between AA and ATI approaches in the data plane (four applications case)	71
Figure 31 – Comparison between AA and ATI approaches in the control plane (four applications’ case)	72
Figure 32 – Data delivery rate, changing the MCR value	73
Figure 33 – Data delay, changing the MCR value	75

Figure 34 – Control overhead, changing the MCR value	76
Figure 35 – Energy consumption, changing the MCR value	77
Figure 36 – Control delivery rate, changing the MCR value	78
Figure 37 – Control delay, changing the MCR value	79
Figure 38 – Data delivery rate, changing the DTR value	80
Figure 39 – Data delay, changing the DTR value	81
Figure 40 – Control overhead, changing the DTR value	82
Figure 41 – Energy consumption, changing the DTR value	83
Figure 42 – Control delivery rate, changing the DTR value	84
Figure 43 – Control delay, changing the DTR value	85
Figure 44 – Data delivery rate, changing the AR value	86
Figure 45 – Data delay, changing the AR value	87
Figure 46 – Control overhead, changing the AR value	88
Figure 47 – Energy consumption, changing the AR value	89
Figure 48 – Control delivery rate, changing the AR value	90
Figure 49 – Control delay, changing the AR value	91
Figure 50 – Data delivery rate, changing the DR value	92
Figure 51 – Data delay, changing the DR value	92
Figure 52 – Control overhead, changing the DR value	93
Figure 53 – Energy consumption, changing the DR value	93
Figure 54 – Control delivery rate, changing the DR value	94
Figure 55 – Control delay, changing the difference rate	94
Figure 56 – Data delivery rate, changing the topology	96
Figure 57 – Data delay, changing the topology	96
Figure 58 – Control overhead, changing the topology	97
Figure 59 – Energy consumption, changing the topology	98
Figure 60 – Control delivery rate, changing the topology	98
Figure 61 – Control delay, changing the topology	99

List of Tables

Table 1 – Flow table example	30
Table 2 – Comparison among the works that enabled several applications over the same WSN infrastructure	37
Table 3 – Scheduling approaches’ classification	40
Table 4 – Comparison among the works that adopted an SDWSN on top of TSCH	43
Table 5 – The adopted convergence time for increasing network sizes	46
Table 6 – Simulation settings	48
Table 7 – Priority levels of the considered application types	61
Table 8 – Default simulation settings	68
Table 9 – Parameter’s values for the considered scenarios	69
Table 10 – AA approach versus ATI approach	72
Table 11 – AA approach versus Application’s QoS requirements	100
Table 12 – Healthcare applications with their data rates (LATRÉ et al., 2011) . . .	102
Table 13 – An example of applying our adopted system in a real healthcare scenario	102

List of abbreviations and acronyms

ASN	Absolute Slot Number
AMUS	Adaptive MUlti-hop Scheduling
AAL	Ambient Assisted Living
AA	Application-Aware
AR	Application's Requirements
ATI	Application Traffic Isolation
ADP	Approximate Dynamic Policy
BPD	Blood Pressure Diastolic
BPS	Blood Pressure Systolic
CSMA	Carrier Sense Multiple Access
CLS	Centralized Link Scheduling
CD	Controller Discovery
CDTI	Control and Data Traffic Isolation
CTP	Collection Tree Protocol
DAP	Data Aggregation Point
DRLs	Data Replication Locations
DSP	Data Service Provider
DTR	Data Traffic Rate
DV	Default values
DAL	Device and resource Abstraction Layer
DR	Difference Rate
DCS	Dilution-based Convergecast Scheduling
ECG	Electrocardiogram

EEG	Electroencephalogram
EMG	Electromyography
VSMS	Energy Efficient Vital Signs Monitoring System
EB	Enhanced Beacon
EMSF	Enhanced Minimal Scheduling Function
FTS-SDN	Forwarding and TSCH Scheduling over SDN
GTO	Gates-to-Overlay
HB	Heart Beat
IWSNs	Industrial WSNs
IoT	Internet of Things
IRTF	Internet Research Task Force
KMB	Kou, Markowsky, and Berman
LLNs	Low-power and Lossy Networks
MAC	Medium Access Control
MCR	Metrics Calculation Rate
MV	Metric value
MSF	Minimal Scheduling Function
MST	Minimum Spanning Tree
ND	Neighborhood Discovery
NSAL	Network Services Abstraction Layer
NB	Northbound
OTF	On The Fly
SpO2	Oxygen Saturation
PDR	Packet Delivery Rate
PCE	Path Computation Element
P-SBC	Priority-based Scheduling using Best Channel

QoS	Quality of Service
RDC	Radio Duty Cycling
RPL	Routing Protocol for Low-power and lossy networks
SPATH	Shortest Path Algorithm
SDN	Software-Defined Networking
SDWSN	Software-Defined Wireless Sensor Network
SB	Southbound
TDMA	Time-division multiple access
TSCH	Time Slotted Channel Hopping
WBAN	Wireless Body Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor network

Contents

1	INTRODUCTION	19
1.1	Motivation	20
1.2	Hypothesis	21
1.3	Objectives	21
1.4	Method	22
1.5	Organization	23
2	BACKGROUND	25
2.1	Network-related concepts	25
2.1.1	Wireless Sensor Networks (WSNs)	25
2.1.2	Software-Defined Networking (SDN)	26
2.1.3	Software-Defined Wireless sensor networks (SDWSNs)	28
2.1.3.1	IT-SDN framework	29
2.2	IEEE 802.15.4e Time Slotted Channel Hopping (TSCH)	31
2.3	Chapter summary	34
3	STATE OF THE ART	35
3.1	Several applications over a single WSN infrastructure	35
3.2	TSCH scheduling approaches	37
3.3	SDWSNs on top of TSCH	41
3.4	Chapter summary	43
4	TRAFFIC ISOLATION IN SDWSNS	45
4.1	TSCH evaluation	45
4.1.1	Network convergence time for IT-SDN based TSCH	45
4.1.2	Comparison between TSCH and ContikiMAC	46
4.2	Control and data traffic isolation	49
4.3	Application's traffic isolation	52
4.4	Chapter summary	58
5	APPLICATION-AWARE (AA) SCHEDULING APPROACH	59
5.1	System architecture	59
5.2	Scheduling calculation	62
5.3	AA approach's advantages	64
5.4	Chapter summary	65
6	PERFORMANCE EVALUATION	67

6.1	Method	67
6.2	AA approach's evaluation	70
6.2.1	Comparison with the ATI approach	70
6.2.2	Comparison with the application's QoS requirements	72
6.2.2.1	Metrics calculation rate (MCR)	72
6.2.2.2	Data traffic rate (DTR)	77
6.2.2.3	Application's QoS requirements (AR)	82
6.2.2.4	Difference rate (DR)	86
6.2.2.5	Topology	91
6.3	Case study	101
6.4	Chapter summary	103
7	FINAL REMARKS	105
7.1	Future work	106
7.2	Publications	107
	BIBLIOGRAPHY	109

1 Introduction

Wireless Sensor networks (WSNs) have become the backbone for several Internet of Things (IoT) (KOCAKULAK; BUTUN, 2017) applications in domains such as security, health, and military. These networks consist of small devices called sensor nodes, which collect information from the physical environments such as forests, human bodies, and others. The main disadvantages of these networks are the limited resources of their devices concerning processing, memory, energy, and communications, in addition to the competition for the limited resources among the considered applications (AKYILDIZ *et al.*, 2002).

Software Defined Networking (SDN) is a networking paradigm that decouples the control plane from the data one. It moves all the control and management decisions to a centralized controller. By this way, SDN increases the network's flexibility and programmability and improves the network management. Since the controller has an external power supply, adopting an SDN with the limited resources networks allows a better usage of the available resources. When an SDN paradigm is applied to a WSN, the resulted architecture is called the Software-Defined Wireless Sensor Network (SDWSN) (KOBO; ABU-MAHFOUZ; HANCKE, 2017). Similar to other wireless networks, control and data messages share the same wireless medium over the SDWSN and compete for the available resources. This makes it difficult to meet the application's requirements (SAYJARI; SILVEIRA; MARGI, 2021).

IEEE 802.15.4 (GUTIERREZ *et al.*, 2011) standard defines the physical and Medium Access Control (MAC) network layers. It was mainly designed for the Wireless Personal Area Networks (WPANs), extending the network lifetime by reducing the energy consumption. IEEE 802.15.4 simplifies the implementation and reduces the overall cost. However, the data transmission rate, which is limited to 250 kbps, in addition to the low reliability and high delay are considered the main challenges of this standard (GUGLIELMO; BRIENZA; ANASTASI, 2016). In 2012, IEEE 802.15.4e, an advanced version of IEEE 802.15.4 standard, appeared to address such challenges. This new version presents the time slotted access, which increases the reliability and makes it possible to control the end-to-end delay (GUGLIELMO; BRIENZA; ANASTASI, 2016). Time Slotted Channel Hopping (TSCH) (IEEE, 2012) is one of the IEEE 802.15.4e modes which improves the reliability and delay metrics through network slicing. It schedules the different processes to determine for each node when to send, receive, and sleep. TSCH was adopted with the WSNs to ensure hard requirements of the industrial applications and proved its efficiency (MUNICIO *et al.*, 2019).

In 2015, the work of Thubert *et al.* (THUBERT; PALATTELLA; ENGEL, 2015)

was the first real attempt to adopt TSCH as the MAC layer for SDWSN. The authors presented a theoretical study and described the proposed architecture about using this combination to handle the interference and multipath fading issues. Next, the using of SDWSN on top of TSCH was adopted in several works (([BADDELEY et al., 2017](#)), ([BELLO et al., 2018](#)), ([OROZCO-SANTOS et al., 2021](#)), ([SAYJARI; SILVEIRA; MARGI, 2021](#)), ([OROZCO-SANTOS et al., 2022b](#)), ([VEISI; MONTAVONT; THEOLEYRE, 2022](#)), ([OROZCO-SANTOS et al., 2022a](#)), ([SAYJARI; SILVEIRA; MARGI, 2022](#))) to address various issues. The obtained results showed that SDWSN's slicing using TSCH improved the WSN's performance.

This work adopts TSCH to slice the SDWSN and isolate the different traffic types. It presents an approach to dynamically ensure the requirements of multiple simultaneously running applications with different priority levels.

1.1 Motivation

In 2017, Baddeley *et al.* ([BADDELEY et al., 2017](#)) argued that since the control messages share the same wireless medium with the data ones, this competition could degrade the performance. They proposed, therefore, that the isolation between these two traffic types could reduce this competition. For this purpose, they used dedicated TSCH slots to create tracks between the network nodes and the controller to accommodate control messages. As the obtained results showed, this procedure reduced the end-to-end delay of the data messages. This conclusion motivated us to investigate the effect of achieving this isolation using another simple way. We used the shared timeslots to ensure this isolation ([SAYJARI; SILVEIRA; MARGI, 2021](#)). Our results showed a clear improvement in the network performance. Unlike the work of Baddeley *et al.* ([BADDELEY et al., 2017](#)), our approach was applicable for scalable SDWSN. It is important to note that till this point, all works evaluated the performance of the data plane considering the traffics of all applications as a single traffic, without evaluating the performance of each application independently.

As a second step, we planned to extend the concept of control and data traffic isolation. In other words, we considered that the traffic of each application should be isolated from the other applications as well as the control messages ([SAYJARI; SILVEIRA; MARGI, 2022](#)). The obtained results showed, again, an improvement in the network performance. However, the growing requirements of the nowadays applications and their increasing priority levels impose the necessity of meeting these requirements.

In 2021, Orozco-Santos *et al.* ([OROZCO-SANTOS et al., 2021](#)) proposed an architecture to ensure the application's requirements in real-time. Their work addressed a very critical issue in the world of the limited resources networks. However, the proposed

solution adopted the dedicated timeslots and was not capable of supporting scalable networks. In this context, the same authors addressed in 2022 the scalability issue in other work (OROZCO-SANTOS et al., 2022a). Nevertheless, they also adopted the dedicated timeslots and used external hardware equipments.

We could say, then, that there is still a big gap, which is ensuring the application's requirements for scalable SDWSNs without using additional hardware.

1.2 Hypothesis

Since SDN completely centralizes the network management, and TSCH slices the network offering several simultaneous frequencies to exchange messages, our conviction is that adopting an SDWSN on top of TSCH could be an ideal combination to ensure the application's requirements. Briefly, our hypothesis could be abstracted in the following points:

- Since the SDN controller has an external power supply, most of the control and management tasks do not consume the limited WSN's resources.
- The considered applications should start to send data messages only after the network convergence period;
- The isolation among the different traffic types could reduce the competition for the limited resources over the SDWSN;
- Unlike the dedicated timeslots, adopting the shared timeslots could reduce the energy consumption and support the network scalability;
- The considered applications should be given different priority levels according to their requirements. In this context, giving higher priority for some application means that more network resources should be allocated to it;
- The proposed approach should be able to dynamically adapt to the application's requirements. This adaptation should include the case when in the run-time, the calculated metrics of some application are much better than the requirements. In this case, some of these application's resources should be removed to save energy.

1.3 Objectives

We aim to meet the requirements of multiple applications with different priority levels running simultaneously over the same scalable SDWSN. Thus, our main objective is to design and evaluate an application-aware scheduling approach by slicing the SDWSN

using IEEE 802.15.4e TSCH. This objective could be divided into the following specific goals:

- Design and evaluate a scheduling approach that isolates control and data messages in a scalable SDWSN framework;
- Improve the previous scheduling by isolating the traffic of each application from the other traffic types;
- Evaluate the effect of using shared timeslots on the network performance;
- Evaluate the final scheduling approach considering all its parameters to discover the effect of each one on the network performance.

There is no standard or well defined definition for the "scalability" term in sensor networks. Since our proposal supports the "scalable" SDWSNs, the "scalability" term has been considered in comparison to the related works that adopted an SDWSN on top of TSCH (these works are presented in Section 3.3).

1.4 Method

First, we aimed to determine the network convergence time for the increasing network sizes (SAYJARI; SILVEIRA; MARGI, 2021). For this purpose, we realized simulations without any running applications. For each network size, we monitored the time required for the controller to configure flows to all the network nodes. The obtained convergence times were adopted for all the results presented in this work. Next, we investigated the effect of control and data traffic isolation using shared timeslots. Thus, we proposed the Control and Data Traffic Isolation (CDTI) approach (SAYJARI; SILVEIRA; MARGI, 2021). The reference case for comparison was the 6TiSCH minimal scheduling (CHANG et al., 2021). The evaluation metrics included the data delivery rate, data delay, control overhead, and energy consumption. All implementation and simulation details are presented in Chapter 4.

As a second step, we aimed to extend the concept of control and data traffic isolation to isolate the traffic of each application from the others. Our work (SAYJARI; SILVEIRA; MARGI, 2022) realized this goal by presenting the Application Traffic Isolation (ATI) approach. We evaluated the proposed isolation approach considering the CDTI approach as the reference case for comparison. The implementation details and parameters values are also included in chapter 4.

Next, we implemented the Application-Aware (AA) scheduling approach, which considers the concept of the application's traffic isolation and aims to ensure the application's requirements in the real-time. This approach was evaluated in comparison to

both ATI approach and the application's requirements. The AA approach was thoroughly evaluated considering the following parameters: metrics calculation rate (MCR), data traffic rate (DTR), application's requirements (AR), difference rate (DR), and Topology. The performance evaluation metrics include the control and data delivery rates, the control and data delays, the control overhead, and the energy consumption.

The experiments were carried out using Contiki OS (DUNKELS; GRONVALL; VOIGT, 2004) and COOJA (ÖSTERLIND et al., 2006a) simulator. All simulations considered up to 4 running applications with different DTR values and networks of up to 225 nodes. Each experiment was repeated 10 times, and the presented value is the average of these repetitions. We evaluated the network performance considering both control and data planes.

1.5 Organization

This thesis includes the following chapters:

- Chapter 2: Background, presents the basic concepts of SDWSN and TSCH.
- Chapter 3: State of the art, highlights the relevant researches in the literature, focusing on those that adopted an SDWSN on top of TSCH.
- Chapter 4: Traffic isolation in SDWSNs, explains the concept of traffic isolation and thoroughly shows our effort to isolate the different traffic types.
- Chapter 5: Application-Aware (AA) scheduling approach, describes the full architecture and contains all design details of our proposed approach.
- Chapter 6: Performance evaluation, presents and analyzes the obtained results.
- Chapter 7: Final remarks, includes the conclusions and future work

2 Background

This chapter aims to provide a theoretical background and highlights the main definitions of the concepts that are used in this thesis. It is divided into two main sections: Section 2.1 that presents the network-related concepts, which are the WSNs, SDN, and the SDWSNs including our SDWSN framework called IT-SDN (ALVES et al., 2017); and Section 2.2 that presents the IEEE 802.15.4e TSCH, our adopted MAC layer.

2.1 Network-related concepts

We adopt an SDWSN framework, which results from applying SDN to the WSN. This section highlights these three main concepts.

2.1.1 Wireless Sensor Networks (WSNs)

Wireless Sensor Networks (WSNs) have become popular and widely adopted in the last decades to serve various applications in domains such as healthcare, security, environment, and military. They are infrastructure-less networks that use the wireless medium to communicate. WSNs are composed of tiny devices (called sensor nodes) with limited resources in terms of energy, processing capacity, memory, and communication. These sensors are used to collect data from the environment such as the temperature, light, sound, noise level, humidity, and pressure. There are two main types of how the sensors collect data from the environment: i) by event detection, where the sensors collect data only when a specified event occurs; and ii) periodically, where the sensors collect data every chosen period of time, regardless of the event detection (AKYILDIZ et al., 2002; KARL; WILLIG, 2007).

Concerning how the node uses the channel to send a message, there are mainly two types of MAC protocols that are used with the WSNs: i) Carrier Sense Multiple Access (CSMA), which minimizes the collision by asking each node to check the medium before sending the message. If the medium is free, the message is sent. Otherwise, the node waits for a period of time (in microseconds) and checks the medium again; and ii) Time-division multiple access (TDMA), which divides the time into timeslots to allow multiple nodes to use the same channel without collision (II; MOHAPATRA, 2007). IEEE 802.15.4 standard, which represents the physical and MAC network layers, was not capable of ensuring the reliability and delay requirements for critical applications. To address this challenge, the IEEE 802.15.4e was provided by features such as TDMA (GUGLIELMO; BRIENZA; ANASTASI, 2016).

Ensuring Quality of Service (QoS) requirements of the nowadays applications have become a big challenge since they have hard requirements in different aspects such as packet loss rate, delay, bandwidth, and jitter. This challenge turns much more complex in the WSNs since they have limited resources. WSN is usually established to serve only one application, which imposes the necessity of assigning a WSN per each application and leads to inefficient usage of the current infrastructure. In this sense, it is useful to use the same WSN infrastructure to serve multiple concurrent applications ([AZEEM et al., 2019](#)). However, when multiple applications with different QoS requirements share the same WSN, they compete for the limited resources, where there is no mechanism to balance, distribute, and control the resource allocation and usage ([CHEN; VARSHNEY, 2004](#); [SAYJARI; SILVEIRA; MARGI, 2021](#)). The works that aimed to share a single WSN infrastructure among several applications are highlighted in Chapter 3.

2.1.2 Software-Defined Networking (SDN)

Software-Defined Networking (SDN) is a network paradigm that separates the control plane from the data one, improving the network's flexibility. All control and management tasks are administrated by a centralized controller which has an external power supply. In this way, SDN saves resource usage and improves the network programmability. Moreover, since the SDN controller has the whole network view and has direct contact with the application layer, this facilitates resource management and makes it possible to ensure the application's QoS requirements ([KOBO; ABU-MAHFOUZ; HANCKE, 2017](#)).

All SDN nodes must know how to reach the controller. If the node does not have a direct connection with it, then the node uses some controller discovery protocol to establish such connection. In parallel, each SDN node collects neighborhood information using some neighbor discovery protocol. This information is sent to the controller, which in turn, constructs its whole network view. At this point, the controller establishes the flow tables, which contain the adequate action for each flow type. In other words, for each incoming packet, it is checked if its identifier matches an entry of the flow table. If so, the associated action (which could be receive, forward, or drop) is performed. Otherwise, the node asks the controller about this "unknown" packet. Since the SDN nodes periodically update the controller with the topology changes, then the controller updates, if necessary, the existing flow tables ([ALVES et al., 2017](#)).

Figure 1 depicts the whole SDN architecture defined by RFC 7426 which was published by the Internet Research Task Force (IRTF) in 2015 ([HALEPLIDIS et al., 2015](#)). The architecture is organized as follows from the bottom to the top: the data plane, the control and management planes (on the same level), and the application plane. The data plane contains the network devices that execute the actions taken by the control plane (such as forward and drop). The operational mode is a part of the data plane and

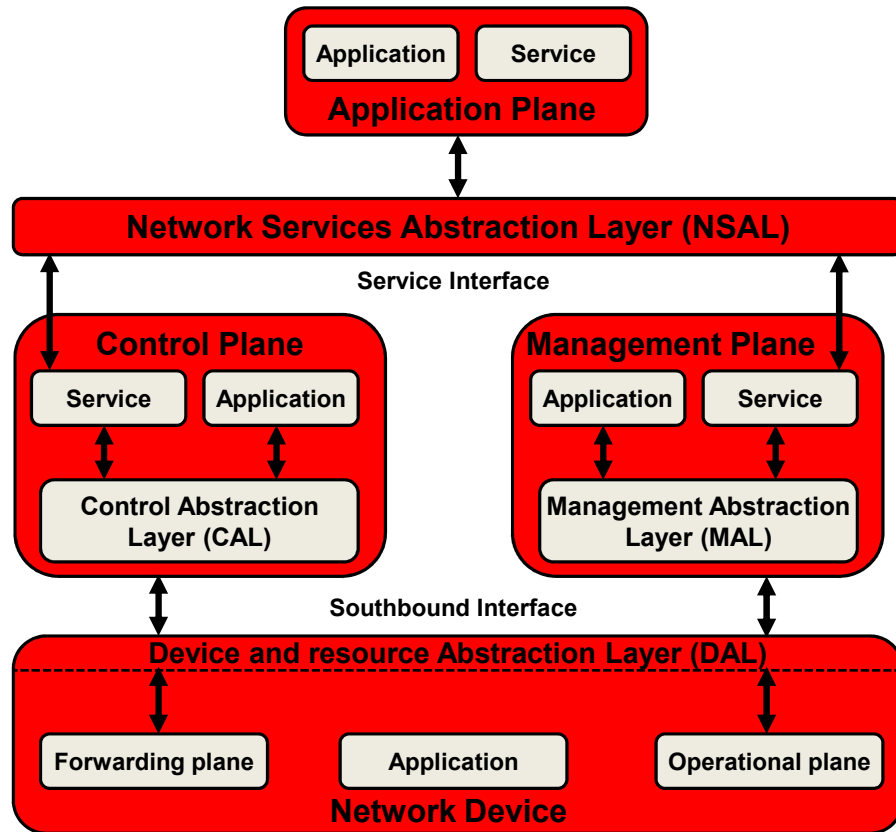


Figure 1 – SDN architecture (HALEPLIDIS et al., 2015)

provides information about the device's state, such as the temperature, quantity of energy, queues, and ports. The control and management planes are responsible for the network administration. The control plane decides the actions executed by the network devices for the different flow types. The management plane monitors the network behavior to optimize the network performance. Using this monitoring procedure, the management plane is capable of detecting when the network performance degrades. Energy consumption, latency, and delivery rate are examples of the metrics that the management plane monitors. Both the control and management planes connect with the network devices using a southbound interface through the Device and resource Abstraction Layer (DAL). The applications and services of the application plane connect with the control and management planes through a northbound interface implemented in the Network Services Abstraction Layer (NSAL).

OpenFlow (MCKEOWN et al., 2008) is the first and most common southbound protocol that ensures communication between the data and control planes. In OpenFlow, the switch has a flow table with entries containing three parts:

- The first part consists of 12 fields that are compared with the packet header. This part is used to detect the match between the incoming packets and the flow table's entries;
- The second part represents the set of actions that defines how the packet is handled

when some incoming packet matches an entry; and

- The third part contains statistics about each flow type, such as the number of the processed packets and the time between each two consecutive processed packets.

OpenFlow was evaluated in the literature and the results showed the effect of the controller's processing capability on the network performance (JARSCHHEL *et al.*, 2011).

2.1.3 Software-Defined Wireless sensor networks (SDWSNs)

Software-Defined Wireless sensor networks (SDWSNs) result from the application of the SDN paradigm to the WSNs. The sensor nodes become forwarders devices without any control or management roles. The controller has an external power supply and needs an interface to access the WSN. This interface could be a direct connection with a WSN node (ALVES *et al.*, 2017). These networks enable resource sharing and reuse and improve network management and configuration (KOBO; ABU-MAHFOUZ; HANCKE, 2017). Figure 2 shows the SDWSNs architecture, which is divided into three planes: i) the infrastructure plane, which includes the sensor nodes that sense the environment and send data messages to the sink (s). It communicates with the control plane using the southbound (SB) protocol; ii) the control plane, which includes a controller, that is responsible for all control and management tasks; and iii) the application plane, which communicates with the control plane using the northbound (NB) protocol. As well as the other wireless networks, the different traffic types (control, data) for the considered applications compete for the limited resources over the SDWSN framework (SAYJARI; SILVEIRA; MARGI, 2022).

In 2011, FlowSensor (MAHMUD; RAHMANI, 2011) was the first attempt to apply an SDN to WSN. The authors used two different approaches to exchange the data and control messages: i) OpenFlow for the communication between the controller and the sensor nodes, and ii) TCP/IP to ensure the connectivity within the data plane (between the sensor nodes and the sinks). Their experimental evaluation showed that FlowSensor improved the performance in comparison to traditional WSN.

Sensor OpenFlow (LUO; TAN; QUEK, 2012) facilitated the management of large-scale WSNs. It is another SDWSN framework that is based on OpenFlow. Sensor OpenFlow completely separated between control and data planes, and ensured the programmability of the control plane. This means that the controller is capable of setting and updating the rules, whereas the sensor nodes only check these rules and apply the associated actions. Unlike Flow-Sensor, no evaluation process was introduced in this paper.

The authors of SDN-WISE (GALLUCCIO *et al.*, 2015) aimed to reduce the exchanged control messages. For this purpose, stateful operations are installed directly on the sensor nodes. These operations are used by the sensor nodes to adopt adequate rules

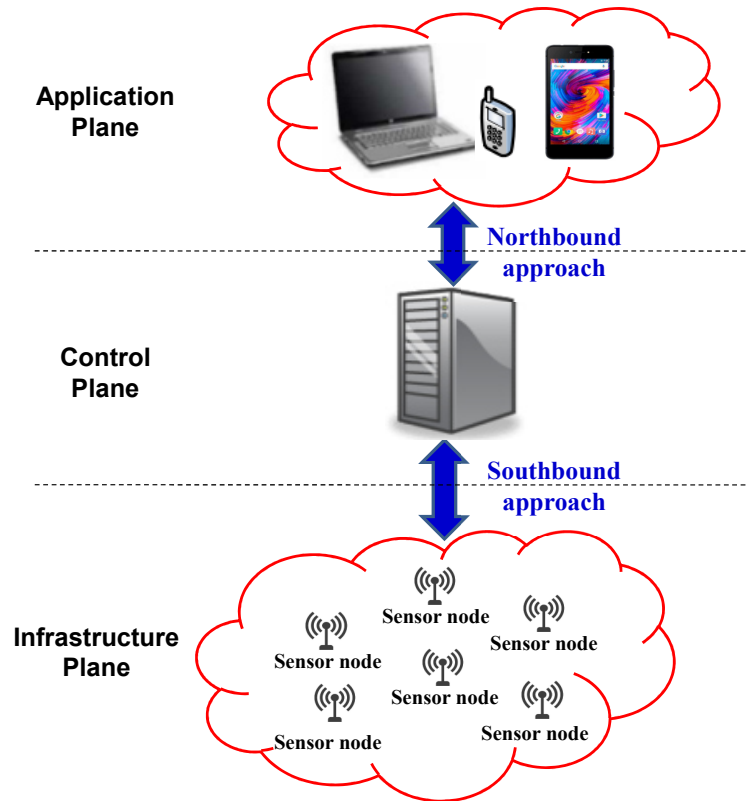


Figure 2 – SDWSN architecture

and execute actions without asking the controller. In this way, the number of exchanged control messages between the controller and the sensor nodes is reduced. The proposed approach was evaluated considering a small network of 5 nodes.

TinySDN (OLIVEIRA; GABRIEL; MARGI, 2015) is an SDWSN framework that is completely based on TinyOS (LEVIS et al., 2005). If a sensor node wants to join the SDWSN, it has to find the controller initially to start exchanging control messages. To achieve this goal, TinySDN adopted Collection Tree Protocol (CTP) (GNAWALI et al., 2013) as its underlying protocol. COOJA simulator (ÖSTERLIND et al., 2006b) was used to implement and evaluate TinySDN. IT-SDN (ALVES et al., 2017) is an improved version of TinySDN and adopts some of its concepts. Moreover, IT-SDN is open source, independent of the operating system, and completely separates its three main protocols: southbound, control discovery, and neighbor discovery. IT-SDN is further detailed in Section 2.1.3.1 since it is our employed SDWSN framework to perform all the experiments presented in this thesis.

2.1.3.1 IT-SDN framework

IT-SDN framework (ALVES et al., 2017) consists of three separated communication protocols: (i) SB protocol that ensures the communication between SDN nodes and the controller; (ii) Neighborhood Discovery (ND) protocol that is responsible for obtaining information about SDN node's neighbors; and (iii) Controller Discovery (CD) protocol

that specifies the next hop on the path between the current node and the controller. The SB protocol consists of the following packet types (ALVES et al., 2017):

- Flow request: the nodes use it to ask the controller about an unknown route;
- Flow setup: it is sent by the controller to the nodes either because of a route recalculation procedure performed by the controller or as a response to a flow request packet;
- Neighbor report: it is sent to the controller to inform it about the neighborhood information. The controller, in turn, uses this information to update the network continuously. This packet is sent if the node detects that some node entered to / exited from its neighborhood range;
- Data: it is the application layer's packet;
- Flow id register: it is used by the nodes to inform the controller that the sender is a potential destination for some flow id;
- Acknowledgement: it is used to confirm the control packets reception.

Each node has a flow table, where Table 1 depicts an example of a flow table that contains five columns: i) flow ID, the flow's identifier; ii) next hop, the next hop address towards the final destination; iii) action, determines how to handle the incoming packet (receive, forward or drop); IV) # Used, represents the number of times this entry (role) was matched; and V) age, informs the number of times this flow was updated. IT-SDN allows to reduce the flow table size through the source-routed flow setup. In this way, the packet contains the whole path and there is no need to configure all the nodes on the path from the controller to the destination node.

Table 1 – Flow table example

Flow ID	Next hop	Action	# Used	Age
13	—	Drop	4	1
5	0x1413	Receive	7	2
4	0x2010	Forward	11	3

Figure 3 shows how the IT-SDN node handles the incoming packet. The node firstly checks its flow table to check whether an entry matches the packet's flow. If such an entry is found, the node checks the packet type and handles it executing the associated action. Otherwise, the node sends a flow request packet to the controller, which responds with a flow setup packet.

Alves et al. (ALVES et al., 2019) evaluated IT-SDN framework varying important parameters for SDWSN, such as controller positioning, Radio Duty Cycling (RDC), number

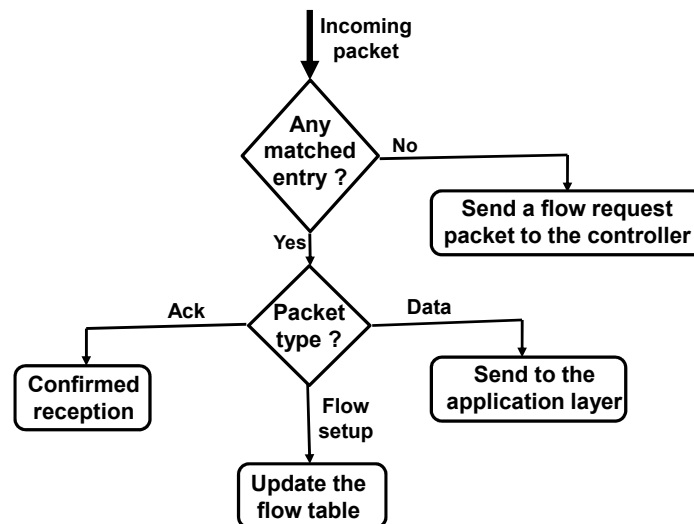


Figure 3 – Packet handling procedure by the IT-SDN node

of data sinks, and use of source routed control messages. Results showed that IT-SDN presented a competitive data delivery ratio and saved energy in comparison to the Routing Protocol for Low-power and lossy networks (RPL) (WINTER et al., 2012).

2.2 IEEE 802.15.4e Time Slotted Channel Hopping (TSCH)

IEEE 802.15.4e Time Slotted Channel Hopping (TSCH) was designed for Low-power and Lossy Networks (LLNs) to provide a reliable MAC layer. It improves reliability, end-to-end delay, and energy conservation. TSCH uses the time-slotted access to divide the time into several timeslots, clustered into one (or more) slotframe(s). In this way, TSCH slices the network into instants of time, and each slice is used to perform one or more processes. Both effects of congestion and collision are reduced by TSCH using the channel hopping feature since different channels (frequencies) are used to achieve the transmit and receive processes (GUGLIELMO; BRIENZA; ANASTASI, 2016; DOHERTY; SIMON; WATTEYNE, 2012). Figure 4 shows the main components of TSCH technology, which are slotframe, channel hopping, network formation, scheduling, and time synchronization (KHARB; SINGHROVA, 2019).

