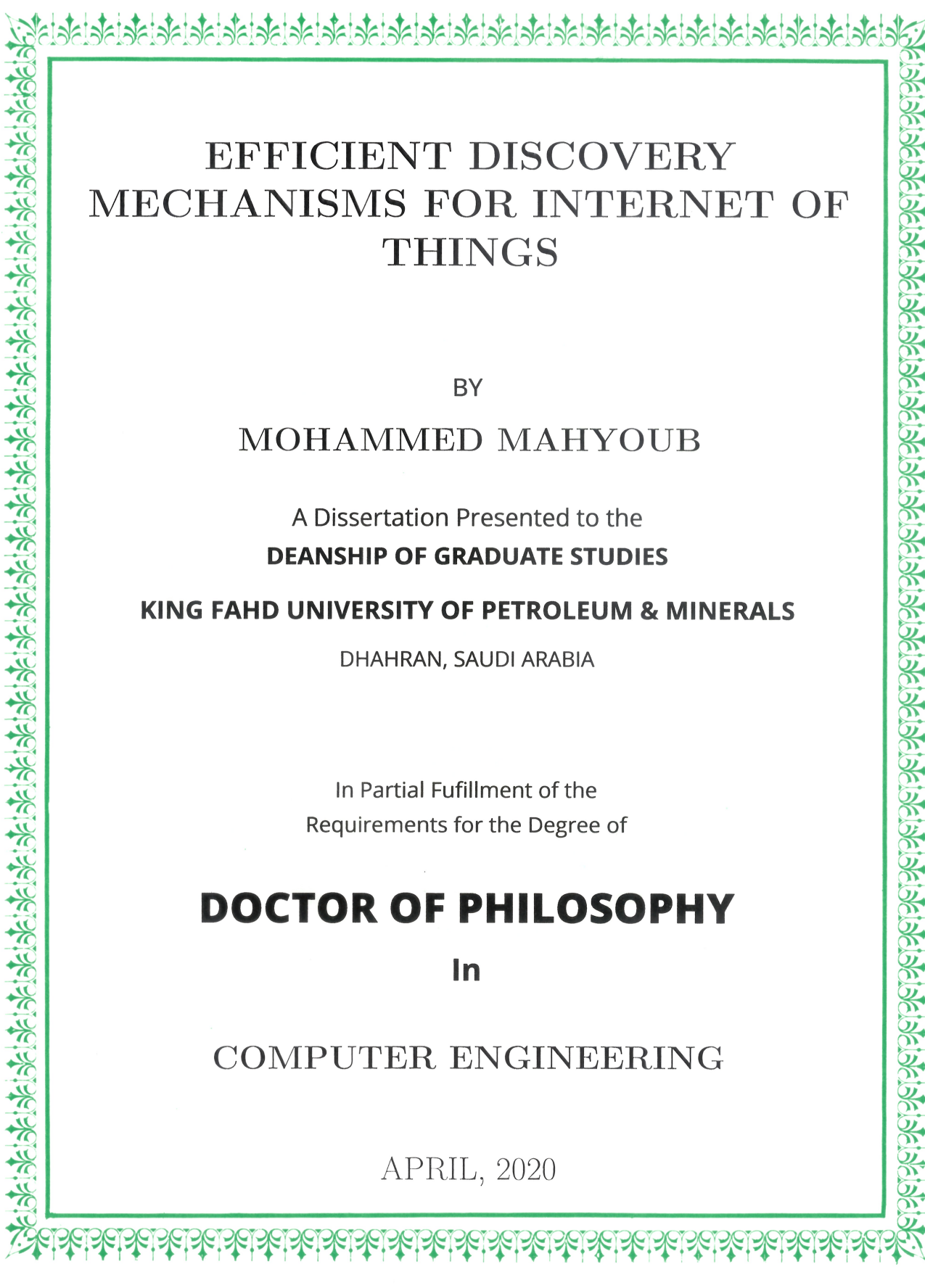




EFFICIENT DISCOVERY MECHANISMS FOR INTERNET OF THINGS

BY
MOHAMMED MAHYOUB

A Dissertation Presented to the
DEANSHIP OF GRADUATE STUDIES
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
In
COMPUTER ENGINEERING

APRIL, 2020

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This dissertation, written by **MOHAMMED MAHYOUB** under the direction of his dissertation adviser and approved by his dissertation committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING**.

Dissertation Committee



August 16th, 2020

Dr. Ashraf Mahmoud (Adviser)

Marwan Abu-Amara August 10, 2020

Dr. Marwan Abu-Amara (Co-Adviser)



Aug 9, 20

Dr. Tarek Sheltani (Member)

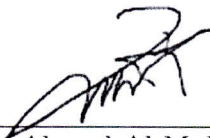
Shokri Selim 9-8-2020

Dr. Shokri Selim (Member)



09/08/2020

Dr. Moataz Ahmed (Member)



Dr. Ahmed Al-Mulhem
Department Chairman



Dr. Suliman S. Al-Homidan
Dean of Graduate Studies



Date

©Mohammed Mahyoub

محمد محيoub

2020

To my beloved father soul

ACKNOWLEDGMENTS

This thesis is the product of a lot of hard work and study, but it is also the result of guidance and support that I have kindly received from many people: my professors, colleagues, family, and friends to whom I am indebted.

I would like to thank my advisor Dr. Ashraf Mahmoud. He has always been there to listen to my questions, to provide me with constructive criticism on my research, and to read my poorly written drafts. I also thank my doctoral committee members: Dr. Tarek Sheltami, Dr. Shokri Salim, Dr. Marwan Abu-Amara, and Dr. Moataz Ahmed for their insightful comments and advice on my thesis, and final defense.

I am so grateful to all my family members, especially my beloved parents, for their continuous love and support. I can't possibly express my gratitude to my beloved wife Aisha Al-Shameri. Her care and love have always been the source of my ambitious aspirations. Also to my children who have been a constant source of inspiration with their positivity and enthusiasm.

My sincere appreciation goes to the Higher Education Ministry of Yemen and to King Fahd University of Petroleum and Minerals (KFUPM) for providing me with the opportunity to continue my graduate studies. Special thanks to the Computer Engineering Department at KFUPM for their invaluable support.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	vi
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF ABBREVIATIONS	xv
ABSTRACT (ENGLISH)	xx
ABSTRACT (ARABIC)	xxii
CHAPTER 1 INTRODUCTION	1
1.1 Internet of Things (IoT)	2
1.2 Thesis Motivations	4
1.2.1 Service Discovery (SD)	5
1.2.2 Routing Discovery (RD)	7
1.3 Thesis Aims and Objectives	9
1.4 Thesis Main Contributions	10
1.5 Thesis Organization and Overview	12
CHAPTER 2 BACKGROUND	14
2.1 Service Discovery in IoT	15
2.1.1 Service Discovery Objectives	16
2.1.2 Service Discovery Requirements	16
2.1.3 The multicast DNS and DNS-SD Protocols	18

2.2	Routing Discovery in IoT	23
2.2.1	Routing Discovery Challenges and Requirements	23
2.2.2	Traditional Routing Protocols for IoT	25
2.2.3	Overview of IPv6 Routing Protocol for LLNs (RPL)	27
2.2.4	Testing the Trickle Algorithm (TA) of ContikiOS	34
2.3	Evaluation Methodologies	40
2.3.1	Technologies for Experiments	40
2.3.2	Calculation of Energy Consumption	43
CHAPTER 3	LITERATURE REVIEW	45
3.1	Service Discovery in the Literature	45
3.1.1	Service Discovery Protocols for IoT	45
3.1.2	Adaptations of mDNS/DNS-SD	48
3.1.3	Enhancements of mDNS/DNS-SD	50
3.2	Enhancements of N2N Routing Discovery	51
CHAPTER 4	ENHANCED MDNS	61
4.1	Chapter Motivations	61
4.2	Chapter Contributions	62
4.3	System Models	63
4.3.1	Node Model	63
4.3.2	Network Model	65
4.4	Overview of Enhanced mDNS (EmDNS)	65
4.4.1	EmDNS Design Overview	66
4.4.2	The Persistent-based Selection Stage (PSS)	66
4.4.3	The Simultaneous Start-up Enhancement (SSE)	69
4.4.4	The Announcement Suppression Mechanism (ASM)	72
4.5	Performance Evaluation	76
4.5.1	Simulation Set-up	76
4.5.2	Impact of RDC on mDNS Performance	78
4.5.3	Implementation Overhead	80

4.5.4	Load Balancing and Fairness	81
4.5.5	Results on Protocol Performance	84
4.5.6	Control Traffic Overhead	89
4.5.7	Energy Consumption	90
4.5.8	EmDNS Backward Compatibility Support	92
4.6	Chapter Summary	94
CHAPTER 5 HYBRID RPL		97
5.1	Chapter Motivations	97
5.2	Chapter Contributions	99
5.3	Network Model	101
5.4	Overview of Hybrid RPL (HRPL)	103
5.4.1	HRPL Core Phases	103
5.4.2	Reporting of Neighboring Information	105
5.4.3	HRPL Neighboring Option	106
5.4.4	Shortest Path Construction	109
5.5	Performance Evaluation	111
5.5.1	Performance Metrics	112
5.5.2	Simulation Model and Configurations Set-up	114
5.5.3	Implementation Overhead	116
5.5.4	Support for Hybrid Networks	118
5.5.5	Comparison with NSRPL and SRPL Modes	120
5.5.6	Comparison with Opportunistic RPL (ORPL)	139
5.6	Chapter Summary	144
CHAPTER 6 CONCLUSIONS AND FUTURE WORK		146
6.1	Thesis Main Contributions	147
6.1.1	Enhanced mDNS (EmDNS)	147
6.1.2	Hybrid RPL (HRPL)	148
6.2	Limitations and Future work	149
6.2.1	Expansion of EmDNS for Multi-hop Networks	149

6.2.2	Expansion of the Analytical Model for Trickle Algorithm (TA) .	151
6.2.3	Expansion of HRPL Experimental Analysis	152
6.2.4	Integrating SD and RD Systems	152
6.3	List of Publications	152
6.3.1	Journal Papers	153
6.3.2	Conference Papers	153
REFERENCES		154
VITAE		176

LIST OF TABLES

2.1	Summary of the RRs describing the service <i>temp1</i> provided by the <i>node1</i>	22
2.2	Explanations of some filed of RRs in Domain Name System (DNS)- Service Discovery (SD)	22
2.3	The satisfaction decisions based on the big- <i>O</i> conditions	26
2.4	Summary for performance satisfaction of the traditional Internet En- gineering Task Force (IETF) protocols	27
2.5	General configurations for experiments	39
3.1	Comparative summary for the SD protocols in Internet of Things (IoT)	49
3.2	Properties summary of Node-to-Node (N2N) enhancements in the lit- erature studies	60
4.1	Used notations summary	63
4.2	Multicast addresses used by multicast Domain Name System (mDNS) protocol	65
4.3	Simulation parameters	77
5.1	The summary of the notations used in this chapter	102
5.2	General configurations for experiments	117
5.3	Time percentage for the components in Hybrid RPL (HRPL) delay . .	136

LIST OF FIGURES

1.1	IoT protocols stack	4
2.1	The mDNS/DNS-SD Operational Phases	19
2.2	The mDNS/DNS-SD Zero-configuration Phase	20
2.3	Example of TA with and without synchronized intervals using $K = 2$ and $I = I_{max}$	31
2.4	The RPL-based DODAG topology with NSRPL and SRPL modes . . .	34
2.5	The typologies of the studied network using different scenarios	37
2.6	The average sending probability $P(S)$ as a function of D for the three scenarios	39
2.7	Some of IoT-enabling Protocols implemented in Contiki Operating Sys- tem (ContikiOS)	42
4.1	The configuration phase for the proposed Enhanced mDNS (EmDNS) .	67
4.2	The Persistent-based Selection Stage (PSS) flow chart	68
4.3	Timelines of mDNS and EmDNS protocols for the simultaneous start- up events	70
4.4	Impact of different Channel Check Rates (CCRs) on mDNS perfor- mance as a function of network size	79
4.5	Memory footprint overhead	81
4.6	Per-node performance evaluation comparing mDNS and EmDNS for η $= 10$	82
4.7	Fairness index using different performance metrics as a function of net- work size η	84

4.8	Identity Uniqueness Percentage (IUP) as a function of network size η . . .	85
4.9	LHR as a function of $MAX_ENTRIES$ when $\eta = 20$	87
4.10	Network Convergence Time (NCT) as a function of network size η . . .	88
4.11	Overall number of control packets as a function of network size η . . .	89
4.12	Total network energy usage by different components as a function of network size η	91
4.13	The hybrid network performance with different EmDNS node deploy- ments.	93
5.1	HRPL Route Request (HRREQ) payload format	104
5.2	HRPL Route Reply (HRREP) payload format	104
5.3	HRPL neighboring information option	107
5.4	The topology of the network under study	114
5.5	Total available memory with Wismote platform and memory foot- print overhead for HRPL, Storing RPL-based mode (SRPL), and Non- Storing RPL mode (NSRPL) implementations	117
5.6	The hybrid network performance with different HRPL node deploy- ments at traffic intensity of 4 Packets Per Minute (PPM)	119
5.7	RPL events frequency under the studied modes as a function of the traffic intensity.	122
5.8	Routing control traffic generated using the studied modes as a function of the traffic intensity.	123
5.9	Amount of DIO packets transmitted per node at traffic intensities of 1PPM and 4PPM	124
5.10	Amount of Destination Advertisement Object (DAO) packets trans- mitted per node at the traffic of 4PPM.	126
5.11	Amount of DAO packets transmitted per node at the traffic of 4PPM. The node IDs on the x-axis are ordered based on the node's distance from the Border Router (BR)	126
5.12	Fairness index for control messages with varying traffic intensity	127

5.13	Packet Delivery Ratio (PDR) using the studied modes as a function of the traffic intensity.	128
5.14	The packet loss percentage caused by routing errors, MAC collisions, and buffer overflow	129
5.15	Cumulative Distribution Function (CDF) for hops and Medium Access Control (MAC) transmissions	132
5.16	The average number of hops and MAC transmissions count per received packet using the studied modes.	134
5.17	The average delay per received packet as a function of traffic intensity.	135
5.18	Energy consumption using HRPL, NSRPL, and SRPL modes as a function of the traffic intensity	138
5.19	PDR using HRPL and Opportunistic RPL (ORPL) as a function of 1PPM and 4PPM traffic intensities	141
5.20	The average MAC transmissions per received packet	141
5.21	The average delay per received packet as a function of traffic intensity .	142
5.22	The Energy consumption per received packet as a function of traffic intensity	143
5.23	The Fairness Index (FI) of the Energy consumption per received packet as a function of traffic intensity	144

LIST OF ABBREVIATIONS

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Network
AS	Announcement Stage
ASM	Announcement Suppression Mechanism
BF	Bloom Filter
BR	Border Router
CA	Collision Avoidance
CCI	Channel Check Interval
CCR	Channel Check Rate
CDF	Cumulative Distribution Function
CoAP	Constrained Application Protocol
ContikiOS	Contiki Operating System
CoRE	Constrained Representational State Transfer Environments
CPS	Cyper-Physical System
CPU	Central Processing Unit
CSMA	Carrier Sense Multiple Access
CTO	Control Traffic Overhead

CTP	Collection Tree Protocol
DAO	Destination Advertisement Object
DHCP	Dynamic Host Configuration Protocol
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DNS	Domain Name System
DODAG	Destination Oriented Directed Acyclic Graph
DRO	Discovery Response Object
EmDNS	Enhanced mDNS
EN	Edge Node
ETX	Expected Transmission Count
EWMA	Exponentially Weighted Moving Average
FI	Fairness Index
GPS	Global Positioning System
HNO	HRPL Neighboring Option
HRPL	Hybrid RPL
HRREP	HRPL Route Reply
HRREQ	HRPL Route Request
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force

IoT	Internet of Things
IP	Internet Protocol
IUP	Identity Uniqueness Percentage
IWT	Initial Waiting Time
KFUPM	King Fahd University of Petroleum and Minerals
LHR	Local Hit Ratio
LLN	Low-power and Lossy Network
LMP	Local Minimum Problem
LQI	Link Quality Indicator
MAC	Medium Access Control
mDNS	multicast Domain Name System
MOP	Mode Of Operation
MPL	Multicast Protocol for Low-Power and Lossy Networks
MTU	Maximum Transmission Unit
N2N	Node-to-Node
NCT	Network Convergence Time
ND	Neighbor Discovery
NSRPL	Non-Storing RPL mode
OF	Objective Function
ORPL	Opportunistic RPL
P2P	point-to-point

PDR	Packet Delivery Ratio
Ph.D.	Doctor of Philosophy
PPM	Packets Per Minute
PS	Probing Stage
PSS	Persistent-based Selection Stage
RAM	Random Access Memory
RCT	Routing Control Traffic
RD	Routing Discovery
RDC	Radio Duty Cycle
RDO	Route Discovery Object
RFC	Request For Comment
RFID	Radio Frequency IDentification
RN	Reference Node
ROLL	Routing Over LLN
ROM	Random Only Memory
RPL	IPv6 Routing Protocol for LLNs
SD	Service Discovery
SOA	Service Oriented Architecture
SRH	Source Routing Header
SRPL	Storing RPL-based mode
SS	Selection Stage

SSE	Simultaneous Start-up Enhancement
SSSP	Single Source Shortest Path
TA	Trickle Algorithm
TCP	Transmission Control Protocol
TTL	Time-To-Live
UDGM	Unit Disk Graph Medium
UDP	User Datagram Protocol
UMHRF	Updated Metric with the Hysteresis Reporting Function
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
WG	Working Group
WSN	Wireless Sensor Network
XML	Extensible Markup Language
ZeroConf	Zero Configuration Networking

DISSERTATION ABSTRACT

NAME: Mohammed Mahyoub
TITLE OF STUDY: Efficient Discovery Mechanisms for Internet of Things
MAJOR FIELD: Computer Engineering
DATE OF DEGREE: April, 2020

Service Discovery (SD) and Routing Discovery (RD) in Low-power and Lossy Networks (LLNs) are crucial in the development of the Internet of Things (IoT). The design of SD and RD can be hampered by the underlying limitations and challenges of LLNs which motivate the quest for efficient solutions. This thesis analyzes the most promising SD and RD protocols to identify some of their potential gaps, and then proposes cost-effective, time-efficient, lightweight, and backward-compatible mechanisms to fill such gaps. Firstly, this thesis proposes an Enhanced multicast Domain Name System (EmDNS) which augments mDNS, the standard SD protocol, with three improvements: the persistent-based selection stage, the simultaneous start-up enhancement, and the announcement suppression mechanism. Through the evaluation of the fairness index for a wide range of number of nodes, we demonstrate that EmDNS

provides better load balancing for the collection of nodes in the network relative to mDNS. Furthermore, results indicate that while EmDNS produces a memory footprint in RAM that is only 4.2% more than that for mDNS, EmDNS manages to significantly reduce the number of control packets by an average of 34.8% for the considered range of network size. Accordingly, the corresponding reduction in energy consumption for radio transmissions and micro-controller is 46.7% and 47.5%, respectively. Secondly, the IPv6 Routing Protocol for LLNs (RPL) has been standardized as the RD protocol for LLNs. However, it does not efficiently route the data between arbitrary nodes within the LLN domain. To deal with such a shortcoming, this thesis proposes a Hybrid RPL (HRPL) extension in which the RPL-based forwarding structure and the link-state routing approach are integrated to discover more efficient routes between arbitrary nodes. The salient features of HRPL include an optimized data forwarding between arbitrary nodes, the use of the available RPL-enabling control messages, and the support of backward-compatibility. HRPL is evaluated in a grid-like network under different traffic intensities. Results show that HRPL increases the packet delivery ratio by 22.3% relative to the RPL. While HRPL consumes RAM size that is 16.7% more than that for RPL, HRPL retains low routing control cost which is 13.8% less than that for RPL. It reduces the number of hops and MAC transmissions to deliver packets between arbitrary nodes by 51.5% and 55.4%, respectively. Accordingly, the corresponding reduction in the total consumed energy is 21.3%. Furthermore, HRPL provides higher balance, relative to RPL, in terms of routing control messages and energy consumption across all nodes in the network. The proposed extensions are implemented on Contiki operating system and evaluated on Cooja emulator.