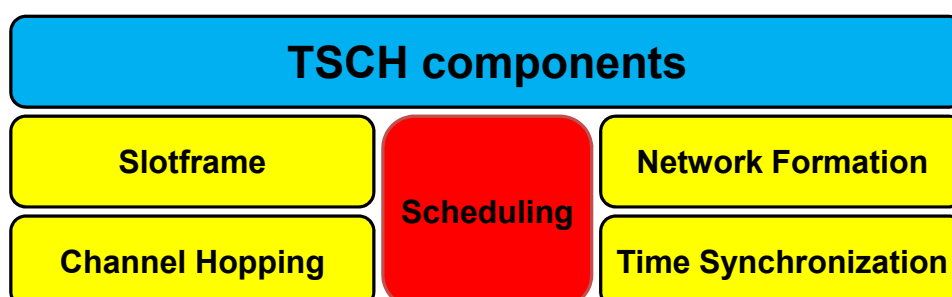


Figure 4 – TSCH components (KHARB; SINGHROVA, 2019)

The slotframe is a determined unit of time that is automatically repeated over time. It could be removed, added, or even modified during the run-time. Each slotframe is associated with a unique identifier and could be assigned to one or more nodes to exchange messages. It is divided into smaller units of time (measured in milliseconds) called timeslots. The Absolute Slot Number (ASN) determines the global number of the timeslots since the beginning of run-time. This means that each timeslot of each slotframe has a unique ASN. Figure 5 shows the first two slotframes since the beginning of run-time with their associated ASNs. We can see that each timeslot has two numbers: i) a local number, which is not changed over the repeated slotframes and a given timeslot has the same local number for all the slotframes; and ii) an ASN number, which is unique for each timeslot. The third timeslot (timeslot 2) of both slotframes 0 and 1 has the same local number which is 2. This same timeslot, on the other hand, has different global numbers which are 2 and 7 for the slotframes 0 and 1, respectively. The timeslot could be: i) shared,

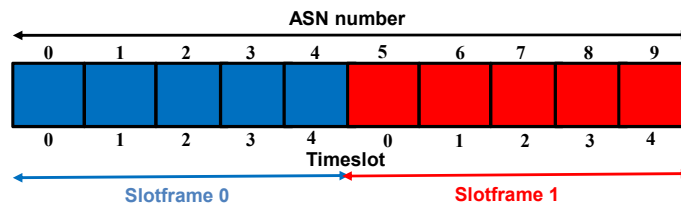


Figure 5 – Timeslot's numbering in TSCH

where several processes (send, receive) could be achieved by several nodes; ii) dedicated, where only one process could be achieved by only one node; and iii) idle, where no process is scheduled to be executed. Figure 6 shows an example of the timeslot's types. The timeslot 0 is shared since node E sends to node C and node H sends to node D, whereas timeslot 1 is dedicated since only one process is realized during it (node C sends to node A), and timeslot 2 is idle.

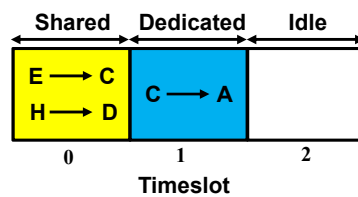


Figure 6 – Timeslot's types in TSCH

The channel hopping feature enables using several frequencies to send and receive messages. In this way, multiple processes could be simultaneously achieved during the same timeslot but using several different frequencies. This improves the network performance by mitigating the effects of interference and multipath fading (WATTEYNE; MEHTA; PISTER, 2009). TSCH offers up to 16 channels (the range varies between 0 and 15) with different frequencies. Each channel is called a channel offset and each pair of timeslot and channel offset is called a cell. In each timeslot, the physical channel is calculated using the

equation 2.1:

$$\text{Channel} = \text{macHoppingSequenceList}[(\text{macASN} + \text{ch}_{of}) \bmod \text{macHoppingSequenceLength}] \quad (2.1)$$

where `macHoppingSequenceList` represents the available channels, `macASN` is the MAC attribute representing the ASN, `chof` is the channel offset, and the `macHoppingSequenceLength` is the length of the `macHoppingSequenceList` (KHARB; SINGHROVA, 2019). Figure 7 shows an example of the channel hopping feature of TSCH, where during the timeslot 0, two processes are achieved (node A sends to node B and node C sends to node D) using both channel offsets 0 and 2, without any potential interference.

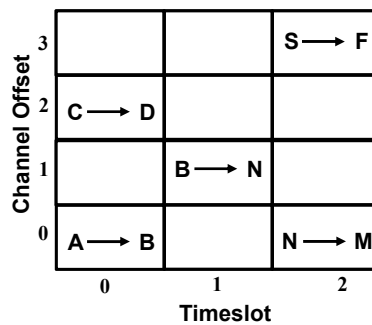


Figure 7 – Channel hopping feature

The TSCH network is formed using a PAN coordinator, which uses the Enhanced Beacon (EB) to advertise the network’s presence. The EB has all the important information that a node needs to join the TSCH network such as the slotframe and timeslot information, channel hopping information and synchronization information. When some node wants to join the TSCH network, it starts the scanning procedure to find an available channel. When such channel is found, the node listens to it until it receives an EB. The information included in the received EB is used to join and synchronize the node to the TSCH network.

TSCH provides the concept of scheduling, which is its most important aspect. The scheduling determines for each node when to transmit, receive or sleep. Figure 8 depicts the minimal mode of operation for TSCH, which is called 6TiSCH minimal scheduling (CHANG et al., 2021). It consists of only one shared cell for all the network nodes, considering a slotframe of 3 timeslots. We classify the scheduling approaches according to the calculation

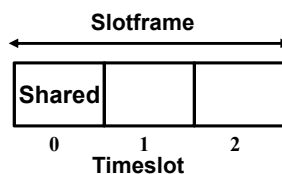


Figure 8 – 6TiSCH minimal scheduling

and control strategies (Figure 9). Considering how the adopted scheduling is calculated, it could be i) static, which is predefined and stays fixed during the network run-time;

and ii) dynamic, which is modified during the run-time to adapt to the network changes. Although the dynamic scheduling could adapt to the network changes in the real-time, this requires either a central entity to control the whole scheduling or a cooperation among the network nodes to decide the adequate scheduling. In both cases, more processing and energy consumption are required. Concerning the scheduling administration and control, it could be: i) centralized, where a central entity is responsible for constructing and maintaining the scheduling; ii) distributed, where each node calculates its own scheduling; and iii) hybrid, which combines both centralized and distributed scheduling; this means that the adopted scheduling is decided by a cooperation between the network nodes and a central entity (JAVAN; SABAIE; HAKAMI, 2019).

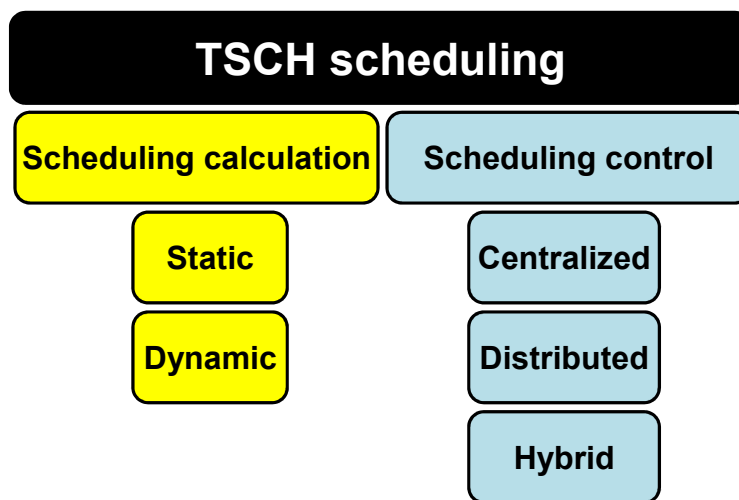


Figure 9 – Scheduling approach’s classification

TSCH was evaluated in the literature in several works (ALVES; MARGI, 2016; DUQUENNOY et al., 2017) considering various scheduling approaches. The obtained results showed that TSCH could present a better performance depending on the adopted scheduling approach. Several detailed scheduling approaches are presented in Chapter 3.

2.3 Chapter summary

We presented in this chapter a theoretical background of the architectures and technologies adopted in this thesis. First, we introduced the WSNs and the SDN, which form the SDWSN architecture. Next, we further detailed the SDWSN architecture and layers, highlighting its advantages. In this context, we presented IT-SDN, our adopted SDWSN framework, providing information about its protocols, exchanged messages, and mechanisms. Moreover, we detailed TSCH technology, our adopted MAC layer, presenting its main five components. All these concepts are fundamental in this work.

3 State of the art

This chapter reviews the literature and highlights the related work. First, Section 3.1 presents the works that used a single WSN infrastructure to serve multiple applications running simultaneously. Next, the TSCH scheduling approaches are introduced and detailed in Section 3.2. Section 3.3 discusses the works that adopted an SDWSN on top of TSCH. We conclude this chapter by summarizing the state of the art to highlight the discovered gaps in Section 3.4

3.1 Several applications over a single WSN infrastructure

The WSN infrastructure should be virtualized to enable multiple concurrent applications. This virtualization imposes an abstraction between the application services and the physical network. This abstraction, in turn, requires abstracting the sensor node's hardware and is called node-level virtualization. Some works also proposed forming a set of isolated groups over the physical network, so that each group is assigned to serve an application. This type of virtualization is called network-level virtualization (KHAN et al., 2015; NKOMO et al., 2018; AZEEM et al., 2019).

In 2006, an early effort to enable several concurrent applications over the same WSN (YU et al., 2006) was presented. It is based on Maté virtual machine (LEVIS; CULLER, 2002). To support the node-level virtualization, each sensor node stores an image code of each supported application with its required space. Then, the maximum number of applications supported by a sensor node depends on the available RAM memory (each sensor node supported up to 5 applications). The authors used the Trickle technique (LEVIS et al., 2004) to select the sensor nodes that would support some application and disseminate its code to them. For the network-level virtualization, a dynamic group is formed for each application, where the maximum number of supported groups is 16. Each sensor node becomes a member of all groups (applications) which their codes has been stored. The authors presented implementation details in addition to mathematical and simulation analysis. The results showed that although the code size is bigger, the obtained delay was smaller than Maté's one.

SenShare (LEONTIADIS et al., 2012), which is TinyOS-based, is another platform to accommodate several applications over a single WSN. Using the Collection Tree Protocol (CTP) (FONSECA et al., 2006), SenShare creates overlay networks over the physical network to enable network-level virtualization. Each overlay represents an application and is isolated from the others. These overlays are deployed using SQL-like commands and managed using their IDs. For node-level virtualization, SenShare uses a hardware

abstraction layer to allow the considered applications to share the same hardware resources. With the increasing number of running applications, more overlay networks will be needed, which increases the complexity of the proposed architecture. Simulation results showed that the application's sampling rate is reduced by about 28% compared to the single application case. Moreover, CPU and memory usage is increased when several applications are running simultaneously.

In 2013, it has been introduced one more approach to accommodate several concurrent applications using the overlay networks (KHAN et al., 2013). The proposed architecture consists of three layers: (i) physical layer, that contains sensors of different generations and capabilities. To enable the older and less capable nodes to connect the overlays, another type of sensor node called "Gates-to-Overlay (GTO)" is used; (ii) virtual sensor layer, that abstracts the different applications executed by the sensor nodes, where each sensor node supports multiple concurrent applications; and (iii) overlay layer, which is a number of independent overlays. No performance evaluation was provided in this paper.

A middleware has been proposed to virtualize the WSN in Home and Ambient Assisted Living (SHAAL) (KHALID et al., 2014). The authors aimed to extend the network lifetime to make the middleware an energy-aware approach. The infrastructure consists of three layers: i) physical layer, which includes various heterogeneous sensor nodes; (ii) virtual sensor layer, which is responsible for the node-level virtualization; and (iii) Application layer. The authors presents a detailed architecture to their middleware, but there was no performance evaluation.

Motley (KATONA et al., 2016) is a middleware that enables multiple concurrent applications over the same WSN. It runs over the Contiki OS and gives priority to some application according to its QoS requirements. Motley's architecture consists of the following main units: (i) resource monitor, which monitors the resource's usage and availability. This information is used by the nodes to ensure the application's QoS requirements; (ii) application monitor, which analyzes the application's behavior in the run-time. It collects information about metrics such as energy consumption, bandwidth and the required processing time to keep the expected level of service; and (iii) application schedule, which organizes the application's access to the available resources according to its priority level (in any given time, only one application can access the available resources). Simulation results showed that the required loading time increases with the re-loading iteration times. However, the results did not show the ability of Motley to ensure the application's requirements.

Another middleware to virtualize the sensor nodes and enable the node level virtualization has been presented in 2019 (KHALID et al., 2019) to serve a healthcare application. The authors aimed to reduce the delay caused by data aggregation in the

Ambient Assisted Living (AAL) application. The middleware’s architecture was described and the performance evaluation was carried out using Testbed. The results showed that the high-priority packets of AAL application had the least delay compared to the other packets and their delay was reduced by 97.08% in comparison to the FIFO queue.

The concept of "Data Replication" has been adopted to serve multiple concurrent applications (HARRINGTON et al., 2019) over the same WSN. The authors aimed to extend the network lifetime by reducing the overall energy consumption. A Data Service Provider (DSP) is assigned to each application and a Data Aggregation Point (DAP), which represents a sink node, collects the data of each DSP. The sensed data by each sensor is replicated to a number of Data Replication Locations (DRLs) and this data is directly uploaded to the related DSPs. Without this "data replication" mechanism, each sensor node should individually send its data to all the related DSPs, which increases the overall energy consumption. The proposed approach was compared to three other approaches: Shortest Path Algorithm (SPATH) (KOU; MARKOWSKY; BERMAN, 1981), Kou, Markowsky, and Berman (KMB) (WINTER, 1987), and Minimum Spanning Tree (MST) (VOSS, 1992). Simulation results showed that the proposed approach reduced the overall energy consumption.

Table 2 shows a comparison of the previous works. It is important to note that none of them was aware of the application’s QoS requirements. This indicates that existing technologies like TSCH with adequate scheduling, could help to meet the requirements.

Table 2 – Comparison among the works that enabled several applications over the same WSN infrastructure

Work	Node level virtualization	Network level virtualization	QoS support	Platform independence	Implementation details	Tool	Performance evaluation
(YU et al., 2006)	VM-based	Connected graph	No	No	Yes	TinyOS	Yes
(LEONTIADIS et al., 2012)	Hardware abstraction layer	Overlay based	No	No	Yes	TinyOS	Yes
(KHAN et al., 2013)	Virtual sensor layer	Overlay based	No	Not discussed	No	—	No
(KHALID et al., 2014)	Virtual sensor layer	Not discussed	No	Not discussed	No	—	No
(KATONA et al., 2016)	OS-based	Not supported	Yes	No	Yes	Contiki OS	Yes
(KHALID et al., 2019)	Middleware	Not supported	Yes	Yes	Yes	Testbed	Yes
(HARRINGTON et al., 2019)	Not discussed	Not supported	No	Yes	Yes	Matlab	Yes

3.2 TSCH scheduling approaches

For the works that adopted TSCH technology as the MAC layer, this section highlights their associated scheduling approaches.

Orchestra (DUQUENNOY et al., 2015) built RPL network protocol over TSCH. Each node autonomously computes its own scheduling that consists of several slotframes of different lengths. Each slotframe is assigned to some traffic type (RPL control messages, application data) without any central scheduling entity. This scheduling is automatically

updated with the network changes. Orchestra's slotframe is composed of a number of virtual Orchestra's timeslots, where Orchestra's timeslot is different from that of traditional TSCH. Timeslot's size here is identified using the information obtained from RPL. The evaluation process was carried out using Testbed. The results showed that Orchestra achieved a delivery rate of over than 99.9% in all the considered scenarios.

On The Fly bandwidth reservation (OTF) algorithm (PALATTELLA et al., 2015) is a distributed schedule for 6top sublayer in 6TiSCH networks. The higher layers are capable of managing and adjusting the schedule depending on the traffic flow. OTF monitors the transmitted data from a node to its neighbor. When it discovers that this transmitted data is not compatible (too large or too low) with the number of reserved cells, it asks the 6top sub-layer to adjust this number (add or remove cells).

The Centralized Link Scheduling (CLS) approach (CHOI; CHUNG, 2016) enables the allocation and de-allocation of some cells without recomputing the whole scheduling. It is designed for the industrial IoT. The sink node represents the scheduler entity. Each node sends a CLS Allocation Request message containing the required number of slots to its parent, which in turn, moves it up to the sink. When the message arrives at the sink, the required slots are allocated. Similarly, when the parent of a node changes, the node de-allocates its own transmission slots to this parent and sends it a CLS De-allocation Request message. The node also sends an CLS Allocation Request message to the new parent, which forwards it toward the sink. This scheduling approach suffers from a single point of failure since the sink is responsible for the whole scheduling process.

The Approximate Dynamic Policy (ADP) (HUYNH; THEOLEYRE; HWANG, 2017) is another centralized scheduling approach that aims to increase the network's reliability. The packet is sent from a node to a set of its neighbors through a specified link. These neighbors listen to this link and the forwarding node is the first neighbor that receives the packet and acknowledges the transmitter node.

In 2018, a scheduling approach to improve Orchestra has been proposed (REKIK et al., 2018). The authors observed that Orchestra assigns only one TX and RX timeslot for each node per slotframe. However, the transmission queue of the transmitter could include several (and sometimes full of) packets. This leads to a drop in the incoming packets. Therefore, the authors proposed a dynamic approach called e-TSCH-Orch, where the transmitter is allowed to transmit several packets to its neighbor per slotframe. The receiver checks the number of transmitted packets and directly schedules the adequate receiving cells.

A hybrid scheduling approach, which is a combination of centralized and distributed scheduling, has been presented in 2018 (KARAAGAC; MOERMAN; HOEBEKE, 2018). Each node calculates its local schedule based on its network view. A centralized scheduler entity collects the local schedules from the nodes and uses them to calculate the final

scheduling and send it to the nodes. The scheduling calculation process is repeated to adapt to the topology changes. This approach avoids any possible collision resulting from selecting the same cell by multiple nodes to transmit data.

The Priority-based Scheduling using Best Channel (P-SBC) algorithm (LEE et al., 2019) has been introduced in the 6TiSCH networks. It monitors the available links between each pair of nodes to estimate their Packet Delivery Rate (PDR). The link with the highest PDR is considered the best channel. When there is a packet with a high priority, it is tagged and its receiver selects the best channel to send it directly in the next active timeslot.

TESLA (JEONG et al., 2019) aimed to minimize the energy consumption and keep the obtained delivery rate level. It allows each node to dynamically modify its slotframe size according to the incoming traffic. Each node monitors the incoming traffic from its neighbors and periodically predicts the contention level, i.e. the level of competition (among the node's neighbors) for sending a packet to the node. If this level is high, the node decreases its slotframe size to ensure a reliable packet delivery. Whereas the node increases its slotframe size when the contention level is low to reduce the energy consumption. In the case of any slotframe's size changes, the node informs its neighbors of the new slotframe size.

The Enhanced Minimal Scheduling Function (EMSF) (HAMZA; KADDOUM, 2019) aimed to dynamically meet the industrial IoT application's requirements in terms of reliability and latency. EMSF is based on two main processes: i) calculate the average number of packets generated by each node, and ii) predict the required scheduling according to this calculation. The proposed approach can be divided into two main stages, where in the first stage the network follows the 6TiSCH minimal scheduling for a specified number of slotframes (n). Starting from the $(n + 1)$ slotframe, the second stage begins. Each node in this stage calculates the average number of packets generated by it during the previous slotframes to predict the number of packets that will be generated in the next slotframe. According to this prediction, the required cells are scheduled.

In 2020, a static scheduling approach to meet the hard requirements in terms of reliability and energy efficiency of the healthcare domain has been proposed (ELSTS et al., 2020). The schedule consists of a single shared timeslot for broadcast communications and multiple blocks of shared timeslots to ensure communication among the different node types (sensors, forwarders, gateways). During the run-time, the transmission timeslot is selected according to the node's position in the network. This means that the routing information will be used to adapt to the topology's changes.

A traffic-aware scheduler that is based on Orchestra has been designed for the IPv6 WSNs (DEAC et al., 2022). The authors argued that Orchestra is not an efficient solution in the case of high traffic. For this purpose, they proposed a scheduler that assigns more

resources (cells) in the case of high traffic to ensure the application’s QoS requirements. The additional cells which should be assigned for the nodes are calculated considering two factors: i) the traffic measured for the root’s children; and ii) the size of the RPL subtree. The evaluation process showed that the proposed scheduler led to a higher delivery rate and lower latency in comparison to Orchestra.

Dilution-based Convergecast Scheduling (DCS) (ASSIS et al., 2023) is a collision-free scheduling algorithm that adopts the Dilution (JURDZINSKI; KOWALSKI; STACHOWIAK, 2013) concept to compute the schedules. Dilution divides the network plane into a number of the grid boxes and each grid box is divided into grid and local trees. DCS uses the RPL protocol to construct a global convergecast tree by connecting the local trees. In DCS, each node has its own weight, which is assigned using a broadcast algorithm. These weights are calculated using RPL protocol, where several nodes could have the same weight. Since DCS requires the weights to be completely ordered, then the weight assigned to each node in DCS is a combination of the weight calculated by RPL and the node ID. DCS has been evaluated in comparison to the 6TiSCH minimal scheduling and Orchestra, and the obtained results showed that DCS performed better in terms of end-to-end delay and delivery rate.

Table 3 classifies the previous scheduling approaches according to the mechanism (static or dynamic) and the control method (centralized, distributed, or hybrid).

Table 3 – Scheduling approaches’ classification

Work	Scheduling approach	Scheduling mechanism	Scheduling control	Scheduler entity
(DUQUENNOY et al., 2015)	Orchestra	Dynamic	Distributed	—
(PALATTELLA et al., 2015)	On The Fly bandwidth reservation (OTF)	Dynamic	Distributed	OTF module
(CHOI; CHUNG, 2016)	Centralized Link Scheduling (CLS)	Dynamic	Centralized	The sink
(HUYNH; THEOLEYRE; HWANG, 2017)	Approximate Dynamic Policy (ADP)	Dynamic	Centralized	Path Computation Element (PCE)
(REKIK et al., 2018)	e-TSCH-Orch	Dynamic	Distributed	—
(KARAAGAC; MOERMAN; HOEBEKE, 2018)	—	Dynamic	Hybrid	Network Manager
(LEE et al., 2019)	Priority based scheduling (P-SBC)	Dynamic	Distributed	P-SBC module
(JEONG et al., 2019)	TESLA	Dynamic	Distributed	—
(HAMZA; KADDOUM, 2019)	Enhanced Minimal Scheduling Function (EMSF)	Dynamic	Distributed	—
(ELSTS et al., 2020)	—	Static	—	—
(DEAC et al., 2022)	—	Dynamic	Distributed	—
(ASSIS et al., 2023)	—	Dynamic	Distributed	—

The centralized scheduling approaches do not have an entity with the whole network view to facilitate scheduling management. Discovering the network topology, then, requires more communication processes and consumes more energy. The distributed scheduling approaches, on the other hand, are not able to combat the potential interference since there is not any central coordinator among the network nodes (or parts) to control the whole scheduling. Concerning the hybrid scheduling approaches, they combat the potential interference by combining a central entity with the whole network view and distributed

scheduling. However, they require more processing and energy capabilities in comparison to the centralized and distributed approaches. In this sense, existing some central entity with the whole network view and an independent energy unit could facilitate the scheduling management and reduce the resource consumption (JAVAN; SABAEI; HAKAMI, 2019).

3.3 SDWSNs on top of TSCH

As aforementioned, TSCH proved its efficiency with the limited resources networks in recent years. SDWSNs, thus, were adopted in several studies on top of TSCH. Thubert *et al.* (THUBERT; PALATTELLA; ENGEL, 2015) used 6TiSCH to construct centralized scheduling managed by the Path Computation Element (PCE) protocol (FARREL; VASSEUR; ASH, 2006). Multipath fading and interference issues were addressed, and the proposed architecture was theoretically presented. However, there is no implementation or evaluation processes in this paper.

Baddeley *et al.* (BADDELEY *et al.*, 2017) constructed dedicated tracks to accommodate the control messages and isolate them from the data ones. Each track is a path consisting of a set of nodes between a node and the controller entity. As for the SDWSN framework, the authors adopted μ SDN, which includes only two control message types. In addition to the tracks, the scheduling approach assigns four shared timeslots for the data messages. To evaluate the efficiency of the proposed approach, a comparison process was carried out between the adopted scheduling with and without the tracks. The simulation results showed that the end-to-end delay of the application layer was reduced in the case of the TSCH tracks.

Lo Bello *et al.* (BELLO *et al.*, 2018) proposed Forwarding and TSCH Scheduling over SDN (FTS-SDN) to address the topology changes caused by the node mobility. The authors adopted SDN-WISE (GALLUCCIO *et al.*, 2015), and the proposed approach uses the 6TiSCH minimal scheduling during the network setup period. Next, two or more timeslots are assigned to each node; one of them is assigned to accommodate the broadcast messages, and the others are assigned to the unicast messages to dynamically adapt to the topology changes. To evaluate the proposed approach, a single mobile node moving at three different speeds is considered. The results showed that the end-to-end data delay is lower when the FTS-SDN approach is adopted.

Orozco-Santos *et al.* (OROZCO-SANTOS *et al.*, 2021) improved the SDN-WISE framework by adopting TSCH as the MAC layer. They aimed to ensure the requirements of packet loss rate and time delay. Different priorities were assigned to the considered applications depending on the application's requirements. The authors added three modules to the application layer, namely Traffic Manager, Routing Process, and TSCH Scheduler. These modules work with the controller to dynamically decide the adequate route and

schedule. All implementation details were provided, and the evaluation process considered network sizes of up to only 10 nodes and was carried out using simulation and Testbed. The results showed that the proposed approach increased the network lifetime and ensured the application's requirements for the considered network sizes.

Orozco-Santos *et al.* (OROZCO-SANTOS *et al.*, 2022b) investigated the importance of SDN on top of TSCH. They compared several TSCH schedulers, namely SDN WISE-TSCH (OROZCO-SANTOS *et al.*, 2021), Adaptive Multi-hop Scheduling method (AMUS) (JIN *et al.*, 2016), 6TiSCH Minimal Scheduling Function (MSF) and Orchestra (DUQUENNOY *et al.*, 2015). All these schedulers were considered in the first stage of the simulation. Then, the two schedulers that presented the best performance in the simulation (SDN WISE-TSCH and Orchestra), were selected to be compared using Testbed. The results showed that SDN WISE-TSCH highly outstands the other schedulers.

Veisi *et al.* (VEISI; MONTAVONT; THEOLEYRE, 2022) proposed the SDN-TSCH approach. They aimed to control the reliability and delay values of the considered flows. The controller entity defines the adequate schedule and reserves the required resources for each flow. SDN-TSCH also adopts the concept of control and data traffic isolation and used dedicated timeslots to construct reliable paths from the nodes to the controller and vice-versa. Each traffic flow is also accommodated using dedicated timeslots. The authors considered up to 15 nodes and compared their approach with Orchestra (DUQUENNOY *et al.*, 2015). The simulation results showed that the proposed approach presented a better delivery rate and delay in comparison to Orchestra, especially for the network size of 15 nodes.

Orozco-Santos *et al.* (OROZCO-SANTOS *et al.*, 2022a) addressed the scalability issue in the industrial WSNs (IWSNs). They considered that the sink is the only node that connects the controller with the SD-IWSN. Thus, the sink suffers from congestion. This, in turn, limits the number of nodes. The proposed approach used the advantages of SDN to add a virtual sink, which allows the network extension. Normally, each node has a single radio interface. This means that the sink node (as well as the other nodes) can receive from only one transmitter node at a timeslot. The virtual sink could be considered as a set of nodes, each one with a radio interface, making the controller consider them as a single sink. In other words, this work considered sinks with several radio interfaces to extend the capabilities of the real sink. A sink node with three radio interfaces, for example, is capable of receiving from three different nodes at the same timeslot. The results showed that virtual sinks allowed the network extension, meeting the application's requirements.

Veisi *et al.* (VEISI; MONTAVONT; THEOLEYRE, 2023) improved the SDN-TSCH framework proposed by their previous work. They aimed to ensure the application's requirements in terms of packet delivery rate and end-to-end delay. The improvements have been realized by allowing the controller to i) improve the link quality estimation

by assigning dedicated resources to the EB messages that are used in the estimation process, and ii) enhance the time synchronization by selecting the time source for each device. The proposed approach was compared to both MSF (CHANG *et al.*, 2019) and SDN WISE-TSCH (OROZCO-SANTOS *et al.*, 2021) and the obtained results showed that SDN-TSCH outperformed and improved both packet delivery rate and end-to-end delay.

Table 4 represents a comparison among the works that adopted an SDWSN on top of TSCH. It shows that none of these works ensured the application's QoS requirements for scalable SDWSN without using additional hardware equipment. Moreover, all of them used dedicated timeslots.

Table 4 – Comparison among the works that adopted an SDWSN on top of TSCH

Work	Issue	Traffic isolation type	Timeslots type	Scalability support
(THUBERT; PALATTELLA; ENGEL, 2015)	Interference and multipath fading	—	—	No
(BADDELEY <i>et al.</i> , 2017)	Control and data traffic competition	Control and data	Dedicated	No
(BELLO <i>et al.</i> , 2018)	Mobility management	—	Dedicated	No
(OROZCO-SANTOS <i>et al.</i> , 2021)	Ensure the QoS level	Control and data / application's traffics	Dedicated	No
(OROZCO-SANTOS <i>et al.</i> , 2022b)	Comparison among TSCH schedulers	—	—	—
(VEISI; MONTAVONT; THEOLEYRE, 2022)	Control the application's requirements	Control and data / application's traffics	Dedicated	No
(OROZCO-SANTOS <i>et al.</i> , 2022a)	Scalability in the IWSNs	—	Dedicated	Yes
(VEISI; MONTAVONT; THEOLEYRE, 2023)	Control the application's requirements	Control and data / application's traffics	Dedicated	No

3.4 Chapter summary

This chapter presented and discussed the state of the art. We started with the works that enabled the WSN to be shared among multiple concurrent applications competing for limited resources. This shows the importance of adopting technologies like TSCH within the WSN to improve performance. Next, we detailed several centralized, distributed, and hybrid TSCH scheduling approaches and concluded that the absence of a central entity with the whole network view and an external energy unit led to more energy consumption and communication processes. This shows the importance of applying the SDN to the WSN. Finally, we presented the works that adopted an SDWSN on top of TSCH. These works addressed several issues; nonetheless, none of them was aware of the application's QoS requirements for scalable SDWSN without using additional hardware equipment. Since Baddeley *et al.* (BADDELEY *et al.*, 2017) improved the performance by isolating control and data traffic, Chapter 4 discusses the isolation among the different traffic types in SDWSNs using TSCH.

4 Traffic isolation in SDWSNs

Given the improvement obtained from the traffic isolation in the SDWSNs as discussed in Section 3.3, we adopt the isolation among the different traffic types using TSCH in this work. This Chapter aims to introduce our methodology to isolate the different traffic types. More precisely, we start by experimentally evaluating TSCH technology in comparison to ContikiMAC (DUNKELS, 2011) RDC strategy in Section 4.1. Next, we investigate the effect of control and data traffic isolation in Section 4.2. We extend the isolation concept to include the considered applications as Section 4.3 shows. For each of these sections, the associated simulation results and discussion are provided. This Chapter is concluded with Section 4.4 that summarizes the obtained results and determines the next step.

4.1 TSCH evaluation

This section aims to evaluate TSCH technology in comparison to ContikiMAC RDC strategy. However, firstly we intend to determine the convergence time for IT-SDN based TSCH. Next, the evaluation process is presented.

4.1.1 Network convergence time for IT-SDN based TSCH

We consider that the application's traffic should start after the network convergence (i.e., after the IT-SDN controller collects information about the network neighborhood and connections, and is able to configure flows to enable communication with the nodes). Since we use TSCH as the MAC layer, each node must associate with the TSCH network to be able to exchange control messages for IT-SDN. Therefore the convergence time depends both on the IT-SDN convergence as well as on the TSCH association process. To obtain such a time for an increasing number of nodes, simulations were carried out without running applications and considering a slotframe length of 3 timeslots. Figure 10 shows the simulation scenarios.

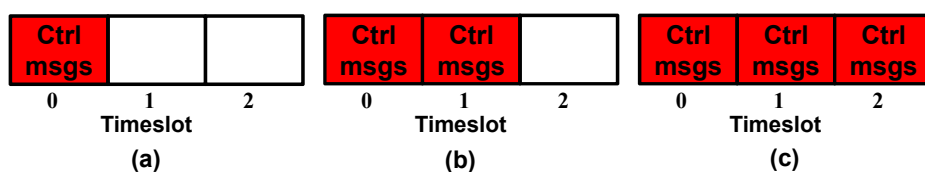


Figure 10 – The considered scenarios to obtain the convergence time considering: (a) One active slot, (b) Two active slots, and (c) Three active slots

Figure 11 shows the convergence time, which is reduced for all cases when more timeslots are assigned to accommodate the control messages. This time increases with the network size, i.e. with the number of hops the messages need to go through. Since the convergence time in the case of one active timeslot (Figure 10 (a)) is higher than the other cases, we adopt it for all the simulations presented in this thesis. Table 5 shows the adopted convergence time values for the increasing network sizes.

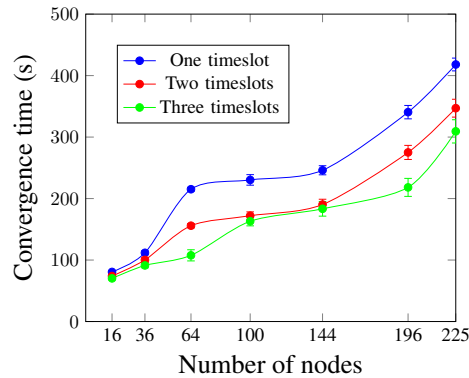


Figure 11 – Convergence time for IT-SDN based TSCH

Table 5 – The adopted convergence time for increasing network sizes

Network size (node)	16	36	64	100	144	196	225
Convergence time (s)	83	115	219	239	254	352	429

4.1.2 Comparison between TSCH and ContikiMAC

In ContikiMAC, the radio is periodically turned on to listen for potential transmissions. If such transmission is detected, the radio is kept on to receive the transmitted packet. Next, the receiver sends a link layer acknowledgment to confirm the reception. To transmit a packet, the node continues sending this packet until it receives an acknowledgment (DUNKELS, 2011).

We adopt IT-SDN with ContikiMAC RDC strategy as the reference case to evaluate IT-SDN based TSCH. The adopted scheduling is the 6TiSCH minimal scheduling depicted in Figure 8. We consider up to 4 applications with different DTR values. An application is a set of sensor nodes periodically sending data messages to one unique sink. Concerning the relation between the sensor nodes and the sinks, all sensor nodes periodically send data messages to all sinks. This means that each sensor node executes up to 4 applications simultaneously. The controller node and TSCH coordinator node (which controls join and departure processes within TSCH network) are selected to be the same node, depicted as number 1 in Figure 12. Sinks are located in the center of the most exterior row or columns, as determined by the number of applications. Transmission range of each node covers four

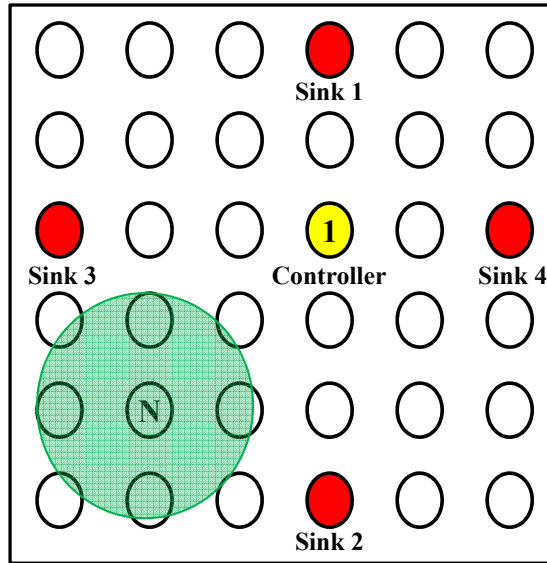


Figure 12 – Grid topology with 36 nodes and 4 applications (sinks)

adjacent nodes (one in each direction), as depicted in Figure 12 for node N. We vary the network sizes from 16 up to 225 nodes. Simulation duration was set to be 60 minutes, and the presented simulation results are the average of 10 repetitions for each experiment. Table 6 shows the simulation settings. Performance metrics include: (i) data delivery rate, which is the ratio between the total number of data messages successfully received and the total number of data messages sent; (ii) data delay, which is the average time a data message takes to reach the destination; (iii) energy consumption, which is the average energy consumed by the node; and (iv) control overhead, which includes IT-SDN’s control messages (Flow request, Flow setup, Source routed flow setup, Acknowledgement, Neighbor discovery, Controller discovery, and Neighbor report messages). The results presented in this section depict 8 curves, each one is identified using the form: “Strategy _ Number of applications”. The “Strategy” could be: TSCH (T) or ContikiMAC (C), whereas we consider up to 4 applications (1A, 2A, 3A, and 4A).

Figure 13a shows that the delivery rate values are inversely proportional to the network size for all cases. These values are reduced for more running applications. ContikiMAC presents higher values than TSCH in all cases. For network size of 225 nodes, ContikiMAC provides an increase by 11% for a single application, compared to 22.92% for the case of 4 running applications. This occurs since in the case of TSCH, all transmission / reception processes are performed using only one shared timeslot.

Concerning delay depicted in Figure 13b, it remains approximately constant for networks up to 100 nodes for all cases. TSCH presents smaller delay values compared to ContikiMAC. For network size of 225 nodes, TSCH decreases the delay by 35.11% in the case of 3 running applications. This occurs since in the case of TSCH, all processes are achieved during predefined scheduled timeslots; whereas in the case of ContikiMAC,

Table 6 – Simulation settings

Variable simulation settings	
Number of nodes	16,36,64,100,144,196,225
Number of applications (sinks)	1,2,3,4
Data traffic rate	1 packet per 1/4/8/10 minutes
Data traffic start time	[83 to 429] s
Fixed simulation settings	
Topology	Square grid
Distance between neighbors	50 m
Compiling mote	Z1
Radio environment	UDGM
Simulation duration	3600 s
Simulation repetition	10 times for each case
Radio module power	0 dB
ContikiMAC channel check rate	16 Hz
IT-SDN version	0.4.1
Controller re-transmission timeout	60 s
ND protocol	Collect-based
CD protocol	None
Link metric	Expected Transmission Count (ETX)
Route recalculation threshold	20 %
Size of the flow table	10 entries
Neighbor report max frequency	1 packet per minute
Route calculation algorithm	Dijkstra
Flow setup	Source routed
Data payload size	10 bytes
Number of TSCH channels	4 channels
Slotframe length	3 timeslots
Timeslot length	15 ms
Enhanced Beacon (EB) transmission rate	2 s

receiving some message occurs only when the receiver node detects the transmission during its wake up.

Figure 13c depicts that the control overhead is directly proportional to the network size for all cases. ContikiMAC and TSCH present similar performance in terms of control overhead. In the case of 3 applications, TSCH increases the control overhead by 9.27% for network size of 196 nodes, whereas decreases it by 5.23% for network size of 225 nodes. This occurs since both strategies use similar retransmission mechanisms, i.e. an IT-SDN's control message is retransmitted when the acknowledgment delays or is lost.

Concerning the energy consumption depicted in Figure 13d shows that whereas the energy consumption in the case of TSCH stays constant regardless of the network size and number of applications, ContikiMAC consumes more energy for larger network sizes and more applications. TSCH consumes more energy than ContikiMAC in most cases. This occurs since in the case of TSCH, each node wakes up at the beginning of each timeslot, then it performs its scheduled process or sleeps until the end of the timeslot. The nodes of ContikiMAC, on the other hand, wake up periodically, and their wake up duration

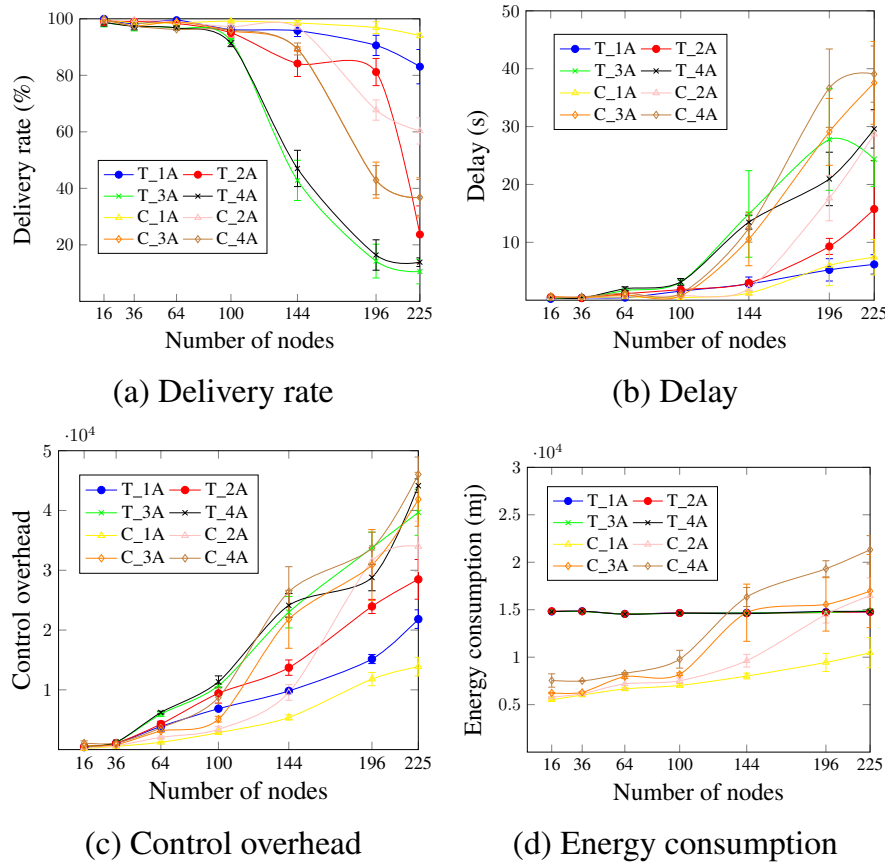


Figure 13 – Comparison between 6TiSCH minimal scheduling and ContikiMAC

depends on detecting (or not) a transmitted message and the required time to receive it.

These results show that although ContikiMAC outperformed TSCH in terms of delivery rate and energy consumption, TSCH was capable of reducing the delay even with its minimal mode of operation.

4.2 Control and data traffic isolation

Since the isolation between control and data messages ([BADDELEY et al., 2017](#)) by creating dedicated tracks using TSCH reduced the end-to-end delay, we here investigate the effect of such isolation adopting another approach. Our main motivation for this investigation is that control and data traffic isolation could help to reduce the competition for resources, which improves the network performance.

The proposed approach is called the Control and Data Traffic Isolation (CDTI). It assigns the first timeslot for the control messages, and one (or more) timeslot is assigned for the data messages. We consider two scenarios for comparison as Figure 14 shows: (i) 2S, where the first timeslot is reserved for the control messages, and the next one is assigned for data ones; and (ii) 3S, where one more timeslot is assigned for the data messages. The reference case for comparison is the 6TiSCH minimal scheduling depicted in Figure 8. We

adopt the same simulation settings and evaluation metrics presented in Section 4.1.2. The results are identified using the form: Strategy __ Number of applications __ Number of active timeslots”, and the curves with "1S" represent the 6TiSCH minimal scheduling. In addition to the data plane, we evaluate the CDTI approach in the control plane considering both i) control delivery rate, which is the total number of received control messages divided by the total number of sent control messages; and ii) control delay, which is the average time a control message takes to reach its destination.

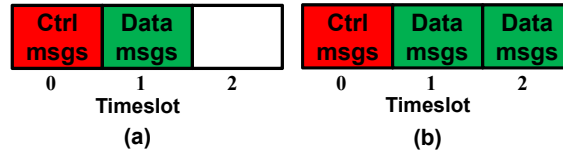


Figure 14 – CDTI approach with two scenarios: (a) 2S and (b) 3S

Figure 15 depicts the CDTI approach’s evaluation in comparison to the 6TiSCH minimal scheduling in the data plane. The higher data delivery rate, as Figure 15a shows, is obtained when more timeslots are assigned for the data messages, regardless of the network size and number of applications. For network size of 196 nodes and 4 running applications, the obtained delivery rate is 16.41% for only one active timeslot, compared to 39.54% and 44.65% for two and three active timeslots, respectively. This occurs since the CDTI approach reduces the probability of collision as well as the number of lost messages resulted from the buffer fullness.

Concerning the data delay depicted in Figure 15b, all cases present similar behaviors up to 100 nodes, regardless of the network size, number of applications, and number of active timeslots. For larger network sizes, the data delay increases with the network size and number of applications. Notice that more active timeslots led to lower delay in the application layer. For network size of 196 nodes and 4 applications, the delay value was 20.95 s for one active timeslot, compared to 15.28 s and 11.03 s for two and three active timeslots, respectively. This occurs since the proposed traffic isolation led to less congestion, thus the data message spent less time to reach its destination.

Figure 15c shows that the control overhead is directly proportional to the network size and number of applications. This overhead is not highly affected by the number of active timeslots, thus the CDTI approach did not affect the control overhead. Concerning the energy consumption, Figure 15d depicts that for the same number of active timeslots, this energy stays approximately constant regardless of the network size and number of applications. The energy consumption increases with the number of active timeslots. For a network size of 144 nodes and 4 applications, the consumption was 14.58 J for one active timeslot, compared to 27.19 J and 39.44 J for two and three active timeslots, respectively. This occurs since more active timeslots mean that the nodes should wake up for more time.

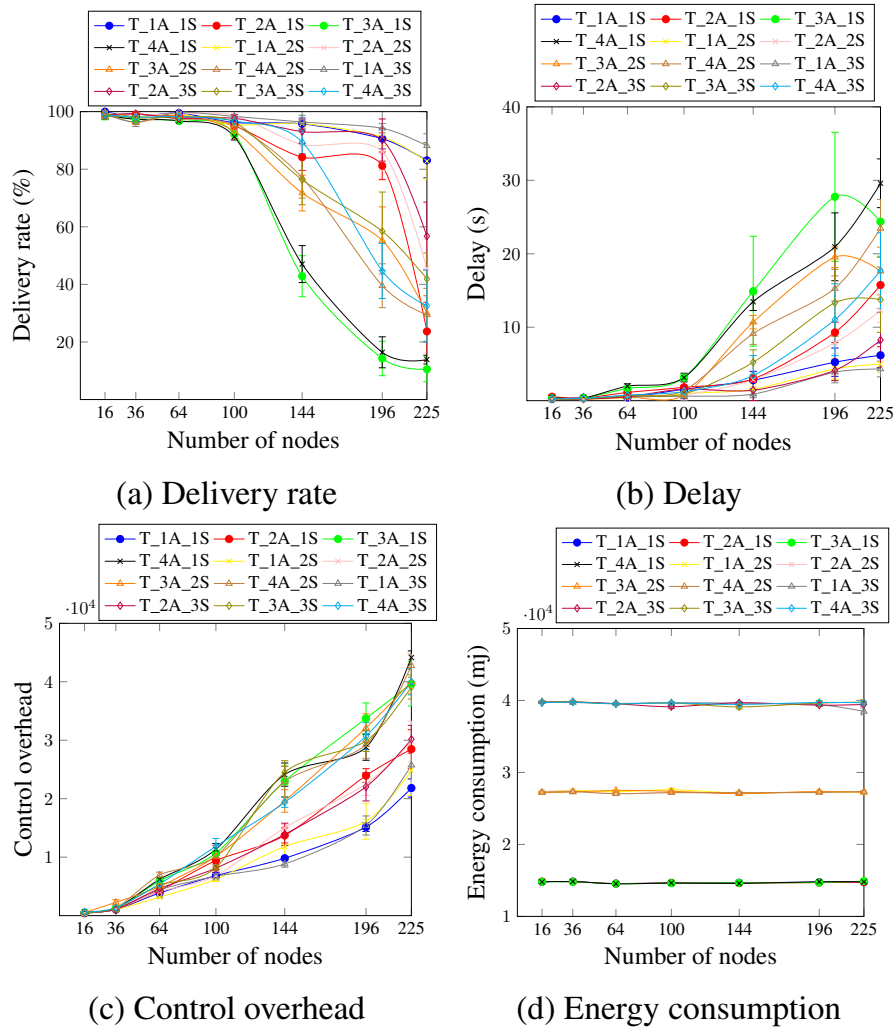


Figure 15 – CDTI approach's evaluation in the data plane

Concerning the control plane, Figure 16a shows that the delivery rate of the control messages is reduced for the increasing network sizes, regardless of the number of applications and the number of active timeslots. This delivery rate increases when the CDTI approach is applied. For a network size of 196 nodes and three applications, the control delivery rate was 33.18% in the case of one active timeslot, compared to 38.39% in the case of two active timeslots. This occurred since the CDTI approach reduces the potential congestion and collision between control and data messages. It is important to note that the obtained values for the cases of two and three active timeslots are similar since the control messages do not highly affect by the number of timeslots assigned for the data messages. Figure 16b shows that the CDTI approach decreased the control delay values. For a network size of 225 nodes and 4 applications, the delay value was 3.34 s in the case of one active timeslot, compared to 2.71 s in the case of two active timeslots. This occurs since the CDTI approach eased the congestion by isolating control messages from data ones. Again, the delay of control messages did not highly affect by the number of active timeslots assigned for the data messages.

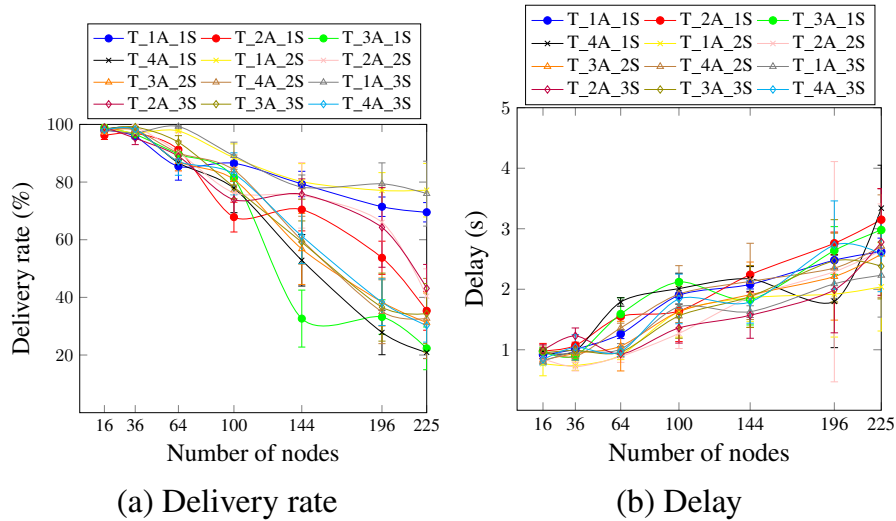


Figure 16 – CDTI approach's evaluation in the control plane

These results show that the proposed CDTI approach improved the delivery rate and delay in the application layer. Furthermore, assigning more timeslots for the data messages also improved the performance in terms of the data delivery rate and data delay.

4.3 Application's traffic isolation

Since the CDTI approach improved the performance in the application layer, this motivated us to extend the isolation concept to include all the traffic types. In other words, we argue that the traffic of each application should be considered as independent traffic and isolated from the other traffic types. For this purpose, we proposed the Application Traffic Isolation (ATI) approach, which isolates the application's traffic through network slices using TSCH technology. ATI is a scheduling approach that assigns a single timeslot per application traffic and adopts the concept of control and data traffic isolation (SAYJARI; SILVEIRA; MARGI, 2021). Figure 17 shows the ATI approach, where the default slotframe size is selected to be three timeslots, and is extended for the increasing number of applications in order to accommodate the additional applications.

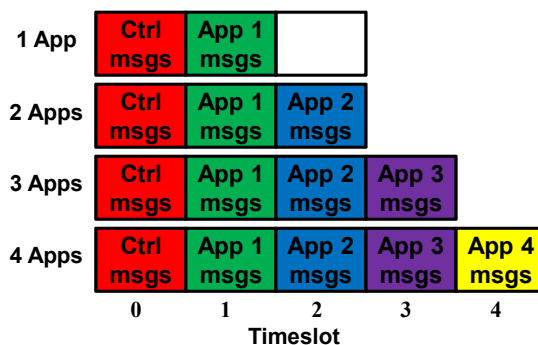


Figure 17 – ATI approach

The ATI approach changes the default node transmission operation mode, implementing the search feature and looking past the first position of the transmission buffer. This search feature looks at the whole buffer to pick up the message associated with the current timeslot and then sends it. The ATI search feature uses the first byte of the IT-SDN header to determine the message type (control or data), and in the case of the data message, the targeted sink address is used to determine the application. Figure 18 shows the search procedure for timeslot 2 considering a buffer size of N elements. This feature reduces the delay and increases the delivery rate by reducing the probability of buffer fullness. Moreover, the ATI approach transmits control messages in all the timeslots during the convergence period.

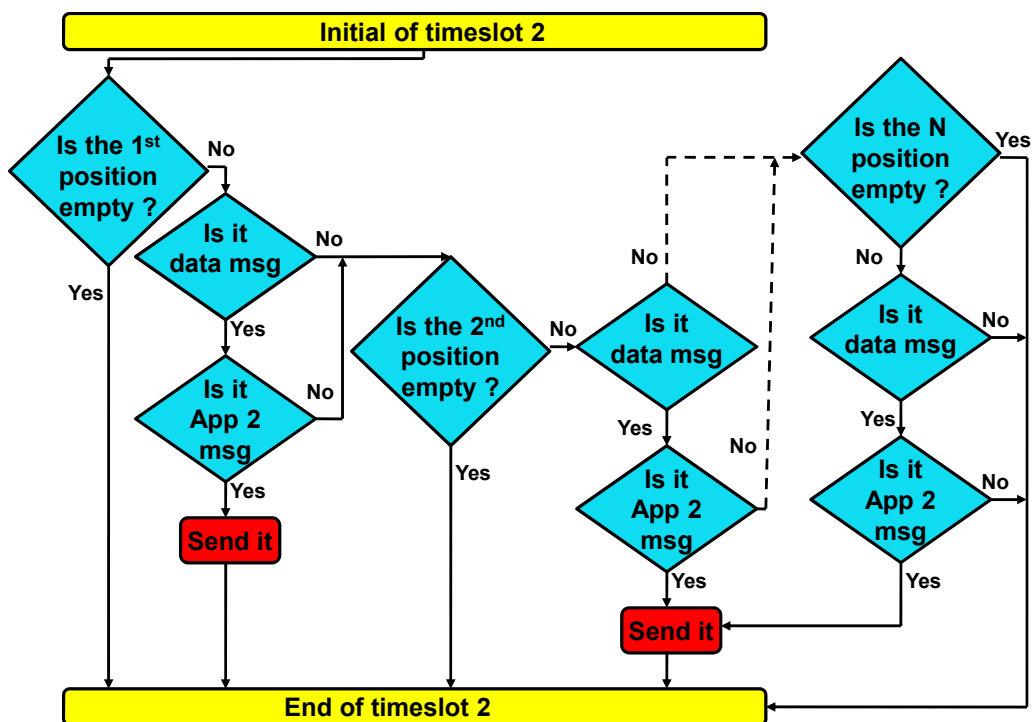


Figure 18 – ATI's search feature

The ATI approach's evaluation was carried out considering the CDTI approach as the reference case for comparison. The adopted comparison scenario was selected to be that is depicted in Figure 14b. We consider both control and data planes and adopt the simulation settings and evaluation metrics presented in both sections 4.1 and 4.2. The sole difference here is considering scenarios with 1, 2, and 4 applications (sinks) with DTR values of 1, 4, and 10 minutes, respectively. Besides the average values, and since the ATI approach isolates each application traffic from the other traffic types, we evaluate the performance of each application individually. In other words, we present and analyze the data delivery rate and data delay for each application in the case of four running applications.

Figure 19 depicts the data delivery rate for up to 4 running applications. In the

case of a single application, shown in Figure 19a, the CDTI approach performs better than the ATI approach. This occurs since the ATI approach sends data messages in only one timeslot compared to two timeslots in the CDTI approach. For all the other cases, the ATI approach presents higher values, as expected. In the case of two applications (Figure 19b), in both approaches two timeslots are assigned for the data messages, and the difference between the two approaches occurs since the search feature is enabled in the case of the ATI approach. For four applications, Figures 19c and 19d show the average and per application values, respectively. Both figures confirm that for more applications, the difference between the ATI and CDTI approaches increases. This could be explained by two reasons: i) the number of active timeslots assigned for the data messages in the cases of the ATI and CDTI approaches, and ii) the importance of the search feature becomes higher for more running applications since more applications mean more data messages in the node's buffer. The search feature of the ATI approach improves the delivery rate in this case since it reduces the probability of packet drop, which results from the buffer fullness.

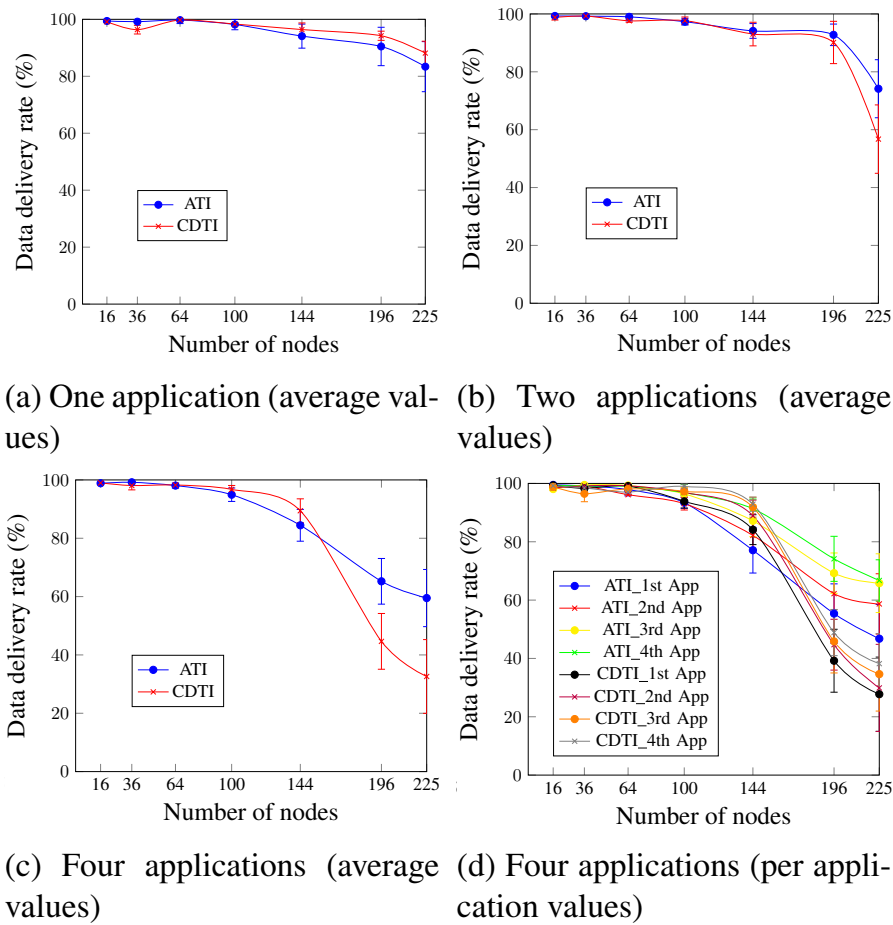


Figure 19 – ATI approach's evaluation: data delivery rate

Concerning the data delay, Figure 20a shows that in the case of a single application, CDTI's delay values are lower (better) than those of the ATI approach because of the

number of timeslots assigned for data messages in each approach. However, for two running applications as Figure 20b shows, the search feature led to better delay values in the case of ATI approach. For four applications, the average (Figure 20c) and per application (Figure 20d) values show that ATI's delay continues to be better than that of CDTI approach. This mainly occurs, as we mentioned before, because of the number of timeslots assigned for the data messages in each approach.

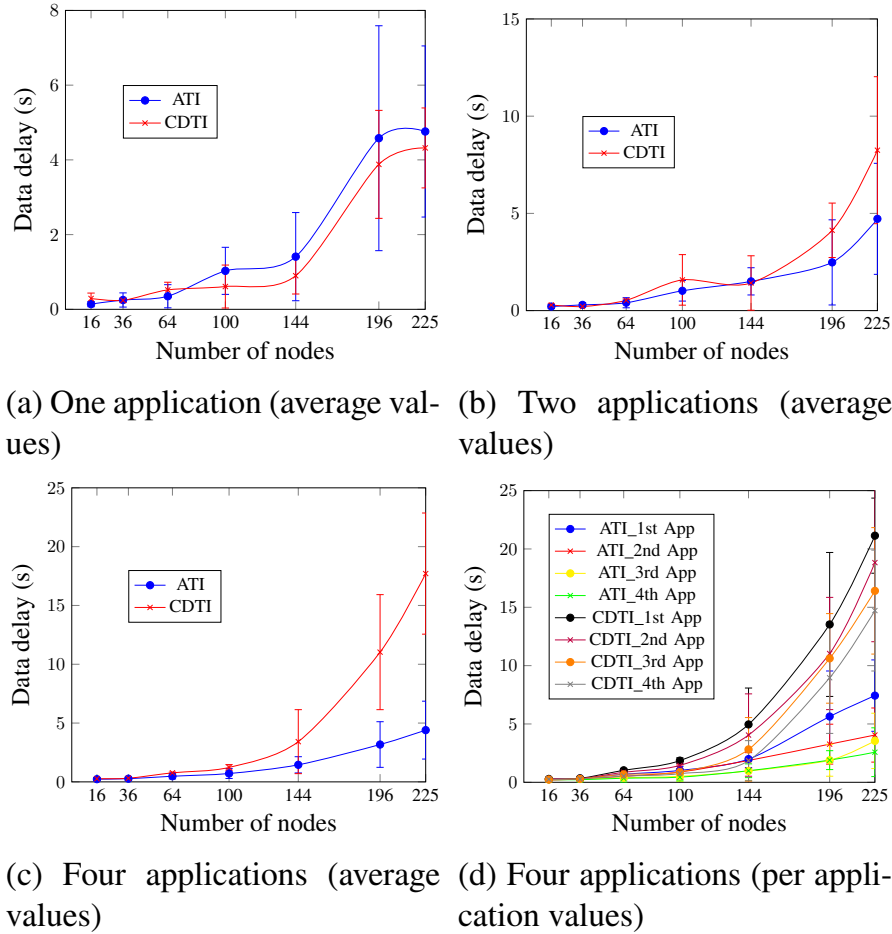


Figure 20 – ATI approach's evaluation: data delay

Figure 21 shows the average control overhead values considering up to four applications. For all cases, these values are directly proportional to the network size. This occurs since more nodes mean more exchanged control messages. For the same network size and the number of applications, ATI and CDTI approaches present similar control overhead values. This occurs since the ATI approach changes the scheduling for the data messages, without adding any new type of control messages.

Figure 22 presents the average energy consumption values for both ATI and CDTI approaches. To show the effect of the increasing number of applications on energy consumption, we adopted the 1 application case of both approaches (ATI: 1 APP and CDTI: 1 APP) as reference cases for comparison. In this way, the case entitled 'ATI: 2

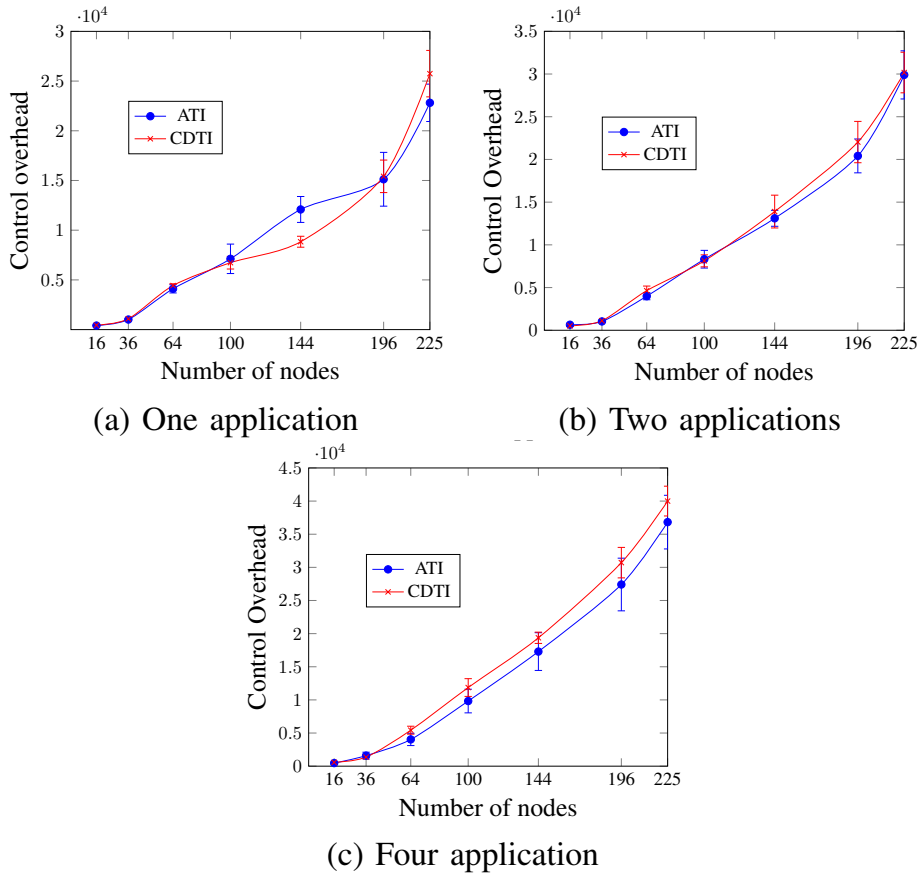


Figure 21 – ATI approach's evaluation: control overhead

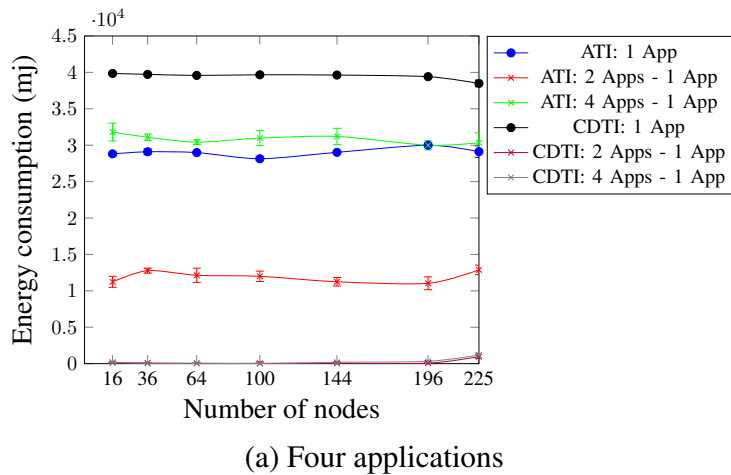


Figure 22 – ATI approach's evaluation: energy consumption

Apps - 1 App' means that in the case of the ATI approach, considering two applications led to more energy consumption of approximately 12000 mj, compared to the case of 1 application. For a single application, the CDTI approach consumes more energy than the ATI approach. This occurs since only one timeslot is assigned for data messages in the case of the ATI approach, compared to two timeslots in the case of the CDTI approach. For two running applications, both approaches present similar values regardless of the network size. This could be justified by the number of timeslots assigned for the data

messages, which is two for both ATI and CDTI approaches. For four applications, the ATI approach consumes a higher quantity of energy in comparison to the CDTI approach. This occurs, again, because of the number of timeslots assigned for the data messages in both approaches. It is important to note that more running applications led to a higher difference between ATI and CDTI approaches in terms of energy consumption since more applications require more timeslots to be accommodated. Since the cases 'CDTI: 2 Apps - 1 App' and 'CDTI: 4 Apps - 1 App' stay at approximately zero, this shows that the increasing number of applications did not affect the energy consumption. This occurs since the CDTI approach uses a fixed number of active timeslots regardless of the number of applications.