ملخص الرسالة

الاسم:

محمد عايض أحمد مهيوب

عنوان الدراسة:

آليات إكتشاف فعالة لشبكات إنترنت الأشياء

التخصص:

هندسة الحاسب الآلي

تاريخ الدرجة العلمية: رمضان 1441

يُعَدُّ إكتشافُ الخدمات والمسارات في الشبكات منخفضة الطاقة أمرًا بالغ الأهمية في تطوير إنترنت الأشياء. هذه الرؤية تهدف إلى توصيل كل الأجهزة المستخدمة في حياتنا بالإنترنت وتمكينهم من التفاعل مع بعضهم البعض ومع البيئة المحيطة. إن لهذا التفاعل عدة أشكال منها على سبيل المثال لا الحصر تبادل الخدمات واتخاذ القرارات ومراقبة البيئة المحيطة والتحكم بها أيضا. من أجل تحقيق هذه الرؤية فإنه يلزم وجود بروتوكول لإكتشاف الخدمات المُقدَّمة في الشبكة ليقوم بتهيئة الأجهزة والإعلان عن خدماتهم للآخرين بطريقة آليّة ويلزم كذلك وجود بروتوكول آخر لإكتشاف المسارات وتسهيل التواصل بين الأجهزة لتبادل هذه الخدمات وبأقل تكلفة. تجدر الإشارة إلى أن تصميم البروتوكولات آنفة الذكر يتأثر بمحدودية الموارد التي تعاني منها هذه الشبكات مثل محدودية الطاقة والمعالجة والتخزين وكذلك الفقد العالي للبيانات وصغر حجم البيانات الممكن إرسالها والمدى الذي يمكن تغطيته. في هذه الرسالة قمنا بتحليل أشهر البروتوكولات المستعملة في هذا الإطار لتحديد مواطن الضعف التي من شأنها تؤثر سلباً على جودة الخدمة المقدمة في هذه الشبكات وتستهلك مواردها المحدودة. بعد ذلك قمنا بتطوير تقنيات جديدة تساهم في التغلب على بعض المشاكل المرتبطة بهذه البروتوكولات في جوانب محدده. وللتحقق من كفاءة التقنيات المقترحة في هذه الرسالة فقد قمنا بإجراء العديد من تجارب المحاكاه حيث أثبتت النتائج فعالية وكفاءة هذه المقترحات من حيث إستهلاك الطاقة ومعدل نقل البيانات وسرعة إيصالها وكذلك توزيع الأحمال بين أجهزة الشبكة.

CHAPTER 1

INTRODUCTION

The concept of the Internet of Things (IoT) was first proposed and coined by the co-founder of the Auto-ID Center at MIT, Kevin Ashton, back in early 2000 [1]. This concept basically envisaged a world where everything in our life can be managed by the Internet, specifically using technology such as Radio Frequency IDentification (RFID). Moving forward, this idea expanded and broadened because of the various advances in related technologies such as wireless networking, sensors and actuators, and microcontrollers. Having said that, many technical challenges should be tackled before this vision is widely accepted. Thus, the main task of developers is to provide efficient IoT solutions while considering such challenges.

In this introductory chapter, a brief introduction to IoT is offered by outlining its main building blocks, for example, low power networks, and its protocols stack. Next, the need for service and routing discovery mechanisms in IoT is discussed. Then, an overview of the main objectives and contributions of this thesis is given. This chapter concludes with the organization of this thesis.

1.1 Internet of Things (IoT)

The IoT paradigm has gained broad-spectrum attention due to its high potential influence on several daily-life aspects [2]. In this context, things are for everyday physical devices embedded with sensing/actuating hardware to incorporate the smartness to these devices. These devices are capable of interacting with the surrounding environment, offering smart services in real-time, and communicating among themselves while they can be uniquely addressed and accessed remotely through the Internet. The smartness and connectivity features integrated with IoT devices bring a plethora of smart applications into our life such as smart grids [3], smart homes [4], smart cities [5], automated industrial monitoring [6], just to name a few.

Most IoT deployments rely on devices with limited resources to meet the large scale and low-cost demands forming Low-power and Lossy Networks (LLNs). Devices of LLNs are battery-powered with low storage and processing capabilities. The networks connecting them, on the other hand, exhibit delivery failure with a high probability, small packet size, low data rates, and variable link quality conditions with short communication ranges. For example, the Wisemote platform is equipped with an 8-bit MSP430 microcontroller working at 16 MHz maximum, a 16 Kbyte Random Access Memory (RAM) and a 128 Kbyte Random Only Memory (ROM) for storage. For communication, Wisemote employs a CC2520 chip and a low-power 2.4 GHz IEEE 802.15.4 transceiver with a theoretical data rate of 250 Kbps and a 127-byte maximum packet size.

Structurally, IoT requires software architectures which take into account the possi-

ble aspects that consume the constrained resources of LLNs. These aspects could be in the form of unneeded packet transmissions, protocol overhead, and/or non-optimized communication patterns. Considering these aspects, various standardization organizations, most notably the Institute of Electrical and Electronics Engineers (IEEE) and Internet Engineering Task Force (IETF), focused their efforts to create a framework of the protocols targeting communications in developing systems.

The standard IEEE 802.15.4 that characterizes the low-power MAC and Physical (PHY) layers, has had the most impact which several IoT technologies have been developed on. The Internet Protocol (IP), on the other hand, has been shown to be a lightweight, stable, and highly scalable architecture for communication. It operates with small, battery-powered devices, and already connects billions of devices. Given that most commercial wireless sensor deployments nowadays are based on the IEEE 802.15.4 standard, a better solution is to reuse the existing IP architecture for IEEE 802.15.4-based devices. However, existing traditional Internet-enabling protocols are not fully optimized for low-power networks. This is due to their lengthy headers and meta-data, implementation complexity and the excessive transmissions required for reliability. Wasting resources this way would limit the capability of these protocols to operate over LLNs unless adequate adaptations are applied [1]. These issues were the catalyst for the creation of different IETF Working Groups (WGs) such as Zero Configuration Networking (ZeroConf), Constrained Representational State Transfer Environments (CoRE), IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN), and Routing Over LLN (ROLL) [7]. The standardized protocols resulting from the works of these groups and their close collaboration were crucial to the development

of IoT from a technological viewpoint. Figure 1.1 depicts some protocols spanned on different layers of the IoT stack.

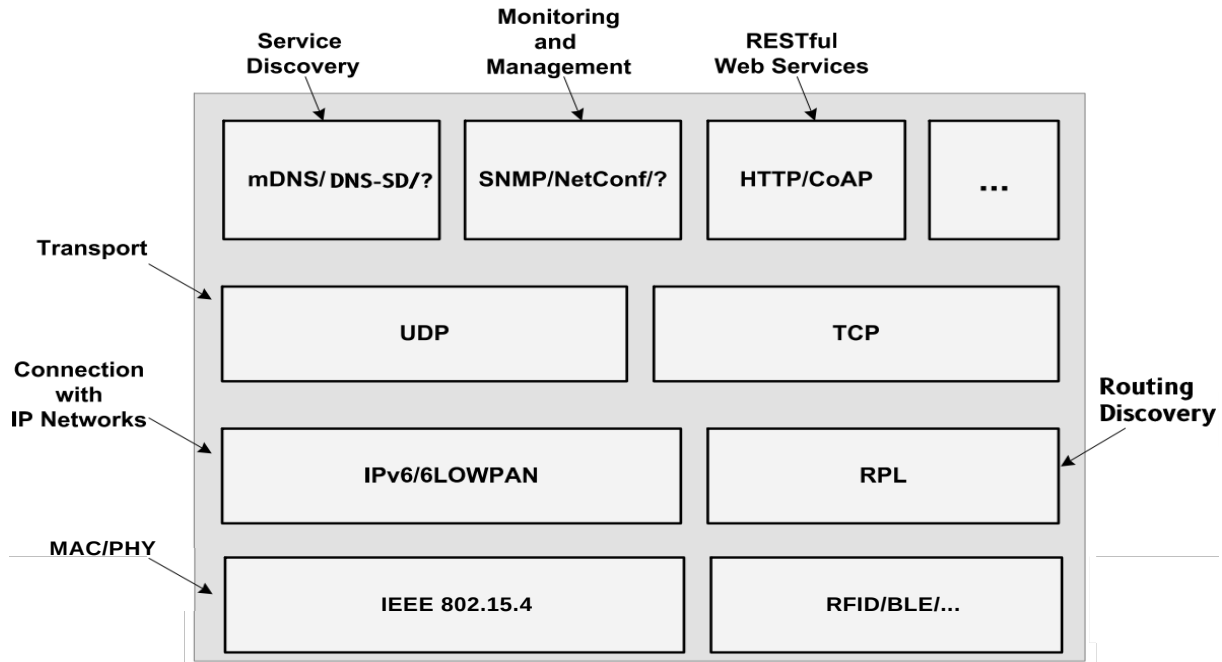


Figure 1.1: IoT protocols stack

1.2 Thesis Motivations

LLN-based applications provide many important daily services for individuals and communities. These services can be introduced locally or globally, for example, the monitoring of the surrounding environment or the actuating of switches (i.e. turning things on and off). In addition, monitoring health [8], buildings, atmospheric phenomena [9], [10] such as pollution levels, temperature and humidity, place noise [11], and smart parking [12] are some of these services. Furthermore, services such as these could be offered in more combined forms to offer more complex functionalities [13].

To support this orientation, the successful deployment of LLN applications requires

two main mechanisms which are a Service Discovery (SD) and a Routing Discovery (RD). On the one hand, the SD mechanism assists devices to configure, register and advertise themselves and their services in a particular LLN. Enabling the SD is a prerequisite for discovering the described devices and services they host [14]. On the other hand, the RD mechanism is needed to enable the discovered devices to communicate using the least costly routes. However, the design of SD and RD solutions can be hampered by the underlying challenges caused by the limitations of LLN resources. This thesis analyzes the most common existing SD and RD solutions considering the challenges and requirements of constrained LLNs, to identify some of their potential gaps and then attempts to propose more efficient mechanisms to fill such gaps. The remaining part of this section is devoted to explaining the SD and RD, and the most common literature protocols supporting them. From now on throughout this thesis, the terms “device” and “node” are used interchangeably

1.2.1 Service Discovery (SD)

Given that LLN nodes are expected to continuously join and leave networks, the SD must provide a mechanism to configure, register and unregister nodes and their services so that nodes can maintain enough information about each other. To realize this goal efficiently, the SD mechanism needs to minimize the required configuration. This could be achieved by automatically detecting nodes and their services in a generic enough way, which enhances usability and reduces the amount of required development to handle different system setup and usage scenarios. Manual configuration, however, has been proven to be ineffective and impractical for the typical low capability LLN

nodes [15]. To this end, auto-configuration is an essential requirement for such an environment, especially when a large number of nodes are connected [16]. On the other hand, the traditional centralized approach where Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) are employed to assign the names, keep all bindings, and to respond to all name inquiries and resolution requests may create considerable traffic. Therefore, resorting to the use of a distributed SD system can best meet the requirements of the constrained nodes deployed in LLNs. [17]–[20].

In this direction, the IETF ZeroConf WG has proposed the use of multicast DNS (mDNS) protocol [17] (RFC 6762) and its companion technology DNS-Service Discovery (DNS-SD) [17] (RFC 6763) as the SD protocols for resource-constrained networks. For more details of mDNS/DNS-SD, please refer to Subsection 2.1.3. The mDNS/DNS-SD protocols support an automatic configuration and discovery for the participating nodes and their services without human intervention. In a sense, they provide “zero-configuration” at the configuration phase and during the joining of new nodes to the network. The most advantages of using mDNS/DNS-SD are that the configuration for nodes can be achieved (i) automatically with no administrative intervention, (ii) in the absence of the infrastructure, and (iii) even during infrastructure failures. Moreover, they utilize roughly the same messaging format and operating semantics as the traditional DNS protocol and possibly can work alongside the widely known DNS protocol in one network [18].

Due to the aforementioned advantages and the low complexity of mDNS/DNS-SD protocols, they are being considered by the IETF ZeroConf WG as the most appropriate candidates for distributed naming and SD systems in constrained networks [21].

That said, mDNS in its current form suffers some issues which hinder its performance in LLNs. In particular, mDNS generates high control traffic during the configuration phase and registration process as reported in several studies [15], [18], [22]. This is attributed, in the first place, to the fact that mDNS was originally designed for conventional computing nodes with nearly no limit of bandwidth, memory, and processor resources [23].

1.2.2 Routing Discovery (RD)

Given the high suitability that IP-based standards offer, the IETF ROLL WG [7] has undertaken the task of identifying a routing solution for LLNs based on IP architecture under specific deployments. The IETF ROLL WG studied existing routing solutions and subsequently published a number of defining documents in the forms of Request For Comments (RFCs). These documents identify the routing requirements of specific deployments pertaining to automation in urban environments [24], industries [25], homes [26], and buildings [27]. The mandatory requirements are related to scalability, low control overhead, bounded end-to-end delay, and quick recovery from any dynamic changes in the topology. Subsection 2.2.1 outlines these requirements in some details.

By considering these requirements, the IETF ROLL WG focused its work on developing a standardized routing protocol adequate for the considered deployments. As a result, IPv6 Routing Protocol for LLNs (RPL) [28] has been standardized in RFC 6550 [29]. Please refer to Subsection 2.2.3 for more details on RPL. RPL limits the physical topology of the LLN to a Destination Oriented Directed Acyclic Graph (DODAG) controlled by a particular user-defined Objective Function (OF). The Des-

termination Oriented Directed Acyclic Graph (DODAG) is rooted at a pre-assigned resourceful node named the DODAG's root or simply the "root". The DODAG is built to optimize routes reaching the root. The OF is defined to minimize a specific routing metric such as, among others listed in [30], hop count or Expected Transmission Count (ETX) to reach the root.

RPL was mainly designed with an assumption that the predominant traffic flow would be gathering data towards a single destination. However, most of the applications enabled by LLNs are not only data generators but also addressable endpoints with actuation capabilities and can be introduced as resources on the Internet. This implies, from a networking point of view, that data not necessarily be routed only towards a pre-assigned destination, but also arbitrarily between nodes within the LLN domain. This kind of communications are commonly known as Node-to-Node (N2N) communications. In applications such as a remote control in building automation and industrial monitoring, almost 30% of the traffic flows are N2N communications [31], [32]. They would be employed to launch data transmissions, send acknowledgments, or to carry infrequent network diagnostics such as ping and trace-route traffic [33]. The importance of such communications increases as nodes tend to become independent [34]. In conclusion, using the RPL-based forwarding structure to handle N2N communications would pose another challenge. In a sense, an N2N packet should be forwarded upwards along the DODAG to the nearest common ancestor of the source and destination. Thereafter, the packet is propagated downwards along the DODAG until it reaches its destination. In the case where the downward routes are maintained only by the root, the packet is forwarded upward all the way to the root whereupon it

is routed downward to its destination. Therefore, based on the current specifications of RPL, it does not cater well to the N2N routing requirements. This shortcoming limits the adoption of RPL in several N2N-based applications [35]–[39].

1.3 Thesis Aims and Objectives

In light of the above challenges and shortcomings, this thesis aims to extend the mDNS and RPL protocols by proposing mechanisms that are able to mitigate their aforementioned-related shortcomings. To quantify these aims, the measurable objectives and requirements related to this ability are summarized by the following:

- **Low energy consumption:** LLN nodes consume energy while processing, sleeping, and radio listening, receiving and transmitting. One of the main concerns in LLNs is energy limitation as most LLN nodes are battery powered. These batteries are not recharged very often and replacing them is not practical, thus they need to run for longer periods [2]. This makes energy a precious resource that should not be wasted and any proposed solutions should require lower energy consumption.
- **Backward compatibility support:** this ensures the compatibility of any proposed solutions with existing systems and is an essential requirement for supporting hybrid networks. This is also to guarantee interoperability which is crucial in practice to enable the convergence of diverse technologies [40].
- **High reliability:** this is another key point to be considered which aims to in-

crease the rate services that are successfully delivered [41]. Furthermore, high reliability is needed to deliver the control packets to efficiently handle the operations that are required by a particular protocol.