The control delivery rate for an increasing number of applications is shown in Figures 23a, 23b, and 23c. For all cases, the ATI approach outperforms the CDTI approach. This is justified by the number of timeslots that control messages are allowed to pass through for each of both approaches. In the case of the CDTI approach, the control messages are allowed to pass during a single timeslot (33.33% of time), compared to all timeslots (100% of time) for the ATI approach. Concerning the ATI approach, and for the same network size, the control delivery rate presents similar values regardless of the number of applications. Taking into account that most control messages are treated during the convergence time, this occurs since for all slotframe sizes (which changes in the ATI approach), control messages are allowed to pass during 100% of time. Since some of the control messages are sent and received after the convergence time, this led to a little difference between the control delivery rate values when more applications are considered. For a network size of 225 nodes, the control delivery rate value was 83.91% for a single application, compared to 80.15% for 4 applications.

Control delay values for up to 225 nodes and 4 applications are shown in Figures 23d, 23e, 23f. The ATI approach outperforms the CDTI approach for all cases regardless of the number of applications. This mainly occurs since control messages are allowed to pass during all the timeslots on the convergence time in the case of the ATI approach. Moreover, this approach activates the search feature, which searches for the control messages after the convergence time. This reduces the time required to send the control messages during its associated timeslot. For two applications (Figure 23e) and a network size of 144 nodes, the control delay value was 1.22 s and 1.57 s for ATI and CDTI approaches, respectively.

We could say that the previous results showed that, except in the case of a single application, the proposed approach led to a higher delivery rate and lower delay in the application layer. However, it caused more energy consumption because of the additional timeslots required to accommodate the considered applications.

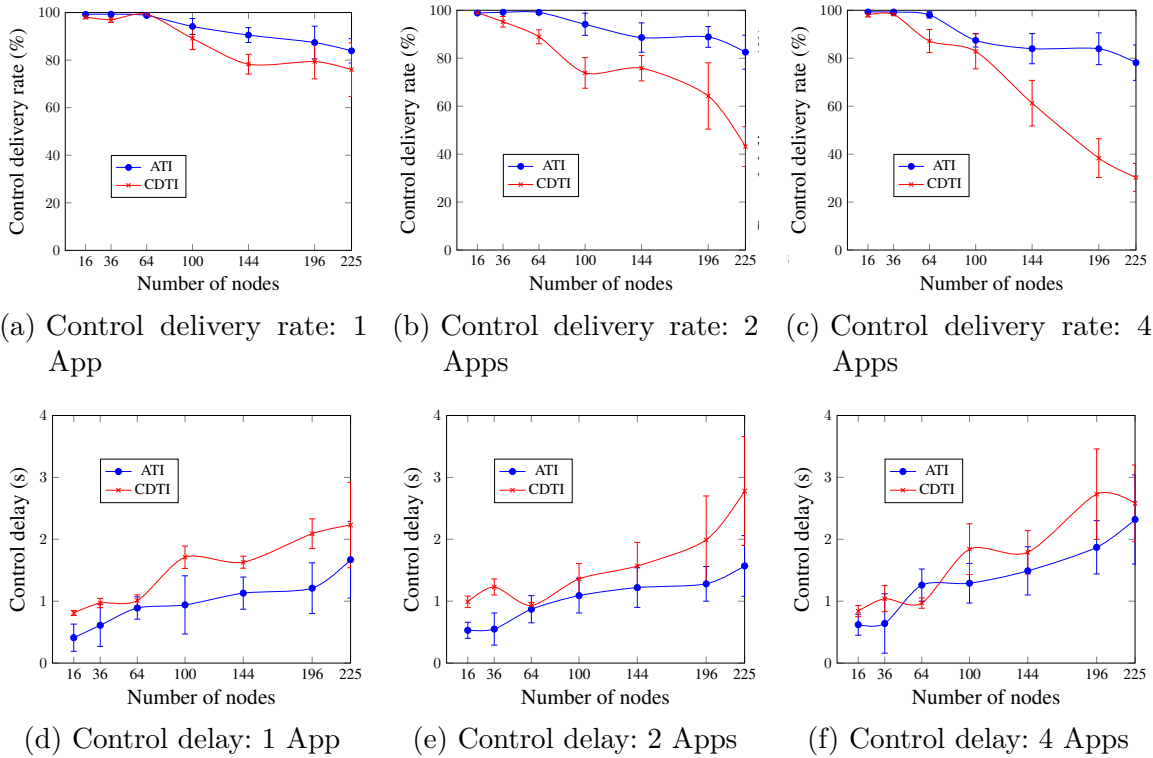


Figure 23 – ATI approach's evaluation: control plane

4.4 Chapter summary

In this chapter, the concept of traffic isolation in the SDWSNs is thoroughly discussed and experimentally evaluated. First of all, we determined the convergence time for IT-SDN based TSCH and evaluated TSCH in comparison to ContikiMAC. Next, two isolation approaches were proposed and evaluated: i) the CDTI approach, which isolates control and data traffics; and ii) the ATI approach which extended the CDTI approach to isolate the application's traffic. Both approaches improved the network performance and confirmed the importance of the traffic isolation over the SDWSNs.

Constructing on the results presented in this chapter and the gap highlighted in Section 3.4, the next step is to present our proposed approach that adopts the traffic isolation concept and meets the application's QoS requirements.

5 Application-Aware (AA) scheduling approach

This chapter details our proposal to ensure the application's QoS requirements for scalable SDWSNs. First, Section 5.1 presents the full system architecture and the considered application types. Next, Section 5.2 provides all information about the scheduling calculation procedure. The advantages of our proposed approach are presented in Section 5.3, and Section 5.4 concludes the chapter by summarizing the main points related to the proposed approach and determining the next step.

5.1 System architecture

The main contribution of this work is the Application-Aware (AA) scheduling approach, which modifies the current scheduling approach in real-time to ensure the application's QoS requirements. Figure 24 shows the system architecture, which consists of three planes: i) the infrastructure plane; ii) the control plane; and iii) the application plane. The control plane contains the IT-SDN controller, which periodically receives the traffic data from the infrastructure plane, calculates both delivery rate and delay metrics, sends them to the application plane, and disseminates the new schedule to the network nodes. Two modules have been added to the application plane: 1) the Application Manager module, which periodically receives the calculated metrics, checks if the application's QoS requirements are met, and asks the TSCH Scheduler to assign adequate scheduling; and 2) the TSCH Scheduler module, which calculates the new scheduling and sends it to the controller.

To be aware of the application's requirements, five types of control messages are exchanged among the different planes:

- Statistical message: it counts the sent/received data messages, besides recording the message's arrival time. It is periodically sent from the sensor nodes and sinks to the controller;
- Calculated metrics message: includes the calculated delivery rate and delay metrics for each of the considered applications. It is periodically sent from the controller to the Application Manager module;
- Rescheduling request message: contains the number of timeslots that should be added to/removed from each application. It is sent from the Application Manager

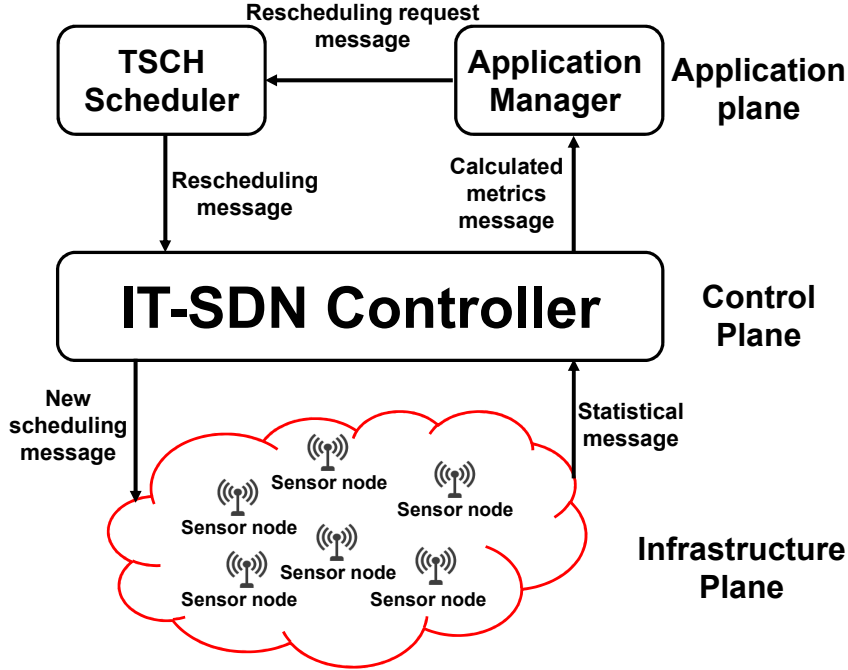


Figure 24 – Application-Aware (AA) IT-SDN system

module to the TSCH Scheduler module;

- Rescheduling message: it is sent from the TSCH Scheduler module to the controller, and contains the new scheduling;
- New scheduling message: it is sent from the controller to the network nodes and contains the new scheduling.

The application plane delivery rate and delay metrics are calculated according to the Metrics Calculation Rate (MCR) value. Each sensor node periodically sends to the controller the tuple (DS, TXt) that represents the number of the data messages sent (DSid) and the timestamp of the data message sent (TXt). Additionally, the sink periodically sends to the controller the tuple (DR, RXt) which similarly represents the number of the data messages received (DRid) and the timestamp of the data message received (RXt). The data delivery rate and delay metrics are calculated using equations 5.1 and 5.2.

$$Delivery\ rate\ (\%) = 100 * \frac{DRid}{DSid} \quad (5.1)$$

$$Delay\ (s) = \frac{\Sigma (RXt - TXt)}{DSid} \quad (5.2)$$

Since the considered applications send data messages at different Data Traffic Rate (DTR) values, the node does not send its statistical message about an application if the number of transmitted messages for this application is zero. To explain this, suppose that the DTR value of Application 2 is 10 minutes. This means that the first data message

for this application is transmitted after 10 minutes of the run-time beginning. During these 10 minutes, the nodes do not send statistical messages about Application 2 to the controller. The calculated metrics are sent to the Application Manager module to verify if the application's QoS requirements are satisfied. If so, the current scheduling continues to be adopted. Otherwise, the TSCH scheduler module assigns a new scheduling and sends it to the controller, which in turn, disseminates it to the network nodes. Figure 25 shows the sequence of all the system operations, where Sch 0 schedule is the initial scheduling and is presented in Section 5.2.

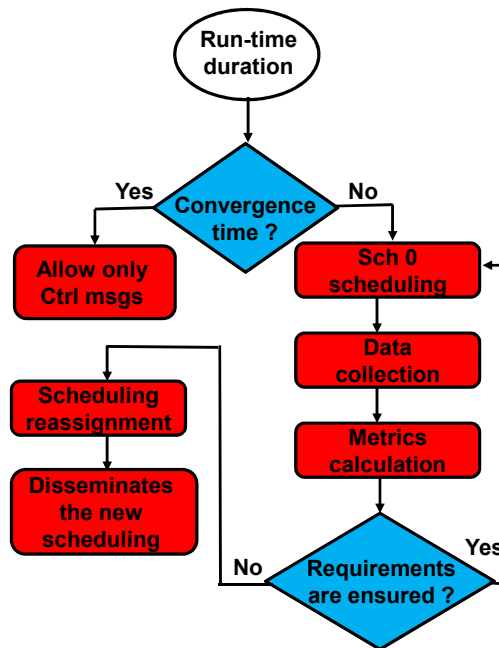


Figure 25 – Sequence of the system operations

We consider applications with four different priority levels as Table 7 shows. The highest priority is assigned to the first application type (Type 1), which has the hardest requirements in terms of delivery rate and delay. The lowest priority is assigned, on the other hand, to Type 4, which has no requirements. The applications are classified according to their delivery rate and delay requirements.

Table 7 – Priority levels of the considered application types

Application type	Application's QoS requirements		
	Delivery rate	Delay	
Type 1	✓	✓	Priority 1
Type 2	—	✓	Priority 2
Type 3	✓	—	Priority 3
Type 4	—	—	Without priority

5.2 Scheduling calculation

Since control and data traffic isolation and the search feature (SAYJARI; SILVEIRA; MARGI, 2022) improved the network performance, we adopt these concepts in our proposed approach. Moreover, to speed up the network convergence, the control messages are transmitted in all the timeslots during the convergence period. After this period, the data messages start to be transmitted using Sch 0 scheduling as Figure 26 shows. The first timeslot (s) is always reserved for the control messages. The slotframe size is selected to accommodate the additional applications and to prioritize the applications with harder requirements. In Figure 26b, where two applications are considered, nine timeslots (56.25% of the slotframe size) are assigned to Application 1, compared to six timeslots (37.5% of the slotframe size) to Application 2.

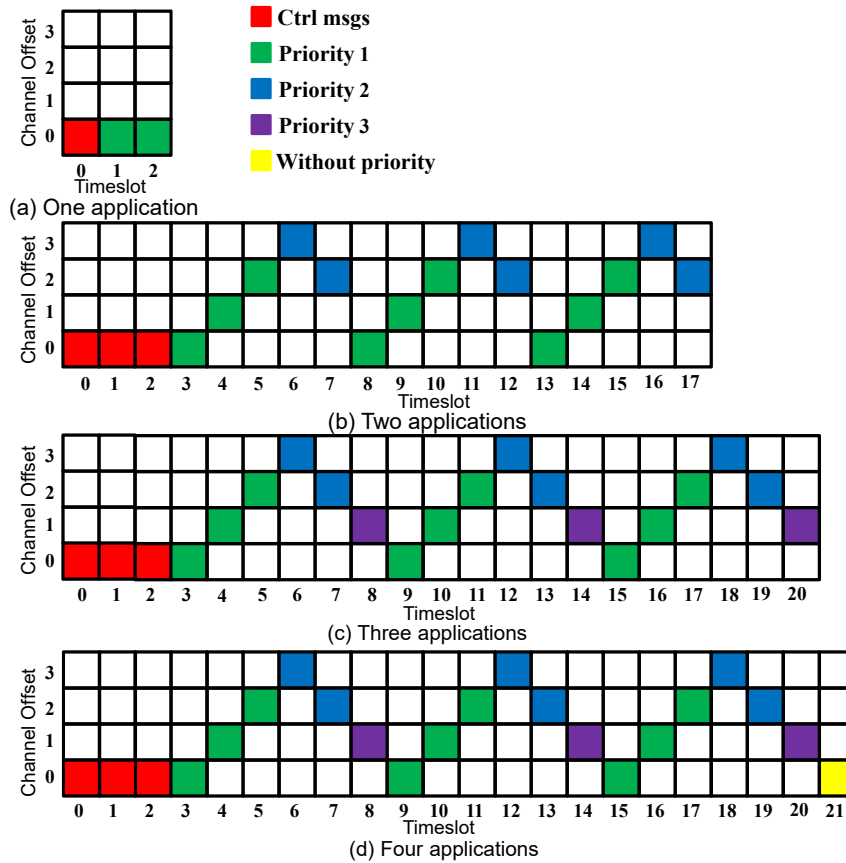


Figure 26 – Sch 0 scheduling

Concerning the new scheduling calculation, more (or fewer) resources are assigned to ensure the application's QoS requirements. The proposed approach adds timeslots to the end of the slotframe depending on the calculated metrics values, as follows:

- The calculated metric is (0 – 20)% worse than the application's requirement; a single timeslot is added to the end of the slotframe;

- The calculated metric is $(20 - 40)\%$ worse than the application's requirement; two timeslots are added to the end of the slotframe;
- The calculated metric is $(40 - 60)\%$ worse than the application's requirement; three timeslots are added to the end of the slotframe.

To generalize the rule of adding more timeslots, we could say that if the calculated metric is from $N\%$ to $(N+20)\%$ worse than the application's requirement, X timeslots are added to the end of the slotframe. This leads to saying that if the calculated metric is from $(N+20)\%$ to $(N+40)\%$ worse than the application's requirement, $(X+1)$ timeslots are added to the end of the slotframe. Similarly, when the calculated metric is better than the application's requirement, the proposed approach removes timeslots from the end of the slotframe as follows:

- The calculated metric is $(0 - 20)\%$ better than the application's requirement; no timeslots are removed;
- The calculated metric is $(20 - 40)\%$ better than the application's requirement; a single timeslot is removed from the end of the slotframe;
- The calculated metric is $(40 - 60)\%$ better than the application's requirement; two timeslots are removed from the end of the slotframe.

As a general rule, for $N \geq 20$, if the calculated metric is from $N\%$ to $(N+20)\%$ better than the application's requirement, X timeslots are removed from the end of the slotframe. This leads to saying that if the calculated metric is from $(N+20)\%$ to $(N+40)\%$ better than the application's requirement, $(X+1)$ timeslots are removed from the end of the slotframe. Figure 27 shows the scheduling calculation procedure considering that MV is the metric value, AR is the application's requirement value and TS represents the timeslot. Assigning multiple timeslots per application at the same time (if needed) could help to reduce the time required to ensure the application's requirements. Removing multiple timeslots (if needed), on the other hand, could help to save energy and increase the network lifetime.

If both requirements of an application are not satisfied, the number of added timeslots will depend on the highest number of required timeslots for each requirement. To explain this, suppose that during a metric calculation cycle, the obtained delivery rate of Application 1 was 10% lower than the delivery rate requirement, and the obtained delay was 30% higher than the delay requirement. The number of added timeslots for Application 1 would, thus, be 2 timeslots. Similarly, if the obtained values of both delivery rate and delay metrics are better than the requirements, the number of removed timeslots will depend on the lowest number of required timeslots for each requirement. For instance, suppose that during a metric calculation cycle, the obtained delivery rate of Application

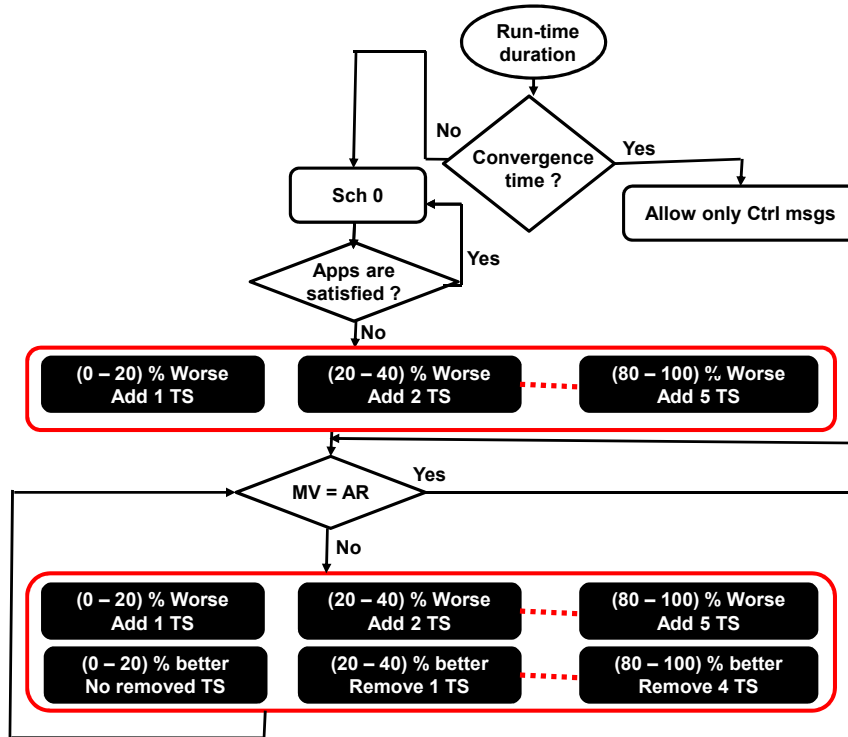


Figure 27 – Scheduling calculation procedure

1 was 10% higher than the delivery rate requirement, and the obtained delay was 30% lower than the delay requirement. The number of removed timeslots for Application 1 would be 1. To clarify how the new scheduling is constructed, suppose that during a metric calculation cycle, multiple applications were not satisfied; the timeslots would then be added depending on the application's priorities. Suppose that in the case of 3 running applications, Applications 1 and 2 were not satisfied, and two more timeslots should be assigned to Application 1 and one more timeslot to Application 2. Then, the sch 0 scheduling depicted in Figure 26c turns into the scheduling depicted in Figure 28, where firstly, two more timeslots were assigned to Application 1 (which has the highest priority), and then one more timeslot was assigned to Application 2.

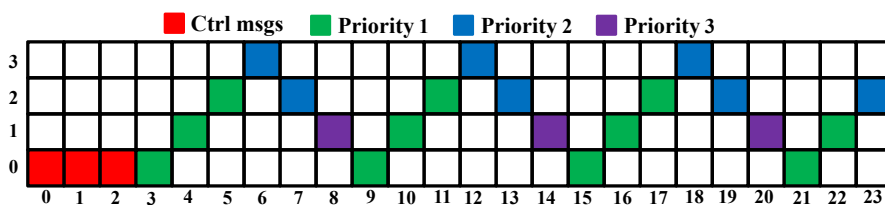


Figure 28 – The new calculated scheduling (three applications case)

5.3 AA approach's advantages

Our approach is the first one to dynamically adapt to the application's QoS requirements for scalable SDWSNs, without any additional hardware. Moreover, the Sch

0 schedule, which is a part of the AA approach, considers the application's priorities, which could delay the need of adding more timeslots. This, in turn, could save both energy and the time required for rescheduling. The AA approach reduces the time required to ensure the application's QoS requirements since the number of added timeslots depends on the difference between the calculated metrics and the application's requirements. Our approach could also save energy since it contains a mechanism to dynamically remove active timeslot (s) when possible. Finally, this approach assigns more shared timeslots per application if needed, without any dedicated timeslots, in which it is difficult to support large networks and more energy is consumed.

5.4 Chapter summary

This chapter was dedicated to present and discuss the Application-Aware (AA) scheduling approach, which is the core contribution of this work. The AA approach is the first to meet the application's QoS requirements for scalable SDWSNs, without any additional hardware. All details about the system architecture, metrics calculation, exchanged messages, and new scheduling calculation were provided. The next step is to evaluate the proposed approach to investigate its efficiency.

6 Performance evaluation

In this chapter, we thoroughly evaluate the AA approach. Section 6.1 presents the experimental method, including the simulation settings, adopted scenarios, and evaluation metrics. Next, we show and analyze the obtained results in Section 6.2 varying several important parameters. Section 6.3 presents a detailed case study where our system could be applied, and Section 6.4 concludes the chapter by summarizing the conclusions obtained from the shown results.

6.1 Method

The experiments aimed to evaluate our proposed approach and investigate its efficiency in the scalable SDWSNs. For this purpose, we compare the AA approach in two different strategies: i) adopting the Application Traffic Isolation (ATI) (SAYJARI; SILVEIRA; MARGI, 2022) approach as a reference case for comparison, and ii) comparing the application's QoS requirements with the obtained results. For all the simulation results, we show the performance for each of the considered applications in both control and data planes. It is important to define two parameters related to the AA approach: i) metric calculation rate (MCR), which is the frequency of metric's (delivery rate and delay) calculations during the run-time, and ii) difference rate (DR), which is the percent of the difference between the calculated metrics and the application's QoS requirements. Table 8 shows the default simulation settings.

We adopt IT-SDN (ALVES et al., 2017) as the SDWSN framework, whereby the simulations were carried out using Contiki OS (DUNKELS; GRONVALL; VOIGT, 2004) and COOJA simulator (ÖSTERLIND et al., 2006a). Each application is represented by a set of sensor nodes that periodically send data messages to a single sink. To execute up to 4 applications with different DTR values, each sensor node periodically sends a data message to all the sinks. Hence, each sensor executes up to 4 applications simultaneously. Both the IT-SDN controller and the TSCH coordinator (which controls join and departure operations within the TSCH network) run on the same node. Figure 12 shows the locations of the sinks and the controller. We consider up to 4 application types with different DTR values, and network sizes of up to 225 nodes. Each experiment is repeated 10 times, and its duration is set to be 1 hour. The shown simulation results are the average of these repetitions.

To compare the ATI approach, which assigns a single timeslot per application, with the AA approach, we consider the case of 4 applications running simultaneously. Figure 29 depicts the simulation scenario for each approach.

Table 8 – Default simulation settings

Topology	Square grid
Distance between neighbors	50 m
Compiling mote	Z1
Radio environment	UDGM
Simulation duration	3600 s
Simulation repetition	10 times for each case
Radio module power	0 dB
ContikiMAC channel check rate	16 Hz
IT-SDN version	0.4.1
Controller re-transmission timeout	2 s
ND protocol	Collect-based
CD protocol	None
Link metric	Expected Transmission Count (ETX)
Route recalculation threshold	20%
Size of the flow table	10 entries
Neighbor report max frequency	1 packet per minute
Route calculation algorithm	Dijkstra
Flow setup	Source routed
Data payload size	10 bytes
Number of TSCH channels	4 channels
Timeslot length	15 ms
Enhanced Beacon (EB) transmission rate	2 s
Number of nodes	16,36,64,100,144,196,225
Number of applications (sinks)	1,2,3,4
Data traffic rate	1 packet per 1/4/8/10 minutes
Metric calculation rate	180 s
Difference rate	20%
Application's requirements (1st App, 2nd App, 3rd App, 4th App)	
Delivery rate	92%, —, 90%, —
Delay	900 ms, 950 ms, —, —
Number of nodes, convergence time (s)	(16,73) (36,96)(64,117) (100,171) (144,196) (196,233) (225,329)

To investigate the ability of the proposed approach to ensure the application's QoS requirements, we compare the obtained results with the application's requirements. Twelve different scenarios have been considered for this evaluation strategy, varying each of MCR, DTR, AR, DR, and the adopted topology. Table 9 shows the parameter values for these scenarios, indicating the default values (shown in Table 8) with "DV".

Performance is evaluated using the following metrics:

- Data delivery rate: it is the total number of received data messages divided by the total number of sent data messages;

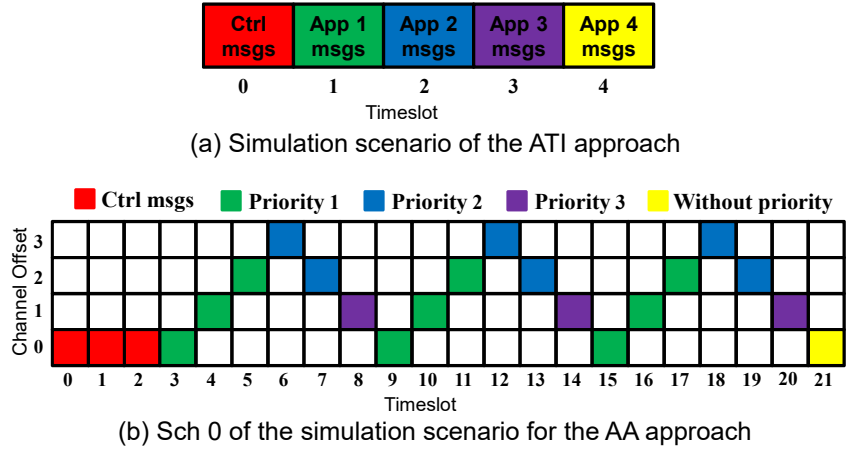


Figure 29 – Simulation scenarios for ATI and AA approaches

Table 9 – Parameter's values for the considered scenarios

Scenario	MCR	DTR	AR	DR	Topology
Scen 1	180 s	1/4/8/10 min	Delivery rate: 92%, –, 90%, – Delay: 900 ms, 950 ms, –, –	20%	Grid
Scen 2	60 s	1/4/8/10 min	Delivery rate: 92%, –, 90%, – Delay: 900 ms, 950 ms, –, –	20%	Grid
Scen 3	300 s	1/4/8/10 min	Delivery rate: 92%, –, 90%, – Delay: 900 ms, 950 ms, –, –	20%	Grid
Scen 4	480 s	1/4/8/10 min	Delivery rate: 92%, –, 90%, – Delay: 900 ms, 950 ms, –, –	20%	Grid
Scen 5	180 s	1/1/1/1 min	Delivery rate: 92%, –, 90%, – Delay: 900 ms, 950 ms, –, –	20%	Grid
Scen 6	180 s	1/4/8/10 s	Delivery rate: 92%, –, 90%, – Delay: 900 ms, 950 ms, –, –	20%	Grid
Scen 7	180 s	1/4/8/10 min	Delivery rate: 95%, –, 90%, – Delay: 450 ms, 500 ms, –, –	20%	Grid
Scen 8	180 s	1/4/8/10 min	Delivery rate: 85%, –, 80%, – Delay: 900 ms, 950 ms, –, –	20%	Grid
Scen 9	180 s	1/4/8/10 min	Delivery rate: 92%, –, 90%, – Delay: 900 ms, 950 ms, –, –	10%	Grid
Scen 10	180 s	1/4/8/10 min	Delivery rate: 92%, –, 90%, – Delay: 900 ms, 950 ms, –, –	30%	Grid
Scen 11	180 s	1/4/8/10 min	Delivery rate: 92%, –, 90%, – Delay: 900 ms, 950 ms, –, –	40%	Grid
Scen 12	180 s	1/4/8/10 min	Delivery rate: 92%, –, 90%, – Delay: 900 ms, 950 ms, –, –	20%	Random

- Data delay: it is the average time a data message takes to reach its destination;
- Control overhead: it includes IT-SDN control messages (Flow request, Flow setup, Source routed flow setup, Acknowledgement, Neighbor discovery, Controller discovery, and Neighbor report messages), in addition to the AA approach control messages (statistical messages, calculated metrics messages, rescheduling request messages, rescheduling response messages, and new scheduling messages);
- Energy consumption, which is represented by the average energy consumed by the

node;

- Control delivery rate: it is the total number of received control messages divided by the total number of sent control messages;
- Control delay: it is the average time a control message takes to reach its destination.

6.2 AA approach's evaluation

This section presents and discusses the obtained results to evaluate the AA approach. The evaluation process is realized by comparing the AA approach with both ATI approach and the application's QoS requirements as Sections 6.2.1 and 6.2.2 show.

6.2.1 Comparison with the ATI approach

We here compare our proposed approach with the ATI approach. Figure 30 depicts the performance of both AA and ATI approaches in the data plane for four applications running simultaneously.

Concerning the delivery rate and delay, Figures 30a and 30b show that for the first three applications (1st application, 2nd application, and 3rd application), the AA approach outperforms the ATI approach. For a network size of 196 nodes and in the case of the 2nd application, the delivery rate and delay of the AA approach are 90.84% and 1.4 s compared to 62.2% and 3.27 s, respectively, for the ATI approach. This occurs since the AA approach assigns priorities and dynamically adapts to the application's requirements. For the 4th application, the ATI approach outperformed the AA approach in terms of both delivery rate and delay since in the case of the AA approach, the 4th application has no priority and is accommodated using only one timeslot as Figure 29b shows.

Figure 30c shows that the control overhead of the AA approach is higher than that of the ATI approach regardless of the number of nodes since the AA approach uses additional control message types to ensure the application's QoS requirements. For a network of 64 nodes, the control overhead value is 10927.1 messages for the AA approach compared to 4036.7 messages for the ATI approach. The AA approach consumed more energy than the ATI approach as Figure 30d shows. For networks of 100 and 144 nodes, the AA approach increased the energy consumption by 14.3% and 16.27%, respectively, in comparison to the ATI approach. This mainly occurs due to the exchanged control messages. Since the ATI approach does not add any control messages, then the number of exchanged control messages after the network convergence is relatively low. This means that the nodes sleep for longer periods during the control timeslots. The nodes in the case of the AA approach, in turn, continue to exchange several control message types after the network convergence and should, thus, stay awake longer.

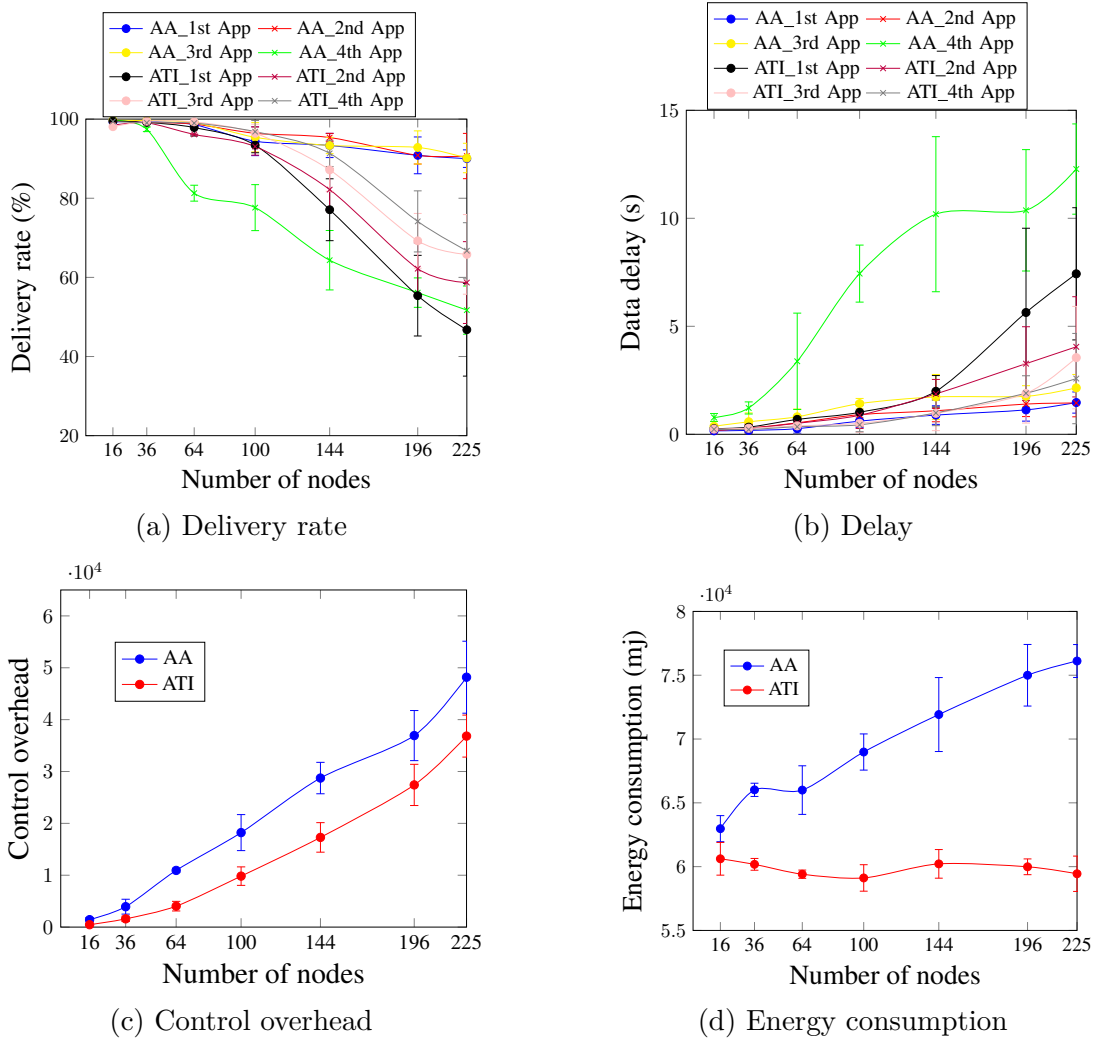


Figure 30 – Comparison between AA and ATI approaches in the data plane (four applications case)

Concerning the control plane, Figure 31 depicts the control delivery rate and control delay for the case of 4 running applications. Figures 31a and 31b show that, in general, both approaches present similar values for the increasing number of nodes, and the ATI approach performs a little better than the AA approach. For a network size of 144 nodes, the ATI approach increased the control delivery rate by 4.65% and decreased the control delay by 7.45% in comparison to the AA approach. This could be justified by the number of exchanged control messages after the network convergence for both approaches. This number is higher for the AA approach, which increases the probability of packet drop due to the buffer fullness and increases the time that the message waits to be processed.