- **Low latency:** most LLN-based applications are very delay-sensitive [25]. Therefore, data transmission and routing should be maintained as timely as possible [42].
- **Even load distribution:** distributing the load evenly among the participating nodes in an LLN is essential to increase the overall network lifetime. Uneven load distribution causes faster depletion of batteries of overloaded nodes which in turn leads to the shortening of the overall network lifetime [39].
- **Low memory footprint:** because of the low storage capacity of LLN nodes, achieving a small memory footprint is considered one of the most important requirements for designing LLN protocols in IoT. Minimizing the memory footprint allows nodes to include all the needed modules with increased features [43].

1.4 Thesis Main Contributions

This thesis aims to make the following contributions:

- **A survey** on existing solutions for SD and RD in IoT outlining their strengths and drawbacks in the context of IoT requirements and challenges. Concerning SD, the survey categorizes the existing SD solutions and outlines their properties individually. Moreover, it presents an account of existing enhancements for

developing SD system with an emphasis on mDNS/DNS-SD optimizations. Regarding RD, the conducted survey gives a detailed comparative analysis for the existing RD solutions. In particular, it focuses on those enhancements targeting N2N communications.

- **An extension to mDNS** named an Enhanced mDNS (EmDNS), is presented in **Chapter 4**. EmDNS is the first work that takes into account the characteristics of constrained LLN nodes in IoT and focuses on the auto-configuration of naming and registration systems. Also, this work presents a thorough analysis of the mDNS configuration phase considering the race condition problem arising from simultaneous start-up events. EmDNS mechanisms are implemented using ContikiOS [44] and evaluated via various experiments on the Cooja emulator [45]. Then, their strengths and shortcomings are evaluated according to the results obtained. The results show that EmDNS is backward compatible with mDNS and achieves a significant reduction in control traffic since less communication is needed to deploy and configure the nodes autonomously. Furthermore, EmDNS outperforms the mDNS in terms of energy consumption, and the ability to provide unique names for deployed nodes while maintaining an acceptable local cache hit ratio.
- **An extension to RPL** called a Hybrid RPL (HRPL), is presented in **Chapter 5**. HRPL is the first work that integrates the concept of link-state routing with the RPL-based forwarding structure. The salient features of HRPL include i) an optimized data forwarding for the N2N communication pattern, ii) the use

of control messages already available with RPL protocol to avoid redundant transmissions, and iii) the support of backward compatibility so that HRPL can seamlessly coexist with conventional RPL implementations in a hybrid network. HRPL is implemented on ContikiOS and evaluated through extensive simulations on Cooja. HRPL has shown a considerable improvement in delivery ratio relative to the standard. Although HRPL consumes an acceptable memory size more than that for RPL, HRPL manages to significantly reduce the control plane packets, the number of hops, and MAC transmissions needed to successfully deliver N2N data packets. Accordingly, these reductions are translated into lower packet delay and less energy consumption.

1.5 Thesis Organization and Overview

The remainder of this thesis is divided into five chapters as follows.

Chapter 2 covers a detailed background of SD and RD which are the main scope of this thesis. This chapter is divided into three parts. The first part outlines the objectives and requirements of SD in IoT and discusses the most common distributed SD protocols, mDNS/DNS-SD, by illustrating their entities and operational phases. The second part is related to the RD. In this part, the requirements and challenges of RD in IoT are presented, traditional RD protocols are discussed by evaluating their capabilities to satisfy such requirements, and RPL protocol is outlined by explaining its routing structure and how it handles N2N communications. Also, this part tests the correctness of the RPL code implemented in ContikiOS. The main purpose of this

test is to confirm the match between the theoretical understanding of RPL operations and the RPL implementation used for the evaluation. The last part discusses the technological tools employed for the implementation and evaluation of the proposed solutions and emphasizes the reasons behind their selection.

Chapter 3 provides an overview of efforts documented in various literature for both SD and RD solutions and protocols in IoT. It first surveys SD protocols and presents the existing enhancements targeting mDNS/DNS-SD protocols. Then, it outlines a comprehensive survey for the relevant N2N routing enhancements summarizing their properties individually.

Chapter 4 discusses the proposed EmDNS extension to the standard mDNS protocol. EmDNS introduces some mechanisms that can reduce the traffic generated and also the energy consumption during the configuration phase and the registration process of mDNS protocol. This chapter outlines the motivation and need for EmDNS, lists the EmDNS-related contributions, formulates the underlying problem, and describes EmDNS and its proposed traffic reduction mechanisms. It also presents an evaluation of the performance of EmDNS.

Chapter 5 describes the HRPL extension which is a modification of the original RPL protocol. This chapter begins with a discussion of the RPL-based N2N routing limitations that motivate the need for the proposed HRPL. Thereafter, the contributions in this chapter are listed. Subsequently, the performance evaluation of HRPL is detailed considering different performance metrics and network conditions.

Chapter 6 reviews and discusses the contributions of this thesis within the context of previous literature studies and outlines the scope for possible future work.

CHAPTER 2

BACKGROUND

The emergence of IoT revolutionizes our daily-lives with countless opportunities in many ways. IoT aims to transform the physical world around us into a sort of information system. This means that numerous services could be offered and a large amount of data would be circulated in such systems. To deal with these aspects, we first need mechanisms to identify and discover the provided services and to route the relevant data to its intended consumers (i.e. IoT nodes, users, applications, etc..). These tasks are undertaken by Service Discovery (SD) and Routing Discovery (RD) mechanisms. This chapter covers the SD and RD mechanisms in detail by discussing their objectives, their design challenges and requirements, and their most promising existing solutions in the literature. At the end of this chapter, the experimental technologies and methodologies that are used to evaluate the mechanisms proposed in this thesis are outlined.

2.1 Service Discovery (SD) in IoT

A mechanism by which IoT nodes can discover and learn about available services in a network is known as an SD mechanism. A new joining node would not function properly in the absence of such a mechanism. In other words, a node would have no way to discover the available services that it might need or to announce the services it hosts. In deployments such as building automation, for instance, light switches need the SD mechanism to discover smart lights available in their proximity. In the same way, lights themselves need such a mechanism to recognize the availability of nearby light switches. The automatic SD, therefore, is an essential requirement for smart nodes due to their limited interactions with end-users.

The importance of the SD mechanism can be seen clearly with deployments running simultaneous applications. In this case, each application has a requirement to find its peer nodes and also those that it needs to report or request information from. Likewise, a single application hard-coded into a system can also benefit from the SD mechanism. In a network employed for collecting temperature data, for example, the sensing application has a requirement to discover the data sink at which the generated data should be gathered. The SD mechanism is of the utmost importance for a network either in its initial bootstrapping state or during its steady-state. This mechanism is required, during the steady-state, to update nodes with information of the new joining nodes or with the new services advertised.

In conclusion, the SD mechanism is closely associated with auto-configuration, that is the procedure which autonomously enables joining nodes to get their interfaces

addresses, services descriptions, and other related configurations that are essential for these nodes to operate. Such a procedure represents an important building block in many SD frameworks to enable a network to have connectivity.

2.1.1 SD Objectives

- **Self-configuration:** An SD protocol should function autonomously with no administrative or human intervention. In other words, the SD protocol is required to support frequent dynamic changes in network topology and service descriptions. In this way, nodes can automatically get updated information about available services.
- **Dynamic Discovery:** The ultimate goal of any SD protocol is to enable users to dynamically find their services of interest according to specific properties. To do so, the SD protocol should identify the essential entities to undertake these operations. Some of these entities are i) an entity to define the operating semantics describing services, ii) an entity to host the information about these services, and iii) an entity to look-up for services requested by nodes. Moreover, the SD protocol should identify the relations and interactions between these entities to achieve the goal mentioned.

2.1.2 SD Requirements

A developed SD protocol must carefully consider the following requirements:

- **Identification:** Identifiers used for nodes and services should be reconfigured au-

tonomously and easily. The reuse of identifiers is beneficial for their recycling due to the limited lifetime of most participating nodes. In addition, the identifiers of the nodes should be independent of the current network because of their frequent movement between different networks.

- Decentralization: Most interactions between nodes are often limited network-wide within their vicinity. Therefore, managing a large number of nodes by a central entity is not practical because it would increase the access delay, communication traffic, and cause privacy and security problems. So, a decentralized approach should be used to avoid a global routing of extreme queries reducing traffic generated and decreasing nodes overheads.
- Service Oriented Architecture (SOA): The SOA interconnects heterogeneous nodes seamlessly by enabling the collaboration between service providers and consumers. This architecture has a minimal overhead requirement for delivering and registering services.
- Automation: The discovery process should take place autonomously without manual administration and be adopted based on the current context. Therefore, the interaction among nodes and also their ability to discover each other should be fully automated.
- Interoperability: To ensure wide acceptance of a developed SD protocol, it must follow a standardized procedure that enables different technologies to converge.

2.1.3 A multicast DNS (mDNS) and DNS-SD Protocols

After surveying the SD protocols which target constrained networks in Subsection 3.1, we select the standardized mDNS/DNS-SD protocols as viable candidates for the SD in LLNs. We emphasize this choice with a consideration of the SD requirements covered in Subsection 2.1 above. The mDNS/DNS-SD are distributed SD protocols that roughly use the same messaging formats, language, and programming interfaces as DNS protocol [18]. They facilitate discovering the available nodes and their services in the neighborhood and maintaining a network automatically. The reuse of DNS protocol can be advantageous in many ways. The most notable of which are 1) DNS has attained outstanding popularity because it has worked with all network sizes for many years, and 2) it avoids deploying and configuring a separate system to be dedicated to SD. Aside from being standard-based protocols, mDNS/DNS-SD also have lightweight memory footprints, broad usage, and good scalability. They have been designed to provide auto-configuration (i.e zero-configuration) with IP-based networking support. They enable a standardized SD of constrained nodes with the same Internet mechanisms which are already used for ordinary computational devices. Interoperability is guaranteed by the established DNS for the IoT at the application layer. Thus, there is no need to use specialized protocol gateways nor to run IoT node-specific code on computational devices.

mDNS/DNS-SD consist of two components: 1) mDNS protocol, which represents a communication protocol. mDNS's functionality is to resolve domain names without the help of any unicast DNS server by sending DNS messages to a multicast group,

and 2) DNS-SD protocol, which defines description formats of nodes and services, and operating semantics of queries and responses. DNS-SD announces a detailed service information (e.g., availability time, access protocol, IP address, port) via so-called DNS Resource Records (RRs, referred to below) to other nodes in the network in a way that applications and users can simply look up or wait for available or needed services. Each node analyzes the released records inside the multicast group and each of them generates a list of entities based on the received records. mDNS/DNS-SD have two operational phases: a zero-configuration phase and a discovery phase, as depicted in Fig. 2.1. These phases are discussed in the following two subsections.

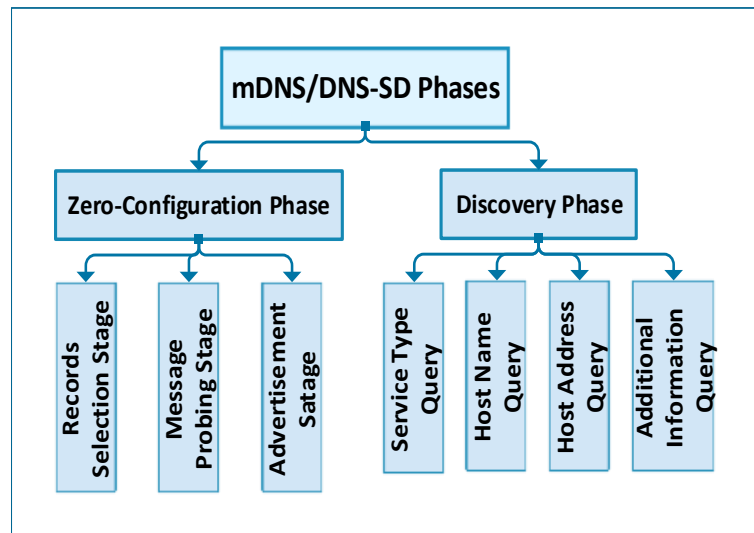


Figure 2.1: The mDNS/DNS-SD Operational Phases

2.1.3.1 The Zero-configuration Phase of mDNS/DNS-SD

The zero-configuration phase enables the networking in nodes, and assigns unique names and descriptions for nodes and for their hosted services. A node performs this phase whenever it starts up, wakes up from sleep, or receives an indication of a

network interface “link change” event. In other words, it is performed when a node has any reason to believe that its network connectivity may have changed in some relevant way [46]. The flowchart in Fig. 2.2 shows the operational flow performed by both a new joining node and nodes that already joined the network. This phase

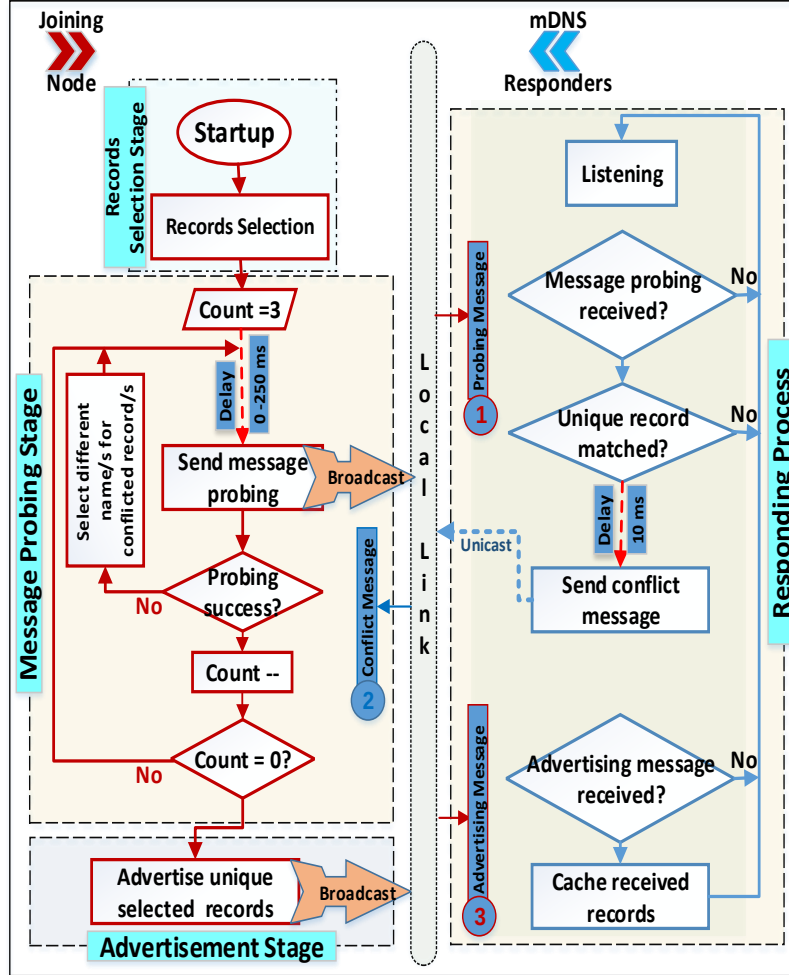


Figure 2.2: The mDNS/DNS-SD Zero-configuration Phase

includes three stages: a record Selection Stage (SS), a Probing Stage (PS), and an Announcement Stage (AS). In the SS, a joining node selects names for all RRs that are needed to be unique in a network. The selection method can be done through any predefined strategy. For example, the RRs that describe interface’s addresses can

be generated as in the RFC 4862 [47]. In the PS, the joining node must multicast an mDNS query asking if the selected RRs are already in use by other nodes. If no conflicting responses are received for three consecutive probing times, the joining node then commits to these records. Having committed to RRs, a joining node advertises them in the networks since this stage is called the AS. The joining node, however, must select different names if a conflict takes place, and then performs the procedure of the PS again. The respondents, on the other hand, store the advertised RRs in case they need to contact the joining node afterwards for some services of interest.

2.1.3.2 The Discovery Phase of mDNS/DNS-SD

The discovery phase enables nodes to discover the services of their interest. DNS-SD defines a set of DNS RRs used for discovery purposes. These RRs are service locator (SRV), pointer (PTR), text (TXT), and IPv6/IPv4 addresses (AAAA/A). Each RR can be seen as a mapping between two parameters, for example, SRV maps the service name to its host-name. Table 2.1 lists these RRs along with their usages and formats. In this table, the temperature service, uniquely identified by “temp1” hosted by “node1” and described by the RRs, is mentioned as an example. The terms included in these formats are explained in Table 2.2. When a node wants to inquire about a name’s service of a specific type, it queries for PTR. The response for the PTR lookup is a list containing all services of that specific type. Then, the requesting node sends a query for SRV to get the host-name and port number of the obtained service. Additional information about the service can be conveyed through a TXT record. The TXT record is represented in a format of key-value pair. Finally, to get

Table 2.1: Summary of the RRs describing the service *temp1* provided by the *node1*

RR	Purpose	Format	Example
PTR	A service to type assignment	<code>_<apptype>._<protocol>.<domain> <class> PTR <service>._<apptype> ._<protocol>.<domain></code>	<code>_coap._udp.local. IN PTR temp1._coap._udp.local.</code>
SRV	A service to host name assignment	<code><service>._<apptype>._<protocol>. <domain>.<TTL> <class> SRV <priority> <weight> <port> <target></code>	<code>temp1._coap._udp.local. 3600 IN SRV 0 0 12345 node1.local.</code>
TXT	service additional information assignment	<code><service>._<apptype>._<protocol>. <domain>.<class> TXT <“Key =value”></code>	<code>temp1._coap._udp.local. IN TXT “PATH=/temp/valve1”</code>
AAAA	An IPv6 to a host name mapping	<code><target>.<class> AAAA <IPv6></code>	<code>node1.local. IN AAAA aaaa :: 8</code>
A	An IPv4 to a host name mapping	<code><target> . <class> A <IPv4></code>	<code>node1.local. IN A 192.168.0.8</code>

Table 2.2: Explanations of some filed of RRs in DNS-SD

Term	Meaning	Example
<code>< service ></code>	Service name	<i>temp1</i>
<code>< apptype ></code>	Service type	CoAP
<code>< protocol ></code>	Service’s accessing protocol	UDP
<i>domain</i>	Domain name of the service, usually .local in mDNS	.local
<code>< TTL ></code>	Time To Live. Record validity period in seconds	3600
<code>< class ></code>	DNS class, usually IN	IN
<code>< priority ></code>	Smaller value indicates higher priority	0
<code>< weight ></code>	Relative weight for entries with the same priority value	0
<code>< port ></code>	Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port number the service listens to	12345
<code>< target ></code>	The host name of the host providing the service	<i>node1</i>

the address of the host-name that provides the service, the requesting node makes a query for the A or AAAA record based on the IP version in use.

2.1.3.3 The Implementations of mDNS/DNS-SD

The mDNS/DNS-SD protocols have been implemented with IP networking support for constrained and also for non-constrained nodes such as PCs, smartphones, and netbooks. Many operating systems and platforms install mDNS/DNS-SD to discover

services advertised in a local network such as print services, file share, Secure Shell (SSH) and web-servers. For example, Avahi [48], jmDNS [49], and Bonjour [50] are different implementations of mDNS/DNS-SD for Linux/BSD/OpenWRT/Android, Java, and Mac/Windows/iOS, respectively. uBonjour [51], on the other hand, is a mini-mized Bonjour-based implementation of mDNS/DNS-SD for constrained nodes running ContikiOS.

2.2 Routing Discovery (RD) in IoT

2.2.1 RD Challenges and Requirements

The RD mechanisms in IoT-based networks poses many challenges that require proper operations by RD protocols. Some of these challenges are i) The nodes inability to store network topology or detailed routing information, ii) The cost of the routing control traffic which would have to be a portion of what is allowed on the Internet, keeping in mind network sizes can be up to thousands of nodes, and 3) The limited power that nodes could have, as they are battery-powered with little to no chance for recharging, being replaced, or harvesting power externally. At the same time, the variation of the deployment scenarios produces different challenges based on the characteristics of their traffic, profiles of their applications, and their critical data. To this end, the IETF ROLL WG identified the RD requirements of a number of specific deployment scenarios. Such requirements published in the form of RFC documents. The studied deployments are related to automation of Buildings in RFC 5867 [27], Homes in RFC 5826 [26], Urban environments in RFC 5548 [24], and Industries in RFC

5673 [25]. The most common RD requirements of these deployments are highlighted in the following:

- Constrained routing: A routing protocol must support predefined constraints during routing. These constraints might be a node constraint, parameter constraint, or other constraints set by the user. For example, choosing the node of mains-operated or battery-operated power, battery level, memory size, or/and micro-controller speed.
- Auto-configuration: The joining process of nodes to the network should be automated. Therefore, a routing protocol should enable new joining nodes to get the network configurations and parameters at the setup phase without human intervention.
- Quick convergence: LLNs have high topological changes and unreliable links with time-varying qualities due to external interference and a harsh environment. Therefore, a routing protocol must spend a reasonable amount of time to converge with minimum control traffic.
- Consideration of constraints: Most LLN deployments are characterized by severe constraints in their resources. A routing protocol, therefore, should retain a low-cost communication rate.
- Supporting different traffic flows: A routing protocol should support different traffic flows that don't only gather data towards a single destination. The routing from a sink towards nodes inside the network and among nodes themselves

should be supported as well.

- Bounded delay: A routing protocol should maintain stringent end-to-end latency as most deployments are delayed sensitive applications.

2.2.2 Traditional RD Protocols for IoT

Routing protocols generally fall into two categories which are link-state and distance-vector approaches. In the link-state approach, a node builds adjacency with its neighbors and floods the topology information to the whole network. Therefore, each node would have a global view of the network. In the distance-vector approach, however, a node exchanges the routing information with its direct neighbors only.

Based on a set of criteria defining costs and benefits, the IETF ROLL WG evaluated the appropriateness of existing protocols towards satisfying the RD requirements previously mentioned. If any protocol cannot satisfy such criteria, it wouldn't be reasonable candidates for routing in major LLN applications. This evaluation has been published in the form of an informational draft in [52]. The considered criteria are derived from the documents that specify the RD requirements of deployments mentioned above. These criteria are 1) Routing state: which measures how the protocol scales in terms of memory occupancy, 2) A loss response: which measures the protocol's ability to maintain link changes or failures without the need of a global repair, 3) A control cost: a protocol fails this criterion if the rate of control traffic per-node (i.e. transmissions plus receptions) is not confined by the data rate plus a small constant. 4) A link cost: a protocol should evaluate links qualities based on an objective

function considering optimizing various metric types such as latency, reliability, and energy, and 5) A node cost: a protocol should account for the properties of the nodes such as memory occupancy and the life of the battery during the construction of the routing structure.

The formal big- O notation is used to precisely evaluate the studied protocols over some of the listed criteria. Notations N , D , L refer to the network size, destination size, and neighborhood size, respectively. R refers to the data rate and e is a small constant. Table 2.3 specifies the pass or fail decision depending on the big- O condition.

Table 2.3: The satisfaction decisions based on the big- O conditions

Criteria	Fail	Pass
Routing state	$O(N)$ or $O(L)$ or more	$O(D)$
Loss response	$O(N)$ or more	$O(1)$ or $O(D)$
Control cost	Otherwise	$O(R \times \log(L) + e)$ or less

Table 2.4 lists the studied protocols and summarizes the satisfaction performance of these protocols in terms of the used criteria. The true mark “✓” indicates that the protocol passes the corresponding performance criterion whereas the false mark “✗” does not. The question mark “?” indicates that the protocol fails but there may be a technique that could be implemented without violating original specifications. However, there are no guidelines on how this should be accomplished. From the table, it is clear that no protocol satisfies all criteria. In particular, it can be seen that all link-state routing protocols flood the network through link updates. These protocols fail to retain low overhead for the control cost because links in LLNs experience a high frequency of failures and dynamic changes [52]. However, the distance-vector protocols fail due to their lack of quick recovery from frequent topology changes and link churn.

Table 2.4: Summary for performance satisfaction of the traditional IETF protocols

Protocol	RFC	Routing Category	Route Creation	Designed for	Routing Metrics	Space Complexity	State Routing	Loss Response	Control Cost	Link Cost	Node Cost
OSPF/IS-IS	2740/1142	LSR	Pro	AS	BW, Delay	$O(N \times L)$	X	X	X	✓	X
OLSRv2	3626	LSR	Pro	MANET	Hops	$O(N \times L + L^2)$	X	?	?	✓	✓
TBRPF	3684	LSR	pro	MANET	Hops	$O(N \times L)$	X	✓	X	✓	?
RIP	2453	DV	pro	AS	Hops	$O(D)$	✓	X	✓	?	X
AODV	3561	DV	Rea	MANET	Hops	$O(D)$	✓	X	✓	X	X
DYMO	Draft	DV	Rea	MANET	N/A	$O(D)$	✓	?	✓	?	?
DSR	4728	DV	Rea	MANET	–	$O(D+L)$	X	✓	✓	X	X
TRIP	2091	DV	Rea	MANET	Hops	$O(D)$	✓	X	✓	?	X

Pro=Proactive, Rea=Reactive, LSR=Link State Routing, DV=Distance Vector,
AS= Autonomous System, BW= Bandwidth

In conclusion, a routing protocol for LLNs is needed. Such a protocol would be either a modification of an existing protocol or a completely new one. To this end, the ROLL IETF WG has designed an IPv6 Routing Protocol for LLNs (RPL), which is specified in RFC 6550 [29], to be the standard RD protocol for LLNs. RPL is explained in the next subsection.

2.2.3 Overview of IPv6 Routing Protocol for LLNs (RPL)

RPL is an IPv6-based RD protocol that has been designed to satisfy the RD requirements previously discussed. RPL is a distance-vector protocol that proactively creates routes. It satisfies the constrained routing requirement by utilizing an OF in which a node can choose its parents confirming a specific routing metric. With RPL, human

intervention is not required as it supports an auto-configuration during the joining of new nodes and the setup phase. Given that some requirements prefer small end-to-end delays, the simulation of RPL in gathering data applications provides satisfactory results in this regard [53]. Also, the study in [54] has shown that RPL can recover quickly in the case of topology changes or link failures. Finally, RPL supports all traffic types such as gathering data from nodes to a sink, traffic from the sink to nodes located inside an LLN, and communication between nodes within an LLN. This subsection discusses how the RPL-based routing structure is constructed and maintained, defines some of the important routing concepts in RPL, and explains how N2N communications are handled in RPL-based networks.

2.2.3.1 RPL-based DODAG Construction

RPL organizes the LLN nodes, in a tree-like structure typically rooted at the BR and named a Destination Oriented Directed Acyclic Graph (DODAG). Three Internet Control Message Protocol (ICMP) messages maintain the DODAG construction process which are 1) A DODAG Information Object (DIO), 2) A DODAG Information Solicitation (DIS), and 3) A Destination Advertisement Object (DAO). The DODAG construction is initiated by having the BR broadcasts a DODAG Information Object (DIO) message containing its rank, implementation-dependent parameters, and the common configuration attributes and policies needed to build the DODAG. Upon receiving the broadcast DIO, the neighboring node performs the following: i) selects its preferred parent depending on the OF, ii) calculates its rank (i.e position) relative to the BR, which indicates the cost of reaching the BR from the node itself, and iii)

updates the received DIO with its rank and re-broadcasts it further. This process is continued until all participating nodes in the network are covered. At the end of this process, the upward routes from each node towards the BR are built.

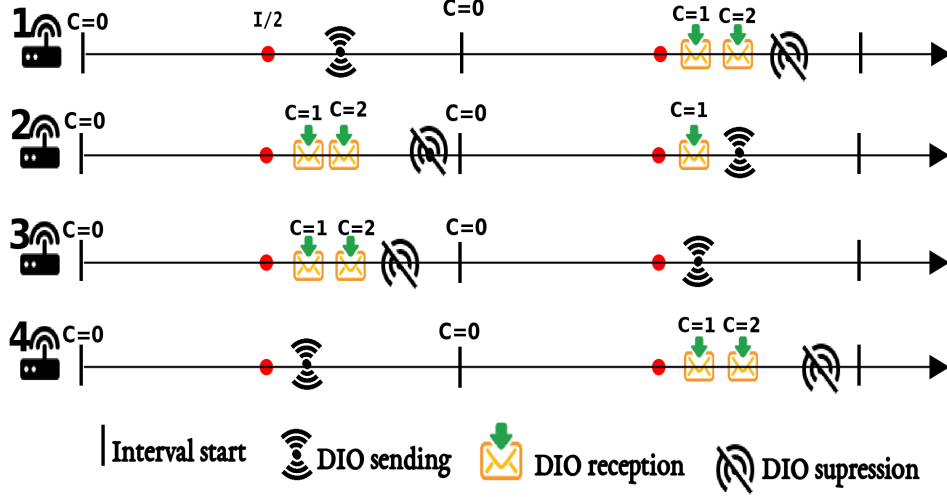
Each node performs a parent selection process (i.e. parent switching) whenever it detects a change in the routing metric to one of its neighbors. It changes its current parent to the best alternative one from its parent set. The HRPL can trigger a global recalculation of the DODAG by increasing a sequence number and a DODAG version in the transmitted DIO messages. To keep the DODAG consistent, each node emits DIO triggered by a Trickle timer [55] to avoid unnecessary traffic. The timer duration increases exponentially but it resets to a minimum value if some events occur. These events are parent unreachability, parent switching, DODAG recalculation, and detecting of routing loops. Each node adopts a local repair mechanism to fix the parent unreachability or unavailability of a route to the BR. It does so by initiating the parent selection process and sending out a DIO message to notify its associated nodes of the need for parent switching. When any node encounters a DIO with the newer sequence number, it again starts the parent selection process as per the updated link costs. In the case of the loop detection, the receiver attempts to repair the loop by sending a unicast DIO back to the sender so that it gets a fresh update of the receiver's rank. Finally, when a node joins the network, it may wait to receive a DIO or it may alternatively multicast the DIS message to proactively solicit a DIO.

2.2.3.2 RPL-based Trickle Algorithm (TA)

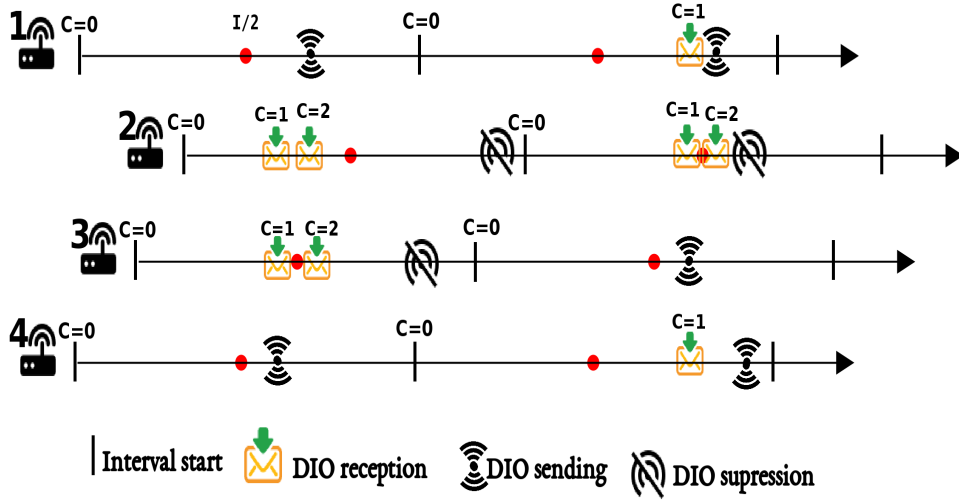
To eliminate the redundant transmission of DIOs, RPL uses the TA. A node running the TA suppresses the DIO transmission upon overhearing the same information it plans to send through that DIO. In this way, energy depletion and bandwidth wasting are reduced significantly. The TA reduces the DIO transmissions exponentially when no inconsistencies are detected in the network and quickly propagates new updates when inconsistencies are detected. The consistency is tested by comparing the DODAG information locally cached with the information in the received DIO.

With the TA, the time is divided into intervals in which the DIO transmission may occur following the Trickle's rules. The intervals may have a variable length (I) ranging from the minimum (I_{min}) to the maximum (I_{max}) length. At each interval, each node selects a tentative transmission time $\mathcal{T}$ in which the DIO message is planned to be sent. The $\mathcal{T}$ is randomly selected from the second half of the interval, i.e. $\mathcal{T} \in [\frac{1}{2}I, I]$. In the first half of an interval which is called a listen-only period, a node never transmits a DIO. Each node manages a local counter C . This counter is reset to *zero* at the beginning of each new interval and incremented whenever a consistent DIO is received. The node broadcasts its DIO at time instant $\mathcal{T}$ if and only if $C < K$ where K is a predefined redundancy constant. If the node receives an inconsistent DIO message, the next interval is reset to I_{min} or otherwise, it is doubled until it reaches I_{max} .

Figure 2.3 is an example that illustrates the behavior of nodes running the TA in a network at a steady state for $K = 2$. Note that only nodes one and four are outside of each other's covering areas. The figure shows sending and suppressing events of



(a) Intervals with a synchronous start times



(b) Intervals with an asynchronous start times

Figure 2.3: Example of TA with and without synchronized intervals using $K = 2$ and $I = I_{max}$

the DIO transmission for two consecutive intervals with maximum length for both synchronous and asynchronous start intervals. For a synchronous startup as depicted in Fig. 2.3(a), all nodes begin their intervals at the same time. In the asynchronous one, on the other hand, each node has a different start interval as shown in Fig. 2.3(b). At the selected transmission time and as long as $C \geq K$, the transmission of a DIO is suppressed as shown in the figures.

2.2.3.3 DAO Emissions and Downward Routing

Nodes joining the DODAG unicast their prefix and prefixes through reachable them to their parents. These prefixes are carried in DAO messages particularly by means of the target and transit options defined with the DAO message in the RPL specifications. In turn, this information is unicast further up to eventually reach the BR to support downward traffic from the HRPL towards non-HRPL nodes within the network. Each node generates a DAO packet in the following cases: the preferred parent has changed, a global repair has taken place, or a downward route refreshment has been triggered. Upon receiving a unicast DAO message, the receiving node schedules a new DAO to be propagated upwards. The received DAO is treated based on the mode of operation used. RPL supports two Mode Of Operations (MOPs). These modes are non-storing and storing modes that are referred to herein by NSRPL and SRPL, respectively.

In **NSRPL mode**, non-BR nodes are not required to manage routing tables in their memory. NSRPL mode can be used either in large networks or in networks of nodes with extremely restricted resources to enhance their scalability. Upon receiving a DAO from direct children or other associated nodes in its sub-DODAG, the receiving node forwards the DAO to its preferred parent, and eventually the DAO will be received by the BR. Once the BR collects information about all nodes, only then will it be able to route packets downward along the proper path. If the BR is not able to build a path to a particular destination, it has no option but to drop the packet directed to that destination. NSRPL involves the BR in constructing and inserting a Source Routing Header (SRH) [56] into packets which are then transmitted on a

hop-by-hop basis. By using the SRH, a full path from a source node to a destination is explicitly listed in the header of the packet sent. Each node that is addressed in the header then forwards the packet to a next address extracted from the header.

In **SRPL mode**, on the other hand, all participating nodes are supposed to maintain tables to store the routing information for nodes in their sub-DODAG. Whenever the node forwards a destination prefix, available via itself or any of its children, it creates a corresponding routing entry for that particular destination. If the node has no free memory, it drops all new associations. This action leaves dropped children unreachable from the BR point of view. Consequently, the BR has no choice but to drop any packet destined to unreachable destinations. Maintaining routing entries for the downward route to each destination at every node is unfeasible even for networks of moderate size [57].

2.2.3.4 Node-to-Node (N2N) Routing in RPL-based Network

In the case of NSRPL mode, N2N packets are forwarded upwards through the preferred parents to the BR whereupon the BR builds an SRH. This SRH specifies the nodes which the packets should traverse downwards the DODAG to successfully reach their final targets. On the other hand, in the case of SRPL mode, N2N packets are forwarded upwards through the preferred parents of the node to the common ancestor of both the source and destination nodes. Afterwards, the packets propagate downwards along the DODAG to their final destination. Figure 2.4 depicts an example of DODAG topology and the N2N-related data routing from node ⑨ to node ⑪ using NSRPL and SRPL modes.