Table 10 summarizes the comparison between the AA and ATI approaches, highlighting the outperformed approach for each of the evaluation metrics.

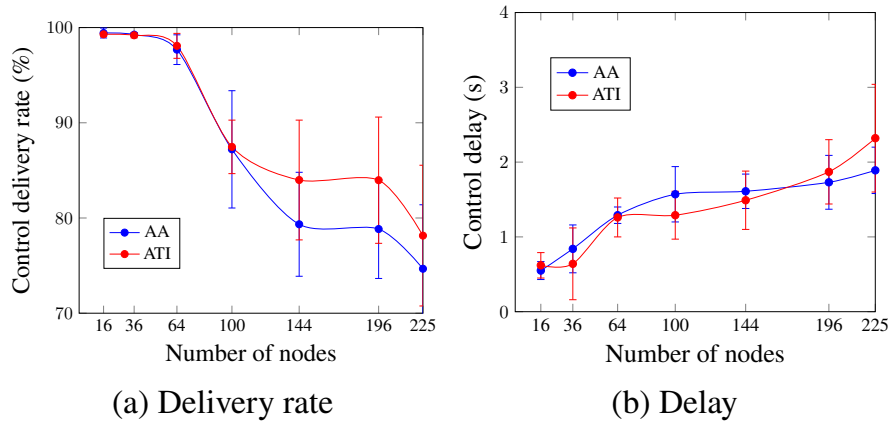


Figure 31 – Comparison between AA and ATI approaches in the control plane (four applications' case)

Table 10 – AA approach versus ATI approach

Evaluation metric	AA Vs ATI	
Data delivery rate	✓	AA outperformed
		Similar
		ATI outperformed
Data delay	✓	AA outperformed
		Similar
		ATI outperformed
Control overhead		AA outperformed
		Similar
	✓	ATI outperformed
Energy consumption		AA outperformed
		Similar
	✓	ATI outperformed
Control delivery rate		AA outperformed
	✓	Similar
		ATI outperformed
Control delay		AA outperformed
	✓	Similar
		ATI outperformed

6.2.2 Comparison with the application's QoS requirements

This section evaluates the AA approach in comparison to the application's QoS requirements in terms of the data delivery rate and data delay. We investigate the effect of MCR, DTR, AR, DR, and the adopted topology on the network performance.

6.2.2.1 Metrics calculation rate (MCR)

Here we evaluate the effect of the MCR value, considering several scenarios. Note that the low values of MCR could speed up ensuring the application's QoS requirements.

However, this leads to high control overhead. High values of MCR, conversely, could delay ensuring the application's QoS requirements, since the AA approach needs more time to discover the unsatisfied application. However, this reduces the control overhead.

Figure 32 depicts the data delivery rate for up to 4 applications, considering 4 different scenarios (Scen 1, Scen 2, Scen 3, and Scen 4). For the case of a single application (Figure 32a), the delivery rate's requirement is ensured for most of the considered scenarios and the number of nodes, except the case of Scen 4. This occurs since the MCR value is relatively high, and the metrics are calculated every 8 minutes during the run-time.

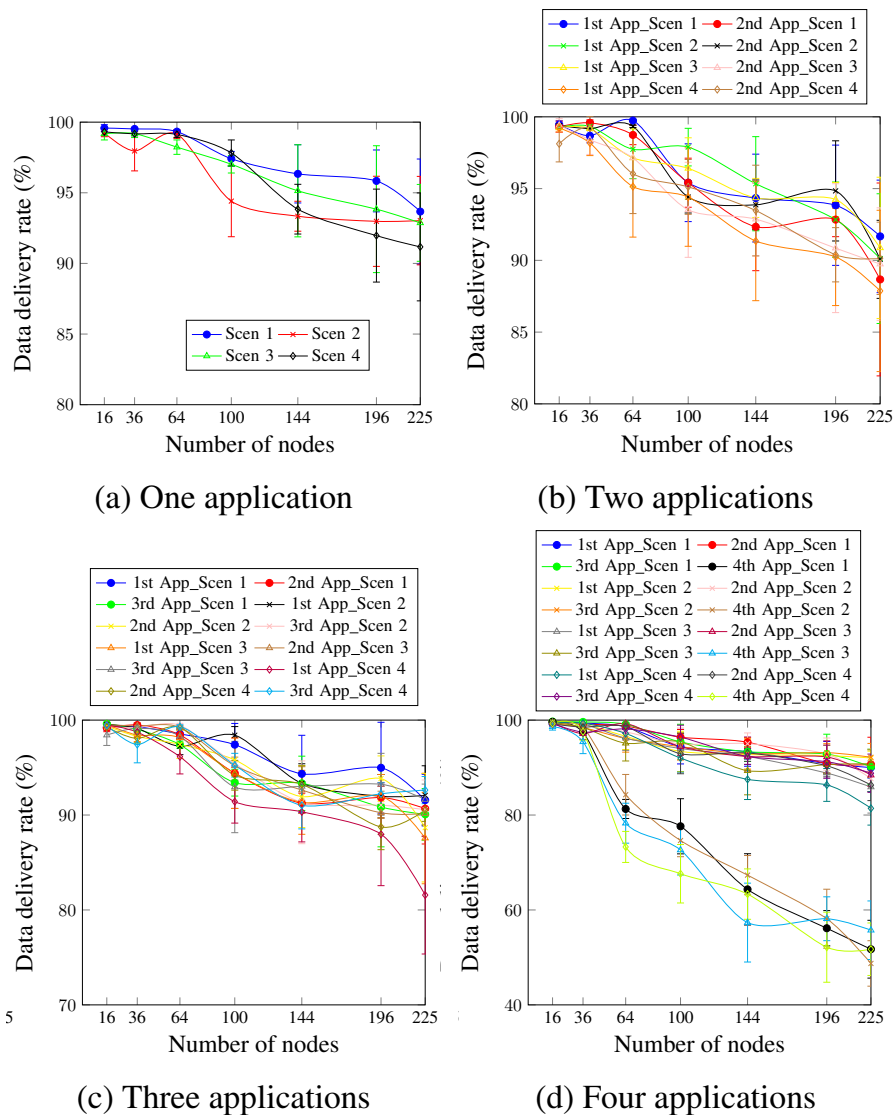


Figure 32 – Data delivery rate, changing the MCR value

Concerning two applications, Figure 32b shows that the 1st application is always satisfied for most of the considered scenarios. However, for the network size of 225 nodes, the obtained delivery rate values are a little lower than the requirement for all the considered scenarios. The 2nd application presents a delivery rate value not lower than 90% for most cases, although the 2nd application does not have a delivery rate requirement. This is

justified by the delay requirement of the 2nd application, which induces the AA approach to be active, This, in turn, improves the delivery rate.

Figure 32c depicts the case of three applications. The delivery rate requirement of the 1st application of all scenarios and for up to 100 nodes is always satisfied. For a higher number of nodes, the delivery rate is a little lower than the requirement for the rest of the cases. This is justified by the probability of packet drop resulting from the high traffic of the three applications. For the 3rd application, the delivery rate requirement is satisfied for all the numbers of nodes and scenarios. This occurs since, in addition to the efficiency of the AA approach, the data traffic rate of the 3rd application is 8 minutes. Therefore, the probability of a packet drop resulting from the buffer fullness is low.

Concerning four applications running simultaneously as Figure 32d shows, we notice that increasing the MCR value resulted in a lower delivery rate, especially for network sizes larger than 144 nodes. For the 1st application, the sole scenario capable of ensuring the delivery rate requirement for all the network sizes is Scen 2, where the MCR value is 1 minute. For the remaining cases, the AA approach was able to ensure the requirement for up to 144 network sizes. For the 3rd application case, the AA approach ensured the requirement for all cases. This mainly occurs since the DTR value of the 3rd application is low (1 data message every 8 minutes). The 4th application presents the worst values for all scenarios since this application has no priority and is accommodated using only one timeslot.

Concerning the data delay, Figure 33a shows that in the case of a single application, the delay requirement, which is 900 ms, is ensured for network sizes up to 144 nodes for all scenarios. For larger networks, the obtained delay values are a little higher than 900 ms. For a network size of 196 nodes, the delay values were 0.94 s and 0.97 s for Scen 1 and Scen 3 respectively.

Figure 33b depicts that for two applications, the delay requirement is ensured for network sizes up to 100 nodes, compared to the 144 nodes in the case of a single application. This occurs since more applications mean more data messages. This increases, in turn, the time the message waits in the buffer to be processed. Concerning the 2nd application, decreasing the MCR value did not significantly improve the obtained values. This is justified by the DTR value of the 2nd application, which is 4 minutes. This, in turn, shows an important trade-off between the selected MCR and DTR values.

Concerning the case of three applications running simultaneously depicted in Figure 33c, the delay requirement of the 1st application is ensured for network sizes of up to 100 nodes when the MCR value varies between 1 to 4 minutes. However, the AA approach ensures the delay requirement only for network sizes up to 64 nodes when MCR is higher (6 and 8 minutes). This occurs because the higher MCR values could delay ensuring the requirement. The requirement is unsatisfied for a longer period during the run-time, which

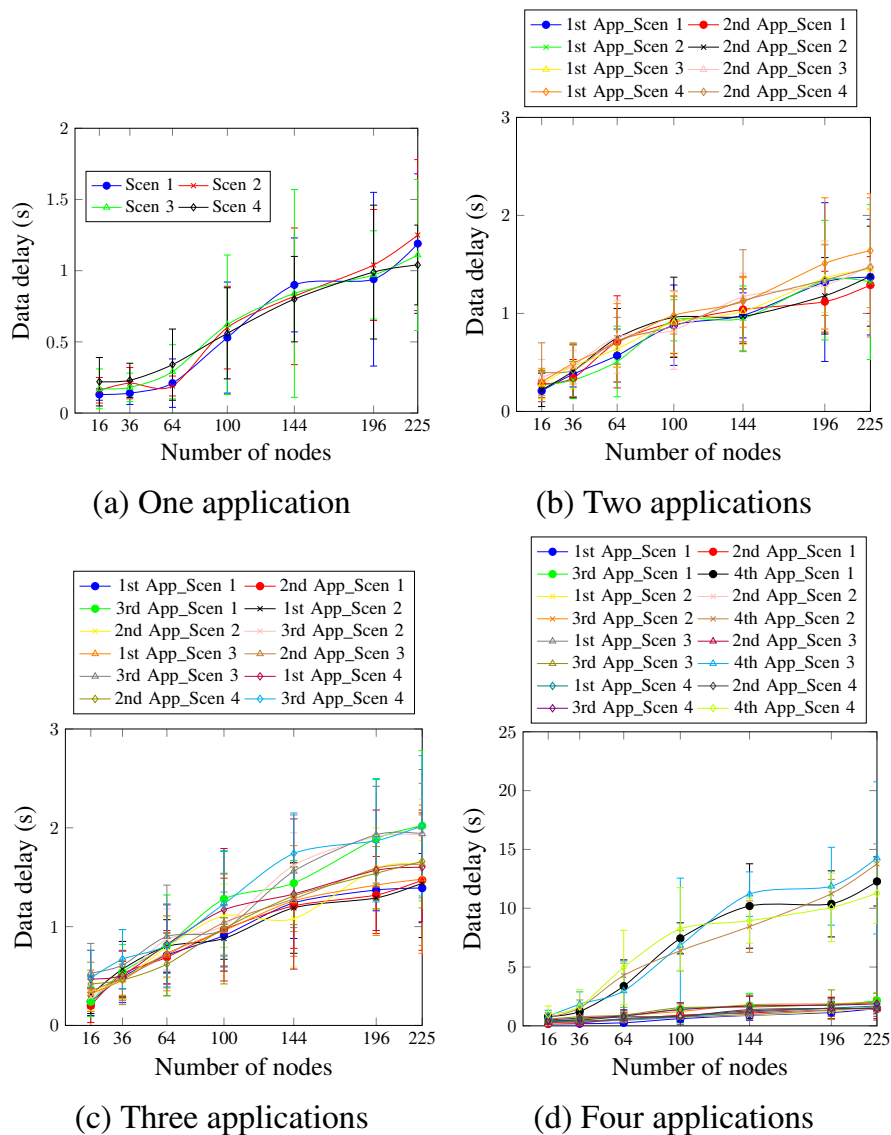


Figure 33 – Data delay, changing the MCR value

affects the final delay result. For the 3rd application, the requirement keeps ensured up to network sizes of 100 nodes for all scenarios, except for Scen 4, where the MCR value is 8 minutes.

For four applications, as depicted in Figure 33d, and for the 1st application, the delay values are lower than 900 ms for up to 144 nodes in Scen 1 and Scen 2, and up to 100 nodes in the Scen 3 and Scen 4, where the MCR values are 6 and 8 minutes, respectively. For the 2nd application, the delay requirement is ensured for network sizes up to 100 nodes for all the considered scenarios, including Scen 3 and Scen 4, where the MCR values are 6 and 8 minutes, respectively. This is justified by the DTR value of the 2nd application, which is 4 minutes. There is, thus, no difference in the number of the 2nd application's messages every 1 or 3 minutes. Scen 1 and Scen 2 did not, therefore, present better delay values than Scen 3 and Scen 4. Similarly to the case of the delivery rate, the 4th application presents the worst delay values, since it is accommodated in a single timeslot and without

any priority.

Figure 34 shows the control overhead for up to four applications. For all network sizes and scenarios, and regardless of the number of applications, the MCR value is inversely proportional to the control overhead. Scen 2, which has the lower MCR value (1 minute), presents the highest control overhead values. This occurs since the lower MCR values mean more exchanged control messages.

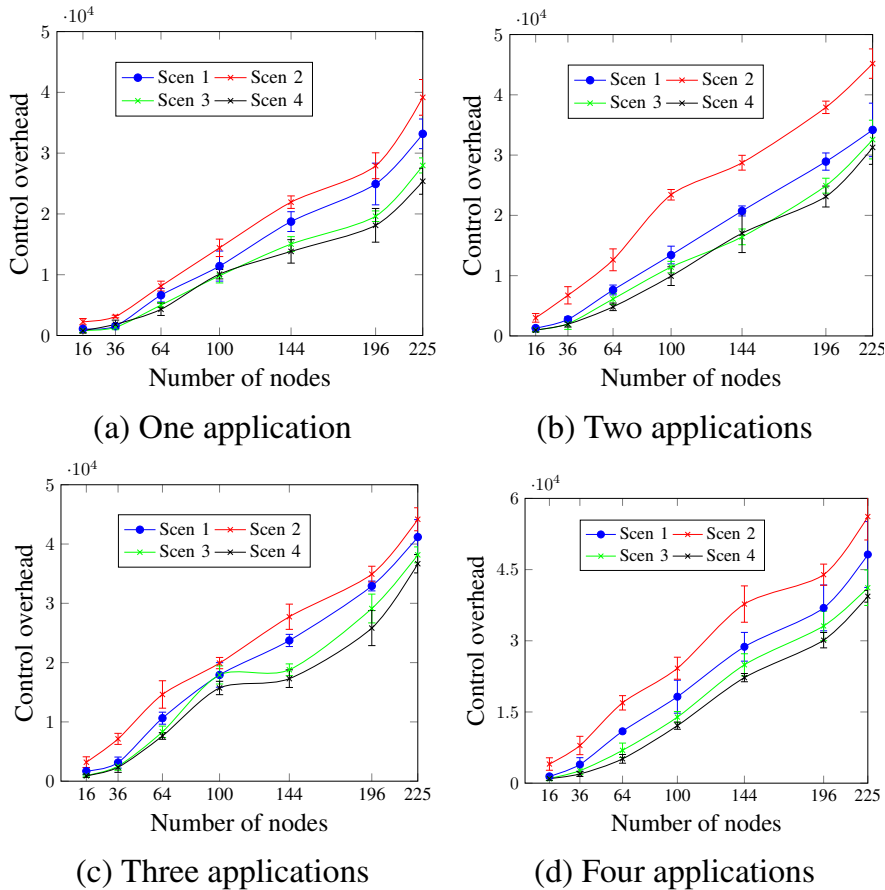


Figure 34 – Control overhead, changing the MCR value

Concerning energy consumption, Figure 35 shows that for all cases, the energy consumption increases with the number of applications. The lower values of MCR led to more energy consumption in most cases. In the case of four applications, as in Figure 35d, and for a network size of 64 nodes, the presented values were 65281.3 mj, 68423.7 mj, 64079.1 mj, and 62819.8 mj for Scen 1, Scen 2, Scen 3, and Scen 4, respectively. This occurs since lower MCR values mean more activities concerning the control messages.

Figure 36 shows that for all cases, the control delivery rate is inversely proportional to the number of nodes. Scen 4, with the highest MCR value, presents the best control delivery rate values, whereas Scen 2, with the lowest MCR value, presents the worst ones. In the case of two applications (Figure 36b), and for a network size of 196 nodes, the control delivery rate values were 88.14% and 81.84% for Scen 4 and Scen 2, respectively.

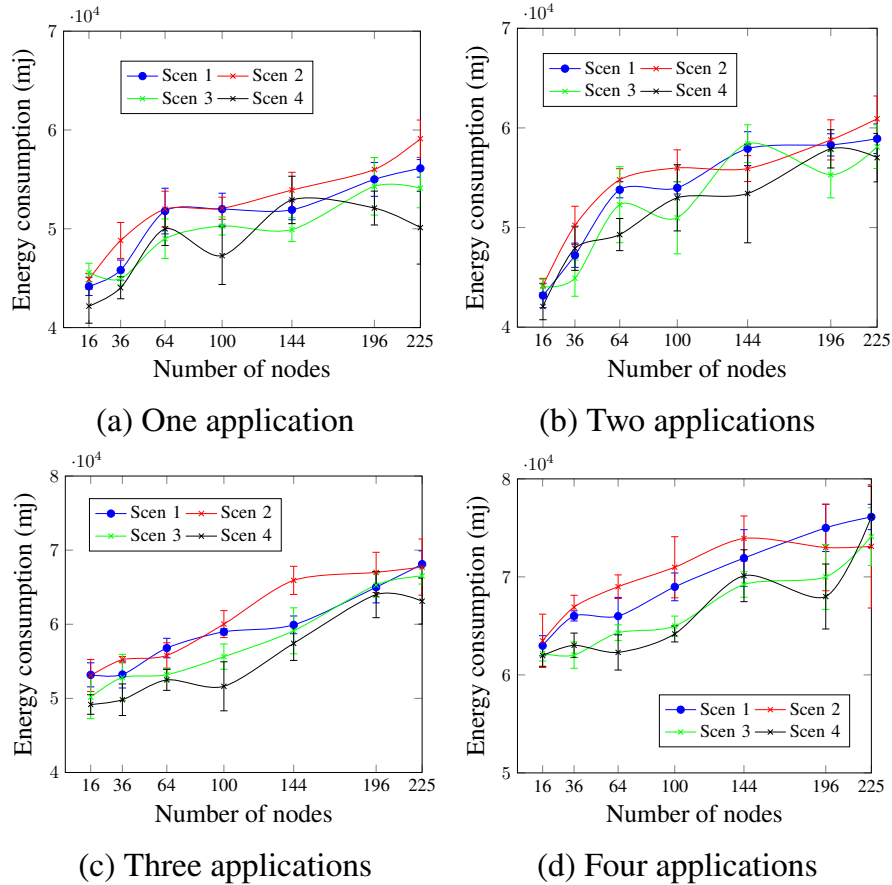


Figure 35 – Energy consumption, changing the MCR value

This is justified by the additional exchanged control messages for higher MCR values, which increases the probability of packet drop as a result of the buffer fullness.

Similarly to the control delivery rate, Figure 37 shows that the higher MCR values led to lower control delay values for all cases. For a network size of 100 nodes and four running applications as depicted in Figure 37d, the control delay values were 1.57 s, 1.65 s, 1.42 s, and 1.29 s for Scen 1, Scen 2, Scen 3, and Scen 4, respectively. This occurs since for lower MCR values, the messages wait for more time in the buffer to be processed.

6.2.2.2 Data traffic rate (DTR)

In this section, we evaluate the effect of the DTR on the network performance. We consider three different scenarios (Scen 1, Scen 5, and Scen 6) with different DTR values as Table 9 shows.

Figure 38 depicts the data delivery rate for up to 4 applications running simultaneously. For a single application (Figure 38a), it is clear that the high DTR values significantly reduced the data delivery rate. Thus, Scen 6, which has the highest DTR value, presented the worst values. This occurs since the higher data traffic rate leads to more congestion and increases the probability of packet drop due to the buffer fullness.

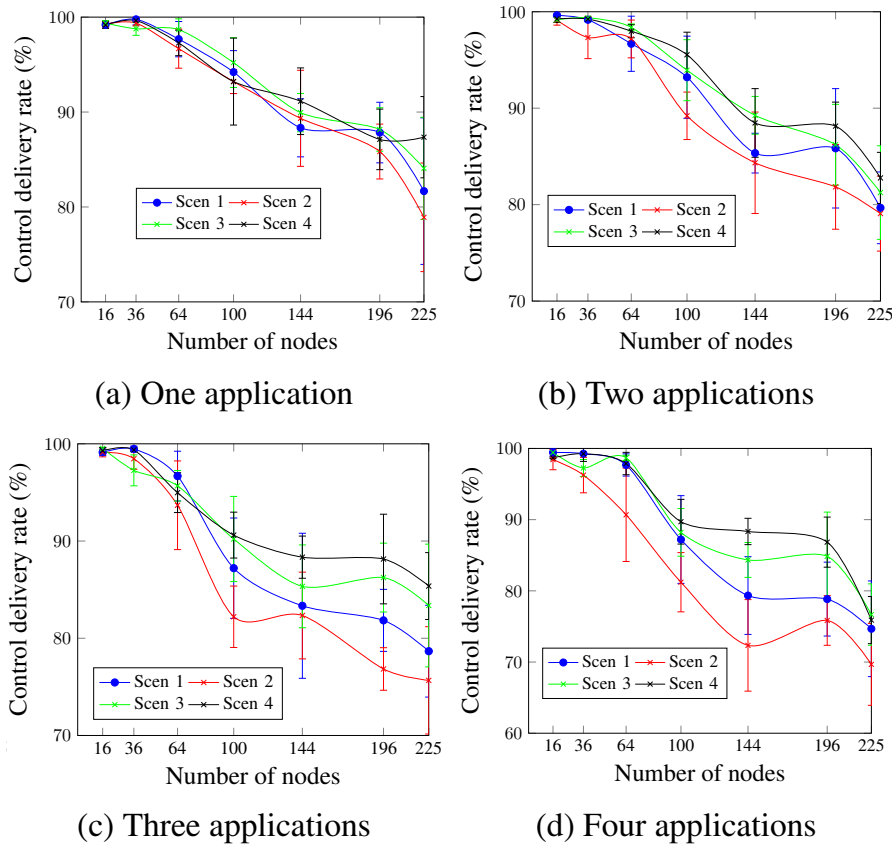


Figure 36 – Control delivery rate, changing the MCR value

For Scen 1 and Scen 5, the AA approach ensured the data delivery rate requirement for network sizes up to 225 nodes, compared to network sizes up to only 36 nodes in the case of Scen 6. To justify these results, it is important to note that for a network size of 100 nodes, for instance, 98 data messages are sent every minute in the case of Scen 1 and Scen 5, compared to 5880 data messages for Scen 6. For two applications, Figure 38b shows that Scen 1, with the lowest DTR value, continues to present better values than Scen 5 and Scen 6 for all cases. For the 1st and 2nd applications, the satisfied network sizes are reduced from 196 nodes for Scen 1 to 100 nodes and 36 nodes for Scen 5 and Scen 6, respectively. Concerning the three applications, Figure 38c depicts that for the 3rd application, which has a delivery rate requirement of 90%, the sizes of the satisfied networks increased from 36 nodes in the case of Scen 6 to 64 nodes and 225 nodes for Scen 5 and Scen 1, respectively. For four running applications as Figure 38d shows, Scen 6, the scenario with the highest DTR value, continues to present the worst data delivery rate values compared to Scen 1 and Scen 5. This is also applied to the 4th application, which has no priority. For a network size of 225 nodes and the 4th application, the data delivery rate was 51.72% for Scen 1, compared to 44.53% and 26.81% for Scen 5 and Scen 6, respectively.

Concerning data delay, Figure 39 depicts that for a single application (Figure 39a), Scen 6 presents the highest delay values since it has the highest DTR value. Scen 1 and

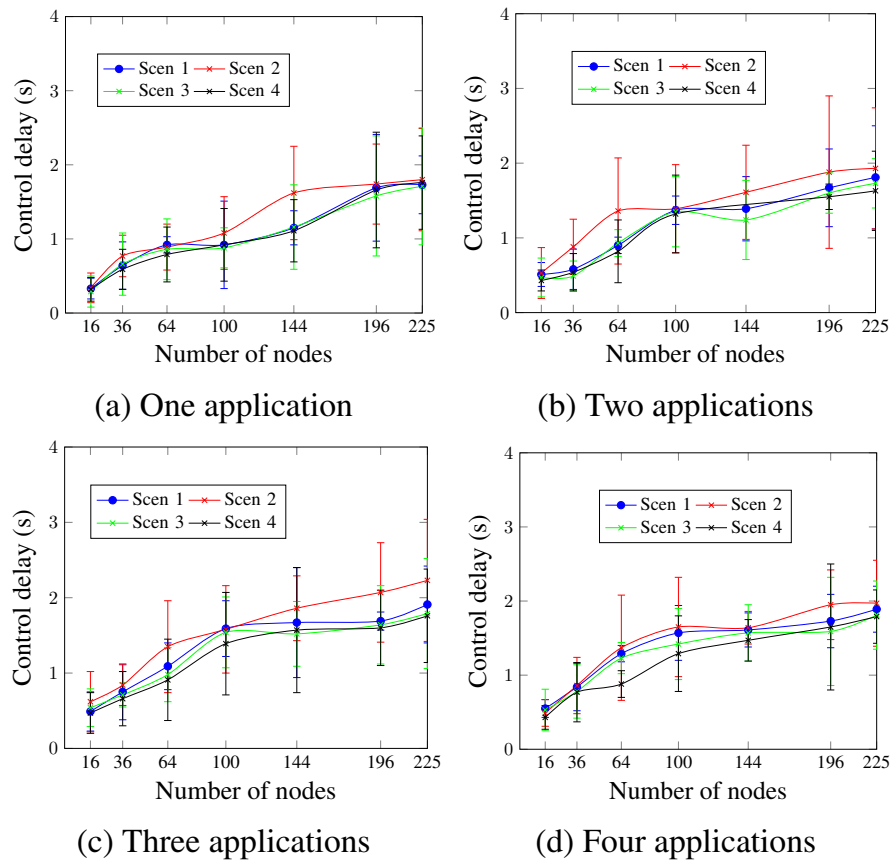


Figure 37 – Control delay, changing the MCR value

Scen 5 present similar values for the different network sizes. This occurs since there is only one application, and the DTR values for both Scen 1 and Scen 5 are hence equal. The delay requirement keeps ensured for network sizes up to 144 nodes for Scen 1 and Scen 5, compared to 64 nodes for Scen 6. For the case of two applications and the 1st application as Figure 39b, the AA approach ensures the delay requirement for network sizes up to 100 nodes for Scen 1, compared to 144 nodes for the single application case. This could be justified by the 2nd application, which raised the network traffic and prevented ensuring the delay requirement for network sizes larger than 100 nodes. The 2nd application has a delay requirement of 950 ms, and the AA approach ensured the requirement for up to 100 nodes, 64 nodes, and 36 nodes for Scen 1, Scen 5, and Scen 6, respectively.

For three and four running applications (Figures 39c and 39d), the DTR value is directly proportional to the obtained delay values for all cases. This means that the scenarios with higher DTR values presented higher delay values. Concerning the 4th application, it presents the worst delay values since it is accommodated using a single timeslot and has no priority. For a network size of 225 nodes, the obtained delay values were 12.28 s, 19.07 s, and 29.13 s for Scen 1, Scen 5, and Scen 6, respectively. These values confirm, again, the high effect of the selected DTR value on the delay requirement.

Figure 40 shows that the control overhead increases with the number of nodes,

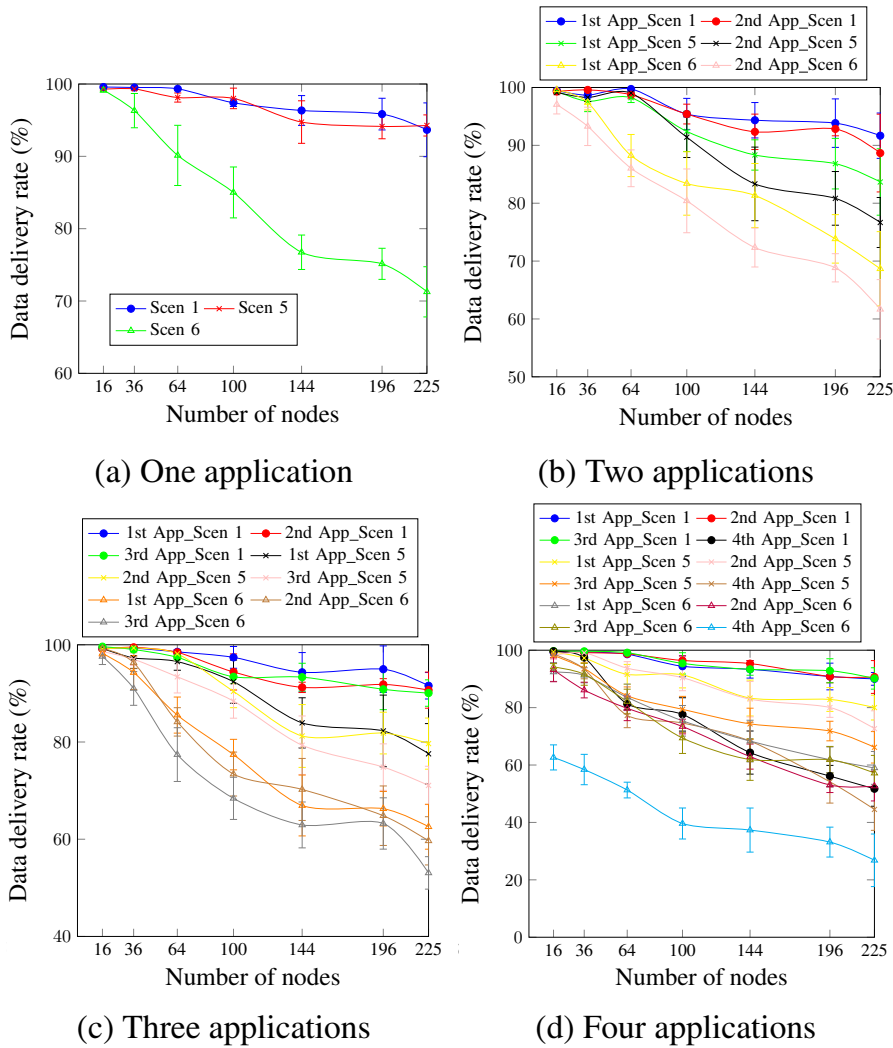


Figure 38 – Data delivery rate, changing the DTR value

regardless of the DTR value and the number of applications. For two and four applications as Figures 40b and 40d show, all scenarios presented similar values for the network sizes up to 100 nodes. For larger network sizes, the control overhead is directly proportional to the DTR value. This means that Scen 6 with the highest DTR presented the highest control overhead values, whereas Scen 1 presented the lowest values. For a network size of 196 nodes and four running applications, the control overhead values were 36926.2 messages, 40301.4 messages, and 47097.8 messages for Scen 1, Scen 5, and Scen 6, respectively. This occurs since higher DTR requires more exchanged control messages (Rescheduling request message, Rescheduling message, and New scheduling message), as an attempt to ensure the application's QoS requirements. Figure 41 shows that, in general, the DTR value did not significantly affect the energy consumption. However, higher DTR values led to a little increase in energy consumption in most cases. For instance, in the case of three running applications and a network size of 100 nodes, the energy consumption values were 58482.5 mj, 59190.6 mj, and 60647.2 mj for Scen 1, Scen 5, and Scen 6, respectively.

Figure 42 shows that Scen 1 and Scen 5 present relatively similar values in terms

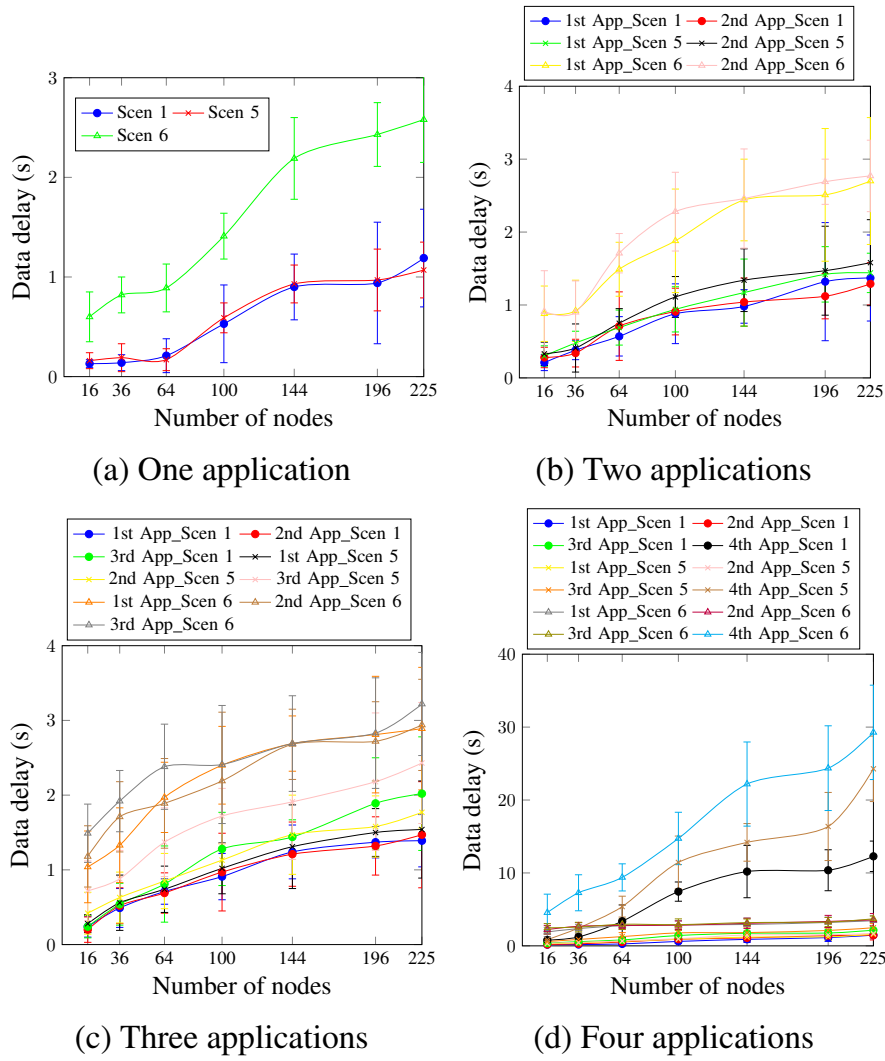


Figure 39 – Data delay, changing the DTR value

of control delivery rate. Scen 6, however, presents the worst values for all cases. This confirms that the control delivery rate is inversely proportional to the DTR value. For four applications and a network size of 144 nodes, Figure 42d shows that the obtained values were 59.64% for Scen 6, compared to 79.16% and 75.34% for Scen 1 and Scen 5, respectively. This is caused by the additional control traffic in the case of Scen 6, which increases the congestion during the control timeslots after the network convergence.

The control delay is shown in Figure 43. Similarly to the control delivery rate, the obtained control delay values show that Scen 6 continues to present the worst values, whereas Scen 1 and Scen 5 present similar values for most cases. For two applications and a network size of 144 nodes, Figure 43b shows that the control delay values were 1.75 s for Scen 6, compared to 1.39 s and 1.47 s for Scen 1 and Scen 5 respectively.

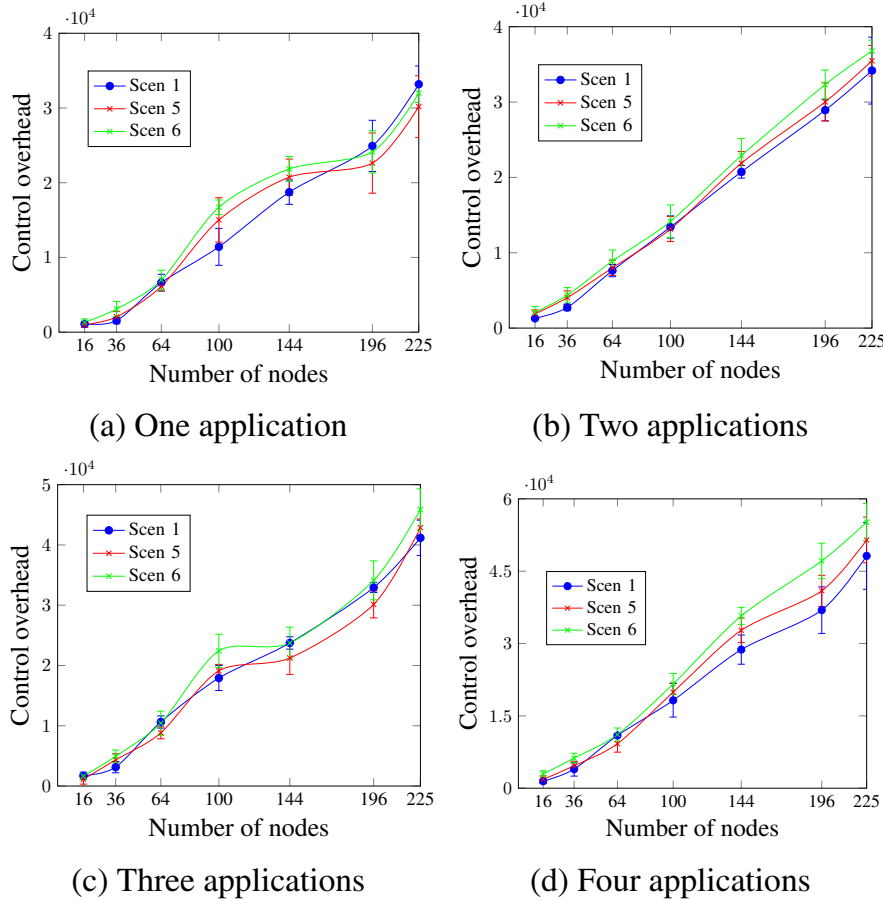


Figure 40 – Control overhead, changing the DTR value

6.2.2.3 Application's QoS requirements (AR)

We evaluate in this section the effect of the application's QoS requirements (AR) on the network performance. We consider three scenarios with different QoS requirements for up to four applications.

Figure 44 shows that for a single application (Figure 44a), the AA approach ensured the data delivery rate requirement for all the considered network sizes for both Scen 1 and Scen 8. This is reduced to network sizes of up to 196 nodes for Scen 7 which has a requirement of 95%. For a network size of 225 nodes, the obtained delivery rate values were 93.67%, 91.42%, and 87.19% for Scen 1, Scen 7, and Scen 8 respectively. This means that the AA approach was more active for higher delivery rate requirement, and thus it increases the delivery rate from 87.19% for Scen 8 to 93.67% for Scen 1. However, although the requirement of Scen 7 is higher than that of Scen 1, the AA approach was not capable of ensuring it for the network size of 225 nodes.

For two applications, Figure 44b shows that for the 1st application, the AA approach continues to ensure the delivery rate for all the network sizes for Scen 1 and Scen 8, compared to network sizes of up to 144 nodes for Scen 7. For the 2nd application, which do not have a delivery rate requirement, the worst delivery rate value for all scenarios

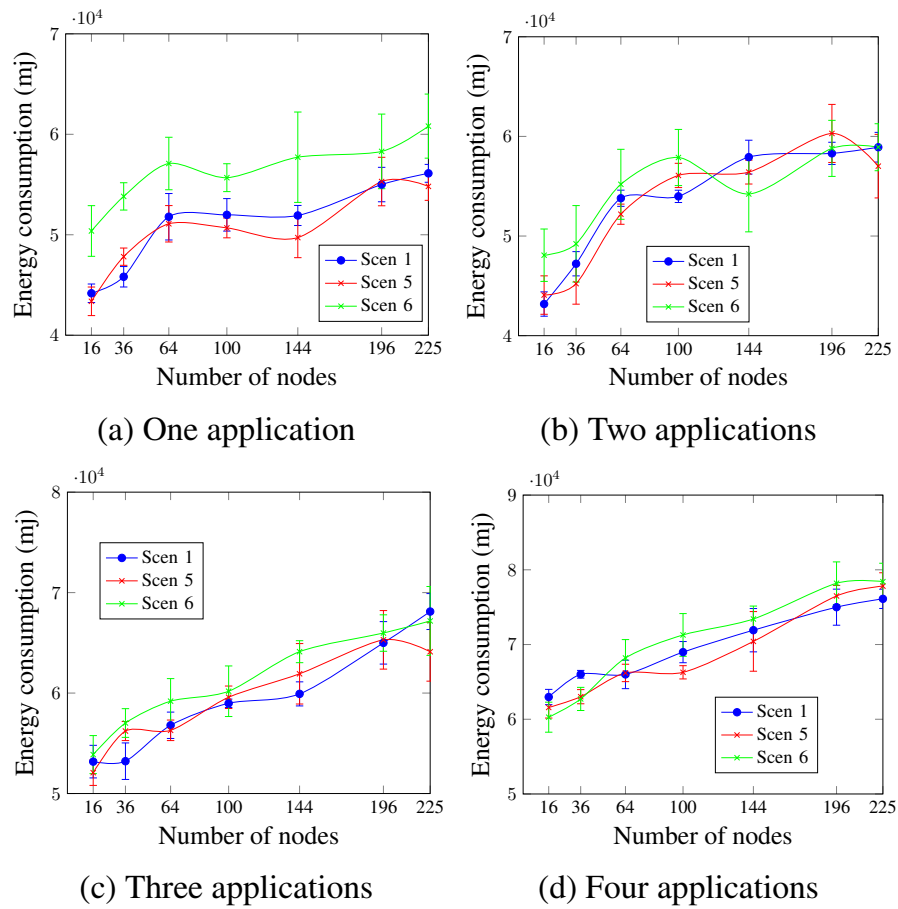


Figure 41 – Energy consumption, changing the DTR value

was 88.18%, knowing that most of the obtained values exceeded 90%. Concerning the three applications as Figure 44c depicts, all network sizes in the case of the 1st application for both Scen 1 and Scen 8 stays to be ensured, compared to network sizes of up to 144 nodes for Scen 7. As for the 3rd application, the AA approach was capable of ensuring the requirement for all the considered scenarios and network sizes. This mainly occurs since: i) the DTR of the 3rd application is relatively low, and then no high additional traffic is added to the network.

For four applications, Figure 44d shows that the unique scenario that ensures the delivery rate requirement for all network sizes is Scen 8. This occurs since the requirement of this scenario is 80%, and then it is easier to be ensured compared to both Scen 1 and Scen 7. For the 3rd application, the AA approach ensures the requirement for all the considered scenarios and network sizes. Finally, the 4th application presents the worst values for all cases, since it has no priority and only one timeslot is assigned to accommodate it.

Figure 45 depicts the data delay variations for the considered scenarios. For a single application as Figure 45a shows, the AA approach ensured the delay requirement for network sizes of up to 144, 100, and 196 nodes for Scen 1, Scen 7, and Scen 8 respectively. For a network size of 100 nodes, the obtained delay values were 0.53 s, 0.44 s, and 0.61 s

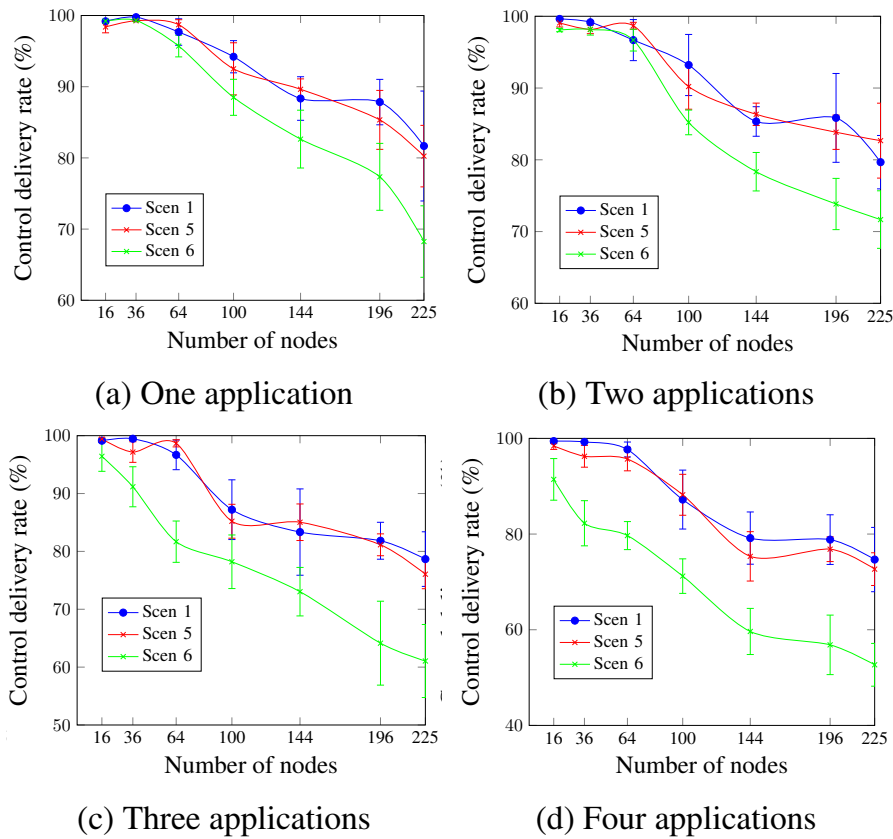


Figure 42 – Control delivery rate, changing the DTR value

for Scen 1, Scen 7, and Scen 8 respectively. This shows the effect of the requirement value on the AA approach's activity level, and proves, again, the efficiency of this approach. For two applications, Figure 45b shows that for the 1st application, the ensured network sizes were 100, 64, and 144 nodes for Scen 1, Scen 7, and Scen 8 respectively. Then the ensured network sizes are reduced for all the considered scenarios. This mainly occurs because of the additional traffic resulting from the 2nd application. For a network size of 196 nodes and for Scen 1 and Scen 7, the obtained values were 1.32 s and 0.9 s respectively. This shows that although the AA approach was not capable of ensuring the delay requirement for Scen 7; however, it improved its data delay compared to Scen 1. For the 2nd application, Scen 7 continues to present better delay values compared to Scen 1 and Scen 8 for the increasing network sizes. This is justified by the application's QoS requirements of Scen 7, which increases the AA approach's activity.

Concerning the three applications, Figure 45c depicts that for both 1st and 2nd applications, the considered scenarios present similar behaviors compared to the case of the two applications. This means that the 3rd application did not add more significant traffic to the network. For the case of four applications (Figure 45d), the 4th application presents the worst data delay values since it is accommodated using only one timeslot.

Figure 46 shows that for all scenarios, the control overhead is directly proportional to the network size. Moreover, the application's QoS requirements did not highly affect

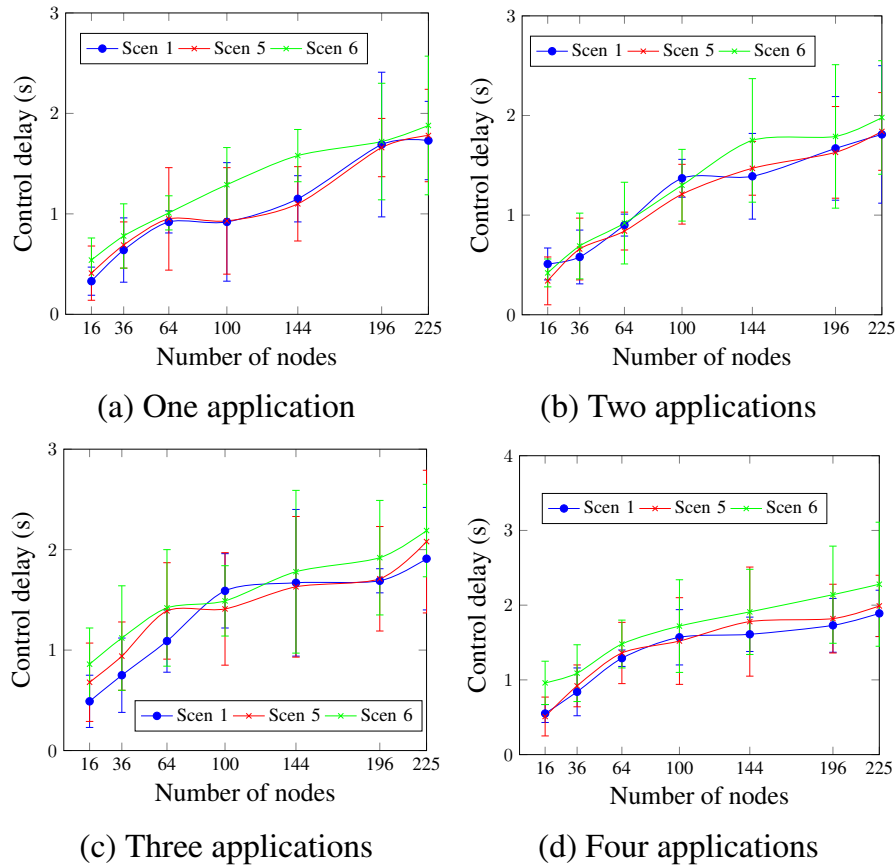


Figure 43 – Control delay, changing the DTR value

the control overhead and all scenarios presented similar performance. However, Scen 7 presented a bit higher values compared to Scen 1 and Scen 8. For three applications and a network size of 196 nodes, the obtained values were 32914.38 messages, 33098.74 messages, and 31683.12 messages for Scen 1, Scen 7, and Scen 8 respectively. This occurs since the harder requirements raise the probability of existing unsatisfied application (s). This, in turn, requires more exchanged control messages to request and disseminate the new scheduling.

Figure 47 depicts that for all cases, the energy consumption is directly proportional to the number of applications. The obtained results do not indicate a clear relationship between the application's QoS requirements and the energy consumption. For two applications (Figure 47b) and a network size of 144 nodes, the energy consumption was 57919.32 mj, 58472.41 mj, and 58985.27 mj for Scen 1, Scen 7, and Scen 8 respectively.

Concerning the control plane, Figures 48 and 49 show that for all cases, a worse performance is obtained for the larger network sizes. The obtained results could be considered similar for all the considered scenarios. For three applications and a network size of 225 nodes, the control delivery rate values were 78.67%, 75.28%, and 79.31%, and the control delay values were 1.91 s, 1.94 s, and 1.77 s for Scen 1, Scen 7, and Scen 8 respectively. Thus we could say that the application's QoS requirements have no significant

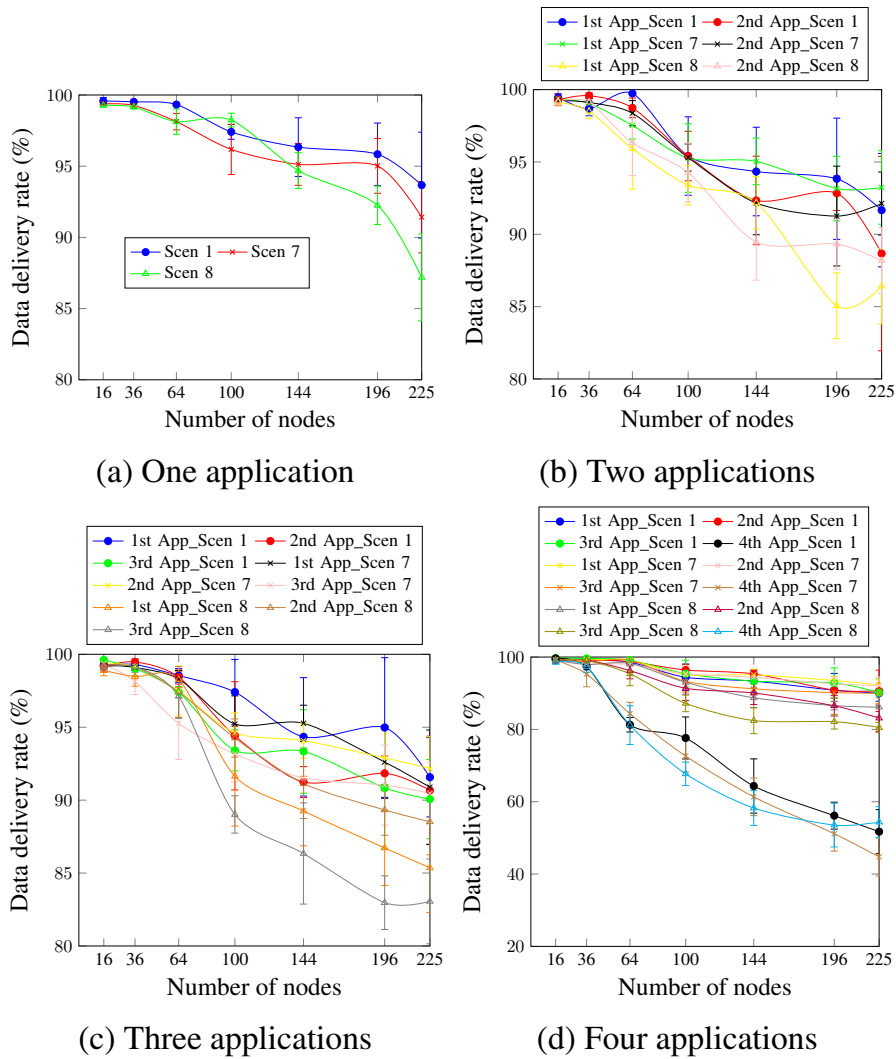


Figure 44 – Data delivery rate, changing the AR value

effect on the control plane. This occurs since, although the harder requirements lead to more exchanged control messages; however, the quantity of the additional control messages is not high enough to make a clear difference among the considered scenarios.

6.2.2.4 Difference rate (DR)

The effect of the difference rate (DR), which is the difference between the calculated metrics during the run-time and the application's QoS requirements, is evaluated in this section. Higher values of the DR could delay ensuring the requirements since more time is required to discover that some application is unsatisfied. However, this reduces the control overhead since fewer amount of rescheduling request messages, rescheduling messages, and new scheduling messages are exchanged. Lower values of the DR, on the other hand, could speed up ensuring the requirements. However, this leads to more control overhead.

Concerning the data delivery rate, Figure 50 depicts that for a single applications (Figure 50a), the AA approach ensured the requirement for all the considered network

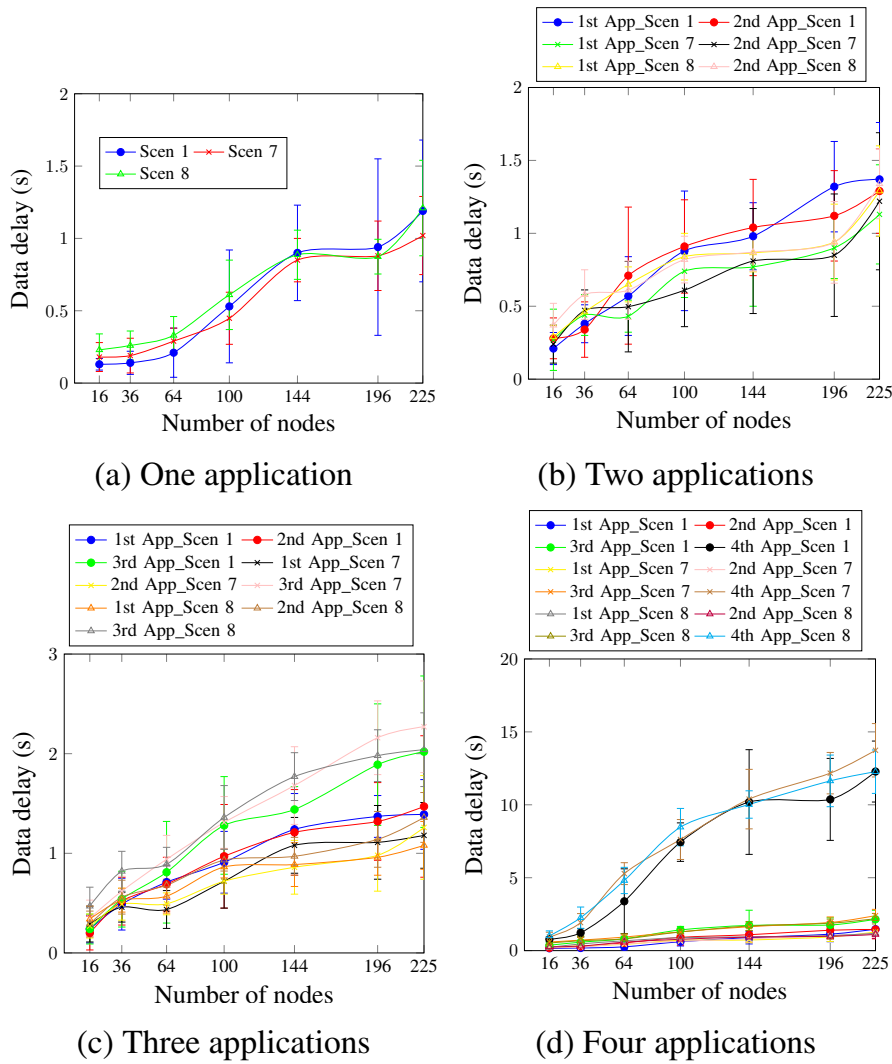


Figure 45 – Data delay, changing the AR value

sizes for Scen 1 and Scen 9 which have lower DR values, compared to network size of up to 100 nodes for both Scen 10 and Scen 11 which have higher DR values. Thus, we could say that Scen 1 and Scen 9, presented better values compared to Scen 10 and Scen 11. For a network size of 196 nodes, the obtained delivery rate values were 95.84%, 93.77%, 90.41%, and 88.31% for Scen 1, Scen 9, Scen 10, and Scen 11. It is important to note that although Scen 9 has the lowest DR value; however, it was not able to outperform the Scen 1. This indicates that lower DR value do not always mean better data delivery rate, and under a determined threshold, the DR values are not useful to improve the data delivery rate.

For two applications, Figure 50b shows that for the 1st application, which has a delivery rate requirement of 92%, the AA approach ensured the requirement for all the network sizes compared to network sizes of up to 100 nodes and 64 nodes for Scen 10 and Scen 11 respectively. For a network size of 144 nodes, the obtained delivery rate was 94.34% for Scen 1 compared to 86.18% for Scen 11. This occurs since in the case of Scen 11, which has a DR value of 40%, more data messages are loosed before assigning a new

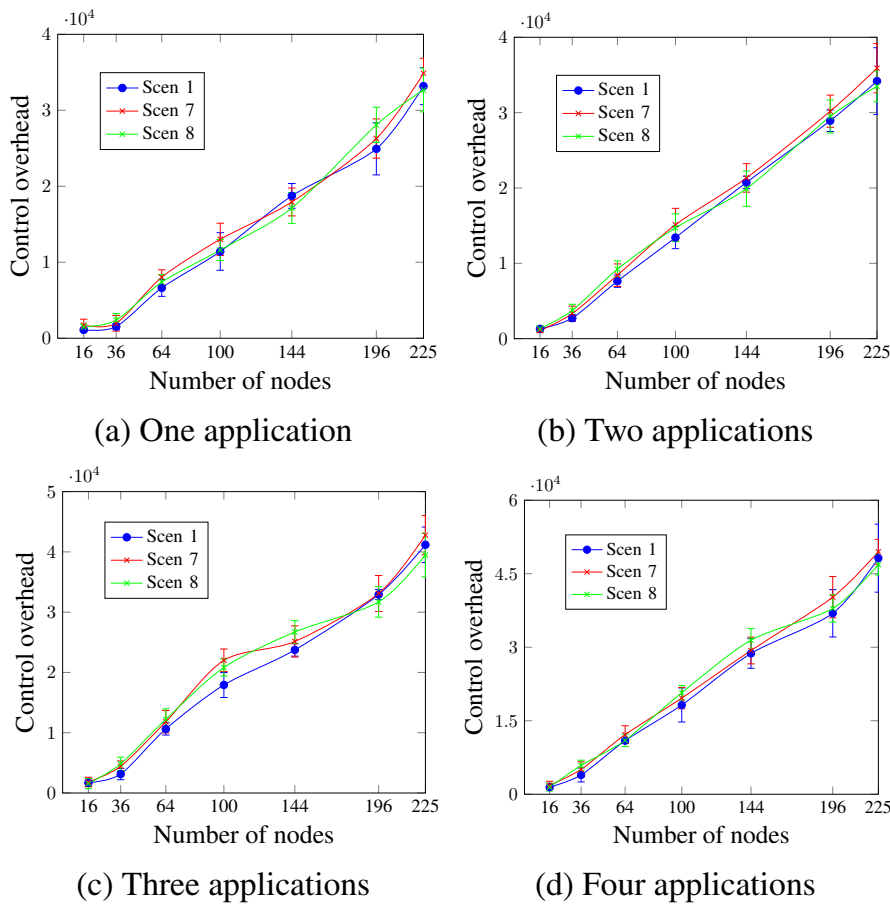


Figure 46 – Control overhead, changing the AR value

scheduling. This, in turn, reduces the overall data delivery rate.

Concerning three applications as Figure 50c shows, Scen 1 and Scen 9 continue to present better values than Scen 10 and Scen 11 for all cases. For the 3rd application, which has a delivery rate requirement of 90%, the requirement is ensured for all the considered network sizes for both Scen 1 and Scen 9, compared to network sizes of only up to 100 nodes for Scen 10 and Scen 11. For a network size of 225 nodes and Scen 1, the delivery rate was 91.57% for the 1st application, compared to 90.05% for the 3rd one. Then the obtained values were similar although the 1st application has higher priority than the 3rd one. This is justified by the DTR of each application. Since the DTR of the 3rd application is 8 minutes, this makes it easier to pick its data messages without losing them because of the buffer fullness. For four applications, Figure 50d shows that for the 1st application, the ensured network sizes are reduced to 144 nodes and 196 nodes for Scen 1 and Scen 9 respectively, compared to the case of three applications. This occurs because more applications mean more traffic and higher congestion. For all the considered scenarios, the 4th application, which is a best-effort, presents the worst data delivery rate values.

Figure 51 shows the obtained data delay values. For a single application (Figure

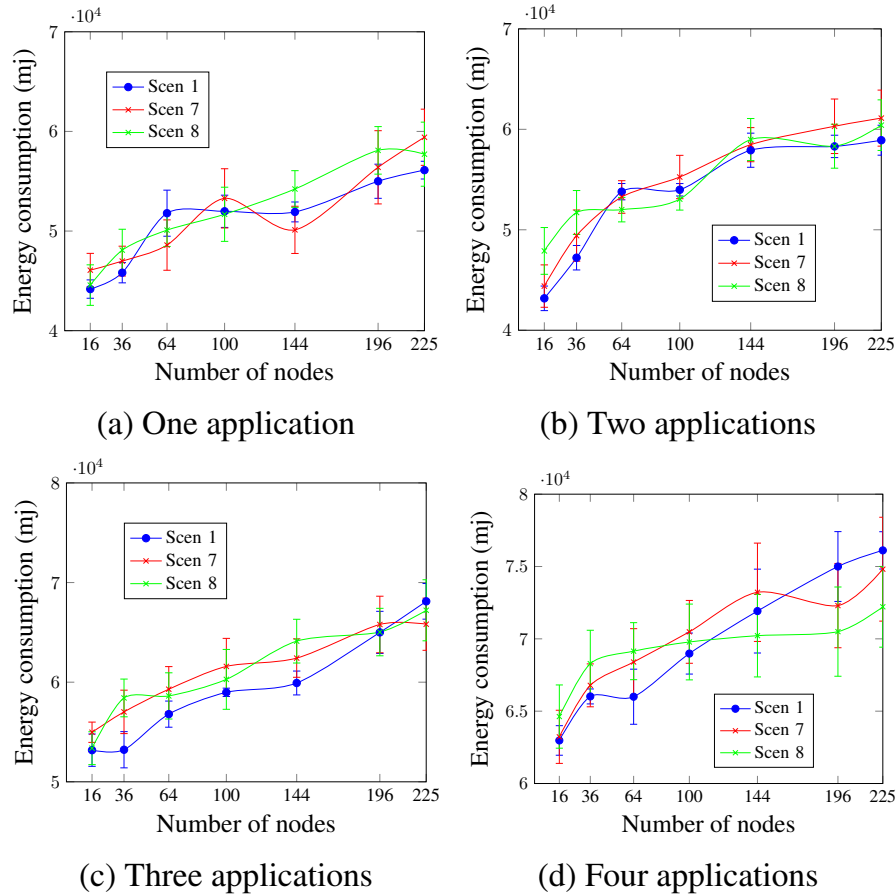


Figure 47 – Energy consumption, changing the AR value

51a), Scen 10 and Scen 11, which have the highest DR values, present the worst delay values since more time is required to change the actual scheduling and ensure the application's QoS requirements. Scen 1 and Scen 9, on the other hand, present similar performance and outperform the other scenarios. For two applications, Figure 51b shows that for the 1st application, Scen 9, the scenario with the lowest DR value, outperformed all other scenarios. The AA approach ensures the delay requirement for network sizes of up to 144 nodes for this scenario, compared to network sizes of up to 100 nodes for Scen 1 and 64 nodes for both Scen 10 and Scen 11. For the 2nd application, Scen 1 and Scen 9 continue to present better delay values with ensured network sizes of up to 144 nodes, compared to network sizes of up to 100 nodes for both Scen 10 and Scen 11. Concerning three applications, Figure 51c shows that for the first two applications, the AA approach ensures the requirement for network sizes of up to 100 nodes for Scen 1 and Scen 9, compared to network sizes of up to 64 nodes for Scen 10 and Scen 11. For four applications as Figure 51d shows, the 4th application, similarly to all the previous cases, presented the worst values for all the considered scenarios.