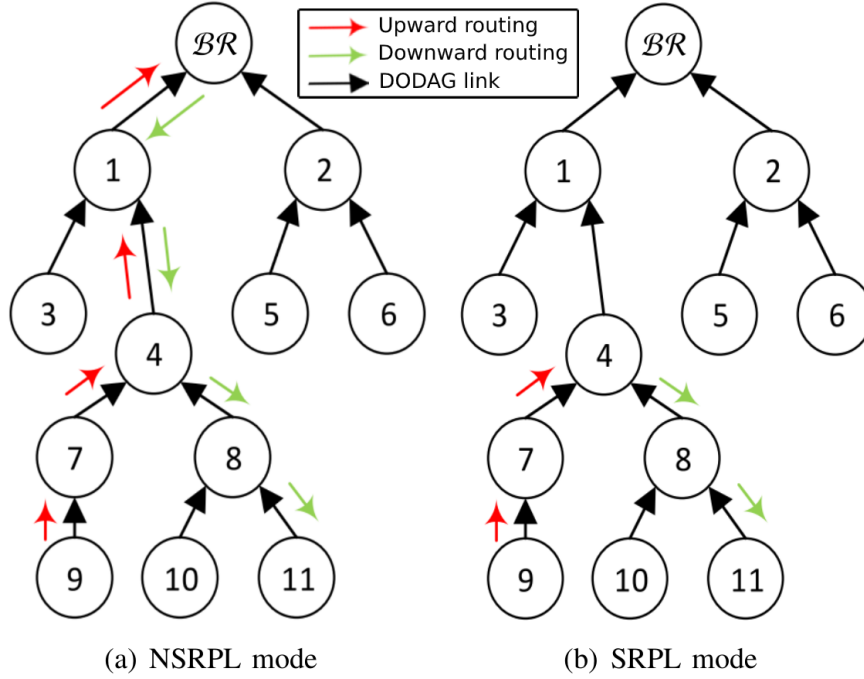


Figure 2.4: The RPL-based DODAG topology with NSRPL and SRPL modes

2.2.4 Testing the Trickle Algorithm (TA) of ContikiOS

This section attempts to check the match between the implementation of RPL in ContikiOS and the theoretical understanding of RPL operations. This is achieved by making use the model presented in [58] which estimates the probability of sending DIO via a node during the steady state of the network (i.e. $I = I_{max}$). This section presents a model of the network being studied, then present the analytical model of the sending probability of interest, and finally compares the results obtained from the analytical model with those from the simulation.

2.2.4.1 Network model and the sending probability of a DIO

Let $\mathcal{N}$ be a set of all nodes in the network with size η . Each node $n \in \mathcal{N}$ has a neighborhood set of size ω_n , which contains all nodes in the one-hop distance of the node n . The following definitions and assumptions are applied: i) The start time of the intervals is synchronized among all participating nodes, ii) Nodes are distributed uniformly in a two-dimensional area of size $\mathcal{R}$, and iii) The coverage area of size r is unified for all participating nodes. Let α_n denotes the probability that a random node in the network belongs to the neighbor set of the node n . Based on these assumptions, α_n can be calculated by (2.1).

$$\alpha_n = \frac{r_n}{\mathcal{R}} = \frac{\omega_n}{\eta} \quad (2.1)$$

The average probability $P(S)$ that the node $n \in \mathcal{N}$ is going to send a DIO in a given interval can be obtained by (2.2), where K is the redundancy constant of the TA.

$$\begin{aligned} P(S) = & \sum_{i=0}^{K-1} \left(\binom{\eta-1}{i} \cdot (\alpha_n)^i \cdot (1 - \alpha_n)^{\eta-1-i} \right) \\ & + \left(1 - \sum_{i=0}^{K-1} \left(\binom{\eta-1}{i} \cdot (\alpha_n)^i \cdot (1 - \alpha_n)^{\eta-1-i} \right) \right) \\ & \left[\left(\sum_{i=K}^{\eta-1} \frac{K}{i+1} \right) + \left(1 - \sum_{i=K}^{\eta-1} \frac{K}{i+1} \right) \left(\sum_{i=K}^{\eta-1} \sum_{j=0}^{K-1} \left(\binom{i}{j} \cdot (P(S))^j \cdot (1 - P(S))^{i-j} \right) \right) \right] \end{aligned} \quad (2.2)$$

2.2.4.2 Results and Discussion

This section first shows how to get the results from the analytical model. Next, the simulation setup is detailed and then, the collected results for both the analytical model and the simulation are compared and discussed.

To get results from the analytical model, a root-finding algorithm is used to solve (2.2) for $P(S)$. We develop a MATHEMATICA code in which the available *FindRoot* function is used to find the HRPL of $P(S)$. The α value of a particular scenario is obtained by dividing the average node degree over the total number of nodes. For instance, the first scenario has an average node degree of 7.568 resulting in α of 0.076. To cope with the effect of the border, we calculate the average node degree by making use of a distance matrix according to the distances determined in the studied scenarios.

To get simulation results, we emulate real nodes running the ContikiOS where RPL specifications and Trickle library are implemented. Cooja emulator is used for this evaluation. In this simulation, we consider a uniform 2D-grid topology including 100 nodes that are distributed in a square area of length 180 meters. Three scenarios are considered in this evaluation as shown in Fig. 2.5. The main difference among these scenarios is the node communication range. For the three scenarios, as shown in Figs. 2.5(a), 2.5(b), and 2.5(c), the coverage area per node has a radius of 35, 40, and 55 meters, respectively. The coverage area of a node is represented by a green circle, so all nodes inside this circle can receive the sent packet correctly. It has different sizes resulting in a different node degree (i.e. a number of neighbors) in each scenario. In this figure, we display the node degree for multiple nodes (i.e. nodes of IDs 1, 45, 51, 92, and 98) to show the border effect. In a sense, the degree of the nodes located on network boundaries such as node 1, for instance, is less than that for nodes located in the middle of the network such as node 45. This difference has effects on the results obtained by the analytical model as the model assumes a unified node degree. To ensure that the sending process controlled by the TA would not intervened

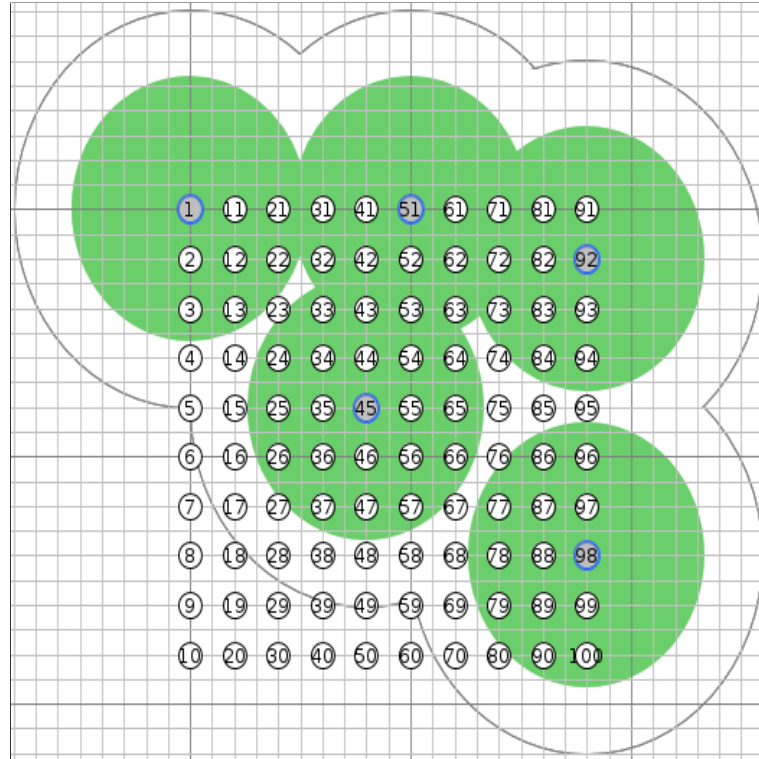
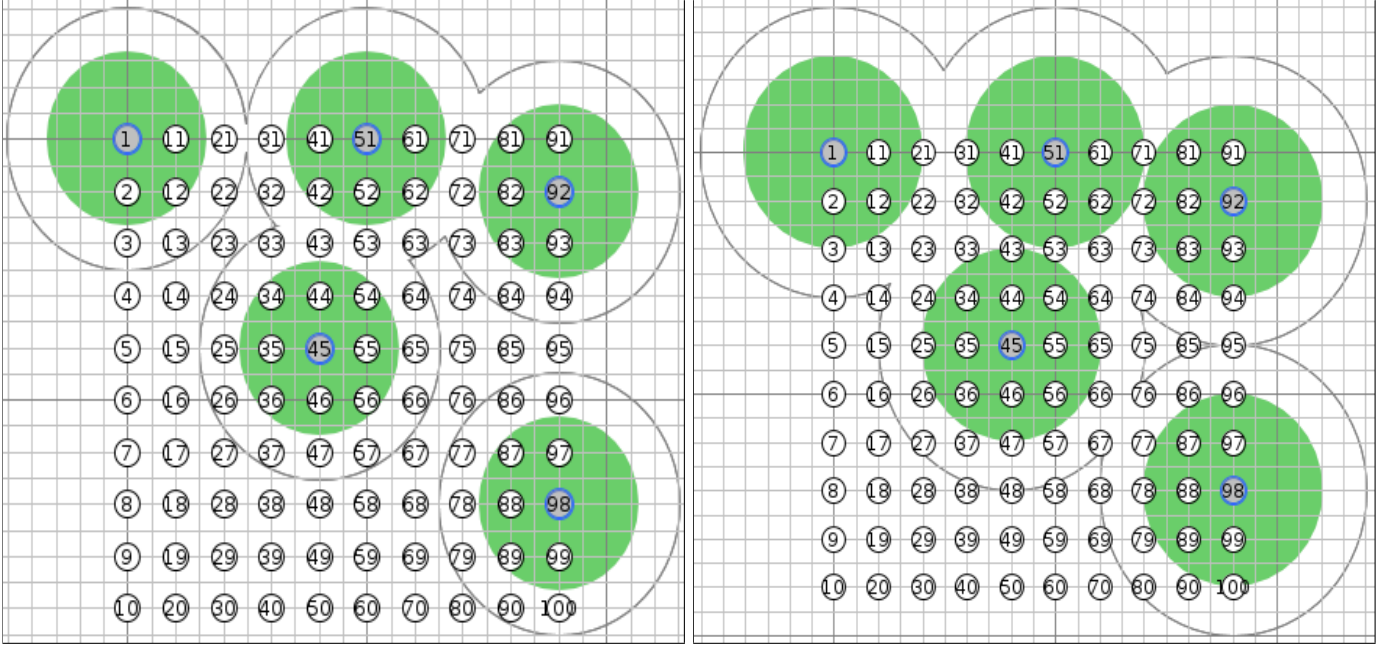


Figure 2.5: The typologies of the studied network using different scenarios

by other processes, we set-up the following configurations: 1) No data packets are sent by the application layer or by Neighbor Discovery (ND) protocol, 2) The downward routing procedure is disabled to guarantee that DAO packets are not sent out during the intervals of interest, 3) For the duty cycling, the Null Radio Duty Cycle (RDC) protocol is used to allow nodes to stay awake for the entire simulation period, and 4) The links are lossless to guarantee the success of sending packets with no loss.

Given that the steady state of the network is considered by this model, the simulator computes the number of sent packets when all nodes reach the interval of the length I_{max} . The sending probability is calculated by dividing the total number of sent packets (i.e. by all participating nodes in a given interval) by the total number of nodes and then is averaged over 20 consecutive intervals. It is important to mention that the start times of the intervals are asynchronized in the simulation. This asynchronous property may participate in increasing the mismatch gap between the simulation and analytical results as the later assumes synchronized start for the intervals. Table 5.2 summarizes the configuration setup of the parameters used throughout this simulation.

Figure 2.6 shows the sending probability obtained by the analytical model (represented by red) compared to that obtained by the simulation (represented by black) as a function of K . The figure includes the results for the three scenarios studied. In the figure, “Model-First” and “Simulation First” legends correspond to the analytical and simulation results, respectively, obtained from the first scenario. It is clear that a higher communication range results in a less sending probability value for the same redundancy value, K . Increasing the communication range means a higher number

Table 2.5: General configurations for experiments

Property	Configuration
Platform	ContikiOS v3.0 / Cooja
Topology	Uniformed 2D-Grid topology
Emulated mote	Wismote with 16KB RAM and 128KB ROM
Network scale	100 nodes
Network dimensions	180m * 180m
Root's position	Upper-left corner of the studied network
Network layer	RPL with Upward mode + uIPv6 + 6LoWPAN
Radio Environment	Unit Disk Graph Medium (UDGM)
Transmission range	Variable (35, 40, and 55 meters)
Minimum length I_{min}	12 which means $2^{12} msec = 4.096 sec$
Maximum length I_{max}	20 which means $2^{12+8} msec = 1048.576 sec$
Redundancy Constant D	Variable in the range [1, 24]
MAC & PHY	Carrier Sense Multiple Access (CSMA)/Collision Avoidance (CA) & IEEE 802.15.4
RDC/CCR	NullRDC/128Hz
Number of intervals	10 intervals

of neighbors for a node. This leads to receiving consistent messages higher than K and then the node suppresses its transmission accordingly. Finally, the figure shows a good match between the theoretical understanding of the RPL-related DIO emissions and the implementation of RPL-based TA on ContikiOS.

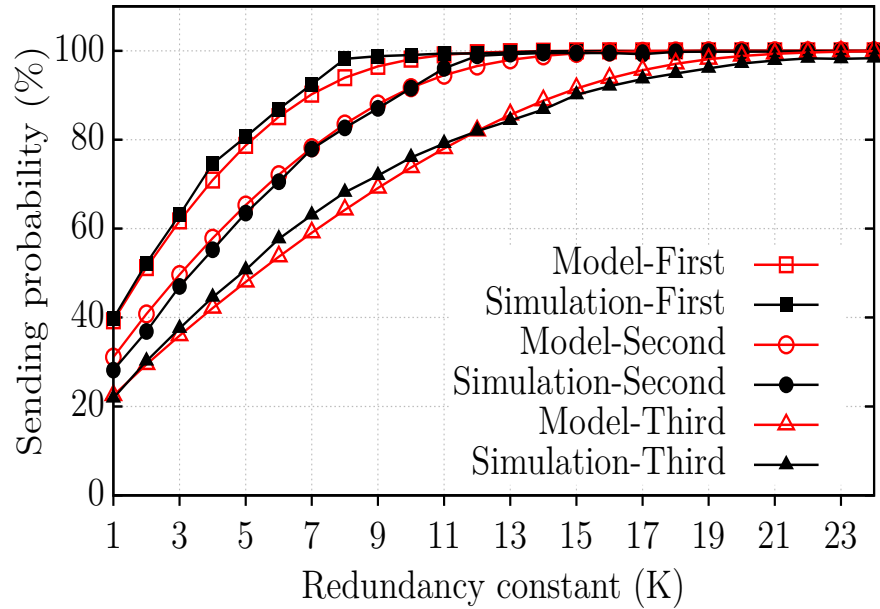


Figure 2.6: The average sending probability $P(S)$ as a function of D for the three scenarios

2.3 Evaluation Methodologies

2.3.1 Technologies for Experiments

It is of utmost importance to choose appropriate tools for implementing and evaluating the performance of any proposed solution. On the one hand, real implementations offer a dependable choice for studying the behavior of these solutions in realistic environments. Although there are various options for such implementations, ContikiOS is an excellent option as it is real and broadly used for implementing and experimenting with IoT protocols [59]–[61]. On the other hand, testing and evaluating the proposed solutions through a real testbed is costly, time-consuming, and a debugging challenge. Therefore, simulators are excellent alternatives that provide effective, low-cost, scalable, time-limited, and easy-to-use tools. Although many simulators have been developed in the recent past, most of these simulators are not conceived for hardware modeling and they do not emulate the same application code installed in the real IoT nodes. Besides, providing reproducibility becomes a major concern of the networking community nowadays [62]. Therefore, emulators lend themselves as very viable solutions for the performance evaluation of IoT protocols. A built-in Cooja emulator represents the de facto emulator for experimenting with IoT protocols [60]. This subsection presents a short review of ContikiOS and Cooja emulator and the advantages of using them.

2.3.1.1 Contiki Operating System (ContikiOS)

ContikiOS is a lightweight and highly portable open-source operating system for IoT written in C programming language. It is a tool box for building complex wireless systems. There are many advantageous features that ContikiOS offers, some of them can be summarized as follows:

- It handles an efficient memory allocation with dynamic support of module loading, and runs using a small memory size (i.e. 10Kbytes RAM and 30Kbytes ROM) [63].
- It supports many hardware platforms such as Tmote Sky, MicaZ, Z1, and Wis-mote; microcontrollers such as TI MSP430 family, Atmel AVR family, and ST STM32w; and radios such as the Texas Instruments CC2420, CC2520, and CC1020, and RFM TR1001.
- It enables an application to use either non-IP networking through Rime stack or IP networking via uIPv4 and uIPv6 stacks where the letter “u” stands for the micro. Figure 2.7 shows some of the IoT protocols that ContikiOS implements.
- It implements the 6LoWPAN adaptation layer and provides numerous implementations of MAC and RDC protocols. For the protocols of interest in this thesis, ContikiOS provides a lightweight implementation for mDNS and RPL protocols named “resolv.c” and “ContikiRPL” respectively.
- It provides a flexible hardware-level simulator called Cooja. We refer below to Cooja with some details.

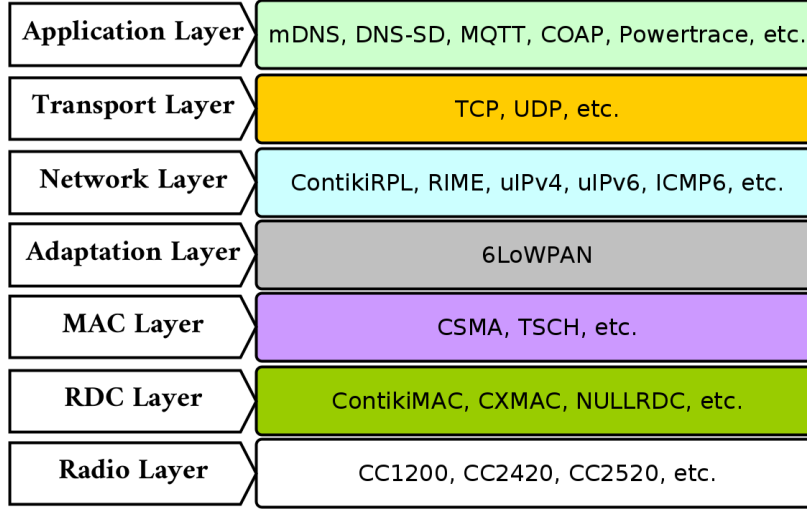


Figure 2.7: Some of IoT-enabling Protocols implemented in ContikiOS

2.3.1.2 Cooja Simulator

Cooja is a cross-layer network simulator developed for IoT nodes running ContikiOS and written in Java programming language. Cooja can simulate ContikiOS applications in two ways, either by running the application as a compiled native code directly on the target Central Processing Unit (CPU) or by running it in MSPSim [64] for the MSP430 microcontroller. Moreover, Cooja can simulate a Wireless Sensor Network (WSN) without a specific IoT node or ContikiOS application focusing only on the network behavior. Using MSPsim, Cooja can also simulate TinyOS [65] applications for MSP430 motes.

The unique features of Cooja can be summarized by the following: 1) It models the fine-grained node behavior by mimicking the exact instruction set of a particular node, 2) It performs a simulation at the hardware-level using the authentic hardware profiles of the emulated nodes, therefore, the developed code can be uploaded directly into the real targeted platform, 3) It allows exploiting available network models such as radio

propagation, medium interference, topologies, and link quality models to evaluate the proposed work under real-life conditions, and 4) It enables the simulation of large scale networks with easy debug through both command line and graphical interfaces.

2.3.2 Calculation of Energy Consumption

It is important to mention that counting the packet transmissions is not necessarily a good evaluator for total energy consumption. This is because IoT nodes operate at different current levels. To perform energy calculations considering this fact, Energest [66], a software-based module, is used to provide a real-time estimation of the consumed energy for the different states of individual nodes. The Energest module maintains a table with entries for node's peripherals such as the micro-controller, and the radio chip. Each table entry contains the total time that the corresponding component has been turned on. To calculate the energy consumed by the micro-controller component, the Energest module tracks the time intervals in which the CPU stays in either the on state referred to as CPU_{on} , or the low-power mode state referred to as CPU_{lpm} . The state of the radio transceiver is also tracked, and the intervals for the transmission state, referred to as Tx , or the listening state, referred to as Rx , are also logged. The energy consumption for each component, E_S , is calculated using (2.3) as follows:

$$E_S = (T_S * C_S * V) / (L) \quad (2.3)$$

where T_S and C_S are the time spent measured in clock ticks and the electrical current consumed, respectively, for the state S . V refers to the battery voltage, while L is the number of clock ticks per second which is equal to 32768 ticks/sec. Using the data-sheet of the emulated Wismote platform, the current requirements for states CPU_{on} , CPU_{lpm} , Tx , and Rx are 2.2 mA, 1.69 μA , 33.6 mA, and 18.5 mA, respectively [67].

CHAPTER 3

LITERATURE REVIEW

3.1 Service Discovery in the Literature

In this subsection, firstly, the SD protocols tailoring IoT are surveyed. Then, some studies that adapt mDNS/DNS-SD are discussed. Finally, a brief account of the related work for mDNS/DNS-SD optimizations is given.

3.1.1 Service Discovery Protocols for IoT

Many SD protocols have been developed in the literature with IoT requirements in mind. For simplicity and relevance, the most current protocols that support IP networking are briefly introduced. The retrieved protocols broadly fall into three categories; centralized, distributed, and hierarchical approach. This categorization is based on the communication pattern that is used among entities of a particular protocol.

In the centralized approach, descriptions of all offered services are stored in a central component called a directory. All nodes should register their services in

this directory so that other nodes can request their interested services from it. The authors in [68] proposed a centralized context-based SD protocol named **TRENDY**. TRENDY chooses the Border Router (BR) as directory and divides nodes to Group Leaders (GL) and Group Members (GM). GLs and GMs are constructed based on their locations (e.g., the nodes in a room are assigned one GL and the remaining become GMs). However, in addition to the single point of failure and bottleneck issues, TRENDY induces high maintenance overhead to manage the formation of GLs and GMs and maintain the network consistent over time. An open service architecture for sensors referred to as **OSAS** is another centralized SD presented in [69]. The most recent centralized work is proposed by Lee *et al.* in [15]. They proposed a DNS-based naming framework named **DNSNA**. This framework utilizes servers running central DNS and DHCP. Nodes use these servers to register their names and services and to discover the other services provided in the network as well. A new format to describe these services is proposed which indeed causes incompatibility issues. Nodes in [15] cannot get the configuration information distributively because the required information is requested from the designated DNS server. Besides, since the DHCP does not have a resolution mechanism for DNS name conflicts, Lee *et al.* framework has the same shortcomings. Therefore, the resolution of these conflicts requires administrative intervention.

In general, the workload in the centralized approach is undertaken by a dedicated directory so that it stores information about all the nodes and services of the network. Therefore, the directory should be powerful enough and be known to the joining nodes beforehand. This approach suffers from a single-point-of-failure problem because of

the use of the central directory. Moreover, high traffic is generated near the directory which affects the performance of the overall network in terms of energy consumption and delay.

In the distributed approach, however, each node stores information about the services it provides locally. Any node interested in a particular service should query all the nodes that could satisfy its request. A protocol named **SDP** is a distributed SD for the Bluetooth technology presented in [70]. Another distributed SD protocol for ad hoc networks named **DEAPSpace** is proposed in [71]. Another distributed SD protocol designed for ad hoc networks is presented in [72] referred to as **Konark**. The authors in [73] introduced a distributed SD protocol named **Sleeper**. The mDNS/DNS-SD that have been introduced in [17], [46] are distributed SD standards widely used in IoT. They are part of the main interests of this thesis and therefore, their details are presented in Subsection 2.1.3. Finally, Djamaa et. al in [74] proposed a distributed SD protocol named **EADP**. In EADP, Joining nodes broadcast their services in the network utilizing the TA to reduce the traffic generated. Also, nodes broadcast their queries in the case they are interested in some services provided by other nodes.

The hierarchical approach combines both centralized and decentralized approaches. It divides the participating nodes into groups or clusters. The nodes in a particular cluster should elect the so-called cluster head (CH) which acts as a master. These CHs should be resource-rich nodes which is something that can't be assumed in constrained networks. Nodes inside a cluster should then register their services with their local CH node. A node contacts the CH if it needs to inquire about a specific service of interest. A CH node should always be in an active mode to answer

SD requests on behalf of non-CH nodes. The frequent maintenance and the election process for the CHs induces a high overhead making this approach unsuitable for constrained networks. The authors in [27] proposed a hierarchical SD protocol for ubiquitous computing referred to as **SANDMAN**. Finally, an electronic number mapping (**ENUM**) introduced in [28] is another hierarchical-based SD protocol. It uses a distributed directory approach based on the idea of resource-rich master nodes holding information about available services in their vicinity. Added to the introduced complexity, delays, and maintenance overhead, ENUM-based SD assumes the availability of powerful nodes to play the role of masters.

To sum up, Table 3.1 summarizes the key relevant characteristics of the SD protocols that are surveyed in this subsection. These properties have been identified in several studies of the literature [75], [76]. In this table, “Discovery” and “Registration” columns define how the services are registered and discovered, respectively, by the nodes in the network. The mark “?” indicates that the corresponding characteristic is not clearly stated in the specification of the protocol. It can be concluded that the lack of full backward compatibility is the main drawback of most of these protocols. This is because they depend highly on special translation protocols, leading to a loss of end-to-end functionality, interoperability, flexibility, and scalability.

3.1.2 Adaptations of mDNS/DNS-SD

The research community has been paying high attention towards adapting mDNS/DNS-SD to provide the SD functionalities. In this subsection, some of such adapting frameworks are sketched out. Jara *et al.* in [79] and [80] present a set

Table 3.1: Comparative summary for the SD protocols in IoT

SD protocol	Architecture	Registration	Discovery	Description	Compatibility
TRENDY [68]	C	M	M	URL	X
OSAS [69]	C	U	U	?	X
DNSNA [15]	C	U	U	?	X
SDP [70]	D	U	U	UUID	X
DEAPSpace [71]	D	B	B	None	X
Konark [72]	D	M	M	XML	X
Sleeper [73]	D	B	B	URL	X
EADP [74]	D	B	B	?	X
mDNS/DNS-SD	D	M	M	DNS-SD	✓
SANDMAN [77]	H	U	U	?	X
ENUM [78]	H	U	U	DNS	✓

C=Centralized, H=Hierarchical, D=Distributed,
M=Multicast, U=Unicast, B=Broadcast

of adaptation layers to extend the current Internet stack for IoT nodes. The authors adapt mDNS/DNS-SD to enable the homogeneous access of applications and services at the application layer over the IPv6 protocol. The SD task in [19] is done through mDNS/DNS-SD protocols. The SOA-governed industrial middleware introduced in [81] employs the mDNS/DNS-SD to discover the published resources in IoT-based networks. The authors in [82] presented a network management system in which the mDNS is used to discover the neighboring nodes. The authors in [83] proposed a distributed service framework in which mDNS/DNS-SD are utilized to discover peer nodes and services in the network. Finally, the ZigBee-IP stack specifies mDNS/DNS-SD as a protocol for realizing SD. The interested reader is referred to, [51], [84]–[91] for more mDNS/DNS-SD adaptation examples.

3.1.3 Enhancements of mDNS/DNS-SD

The study in [50] presents an implementation of mDNS for conventional nodes that is referred to by Bonjour. Subsequently, it is minimized, resulting in a version referred to as uBonjour, in [51] for constrained nodes and is integrated with ContikiOS. The authors of [92] implement further optimizations of uBonjour that are based on *DNS message compression*, and *class code* and *Time-To-Live (TTL) field compression*.

Stolkij *et al.* in [93] propose a context-aware extension for mDNS/DNS-SD to improve the scalability in large IoT networks. A resource discovery solution based on mDNS/DNS-SD for low power networks is presented in [94]. The proposed solution in [94] extends mDNS/DNS-SD to include support of proxy servers for service discovery in low power networks of resource-constrained nodes.

A distributed naming system referred to as DINAS is proposed in [18]. With DINAS, the names of nodes are created by encoding their descriptors in Bloom Filters (BFs) of predefined sizes. DINAS enables mDNS/uBonjour as an extension in case some nodes may want to discover the offered services in the network. The evaluation for DINAS highlights that mDNS still suffers from high communication overhead despite its message compression optimization done by the work in [92] mentioned above. Furthermore, the study points out that mDNS still needs improvements for the efficient support of multicast in multi-hop wireless networks.

Jara *et al.* in [88] present a lightweight implementation of mDNS/DNS-SD, referred to as lmDNS-SD, that is based on optimization strategies that are meant to fulfill the requirements of constrained nodes. The lmDNS-SD focuses mainly on the following

items. First, the number of records in authority and additional sections are pruned at the discovery stage. Second, the TXT records are minimized and the CoRE format [95] is utilized, while the TXT entries themselves are compressed. The study in [88] does not provide any implementation for the proposed strategies nor does it provide any relevant performance evaluation. Note that such strategies are incompatible with the DNS standard, and they depend on compression algorithms that produce high memory footprints. Hence, these strategies negatively affect the performance of the constrained networks.

3.2 Enhancements of N2N Routing Discovery

Routing enhancements for N2N-based communications in LLNs have been proposed by many studies in the literature. Please note that N2N traffic has various conventional names in the literature, for example, point-to-point, any-to-any, device-to-device, machine-to-machine, arbitrary, and bi-directional traffic. In this subsection, the most relevant studies for routing N2N traffic are surveyed. Overall, the related studies fall into four approaches which are: geographic, compact, hierarchical, and spectrum-based routing approach.

In geographic routing, some or all participating nodes are assigned a coordinate to reflect their geographic location within the network. The coordinate for the node may be obtained by a Global Positioning System (GPS) receiver or set statically for the case stationary nodes. In this approach, a packet is forwarded to the neighbor which is geographically closer to the destination. The proposals utilizing this approach

include GeoRank [96] and ER-RPL [32]

The authors in [96] present **GeoRank** in which SRPL is integrated with the position-based routing protocol, namely, Greedy Other Adaptive Face Routing (GOAFR) [97]. In GeoRank, a pre-configuring service or GPS receivers are used to determine the node's position. GeoRank considers a smart street lighting system as a motivating application. In GeoRank, N2N packets are greedily forwarded to the node nearest to the destination. A position list of pre-assigned DODAG roots is stored by all nodes to handle a Local Minimum Problem (LMP). The LMP occurs when no neighbor is closer to the destination other than the current forwarding node itself [98]. To deal with this, GeoRank selects the root with the smallest absolute angle difference to the source-destination line as the next hop to deliver N2N packets.

A reactive energy-efficient region-based RPL routing named **ER-RPL** is presented in [32]. In ER-RPL, the N2N routes are discovered by flooding a route request over some of participating nodes utilizing some Reference Nodes (RNs) with the location-awareness capability. To realize that, various temporary DODAGs are created by making use of a region formation object (RFO). Creating extra DODAGs incurs high traffic and makes the system even more complex. For N2N routing discovery, ER-RPL utilizes four types of message request objects (MROs) namely MRO(0), MRO(1), MRO(2), and MRO(4). ER-RPL is evaluated using the IEEE 802.11 which is usually not considered to be the best candidate for LLNs since the IEEE 802.15.4 is viewed as the optimal choice. Furthermore, ER-RPL neglects the buffer overflow issue which is an unrealistic assumption in memory-constrained nodes.

As a summary, the geographical-based approach has at least four challenges that

limit its adoption for providing appropriate reliability support for LLNs; these challenges are: 1) Increased system complexity due to the added extra hardware required to obtain the accurate geolocation of a node, 2) Careful addressing of the LMP; a problem caused by the incomplete connectivity of the coordinate space. The LMP occurs when no neighbor is closer to the destination other than the current forwarding node itself [98], 3) Since this approach does not account for the lossy nature of LLNs during the next-hop selection, it may end up with poor performance routes, and 4) To maintain the coordinates, nodes need to exchange one- or two-hop routing information which significantly increases the energy depletion.

The compact-based approach aims to achieve scalability through maximizing the memory capability to store more route entries. To do so, compact data structures such as a BF are used to represent the route entries rather than using flat information. Example studies in this category include BVR [99], CBFR [100] and ORPL [101].

A beacon vector routing named **BVR** [99] is a compact-based discovery protocol supporting N2N routing in LLNs. In BVR, predefined beacon nodes are selected to construct spanning trees using the flooding approach. These trees are constructed from every beacon node to all other ordinary nodes and subsequently used in the N2N routing process. Each node maintains a coordinate for itself and a number of coordinates for its neighbors. The coordinate is a vector containing information of distances from the node to all beacon nodes. For N2N routing, a node greedily forwards packets to the beacon node that minimizes the distance to a destination. A scoped flooding is used to manage the LMP.

A counting bloom filter routing referred to as **CBFR** scheme presented in [100]

is a routing mechanism targeting N2N communication and using a Collection Tree Protocol (CTP) [102] as an underlying protocol. The CTP is a collection protocol designed to work with applications for gathering data in WSNs. In CBFR, each node stores routing information, utilizing BFs, for all nodes in its sub-tree. Using BFs, rather than using flat information, decreases memory usage. N2N packets are forwarded towards a root via an up-tree path in which every node along the path check the presence of the destination address of the packet in its filter. If it does, the packet is then forwarded down-tree until it reaches its destination.

An opportunistic RPL-based routing scheme named **ORPL** is presented in [101]. ORPL uses the opportunistic concept to enhance the scalability of RPL in downward and N2N routing. Like SRPL, a node in ORPL stores routing information for all nodes in its sub-DODAG. Unlike SRPL, ORPL represents this information as either a bitmap or as a BF rather than the flat information used in SRPL. ORPL handles N2N traffic by first routing packets upwards to any common ancestor, and then downwards to the destination. A node anycasts N2N packets to a set of nodes instead of unicasting it to the a particular next-hop. This multi-path routing exploits more nodes as common ancestors where the opportunistic routing comes into the picture. When a node receives a packet, it decides whether or not to forward it. If it does, it will send back a link-layer acknowledgment.

To sum up, despite that the compact approach presents a memory-efficient way as opposed to the conventional use of the routing table, the problem of routing the packets through the root still exists. This is due to the possibility of false-positives lookups inherited by the BF operations [98].

The hierarchical-based approach aims to reduce the routing table size through hierarchically assigning addresses for the participating nodes in the network. In a sense, the addresses for close-by nodes are aggregated in a single routing entry. Examples of this approach include HiLow [103], SPR [104], MHCL [105] and MATRIX [57].

A hierarchical Routing over 6LoWPAN (**HiLow**) presented in [103] was suggested by the 6LoWPAN WG as an internet-draft. Nodes in HiLow are assigned short addresses during an association process with their parents. Assigning Addresses for nodes follows a predefined scheme considering the parent address and the number of associated children the parent already has. Subsequently, only these short addresses are used for communication. Each node should maintain a neighbor table containing information about its children and parent. The hierarchical addressing assignment enables nodes to know their depth in the constructed tree. To forward a packet, the receiving node determines whether it is ascendant or descendant to the destination. Following that, it selects the next-hop according to a predefined formula to which it should forward the packet.

The authors in [104] propose a stateless point-to-point (P2P) routing protocol named **SPR**. This proposal is based on a shortcut tree routing (STR) algorithm, developed by the authors to construct short N2N routes in terms of hop count. The STR constructs the forwarding structure of SPR by having nodes broadcast beacons at the bootstrapping stage. From a neighbor list, each node greedily selects a parent with the lowest depth and highest Link Quality Indicator (LQI). The parent selection process takes place only at the bootstrapping or link failure events. To discover the one-hop distance neighbors, a modified ND procedure with asymmetric link check is

employed. Neighbor Solicitation messages are also used to estimate the LQI in terms of ETX. Moreover, nodes periodically broadcast Neighbor Solicitation (ND) packets to monitor the reachability of their neighbors. SPR utilizes the hierarchical addressing structure to determine the short route a packet should be forwarded through. To recover the connectivity on link failure, an Orphan Notification (ON) packet is broadcasted which indeed increases the overhead.