Figure 52 depicts that the DR value did not have a significant effect on the control overhead, and all scenarios presented approximately similar performance. For network sizes of up to 100 nodes, the considered scenarios presented similar values. For larger network

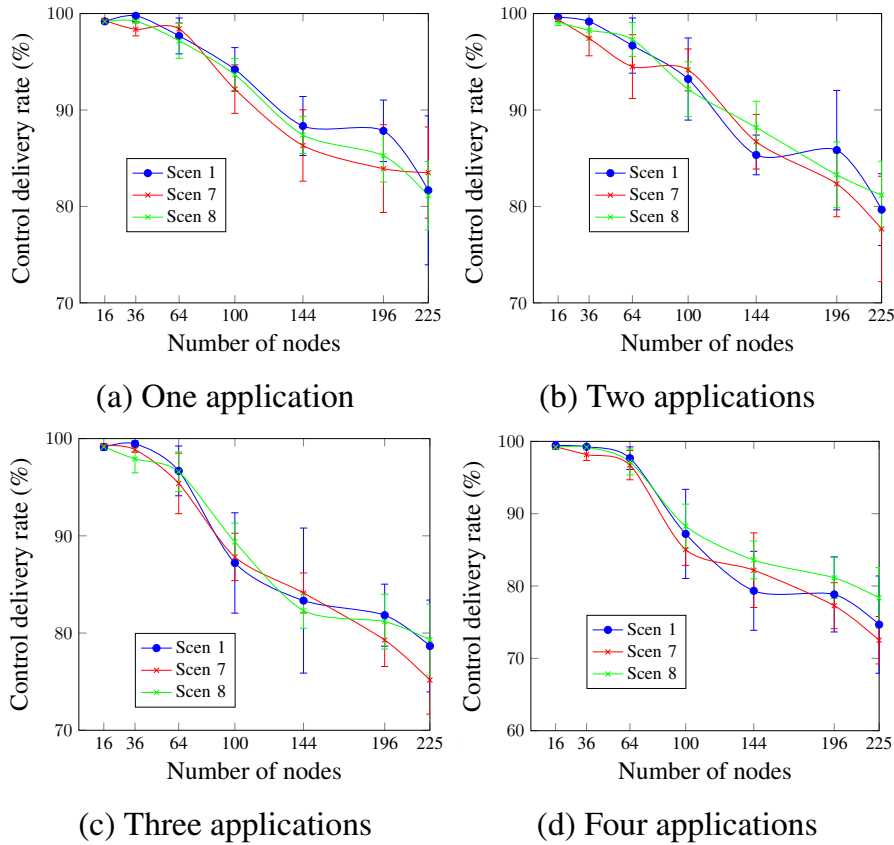


Figure 48 – Control delivery rate, changing the AR value

sizes, the DR value had a bit higher effect on the control overhead. For a network size of 144 nodes and four applications, the obtained values were 28752.11 messages, 29236.46 messages, 26192.7 messages, and 24025.19 messages for Scen 1, Scen 9, Scen 10, and Scen 11 respectively. This occurs since for lower DR values, more control messages are exchanged to construct and disseminate the new scheduling. Similarly, Figure 53 shows that the DR value had not a significant effect on the energy consumption. For a network size of 144 nodes and three applications (Figure 53c), the obtained values were 59681.08 mj, 61472.26 mj, 60245.53 mj, and 63057.18 mj for Scen 1, Scen 9, Scen 10, and Scen 11 respectively.

Concerning the control plane, Figures 54 and 55 show that for both delivery rate and delay and for all the considered scenarios, a worse performance is obtained for the increasing number of nodes. The DR value did not have a high effect on the control plane. Nevertheless, since lower DR values lead to more exchanged control messages to satisfy the QoS requirements, the scenarios with the highest DR values (Scen 10 and Scen 11) outperformed the other scenarios. For a network size of 100 nodes, the obtained delivery rate / delay values were 87.21% / 1.59 s, 84.97% / 1.57 s, 91.72% / 1.16 s, and 93.02% / 1.04 s for Scen 1, Scen 9, Scen 10, and Scen 11 respectively.

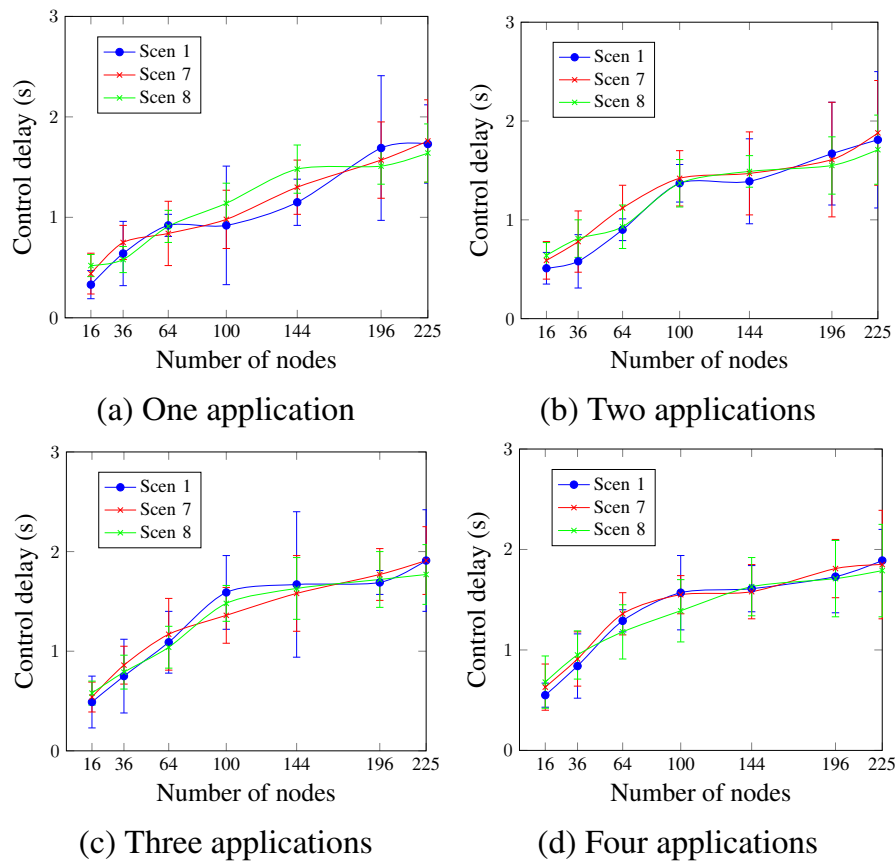


Figure 49 – Control delay, changing the AR value

6.2.2.5 Topology

This section investigates the effect of the adopted topology on the network performance. For this purpose, we consider two scenarios: i) Scen 1, which adopts the grid topology; and ii) Scen 12, which adopts the random topology.

Figure 56 shows that for a single application (Figure 56a), the AA approach ensured the requirement for all the considered network sizes for Scen 1 (grid topology), compared to network sizes of up to 144 nodes for Scen 12 (random topology). This could be justified by the sink position. For two applications, Figure 56b shows that for two applications, Scen 1 performs better than Scen 12. For the 1st application, the AA approach ensured the requirement for network sizes of up to 196 nodes for Scen 1, compared to 100 nodes for Scen 2. Concerning the three applications (Figure 56c), the grid topology continues to present better values than the random one. For the 3rd application, the AA approach ensured the delivery rate requirement for all the considered network sizes for Scen 1, compared to network sizes of up to 196 nodes for Scen 12. For four applications, Figure 56d depicts that for the 1st application, the ensured network sizes are up to 144 nodes and 100 nodes for the grid and random topologies respectively. Then the ensured network sizes are reduced for Scen 1 compared to the case of two applications. This is justified by the additional traffic caused by the increasing number of applications. For the 3rd application,

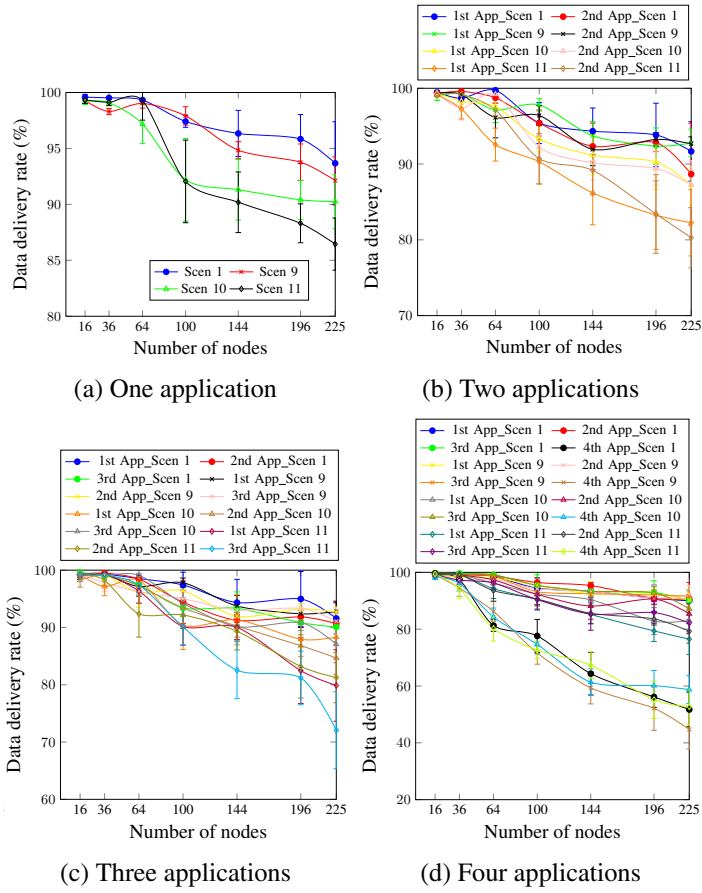


Figure 50 – Data delivery rate, changing the DR value

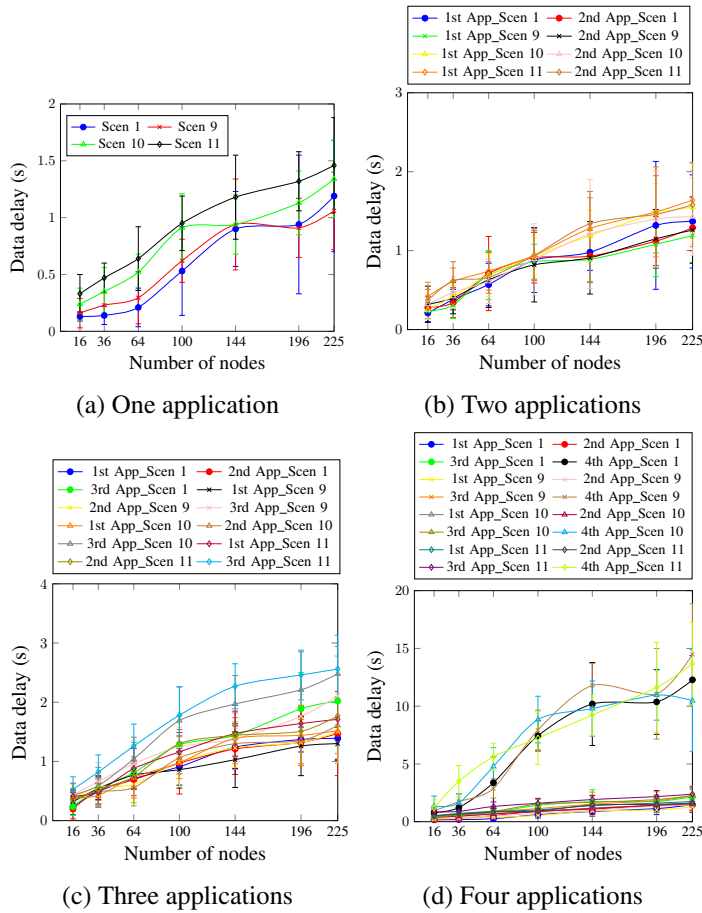


Figure 51 – Data delay, changing the DR value

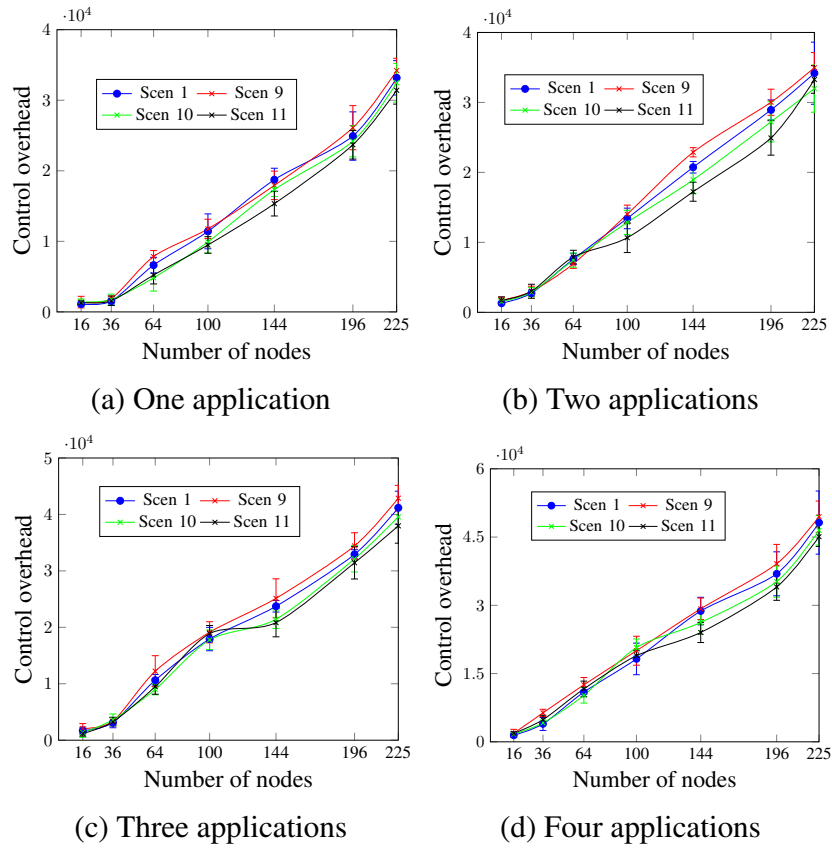


Figure 52 – Control overhead, changing the DR value

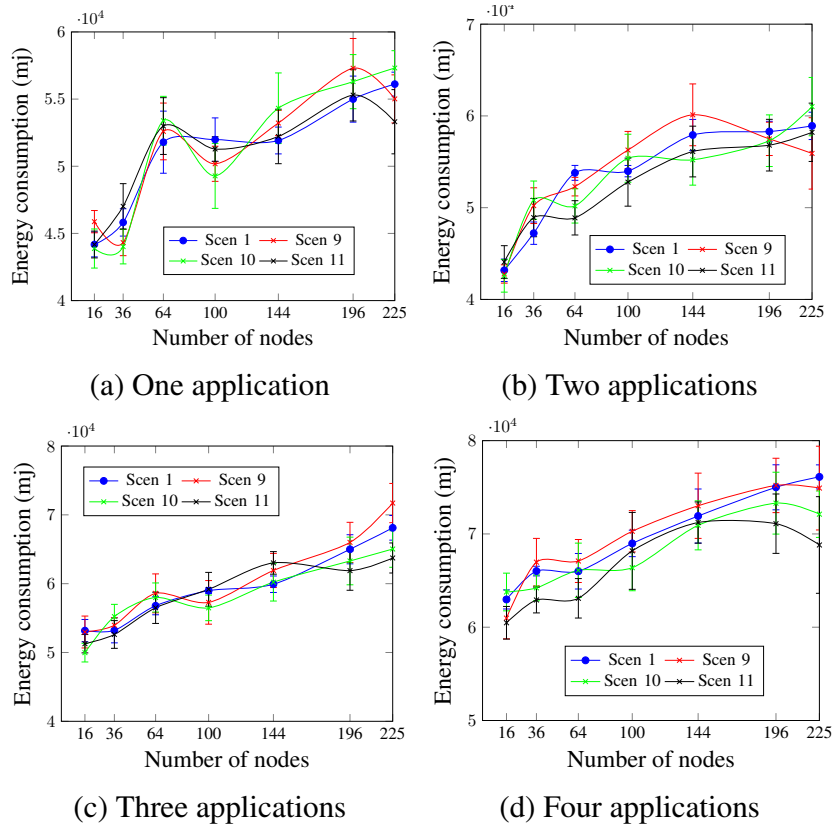


Figure 53 – Energy consumption, changing the DR value

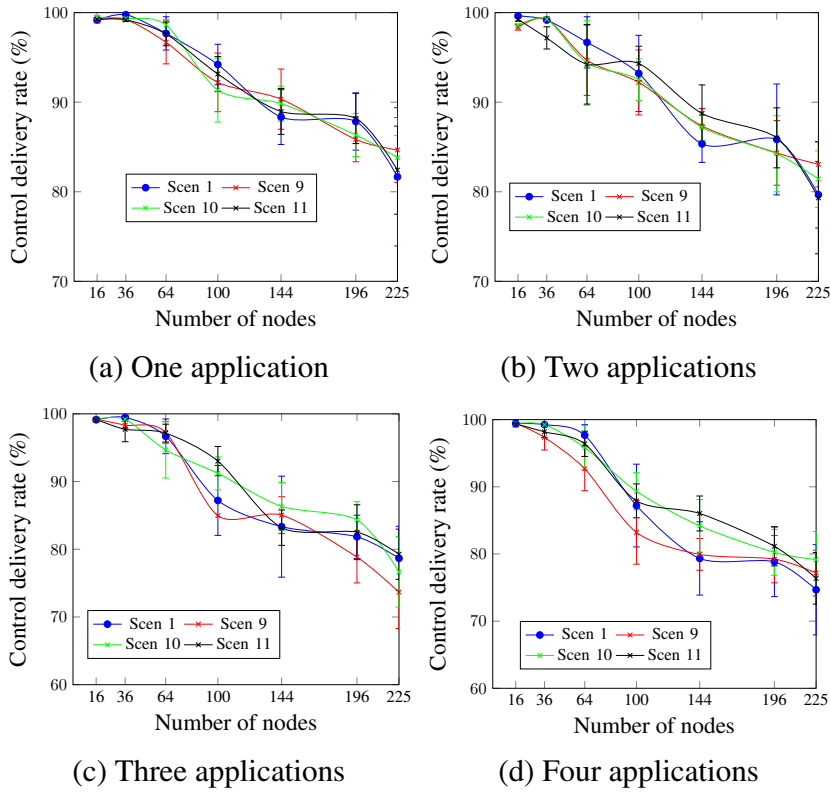


Figure 54 – Control delivery rate, changing the DR value

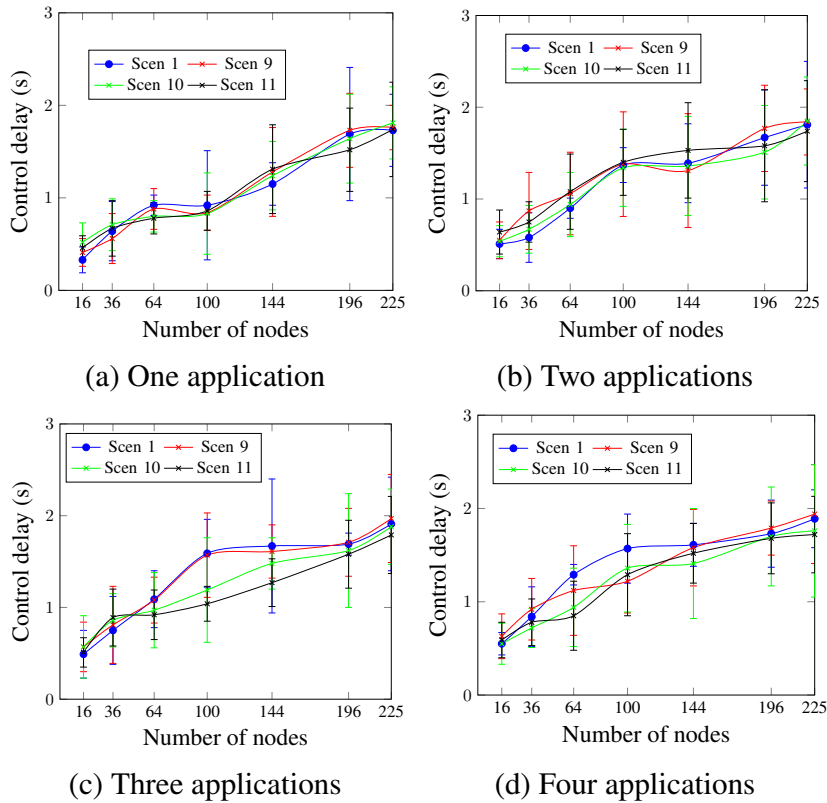


Figure 55 – Control delay, changing the difference rate

all network sizes are ensured for Scen 1, compared to up to 144 nodes for Scen 12. We notice that regardless of the number of applications, the obtained delivery rate values reduce for larger network sizes in the case of grid topology, whereas they vary (increase and decrease) for random topology. Moreover, the grid topology performed better than the random one for most of cases. This occurs since the sink and controller positions have an important effect on the network performance.

Concerning the data delay, Figure 57 depicts that for a single application (Figure 57a), the grid topology presents better values than the random one for the increasing network sizes. The AA approach ensures the delay requirement for networks sizes of up to 144 nodes and 100 nodes for Scen 1 and Scen 12 respectively. For the increasing number of applications, Figures 57b, 57c, and 57d show that the grid topology continues to present lower delay values than the random one. The 4th application, as Figure 57d shows, presents the worst values for both Scen 1 and Scen 12. This occurs since this application is best-effort and only single timeslot is assigned for it.

Figure 58 shows that the control overhead did not highly affect by the topology. For all cases, the obtained values are directly proportional to the network size. Both grid and random topologies present similar control overhead values regardless of the network size and number of applications. For three applications and a network size of 144 nodes, the obtained values were 23736.52 messages and 24118.74 messages for Scen 1 and Scen 12 respectively. This occurs since regardless of the topology, no additional control messages are used. Similarly, Figure 59 depicts that the consumed energy varies for both scenarios with the network size and none of the considered topologies outperform the other. Then we could say that there is no clear relation between the energy consumption and the topology type.

Concerning the control plane, Figures 60 and 61 show the delivery rate and delay values. Regardless of the topology and number of applications, lower delivery rates and higher delays are obtained for larger network sizes. In most cases, the topology had no significant effect on the control plane. For four applications and network sizes of 144 and 196 nodes, the obtained delivery rate / delay values were 79.34% / 1.65 s and 78.62% / 1.73 s for Scen 1, compared to 82.27% / 1.6 s and 77.19% / 1.78 s for Scen 12 respectively. This occurs since changing the topology from grid to random did not change the activity level of the AA approach, and then, did not lead to more / less exchanged control messages.

Table 11 summarizes the previous results to give an overview of the AA approach's performance in comparison to the application's requirements. All the data of Table 11 has taken adopting the obtained results for Scen 1 (with the default values of MCR and DTR as Table 8 shows), and considering the first three applications (1st App, 2nd App, and 3rd App), which have QoS requirements.

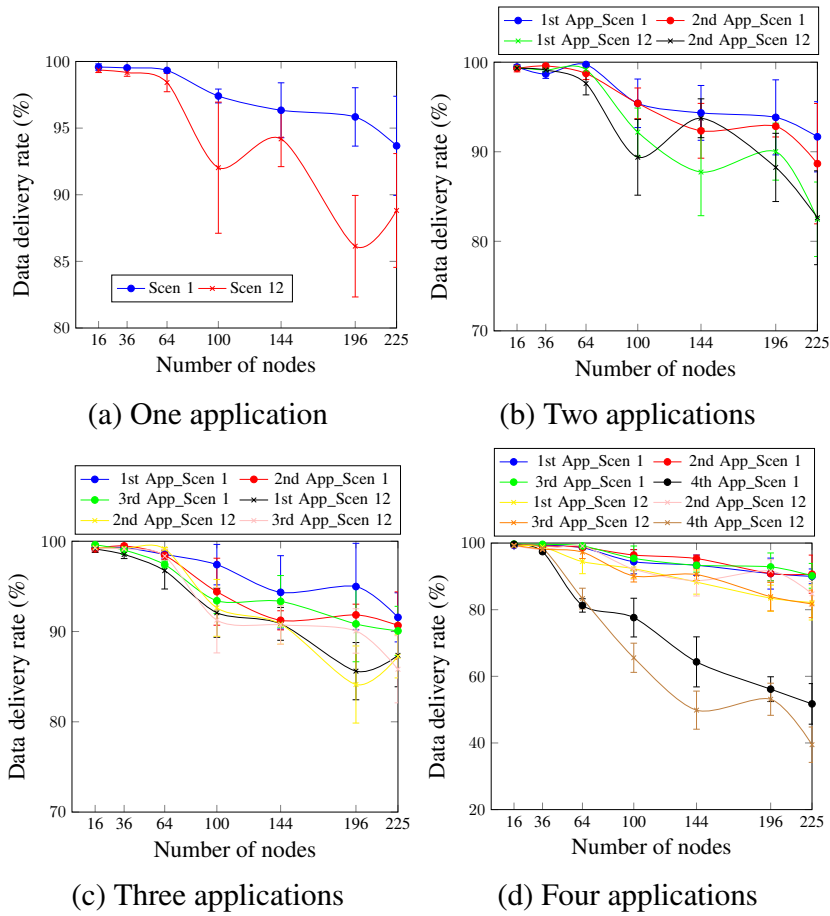


Figure 56 – Data delivery rate, changing the topology

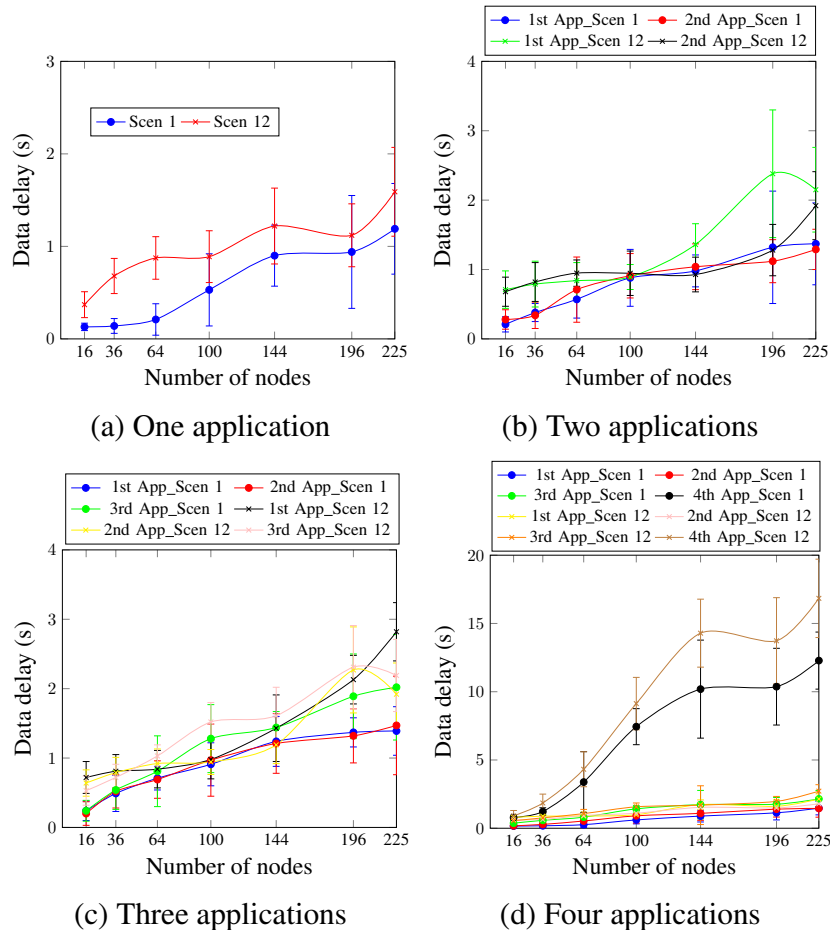
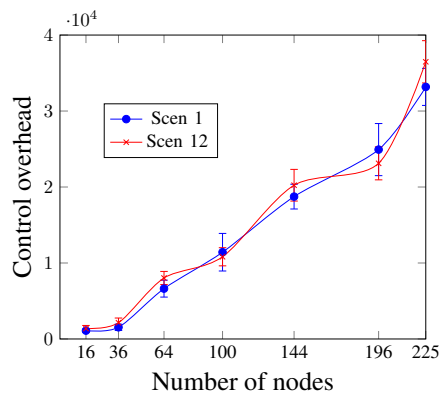
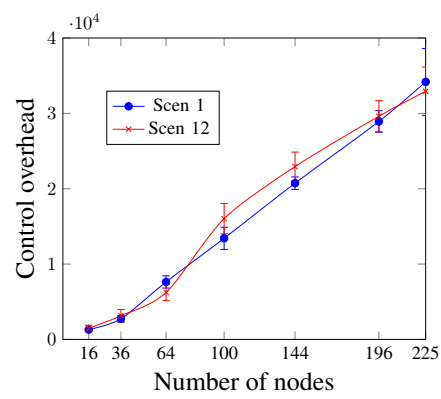


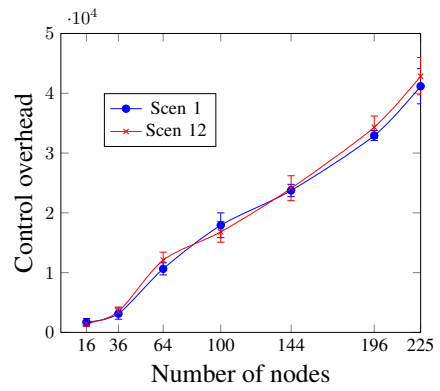
Figure 57 – Data delay, changing the topology



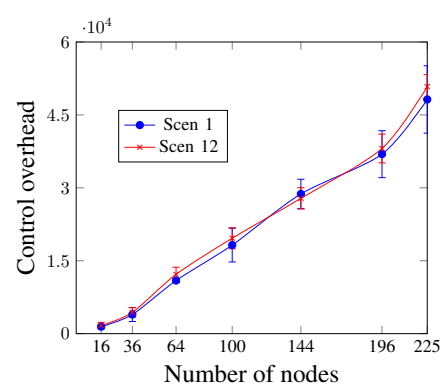
(a) One application



(b) Two applications



(c) Three applications



(d) Four applications

Figure 58 – Control overhead, changing the topology

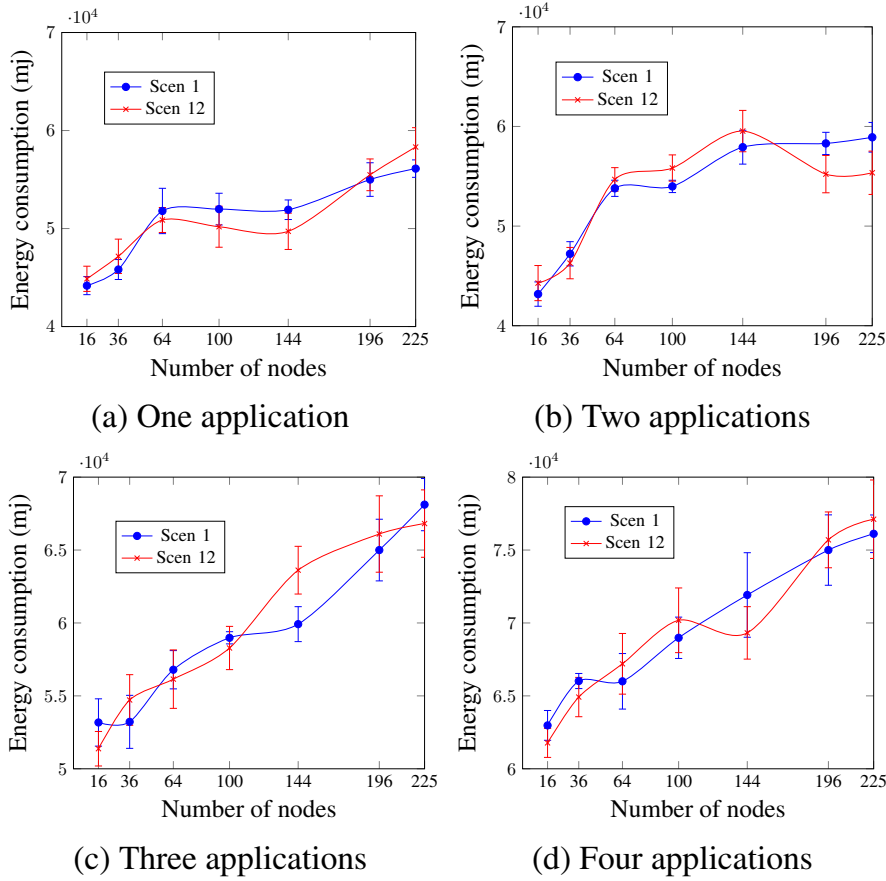


Figure 59 – Energy consumption, changing the topology

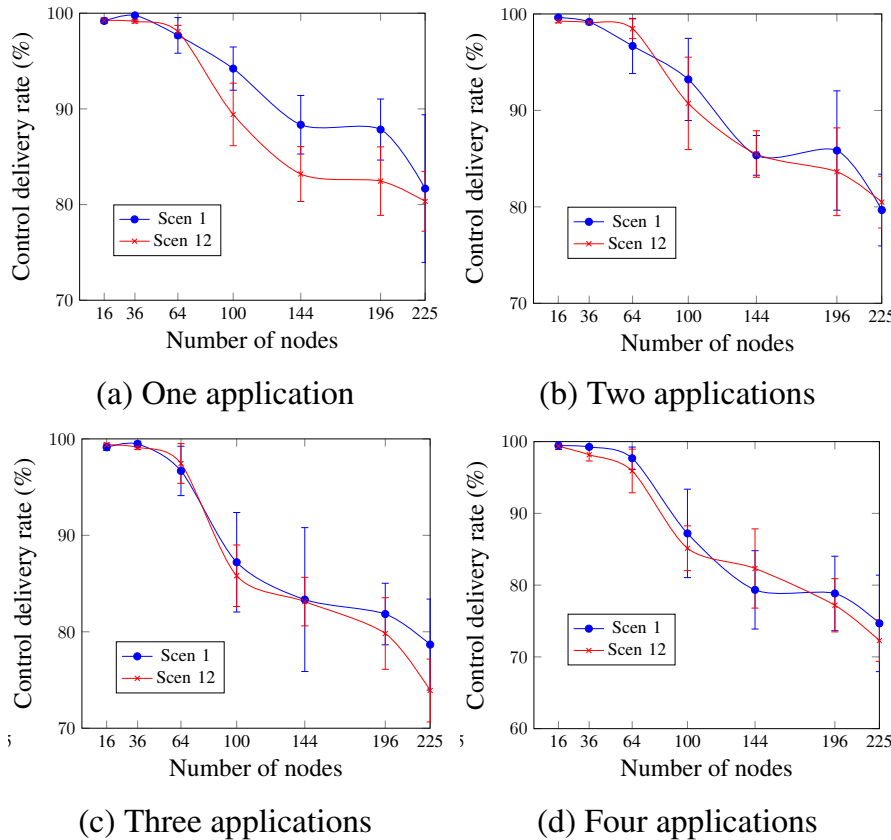


Figure 60 – Control delivery rate, changing the topology

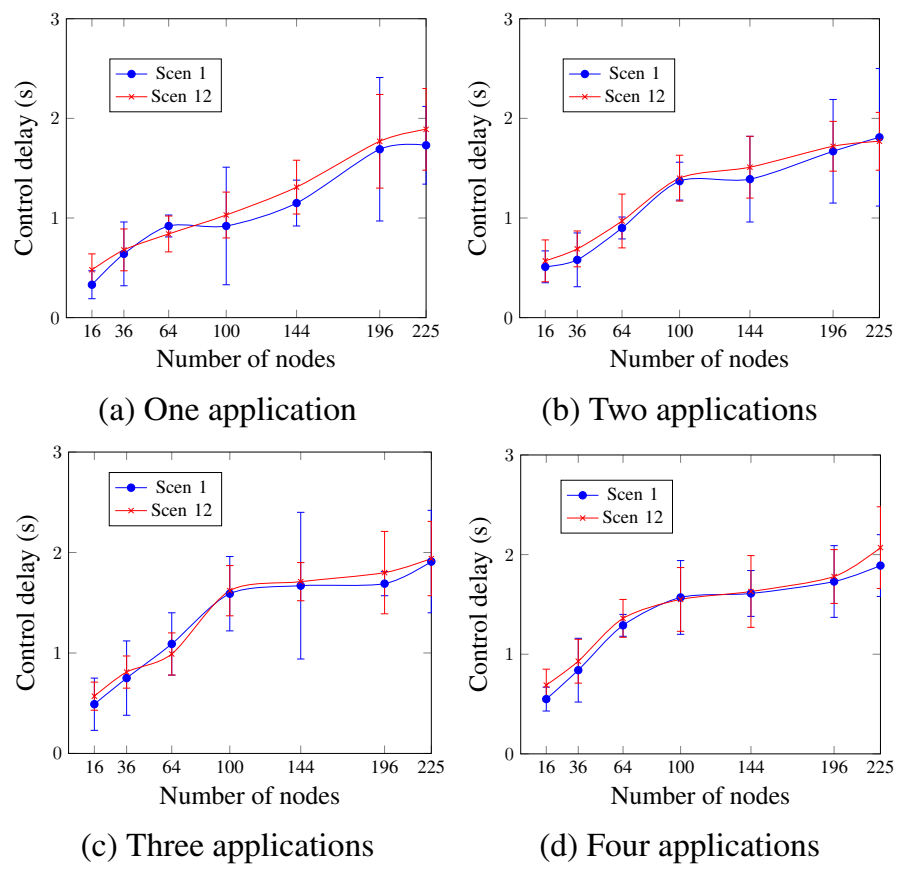


Figure 61 – Control delay, changing the topology

Table 11 – AA approach versus Application's QoS requirements

Evaluation metric	AA approach Vs Application's QoS requirements	Number of applications	
Data delivery rate	Ensured of up to 225 nodes	One App	Performance of the 1st App
	Ensured of up to 196 nodes	Two Apps	
	Ensured of up to 196 nodes	Three Apps	
	Ensured of up to 144 nodes	Four Apps	
Data delay	Ensured of up to 144 nodes	One App	
	Ensured of up to 100 nodes	Two Apps	
	Ensured of up to 100 nodes	Three Apps	
	Ensured of up to 144 nodes	Four Apps	
Data delivery rate	—	Two Apps	Performance of the 2nd App
	—	Three Apps	
	—	Four Apps	
Data delay	Ensured of up to 100 nodes	Two Apps	
	Ensured of up to 64 nodes	Three Apps	
	Ensured of up to 100 nodes	Four Apps	
Data delivery rate	Ensured of up to 225 nodes	Three Apps	Performance of the 3rd App
	Ensured of up to 225 nodes	Four Apps	
Data delay	—	Three Apps	
	—	Four Apps	

6.3 Case study

In this section, we present a detailed real case study where our system could be useful. The considered applications have been selected from the healthcare domain. Aiming to improve the patient's quality of life, WSNs were widely employed in the healthcare domain to continuously achieve the remote monitoring of the vital signs (TENNINA *et al.*, 2016; NITULESCU *et al.*, 2015; HASHMI *et al.*, 2014).