A hierarchical IPv6 address allocation scheme is presented in [105] and referred to by **MHCL**. The objective of MHCL is to sufficiently enable downward and N2N traffic with small routing tables. The main idea of MHCL is to partitioning the IP address hierarchically among participating nodes connected with the BR. During DODAG construction process enabled by RPL, each node receives an address range from its parent and partitions it among its children. A node stores the address range of the whole subtree rooted in itself. MHCL modifies the control messages defined by RPL to contain the address allocation and partitioning policies.

A multi-hop address allocation and dynamic any-to-any routing for 6LoWPAN named **MATRIX** is presented in [57] and [106]. MATRIX is built upon the idea of using hierarchical IPv6 address allocation. It uses the underlying collection tree topology provided by CTP protocol. The authors claim that their previous work, MHCL, could not deal with network faults and dynamic networks. Therefore, MATRIX comes to fill this gap. It considers nodes with static locations and enables them to choose different parents based on LQI. MATRIX uses only one-hop information in the routing tables. The memory footprint of the MATRIX nodes is in order of the number of children at any given time. Finally, The authors also presented μ MATRIX

in [107] as a mobility extension support for MATRIX.

In general, the IPv6 addresses for LLN nodes are autonomously derived from their unique MAC addresses. Therefore, the IPv6 addresses are unstructured and do not reflect the underlying network topology. With the hierarchical approach, two nodes that are physically close to each other may result in an arbitrarily longer path between them compared to the shortest path. Furthermore, building and maintaining structured IPv6 addresses require transmitting many control packets especially in high dynamic networks such as LLNs which leads to high congestion.

The spectrum-based routing approach assumes an underlying distributed cognitive radio architecture to sense and build a spectrum map to determine the best available N2N routes in terms of hop count. Debroy *et al.* in [108] propose spectrum-aware energy-efficient routing for device-to-device communication in IoT termed **SpEED-IoT**. In SpEED-IoT, extra special spectrum sensors are deployed geographically to cover the network. The uncovered nodes stay unreachable even if they can communicate with other IoT nodes. Moreover, the unreachability issue is not addressed in SpEED-IoT. The deployed sensors alongside some ordinary nodes, called Edge Nodes (ENs), are utilized to discover N2N routes. The discovery mechanism, initiated by the source node, employs a selective flooding approach to discover a path towards the destination. While SpEED-IoT may end-up with sub-optimal N2N routes, the following are some drawbacks SpEED-IoT may suffer from: 1) The use of extra special sensors adds more complexity to the network, 2) SpEED-IoT still partially inherits the drawbacks of the flooding approach in terms of excessive consumption of network resources, and 3) SpEED-IoT suffers from an uneven load distribution issue

since the ENs forward more packets, in terms of discovery-related messages, relative to other ordinary nodes.

The authors in [109] proposed a reactive N2N route discovery named **P2P-RPL**. To discover an N2N route in P2P-RPL, a source node initiates a route discovery process by constructing a new DODAG rooted at the source node itself. Through this process, a Route Discovery Object (RDO) is disseminated to the whole network until it reaches the intended destination. Upon receiving the RDO, the destination node sends back a Discovery Response Object (DRO) via the reverse path. Once the N2N route is discovered, the temporarily constructed DODAG is removed, and packets can flow between the source and destination nodes, along the installed route utilizing the source routing approach. The main drawbacks of P2P-RPL are the following: 1) It discovers an N2N route via flooding route request packets throughout the whole network which increases traffic congestion and energy consumption, 2) It creates an extra temporary DODAG in which the source node acts as a root. The discovered path to the destination, therefore, might not be the optimal one especially under asymmetric links, 3) It adds an extra overhead due to maintaining an additional temporary DODAG by all participating nodes, and 4) it defines new control messages requiring changes to the specifications of the standard protocol which lead to possible compatibility issues.

Finally, [110] and [111] propose enhancements to RPL, named **ERPL**, that are limited to work only when the destination node is either a direct neighbor of the source node or a direct neighbor of a node relaying the data packet. However, multihop N2N communications are not supported in ERPL, where ERPL falls back to using the

original RPL protocol. In the latter case, ERPL resorts to using the original RPL protocol.

Table 3.2 summarizes the properties of the studies mentioned in this subsection. In the table, the “N/A” term means either the study has no baseline protocol such as in “Base” column or the property is not stated clearly in the corresponding study.

Table 3.2: Properties summary of N2N enhancements in the literature studies

Work	Year	Type	Base Protocol	MOP	Approach Used	Routing Metric	Performance Metrics	Compared With
GeoRank [96]	2015	Pro	RPL	S	Geographical	Hops	Hops, Cache	SRPL
ER-RPL [112]	2016	Rea	RPL	NS	Geographical	ETX	PDR, Hops, Energy	NSRPL
BVR [99]	2005	Pro	N/A	S	Geographical	Hops	PDR, CTO	N/A
CBFR [100]	2012	Pro	CTP	S	Compact	ETX	PDR, Delay	CTP
ORPL [101]	2013	Pro	RPL	S	Compact	ETX	PDR, Delay, Energy	SRPL, CTP
HiLow [103]	2007	Pro	N/A	S	Hierarchical	Hops	N/A	N/A
SPR [104]	2012	Pro	N/A	NS	Hierarchical	Hops	PDR, Hops, RTT	HiLow, SRPL
MHCL [105]	2016	Pro	RPL	S	Hierarchical	ETX	PDR, CTO	SRPL
MATRIX [57]	2018	Pro	CTP	S	Hierarchical	ETX	PDR, CTO	SRPL, XCTP
SPeED-IoT [108]	2019	Rea	N/A	S	Flooding	Hops	PDR, Hops, Energy	[113], MHCL
P2P-RPL [109]	2011	Rea	RPL	NS	Flooding	ETX	Hops	SAMER [114], LinGO [115]
ERPL [111]	2019	Pro	RPL	S	As SRPL	ETX	PDR, ETX, Delay	NSRPL, SRPL

Pro=Proactive, Rea=Reactive, S=Storing, NS=Non-Storing

CHAPTER 4

ENHANCED MDNS

4.1 Chapter Motivations

Service Oriented Architecture (SOA) has been considered the potential architecture for the IoT vision [116]–[118]. In the simplest form, resources on LLNs nodes can be introduced locally or globally as services for monitoring the surrounding environment or controlling the actuating switches. Furthermore, these services may be presented in more composite forms to offer more complex functionalities [119]. In SOA, LLN nodes require standard mechanisms with high scalability to name and discover such nodes and their offered services. The denoted names can be human-readable or even appropriate for N2N communications [18], [120], [121]. The successful realization of the SOA depends heavily on the availability of an SD protocol [74]. The main rule of the SD protocol is to uniquely configure and advertise the participating nodes along with their hosted services throughout the network. The mDNS protocol and its companion technology the DNS/SD [17] are the most common distributed SD protocol for constrained nodes nowadays. However, because it was mainly designed

for conventional computing nodes with nearly unlimited capabilities, mDNS in its current form is not optimized enough for constrained LLN nodes in IoT. In particular, mDNS generates high control traffic during the configuration phase and registration process [15], [18], [22]. To the best of the authors' knowledge, existing researches focus on enhancing the discovery phase and no work has considered the configuration phase of the mDNS protocol. This chapter analyzes the operations of mDNS and introduces some mechanisms that can reduce the traffic generated and thereby the energy consumed.

4.2 Chapter Contributions

The following points summarize the key contributions included in this chapter:

1. This chapter presents a thorough analysis of the mDNS configuration phase considering the race condition problem arising from simultaneous start-up events.
2. We propose enhanced mDNS, referred to as EmDNS, the first work considering the characteristics of constrained LLN nodes by focusing on the auto-configured naming and registration processes. A significant reduction in control traffic is achieved since less communication is needed to deploy and configure the nodes autonomously.
3. We implement EmDNS in ContikiOS version 3.0 and provide extensive simulations using Cooja. The simulation results demonstrate that the EmDNS outperforms the mDNS in terms of control traffic overhead, energy consumption, and

the ability to provide unique names for deployed nodes while maintaining an acceptable local cache hit ratio. Furthermore, EmDNS nodes work seamlessly in a hybrid network with mDNS nodes which confirms the EmDNS backward compatibility support.

4.3 System Models

This section covers the network and node models presented in this chapter. Table 5.1 lists the main notations used in this chapter and their definitions.

Table 4.1: Used notations summary

η	Total number of nodes in the network
$\mathcal{R}$	Network area size
N_i	The i^{th} node
N_i^H	Node i^{th} 's <i>host_name</i>
$N_i^{S_j}$	The j^{th} <i>service_name</i> provided by i^{th} node
$\mathcal{L}_i$	Set of all services provided by i^{th} node
Ω	Set of all network services
Ψ	Number of all <i>services_names</i> provided in the network
IWT_{iq}	Initial Waiting Time picked by i^{th} node at q^{th} PS stage.
P_{ik}	Probing message by i^{th} node at time k
RP_{ik}	Probing message responding to P_{ik}
RA_{ik}	Announcement message responding to P_{ik}
T_{ik}	Time between sending P_{ik} and receiving either RP_{ik} or RA_{ik}

4.3.1 Node Model

The nodes deployed in the network are denoted by $N_0, N_1, \dots, N_{\eta-1}$. The LLN nodes are integrated with the IEEE 802.15.4 standard as the MAC and PHY layers operate at 2.4 GHz band and provide 250 kB/sec. Each node N_i , for $0 \leq i < \eta - 1$, has to be assigned a unique *host-name* N_i^H and a list of services $\mathcal{L}_i$ where $|\mathcal{L}_i| > 0$ with

unique *service-name* $N_i^{\mathcal{S}^j}$ for each service where $0 \leq j < |\mathcal{L}_i| - 1$. $|\mathcal{L}|$ is defined as the cardinality of the set $\mathcal{L}$. It is worth mentioning that the naming task is not limited to the assignment of the $N_i^{\mathcal{H}}$ and $N_i^{\mathcal{S}^j}$, but also can be extended to include all host- or services-related attributes. Thus, all conditions and assumptions for these assignments are also valid for other related attributes. The number of services $|\mathcal{L}_i|$ provided by each node varies from one node to the other, but each node provides at least one service. The set denoted by Ω is the set of all services provided by all nodes. The total number of all *hosts_names* is equal to η , as mentioned earlier, whereas the total number of all provided *services_names*, denoted by Ψ , in the network of interest is equal to $\Psi = |\Omega| = \sum_{i=0}^{\eta-1} |\mathcal{L}_i|$. From now on throughout this chapter, the terms “configurations” and “identifications” are used interchangeably and are used to mean $N_i^{\mathcal{H}}$ and $N_i^{\mathcal{S}^j}$ assignments. The ultimate goal of any zero-configuration protocol is to satisfy the three following conditions 1) Each node should have its unique *host_name* within the local network, 2) Each service should have its unique *service_name* within the node, and 3) The services should have their unique *service_name* across all network nodes. These three conditions are summarized in the formulas below:

$$N_k^{\mathcal{H}} \neq N_\ell^{\mathcal{H}}, \forall k \neq \ell \quad (4.1)$$

$$N_\ell^{\mathcal{S}^a} \neq N_\ell^{\mathcal{S}^b}, \forall a \neq b \quad (4.2)$$

$$N_k^{\mathcal{S}^a} \neq N_\ell^{\mathcal{S}^b}, \forall k \neq \ell, \forall a \neq b \quad (4.3)$$

where $0 \leq k, \ell < \eta - 1$ and $a, b \in \Omega$

4.3.2 Network Model

We consider a large number, denoted by η , of heterogeneous IoT nodes. These nodes are deployed in an area of predefined size $\mathcal{R}$ squared meters. The *stateless address auto-configuration* mechanism and the *duplicate address detection* procedure specified in RFC 4862 [47] are exploited to generate the node’s local and global link IPv6 addresses. The mDNS protocol has been designed to work in a local link infrastructure-less network where the nodes communicate distributively in a fully connected mesh-like network. According to the protocol specifications, simply messages should be received by all participating nodes in the network. However, multicast messaging can be used to distribute control messages in multi-hop networks efficiently. Addressing details are specified in Table 4.2.

Table 4.2: Multicast addresses used by mDNS protocol

	IPv4	IPv6
Mac address	01:00:5E:00:00:FB	33:33:00:00:00:FB
IP address	224.0.0.251	FF02::FB
UDP port	5353	

4.4 Overview of Enhanced mDNS (EmDNS)

In this section, we overview and discuss the key traffic reduction mechanisms and the definitions corresponding to the algorithms proposed in EmDNS.

4.4.1 EmDNS Design Overview

The configuration phase in the mDNS protocol includes three stages: a Selection Stage (SS), a Probing Stage (PS), and an Announcement Stage (AS). Upon every start-up, the nodes must select their unique configurations such as the *host_name* and *services_names*, the addresses for their interfaces, and any additional information [17]. This procedure is referred to as the SS and is considered the very first stage of the configuration phase. Upon completion of the SS, the starting-up node proceeds to probe the network with these configurations after some delay. The purpose of this PS is mainly to check the configurations redundancy within the network. If no conflict is received for three consecutive probing cycles, the node proceeds to execute the AS. In this chapter, we propose EmDNS consisting of the following three main enhanced mechanisms: a *Persistent-based Selection Stage (PSS)*, a *Simultaneous Start-up Enhancement (SSE)*, and an *Announcement Suppression Mechanism (ASM)*. Figure 4.1 shows the flow chart for the proposed mechanisms in EmDNS.

4.4.2 The Persistent-based Selection Stage (PSS)

Using the original procedure of the mDNS protocol typically results, with a high percentage, in selecting configurations during the SS that are not unique across the network. These conflicts subsequently must be resolved during the PS. Therefore, performing the selection process frequently at each configuration phase using the steps recommended by the original mDNS procedure causes the following negative impacts:

- i) increasing the convergence time as the node needs to begin a new round of SS in

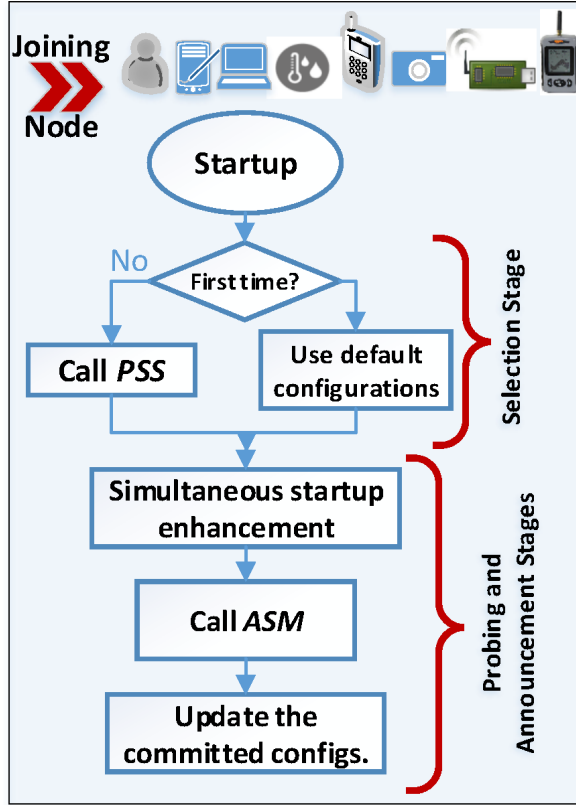


Figure 4.1: The configuration phase for the proposed EmDNS

case a conflict occurs, ii) increasing the power consumption by the CPU of the node as the SS needs to be executed multiple times, and iii) increasing the generated traffic in the network as the selected configuration will be used for probing at the PS leading to increased energy usage by the radio transceiver.

To remedy the above mDNS shortcoming, the work of this chapter proposes using the unique committed configuration selected from the previous start-up. Once an available unique configurations have been determined, by probing without receiving any conflicting response, record this newly chosen name in persistent storage so that the node will use the same configurations the next time it is power-cycled. This technique requires that the committed configuration be stored in persistent storage,

and hence the selection process is not needed at every start-up except when the probing stage fails. Figure 4.2 shows a flow chart of the proposed PSS. The PSS mechanism is implemented using two functions: one is the *Write_configuration* function to store the unique committed configuration in the persistent memory, such as a one-megabyte external serial flash memory. The second is the *Read_configuration* function to restore this configuration at the beginning of each start-up event.