Rajba *et al.* (RAJBA *et al.*, 2013) used wireless sensors to monitor the patient in the hospital. Each hospital bed is provided with five sensors to monitor oxygen saturation, pulse rate, respiratory rate, arterial pressure, and body temperature. The data transmission time was selected depending on the patient's situation. Hashmi *et al.* (HASHMI *et al.*, 2014) proposed the Energy Efficient Vital Signs Monitoring System (VSMS) to continuously monitor the blood pressure diastolic (BPD), the blood pressure systolic (BPS), the oxygen saturation (SpO₂), and the heart beat (HB). The monitoring environment was also the hospital and the wireless sensors are attached with wearable belts. Nitulescu *et al.* (NITULESCU *et al.*, 2015) proposed a system to continuously monitor the respiration rate and oxygen saturation for pregnant women in the home during the last three months of their pregnancy. The proposed system notifies the medical staff when the measured signs are out of the normal range, and allows the pregnant woman to track her pregnancy situation in the real-time. Tennina *et al.* (TENNINA *et al.*, 2016) presented WSN4QoL, a WSN-based system to monitor the remote patient's vital signs. The proposed system monitors the body temperature in indoor environments. Attaoui *et al.* (ATTAOUI *et al.*, 2020) used wearable wireless sensors to monitor the electrocardiogram (ECG) in real-time. The experiments were carried out in a laboratory to validate the proposed approach, and the measured signs have been transmitted through WI-FI technology.

However, because of their limited resources, using the WSNs to ensure the hard requirements of the healthcare applications still represents a big challenge (TENNINA *et al.*, 2016). Moreover, it is important to note that some healthcare applications require high data rates. Since the data rate of the IEEE 802.15.4 standard is limited to 250 kbps (MUÑOZ *et al.*, 2018), not all healthcare applications could be satisfied using this standard. Table 12 (LATRÉ *et al.*, 2011) shows several healthcare applications with their required data rates. Applications such as electroencephalogram (EEG) and blood saturation require data rates lower than 250 kbps, whereas other applications such as electromyography (EMG) and artificial retina require higher data rates.

Suppose that some medical center is composed of several adjacent medical units. Each medical unit consists of several departments (ambulance, emergency, cardiology, respiratory, etc.). A nursing staff is available in each medical unit, and a single medical staff for the whole medical center is available and located in one of the units. All patients in the medical center are in a critical situation and need to be monitored continuously.

Table 12 – Healthcare applications with their data rates (LATRÉ et al., 2011)

Healthcare application	Data rate
ECG (12 leads)	288 kbps
ECG (6 leads)	71 kbps
EMG	320 kbps
EEG (12 leads)	43.2 kbps
Blood saturation	16 bps
Glucose monitoring	1.6 kbps
Temperature	120 bps
Cochlear implant	100 kbps
Artificial retina	50-700 kbps

According to his/her situation, a different set of vital signs for each patient will be monitored periodically. These signs are collected through a Wireless Body Area Network (WBAN) which consists of wireless sensors that could be either implanted inside the body or wearable by the patient's body (TENNINA et al., 2016). The measured signs of all the patients are periodically sent to the medical staff, which analyzes the received data and, if necessary, contacts the associated nursing staff to apply the adequate action (treatment). Each vital sign represents an application, and the frequency of sending data for each vital sign is determined by the medical staff and could be changed according to the patient's situation (HASHMI et al., 2014).

To show a real scenario in the healthcare domain where the AA approach could be applied, Table 13 depicts an example. Four vital signs (LATRÉ et al., 2011) (four applications running simultaneously) with data rates less than 250 kbps were considered. The DTR of these applications was adopted to be as Scen 1 (Table 8) and three different network sizes were considered: 64, 100, and 144 nodes. For each application and network size, Table 13 shows the obtained results in terms of delivery rate and delay.

Table 13 – An example of applying our adopted system in a real healthcare scenario

Vital signs	Data rate	Application's requirements	Data traffic rate	Network size	Obtained results	
					Delay (s)	Delivery rate (%)
1st App: EEG (12 leads)	43.2 kbps	Delivery rate Delay	1 packet per minute	64	0.26 s	98.59%
				100	0.61 s	94.41%
				144	0.89 s	93.74%
2nd App: Glucose monitoring	1.6 kbps	Delay	1 packet per 4 minutes	64	0.53 s	98.35%
				100	0.92 s	96.19%
				144	1.12 s	95.18%
3rd App: Blood saturation	16 bps	Delivery rate	1 packet per 8 minutes	64	0.81 s	99.13% -
				100	1.46 s	95.32% -
				144	1.73 s	93.25% -
4th App: Temperature	120 bps	Best-effort	1 packet per 10 minutes	64	3.38 s	81.28% -
				100	7.29 s	77.63% -
				144	10.04 s	64.12% -

Since the centralized controller has an external power supply, adopting an SDWSN system reduces the energy consumption and extends the network lifetime. Moreover, the AA approach dynamically adapts to the application's requirements.

6.4 Chapter summary

The AA approach was thoroughly evaluated in this Chapter. The evaluation process was carried out in two different strategies: i) in comparison to the ATI approach, where our approach increased the delivery rate by up to 28% and decreased the delay by up to 57%; and ii) in comparison to the application's QoS requirements, where several important parameters were considered and the AA approach was capable of ensuring the requirements for an increasing number of nodes. We could say, then, that the AA approach dynamically adapted to the application's requirements and proved its efficiency. Next, we presented a detailed real case study in the healthcare domain as an example to show where our proposed approach could be useful.

7 Final remarks

One of the main challenges of the WSNs is their limited resources in terms of communications, memory, processing, and energy. The major part of these resources is consumed in control and management tasks. The SDN paradigm moves the responsibility of all these tasks from the network nodes to a centralized controller, which has an external power supply. Since the controller has the whole network view, the SDN paradigm improves network flexibility and facilitates management, programmability, and resource sharing. Therefore, SDN has been highly adopted in the WSNs. However, the resulting SDWSN suffers from the competition for the limited resources between the control and data traffics.

TSCH technology slices the network into a number of timeslots and offers the possibility of sending the messages using several channel offsets (frequencies) in the same timeslot. Through this slicing, TSCH proved its efficiency with the limited resource's networks and was capable of improving their performance in terms of reliability and end-to-end delay. SDWSNs, thus, has been adopted on top of TSCH in several works in the literature. Revision of these works showed that the main gap is that none of them was aware of the application's QoS requirements for scalable SDWSNs without additional hardware. Furthermore, some works highlighted the benefits and importance of the traffic isolation over the SDWSNs using TSCH.

Our main objective was to design and evaluate an approach to adapt to the application's QoS requirements in the real-time. To achieve this goal, our contributions has been divided into several steps. First, as an initial step, we aimed to evaluate TSCH with its minimal mode of operation in comparison to ContikiMAC RDC strategy. Next, in order to evaluate the effect of control and data traffic isolation, the Control and Data Traffic Isolation (CDTI) approach has been proposed and evaluated. Since this approach improved the performance, we extended the isolation's concept to include the application's traffics by presenting the Application Traffic Isolation (ATI) approach. The main contribution of this work was the Application-Aware (AA) scheduling approach, which dynamically adapts to the application's requirements in terms of delivery rate and delay for scalable SDWSNs. It assigns more (or less) resources for each application according to its necessity, and the number of the added (or removed) resources is determined to maintain the requirement ensured without wasting the energy.

The AA approach has been thoroughly evaluated varying several parameters and considering up to 4 applications with different QoS requirements and network sizes of up to 225 nodes. Evaluation process considered the control and data planes and was carried out in comparison to both the ATI approach and application's requirements. The obtained

results showed that the AA approach outperformed the ATI approach by increasing the data delivery rate by up to 28% and decreasing the data delay by up to 57%. Moreover, it was capable of meeting the application's requirements in most cases. In this way, we can say that the research's objective has been achieved.

It is worth mentioning that these results were obtained adopting hard considerations. For instance, each of the sensor nodes executes up to 4 applications simultaneously. This highly increases the network traffic and makes it more difficult to meet the application's requirements.

Several works in the literature used dedicated timeslots to compact the congestion and collision. By this way, the performance is improved since there is no competition as each timeslot is assigned to an unique process (send, receive). However, such solutions could only adequate the small network sizes (most of them were evaluated considering network sizes of 5 to 10 nodes). Our approach, on the other hand, adopted shared times and was capable of ensuring the application's requirements for network sizes of up to 225 nodes. Although the potential congestion and collision are not completely removed using shared timeslots; nonetheless, their effect was significantly reduced using the traffic isolation concept. This shows a trade-off between the use of dedicated and shared timeslots.

7.1 Future work

To continue and improve this research, we present some suggestions which could be a future work:

- Modify the AA approach's scheduling calculation procedure by assigning more (or less) resources vertically (and not horizontally). In other words, our proposed approach adds (or removes) cells to the end of the slotframe, and each added cell has the same channel offset but with different timeslot number. The suggestion is to investigate the possibility and efficiency of adding (or removing) cells with the same timeslot number but with other channel offset.
- Improve the AA approach to consider other application's requirements such as the throughput and jitter.
- Design and evaluate another schedules to play the role of the Sch 0 depicted in Figure 26, as an attempt to optimize the performance and delay the necessity of a new scheduling.
- Evaluate the effect of combining both shared and dedicated timeslots to serve the considered applications. In this way, the dedicated timeslots could be used by the

nodes that execute a specific application with very hard requirements, and the shared timeslots could be used by the other nodes.

7.2 Publications

Four research papers have stemmed from the findings presented in this work. Among them, three have already been published, and one has been accepted for publication. These papers are enumerated below in chronological order:

1. "Control and data traffic isolation in SDWSN using IEEE 802.15.4e TSCH", published at the 2021 IEEE Statistical Signal Processing Workshop (SSP). Available online: <https://ieeexplore.ieee.org/document/9513822>. This paper presented the Control and Data Traffic Isolation (CDTI) approach, which is included in Chapter 4.
2. "Ensuring applications' traffic isolation using IEEE 802.15.4e TSCH through SDWSN slicing", published at the 2022 Symposium on Internet of Things (SIoT). Available online: <https://ieeexplore.ieee.org/document/10070197>. This paper presented the Application Traffic Isolation (ATI) approach, which extended the isolation approach to include the application's traffic. Results of this paper were presented in Chapter 4.
3. "Application-Aware Scheduling for IEEE 802.15.4e Time-Slotted Channel Hopping Using Software-Defined Wireless Sensor Network Slicing", published at the Sensors MDPI journal. Available online: <https://www.mdpi.com/1424-8220/23/16/7143>. The Application-Aware (AA) Scheduling approach as well as a throughout evaluation of both MCR and DTR factors have been provided in this paper. Results of this paper were included in Chapter 6.
4. "IEEE 802.15.4e TSCH traffic isolation approach impact on SDWSN's control plane performance", has been accepted for publication at 2023 IEEE Latin-American Conference on Communications (LATINCOM). This paper evaluated the impact of the application's traffic isolation on the control plane considering both CDTI and ATI approaches. Results of this paper have been shown in Chapter 4.

Bibliography

- AKYILDIZ, I. F. et al. Wireless sensor networks: a survey. *Computer networks*, Elsevier, v. 38, n. 4, p. 393–422, 2002. Cited 2 times on pages 19 and 25.
- ALVES, R. et al. IT-SDN: Improved architecture for SDWSN. In: *Proceedings of the XXXV Brazilian Symposium on Computer Networks and Distributed Systems, Belem, Brazil*. [S.l.: s.n.], 2017. p. 15–19. Cited 6 times on pages 25, 26, 28, 29, 30, and 67.
- ALVES, R. C.; MARGI, C. B. Ieee 802.15. 4e tsch mode performance analysis. In: IEEE. *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. [S.l.], 2016. p. 361–362. Cited on page 34.
- ALVES, R. C. et al. The cost of software-defining things: A scalability study of software-defined sensor networks. *IEEE Access*, IEEE, v. 7, p. 115093–115108, 2019. Cited on page 30.
- ASSIS, F. et al. DCS: Dilution-based Convergecast Scheduling in a TSCH network. *Ad Hoc Networks*, Elsevier, v. 146, p. 103173, 2023. Cited on page 40.
- ATTAOUI, A. E. et al. Wearable wireless sensors network for ECG telemonitoring using neural network for features extraction. *Wireless Personal Communications*, Springer, v. 111, p. 1955–1976, 2020. Cited on page 101.
- AZEEM, N. S. A. et al. Shared sensor networks fundamentals, challenges, opportunities, virtualization techniques, comparative analysis, novel architecture and taxonomy. *Journal of Sensor and Actuator Networks*, Multidisciplinary Digital Publishing Institute, v. 8, n. 2, p. 29, 2019. Cited 2 times on pages 26 and 35.
- BADDELEY, M. et al. Isolating sdn control traffic with layer-2 slicing in 6tisch industrial iot networks. In: IEEE. *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. [S.l.], 2017. p. 247–251. Cited 4 times on pages 20, 41, 43, and 49.
- BELLO, L. L. et al. Software-defined networking for dynamic control of mobile industrial wireless sensor networks. In: IEEE. *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*. [S.l.], 2018. v. 1, p. 290–296. Cited 3 times on pages 20, 41, and 43.
- CHANG, T. et al. 6tisch minimal scheduling function (msf). *Internet Engineering Task Force, Internet-Draft draft-ietf-6tischmsf-02*, 2019. Cited on page 43.
- CHANG, T. et al. *6TiSCH Minimal Scheduling Function (MSF)*. RFC 9033. [S.l.]: Internet Engineering Task Force Fremont, CA, USA, 2021. Cited 2 times on pages 22 and 33.
- CHEN, D.; VARSHNEY, P. K. Qos support in wireless sensor networks: a survey. In: CITESEER. *International conference on wireless networks*. [S.l.], 2004. v. 233, p. 1–7. Cited on page 26.

CHOI, K.-H.; CHUNG, S.-H. A new centralized link scheduling for 6tisch wireless industrial networks. In: *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*. [S.l.]: Springer, 2016. p. 360–371. Cited 2 times on pages 38 and 40.

DEAC, D. et al. Traffic aware scheduler for time-slotted channel-hopping-based ipv6 wireless sensor networks. *Sensors*, MDPI, v. 22, n. 17, p. 6397, 2022. Cited 2 times on pages 39 and 40.

DOHERTY, L.; SIMON, J.; WATTEYNE, T. Wireless sensor network challenges and solutions. *Microwave Journal*, Horizon House Publications, Inc., 685 Canton Street Norwood MA 02062 United . . . , v. 55, n. 8, p. 22–34, 2012. Cited on page 31.

DUNKELS, A. *The contikimac radio duty cycling protocol*. [S.l.]: Swedish Institute of Computer Science, 2011. Cited 2 times on pages 45 and 46.

DUNKELS, A.; GRONVALL, B.; VOIGT, T. Contiki-a lightweight and flexible operating system for tiny networked sensors. In: IEEE. *29th annual IEEE international conference on local computer networks*. [S.l.], 2004. p. 455–462. Cited 2 times on pages 23 and 67.

DUQUENNOY, S. et al. Tsch and 6tisch for contiki: Challenges, design and evaluation. In: IEEE. *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. [S.l.], 2017. p. 11–18. Cited on page 34.

DUQUENNOY, S. et al. Orchestra: Robust mesh networks through autonomously scheduled tsch. In: ACM. *Proceedings of the 13th ACM conference on embedded networked sensor systems*. [S.l.], 2015. p. 337–350. Cited 3 times on pages 37, 40, and 42.

ELSTS, A. et al. Tsch networks for health iot: Design, evaluation, and trials in the wild. *ACM Transactions on Internet of Things*, ACM New York, NY, USA, v. 1, n. 2, p. 1–27, 2020. Cited 2 times on pages 39 and 40.

FARREL, A.; VASSEUR, J.-P.; ASH, J. *A path computation element (PCE)-based architecture*. [S.l.], 2006. Cited on page 41.

FONSECA, R. et al. The collection tree protocol. *TinyOS TEP*, v. 123, n. 2, 2006. Cited on page 35.

GALLUCCIO, L. et al. Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks. In: IEEE. *2015 IEEE Conference on Computer Communications (INFOCOM)*. [S.l.], 2015. p. 513–521. Cited 2 times on pages 28 and 41.

GNAWALI, O. et al. Ctp: An efficient, robust, and reliable collection tree protocol for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, ACM, v. 10, n. 1, p. 16, 2013. Cited on page 29.

GUGLIELMO, D. D.; BRIENZA, S.; ANASTASI, G. Ieee 802.15. 4e: A survey. *Computer Communications*, Elsevier, v. 88, p. 1–24, 2016. Cited 3 times on pages 19, 25, and 31.

GUTIERREZ, J. A. et al. *Low-rate wireless personal area networks: enabling wireless sensors with IEEE 802.15. 4*. [S.l.]: John Wiley & Sons, 2011. Cited on page 19.

HALEPLIDIS, E. et al. *Software-defined networking (SDN): Layers and architecture terminology*. [S.l.], 2015. Cited 3 times on pages 9, 26, and 27.

- HAMZA, T.; KADDOUM, G. Enhanced minimal scheduling function for IEEE 802.15.4e TSCH networks. In: IEEE. *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. [S.l.], 2019. p. 1–6. Cited 2 times on pages 39 and 40.
- HARRINGTON, B. et al. Energy-efficient data replication in virtualized sensor networks with decoupled service model. In: IEEE. *2019 SoutheastCon*. [S.l.], 2019. p. 1–8. Cited on page 37.
- HASHMI, S. W. A. et al. Energy efficient vital signs monitoring system (vsms) using wireless sensor networks. *Wireless personal communications*, Springer, v. 76, p. 489–501, 2014. Cited 2 times on pages 101 and 102.
- HUYNH, T.; THEOLEYRE, F.; HWANG, W.-J. On the interest of opportunistic anycast scheduling for wireless low power lossy networks. *Computer Communications*, Elsevier, v. 104, p. 55–66, 2017. Cited 2 times on pages 38 and 40.
- IEEE. *802.15.4e-2012 - IEEE Standard for Local and metropolitan area networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer*. [S.l.]: IEEE Standard, 2012. Cited on page 19.
- II, K. K.; MOHAPATRA, P. Medium access control in wireless sensor networks. *Computer networks*, Elsevier, v. 51, n. 4, p. 961–994, 2007. Cited on page 25.
- JARSCHEL, M. et al. Modeling and performance evaluation of an openflow architecture. In: IEEE. *2011 23rd International Teletraffic Congress (ITC)*. [S.l.], 2011. p. 1–7. Cited on page 28.
- JAVAN, N. T.; SABAEI, M.; HAKAMI, V. IEEE 802.15.4e TSCH-based scheduling for throughput optimization: A combinatorial multi-armed bandit approach. *IEEE Sensors Journal*, IEEE, v. 20, n. 1, p. 525–537, 2019. Cited 2 times on pages 34 and 41.
- JEONG, S. et al. Tesla: Traffic-aware elastic slotframe adjustment in TSCH networks. *IEEE Access*, IEEE, v. 7, p. 130468–130483, 2019. Cited 2 times on pages 39 and 40.
- JIN, Y. et al. A centralized scheduling algorithm for IEEE 802.15.4e TSCH based industrial low power wireless networks. In: IEEE. *2016 IEEE Wireless Communications and Networking Conference*. [S.l.], 2016. p. 1–6. Cited on page 42.
- JURDZINSKI, T.; KOWALSKI, D. R.; STACHOWIAK, G. Distributed deterministic broadcasting in uniform-power ad hoc wireless networks. In: SPRINGER. *Fundamentals of Computation Theory: 19th International Symposium, FCT 2013, Liverpool, UK, August 19-21, 2013. Proceedings 19*. [S.l.], 2013. p. 195–209. Cited on page 40.
- KARAAGAC, A.; MOERMAN, I.; HOEBEKE, J. Hybrid schedule management in 6TISCH networks: The coexistence of determinism and flexibility. *IEEE Access*, IEEE, v. 6, p. 33941–33952, 2018. Cited 2 times on pages 38 and 40.
- KARL, H.; WILLIG, A. *Protocols and architectures for wireless sensor networks*. [S.l.]: John Wiley & Sons, 2007. Cited on page 25.
- KATONA, R. et al. Challenges in supporting diverse applications in a shared WSN: The motley middleware. In: IEEE. *2016 27th Irish Signals and Systems Conference (ISSC)*. [S.l.], 2016. p. 1–6. Cited 2 times on pages 36 and 37.

- KHALID, Z. et al. System design in sensor network virtualization for shaal. In: IEEE. *2014 5th International Conference on Intelligent Systems, Modelling and Simulation*. [S.l.], 2014. p. 636–641. Cited 2 times on pages 36 and 37.
- KHALID, Z. et al. Sensor virtualization middleware design for ambient assisted living based on the priority packet processing. *Procedia Computer Science*, Elsevier, v. 151, p. 345–352, 2019. Cited 2 times on pages 36 and 37.
- KHAN, I. et al. A multi-layer architecture for wireless sensor network virtualization. In: IEEE. *6th Joint IFIP Wireless and Mobile Networking Conference (WMNC)*. [S.l.], 2013. p. 1–4. Cited 2 times on pages 36 and 37.
- KHAN, I. et al. Wireless sensor network virtualization: A survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 18, n. 1, p. 553–576, 2015. Cited on page 35.
- KHARB, S.; SINGHROVA, A. A survey on network formation and scheduling algorithms for time slotted channel hopping in industrial networks. *Journal of Network and Computer Applications*, Elsevier, v. 126, p. 59–87, 2019. Cited 3 times on pages 9, 31, and 33.
- KOBO, H. I.; ABU-MAHFOUZ, A. M.; HANCKE, G. P. A survey on software-defined wireless sensor networks: Challenges and design requirements. *IEEE access*, IEEE, v. 5, p. 1872–1899, 2017. Cited 3 times on pages 19, 26, and 28.
- KOCAKULAK, M.; BUTUN, I. An overview of wireless sensor networks towards internet of things. In: IEEE. *2017 IEEE 7th annual computing and communication workshop and conference (CCWC)*. [S.l.], 2017. p. 1–6. Cited on page 19.
- KOU, L.; MARKOWSKY, G.; BERMAN, L. A fast algorithm for steiner trees. *Acta informatica*, Springer, v. 15, n. 2, p. 141–145, 1981. Cited on page 37.
- LATRÉ, B. et al. A survey on wireless body area networks. *Wireless networks*, Springer, v. 17, p. 1–18, 2011. Cited 3 times on pages 11, 101, and 102.
- LEE, T.-H. et al. Priority-based scheduling using best channel in 6tisch networks. *Cluster Computing*, Springer, v. 22, n. 1, p. 1023–1033, 2019. Cited 2 times on pages 39 and 40.
- LEONTIADIS, I. et al. Senshare: transforming sensor networks into multi-application sensing infrastructures. In: SPRINGER. *European Conference on Wireless Sensor Networks*. [S.l.], 2012. p. 65–81. Cited 2 times on pages 35 and 37.
- LEVIS, P.; CULLER, D. Maté: A tiny virtual machine for sensor networks. *ACM Sigplan Notices*, ACM New York, NY, USA, v. 37, n. 10, p. 85–95, 2002. Cited on page 35.
- LEVIS, P. et al. Tinyos: An operating system for sensor networks. In: *Ambient intelligence*. [S.l.]: Springer, 2005. p. 115–148. Cited on page 29.
- LEVIS, P. et al. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In: *Proc. of the 1st USENIX/ACM Symp. on Networked Systems Design and Implementation*. [S.l.: s.n.], 2004. v. 25. Cited on page 35.
- LUO, T.; TAN, H.-P.; QUEK, T. Q. Sensor openflow: Enabling software-defined wireless sensor networks. *IEEE Communications letters*, IEEE, v. 16, n. 11, p. 1896–1899, 2012. Cited on page 28.

- MAHMUD, A.; RAHMANI, R. Exploitation of openflow in wireless sensor networks. In: IEEE. *Proceedings of 2011 International Conference on Computer Science and Network Technology*. [S.l.], 2011. v. 1, p. 594–600. Cited on page 28.
- MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 38, n. 2, p. 69–74, 2008. Cited on page 27.
- MUNICIO, E. et al. Simulating 6tisch networks. *Transactions on Emerging Telecommunications Technologies*, Wiley Online Library, v. 30, n. 3, p. e3494, 2019. Cited on page 19.
- MUÑOZ, J. et al. Overview of ieee802. 15.4 g ofdm and its applicability to smart building applications. In: IEEE. *2018 Wireless Days (WD)*. [S.l.], 2018. p. 123–130. Cited on page 101.
- NITULESCU, A. et al. Integrated wireless sensor network for monitoring pregnant women. In: *MIE*. [S.l.: s.n.], 2015. p. 354–358. Cited on page 101.
- NKOMO, M. et al. Overlay virtualized wireless sensor networks for application in industrial internet of things: A review. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 18, n. 10, p. 3215, 2018. Cited on page 35.
- OLIVEIRA, B. T. D.; GABRIEL, L. B.; MARGI, C. B. Tinysdn: Enabling multiple controllers for software-defined wireless sensor networks. *IEEE Latin America Transactions*, IEEE, v. 13, n. 11, p. 3690–3696, 2015. Cited on page 29.
- OROZCO-SANTOS, F. et al. Enhancing sdn wise with slicing over tsch. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 21, n. 4, p. 1075, 2021. Cited 4 times on pages 20, 41, 42, and 43.
- OROZCO-SANTOS, F. et al. Scalability enhancement on software defined industrial wireless sensor networks over tsch. *IEEE Access*, IEEE, v. 10, p. 107137–107151, 2022. Cited 4 times on pages 20, 21, 42, and 43.
- OROZCO-SANTOS, F. et al. TSCH Multiflow Scheduling with QoS guarantees: A Comparison of SDN with common schedulers. *Applied Sciences*, Multidisciplinary Digital Publishing Institute, v. 12, n. 1, p. 119, 2022. Cited 3 times on pages 20, 42, and 43.
- ÖSTERLIND, F. et al. Cross-level sensor network simulation with cooja. In: *First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*. [S.l.: s.n.], 2006. Cited 2 times on pages 23 and 67.
- ÖSTERLIND, F. et al. Cross-level sensor network simulation with cooja. In: *First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*. [S.l.: s.n.], 2006. Cited on page 29.
- PALATTELLA, M. R. et al. On-the-fly bandwidth reservation for 6tisch wireless industrial networks. *IEEE Sensors Journal*, IEEE, v. 16, n. 2, p. 550–560, 2015. Cited 2 times on pages 38 and 40.
- RAJBA, S. et al. Wireless sensor networks in application to patients health monitoring. In: IEEE. *2013 IEEE Symposium on Computational Intelligence in Healthcare and e-health (CICARE)*. [S.l.], 2013. p. 94–98. Cited on page 101.

- REKIK, S. et al. Autonomous and traffic-aware scheduling for tsch networks. *Computer Networks*, Elsevier, v. 135, p. 201–212, 2018. Cited 2 times on pages 38 and 40.
- SAYJARI, T.; SILVEIRA, R. M.; MARGI, C. B. Control and data traffic isolation in SDWSN using IEEE 802.15.4e TSCH. In: IEEE. *2021 IEEE Statistical Signal Processing Workshop (SSP)*. [S.l.], 2021. p. 126–130. Cited 5 times on pages 19, 20, 22, 26, and 52.
- SAYJARI, T.; SILVEIRA, R. M.; MARGI, C. B. Ensuring applications' traffic isolation using IEEE 802.15.4e TSCH through SDWSN slicing. In: IEEE. *2022 Symposium on Internet of Things (SIoT)*. [S.l.], 2022. p. 1–4. Cited 5 times on pages 20, 22, 28, 62, and 67.
- TENNINA, S. et al. WSN4QoL: Wsns for remote patient monitoring in e-health applications. In: IEEE. *2016 IEEE International Conference on Communications (ICC)*. [S.l.], 2016. p. 1–6. Cited 2 times on pages 101 and 102.
- THUBERT, P.; PALATTELLA, M. R.; ENGEL, T. 6TiSCH centralized scheduling: When SDN meet IoT. In: *Proc. of IEEE Conf. on Standards for Communications & Networking (CSCN'15)*. [S.l.: s.n.], 2015. Cited 3 times on pages 19, 41, and 43.
- VEISI, F.; MONTAVONT, J.; THEOLEYRE, F. SDN-TSCH: Enabling Software Defined Networking for Scheduled Wireless Networks with Traffic Isolation. In: IEEE. *2022 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.], 2022. p. 1–7. Cited 3 times on pages 20, 42, and 43.
- VEISI, F.; MONTAVONT, J.; THEOLEYRE, F. Enabling centralized scheduling using software defined networking in industrial wireless sensor networks. *IEEE Internet of Things Journal*, IEEE, 2023. Cited 2 times on pages 42 and 43.
- VOSS, S. Steiner's problem in graphs: heuristic methods. *Discrete Applied Mathematics*, Elsevier, v. 40, n. 1, p. 45–72, 1992. Cited on page 37.
- WATTEYNE, T.; MEHTA, A.; PISTER, K. Reliability through frequency diversity: why channel hopping makes sense. In: *Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*. [S.l.: s.n.], 2009. p. 116–123. Cited on page 32.
- WINTER, P. Steiner problem in networks: a survey. *Networks*, Wiley Online Library, v. 17, n. 2, p. 129–167, 1987. Cited on page 37.
- WINTER, T. et al. Rpl: Ipv6 routing protocol for low-power and lossy networks. *rfc*, v. 6550, p. 1–157, 2012. Cited on page 31.
- YU, Y. et al. Supporting concurrent applications in wireless sensor networks. In: *Proceedings of the 4th international conference on Embedded networked sensor systems*. [S.l.: s.n.], 2006. p. 139–152. Cited 2 times on pages 35 and 37.