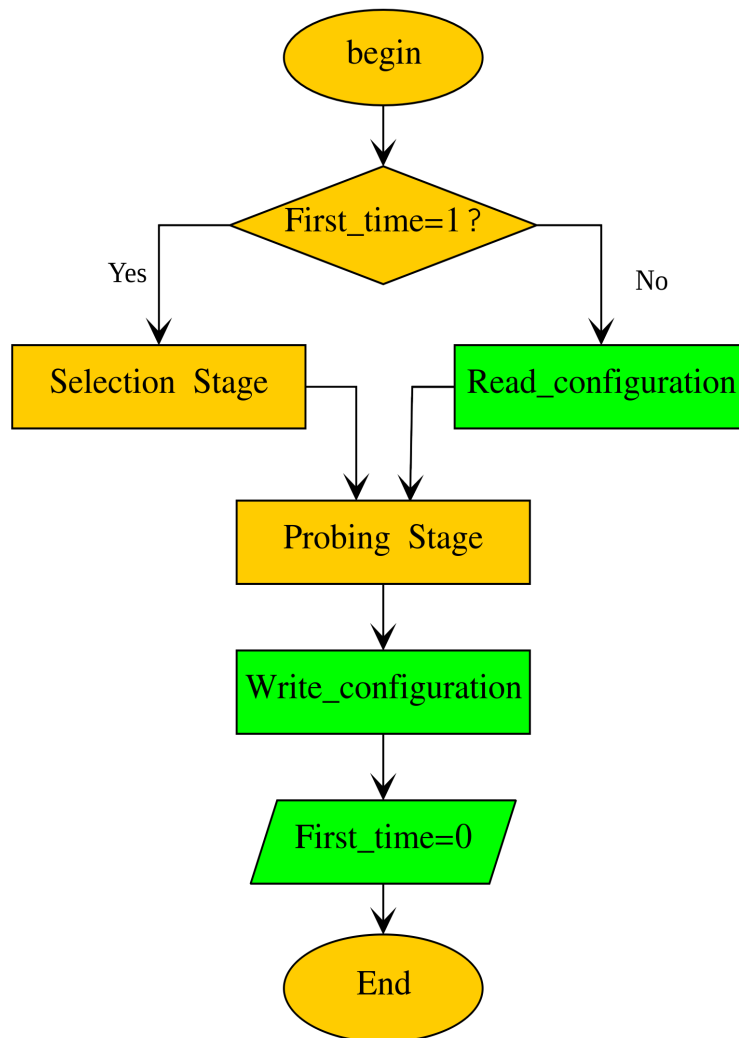
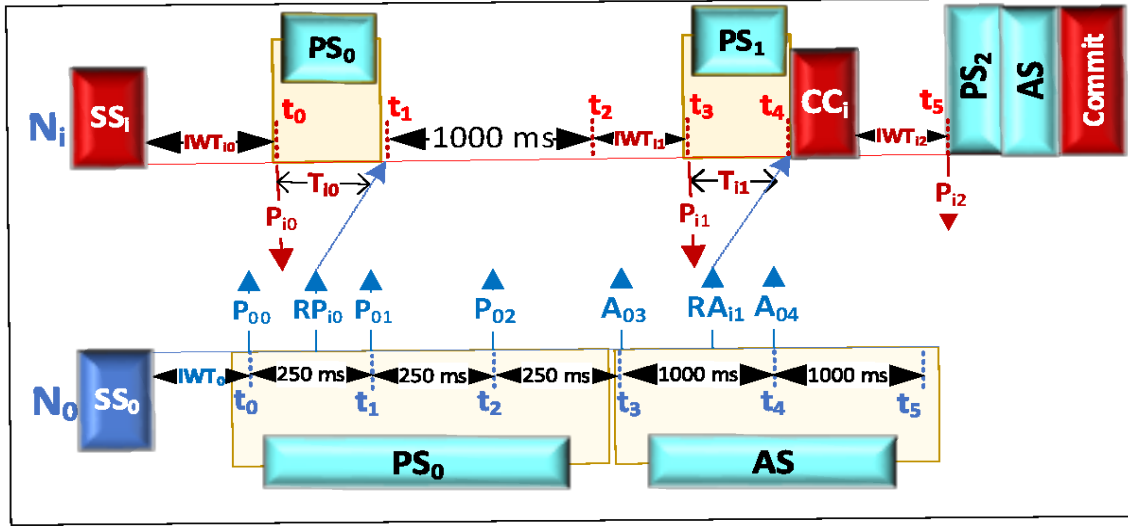


Figure 4.2: The PSS flow chart

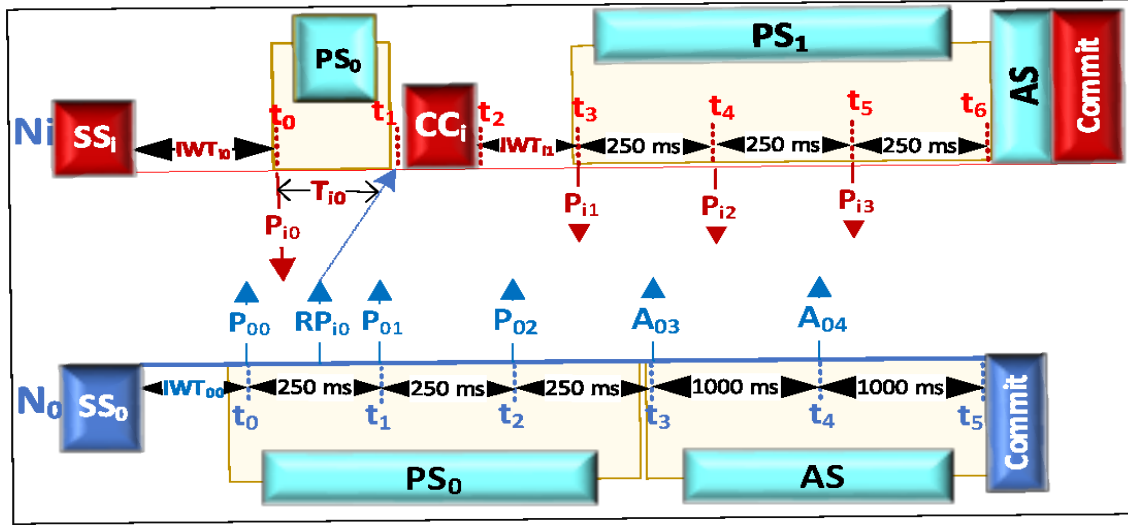
4.4.3 The Simultaneous Start-up Enhancement (SSE)

Having finished the SS successfully, the mDNS nodes proceed to probe the network with the selected configuration after some Initial Waiting Time (IWT), denoted by IWT_{iq} , as shown in Fig. 4.3(a) and Fig. 4.3(b). As multiple PSs may be needed until a unique configuration is successfully reached, the IWT_{iq} refers to the IWT picked by i^{th} node at q^{th} PS stage. The IWT in milliseconds is a random value chosen uniformly from the interval $[0, 250]$. In the case of simultaneous start-ups, it happens that some nodes are starting-up and probing with the same configuration roughly at the same time. This case can occur when several nodes are powered on simultaneously, or when some external events happen to cause a group of hosts to send probes approximately at the same time. In this subsection, we first discuss in details the behavior of the mDNS protocol in the presence of such simultaneous start-up events as depicted in Fig. 4.3(a). Then, the relevant EmDNS improvement, shown in Fig. 4.3(b), is discussed.

Figure 4.3(a) shows that N_i went to probing while N_0 was probing. If this is the case, then a race condition occurs as the two nodes N_0 and N_i are probing for the same configuration simultaneously [17]. The probing messages by N_0 and N_i will be received by each after an amount of time T_{i0} depending on the network processing and propagation speed. Once both of them are probing for the same configuration, eventually one of them will win to use that configuration. The probing message includes multiple records in which $N_i^{\mathcal{H}}$ or $N_i^{S_j}$ is only one of these records. Comparing the data in these records of the two probing messages will help in deciding which node will eventually win with the probed configuration.



(a) mDNS



(b) E-mDNS

Figure 4.3: Timelines of mDNS and EmDNS protocols for the simultaneous start-up events

Figure 4.3(a) shows that N_0 wins with the probed configuration, and therefore N_0 will continue with sending the second P_{01} and the third P_{02} probing messages. P_{ik} refers to the $(k + 1)^{st}$ message sent by i^{th} node. It should be mentioned here that while N_0 is in the probing state, it has to respond to all probing messages sent by other nodes that are trying to probe for the same configurations. The probing node

will respond only if it finds that its data is lexicographically greater than that in the received probing message, an implementation-dependent condition, otherwise the node will flip its behavior and act like N_i . After 750 *ms* at time t_3 , N_0 will begin the *AS* by sending its first announcement message A_{03} notifying other nodes that it has committed to $N_0^{\mathcal{H}}$ or $N_0^{\mathcal{S}_j}$ as its unique configuration. Furthermore, N_0 will send one more announcement message A_{04} and finally finishes the procedure by committing to the selected unique configuration.

On the other hand regarding N_i , 1000 *ms* later at t_2 , N_i has to begin a new round of probing (PS_1) with the same previously probed configuration. Definitely, when N_i starts the new probing cycle, node N_0 will be in the announcement stage. Therefore, N_0 will respond with the announcement message RA_{0i} declaring that it is already using $N_i^{\mathcal{H}}$ or $N_i^{\mathcal{S}_j}$ as its unique configuration. When N_i receives RA_{0i} , it will change its $N_i^{\mathcal{H}}$ or $N_i^{\mathcal{S}_j}$ in the change configuration (CC_i) stage. Once the N_i 's configuration is changed, a new probing stage PS_2 starts again by sending a probing message P_{i2} at time t_5 , but now with a new (i.e. different) configuration. Figure 4.3(a) assumes, for sake of simplicity, that the probed configuration is unique at this stage, and no more conflicts, i.e. identical configurations, will be received by N_i . Eventually, N_i will proceed to probe again two more times and continues to the *AS* similar to N_0 and finally commit to the selected unique configuration.

The proposed improvement attempts to minimize the time needed to distribute unique configurations for all nodes in the network and to reduce the number of the packets required to achieve this goal. By recalling Fig. 4.3(a), N_i has to go through the operations between t_1 and t_4 to satisfy the standard procedure of the mDNS

protocol. In mDNS, it is assumed that the received probing messages after entering first PS_0 , i.e. after sending P_{i0} message, might be either a real probing message or an orphan message echoed by some switch in the network. For this reason, N_i has to probe again with the same configuration after 1000 *ms* delay which is hopefully enough to eliminate the above ambiguity. The foregoing assumption is applicable for Ethernet networks for which mDNS is originally designed. However, this assumption is not valid for single-hop wireless networks where echos cannot occur. Even with multi-hop communication, the used multicast mechanism should apply strict policies to prevent the occurrence of these kinds of echos. While this chapter focuses the single-hop communication, the work can be extended for multi-hop communication networks as discussed in the future work Subsection 6.2.1. Therefore for the proposed enhancement, the received probing message is considered to be a real one sent by another node after start-up and there is no need to delay and probe with the same configuration later on. When two or more nodes detect a conflict at the probing stage, then the node which has data that is lexicographically greater will ultimately win. The other probing nodes have to begin a new round of PS directly, i.e. with no 1000 *ms* delay, using a different configuration. Figure 4.3(b) summarizes the interaction using the revised configuration procedure.

4.4.4 The Announcement Suppression Mechanism (ASM)

Having finished the PS successfully, the node must send a chain of unsolicited announcements to update its neighbor's caches with its new committed configurations. The main purpose of this AS is to notify the local network with the services provided

by the joining node. Upon reception of these announcement messages, the neighboring nodes will decide to store this information according to the availability condition in their local caches. The availability condition can be evaluated by comparing the number of entries currently used with the maximum possible entries. The cached information might be used later in case the node is interested in the advertised services. In this case, the node can initiate the connection with the target address fetched from its local cache instead of broadcasting a query to get this information. Each cached entry consumes approximately 66 bytes of the node's memory divided into three parts: 18, 16, and 32 bytes for control fields, the IPv6 address field, and the name field, respectively.

Due to the memory limitation of the typical IoT nodes, a small number of entries in their caches can be allocated to store the announced information. As a result, most of the received advertisements will be ignored or dropped out due to the lack of enough memory capacity or buffer overflow issues [122]. Furthermore, the high loss rate for LLNs causes some of these advertisements to be lost. As a consequence, the benefits of the AS are circumvented by these three mentioned factors; limited cache, buffer overflow, and high loss rate. The proposed ASM attempts to suppress the AS without any adverse effects on the original protocol correctness or functionalities. The other nodes will still be able to store the needed configurations of their neighbors even with the use of the ASM mechanism. The ASM utilizes the three consecutive probe messages, generated at the PS, as an implicit indication that the probing node is going to commit on the probed configuration. The receiving node keeps track of the probing nodes by assigning a tentative entry for each of the received probes in

its local cache. The allocated tentative entries will be identified as consistent entries when three consecutive probe messages, for the same configuration are received, and no more conflicts on this particular configuration are observed. The availability condition mentioned above still should be considered before allocating any tentative entries in the respective cache of the host node. Using the above reasoning, the configuration phase can still function with the AS being suppressed. The ASM mechanism is described in Algorithm 1 and explained in detail throughout the remaining part of this subsection.

Algorithm 1 ASM ALGORITHM

Input: *prob_msg*, *entry*

Output: Service's records resolution and caching

when new prob_msg is received and no conflict :

```

1: for each entries in the cache do
2:   if ( entry[name] = prob_msg[name]
        && entry[addr] = prob_msg[addr]
        && entry[count] < MAX_PROB_MSGS) then
3:     entry[count] ++
4:     if entry[count] = MAX_PROB_MSGS then
5:       entry[state] = DONE
6:     end if
7:   else
8:     goto create_new_entry
9:   end if
10: end for
    create_new_entry :
11: if CURRENT_ENTRIES < MAX_ENTRIES then
12:   create entry
13:   entry[count] = 1
14:   entry[state] = TENTATIVE
15:   entry[TTL] = 1000 ms
16:   entry[name] = prob_msg[name]
17:   CURRENT_ENTRIES ++
18: else
19:   Ignore prob_msg
20: end if

```

Each node allocates some entries in its memory to cache information about its neighboring nodes. This information includes fields such as configuration (i.e., *name*) and address *addr* appended with an expiry time (i.e. TTL), the current *state* (*TENTATIVE*, *DONE*, ... etc.) and a probe counter (*count*). These fields are among other fields comprising the cache entry, and they are of the utmost relevance to the ASM algorithm. Upon receiving a probe message *prob_msg* with no conflict with the receiving node, the ASM algorithm is initialized. The ASM compares the received probe information with all entries kept in its local cache. If the probe information is found, then a count field is incremented and the process repeats until the probe sending node reaches the maximum number of probes. At this point, the entry is marked as *DONE* and the stored configuration is deemed valid. On the other hand, if the probe is not found in the cache, then a corresponding entry is created and added to the list of entries. The algorithm allows the creation of multiple entries that are conflicting with each other, i.e. the same name but from different nodes. However, only one of these records will eventually be completed and be marked as *DONE*. The fact that the conflicting nodes will be delayed by 1000 *ms*, except for the node that is going to win the competition, and the use of the 1000 *ms* as the TTL for these entries, ensure that only one valid entry is kept at the end.

4.5 Performance Evaluation

4.5.1 Simulation Set-up

EmDNS is implemented on ContikiOS version 3.0 and evaluated using Cooja emulator. In this simulation, we consider a random topology in which the deployed nodes can reach each other in the single-hop distance through the same wireless channel. To model the lossy environment, a UDGM model is used with a transmission range of 100 meters and an interference range of 120 meters. For the sake of simplicity, the transmission range is unified for all nodes in the local network under study. This evaluation uses the Wismote platform as the target mote for the proposed enhancements in all experiments.

Given that the energy consumption is highly dependent on the routing overhead and selected MAC and RDC protocols [123], these parameters are configured accordingly. To focus on the properties of the naming protocol mDNS, no routing protocol is used in this simulation since all nodes can reach each other in one-hop communication. The UDP, which does not involve re-transmissions, is used as a default transport layer in the IoT platform. For the same mentioned reason, we choose to use the IEEE 802.15.4 protocol at the MAC layer accompanied by unslotted CSMA/CA in the beacon-less mode.

For the RDC, we employ the ContikiMAC protocol [124]. ContikiMAC utilizes several optimizations to reduce the consumed energy by turning off the node's radio interface for most of the time. For the implemented asynchronous RDC protocol, receivers wake up periodically to search for potential transmissions by a rate referred

to as the CCR. The senders have to seize the moment when the targeted receiver is awake. The time-lapse between two periodic wake-ups is called the Channel Check Interval (CCI) or wake-up period. The CCR value used in the simulations is 32 Hz.

The simulation considers various sizes for the tested network, ranging from 5 nodes to 30 nodes in steps of 5, for all evaluations. All the experiments are repeated with 10 different random seed values, and the overall number of packets produced in the network, the convergence time, and all needed statistics are logged into a text file. The 95% confidence interval is used to plot the error bars in some of the presented results. Overall, the simulation parameters summarized in Table 5.2 are used throughout this work unless specified otherwise.

Table 4.3: Simulation parameters

parameter	Configuration
Platform / Simulator	ContikiOS v3.0 / Cooja
Topology	Random topology with single-hop distance
Network size	Variable (5, 10, 15, 20, 25, and 30 nodes)
Radio environment	UDGM
Loss model	Distance loss
Transmission range	Transmission: 100m, Interference: 120m
Emulated mote	Wismote with 16KB RAM and 128KB ROM
Transport	UDP over IPv6
MAC & PHY	CSMA/CA & IEEE 802.15.4
Radio interface	CC2520 2.4 GHz
RDC / CCR	ContikiMAC / 32Hz
Buffer size	8 packets (8 * 127 bytes)
Number of iterations	10 iterations
Confidence interval	95%

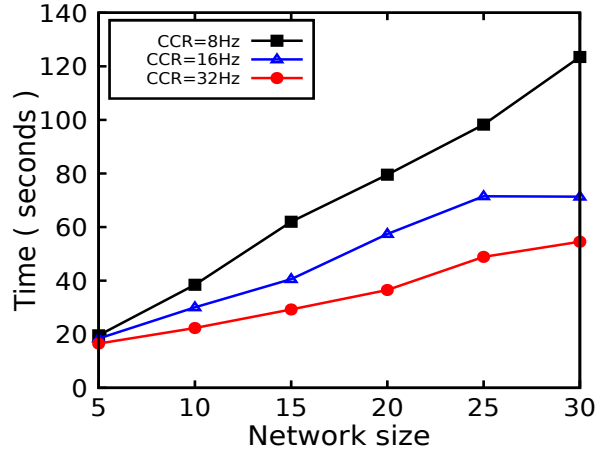
The simulation scenario runs two stages as explained below. In the first stage, all η nodes are started-up simultaneously with a unified configuration and begin to compete on getting the unique configurations for their *host_name*. At the end of this first stage, each node will, hopefully, get its unique configuration. All obtained

respective configurations are logged into an output file to be used in the second stage of the simulation. The uniqueness of the obtained configurations, evaluated in Subsection 4.5.5.1, at this stage depends on the employed protocol, i.e. mDNS or EmDNS. To simulate the case where the nodes rejoin the network after the network converges, the second stage of the simulation assumes a random $\eta/2$ nodes are selected to restart and rejoin the network. These restarting nodes are to use the unique configuration previously logged in case of EmDNS, or the default configuration for the case of mDNS. This step of the simulation aims to evaluate the PSS mechanism. Furthermore, after the completion of the first stage, each node uses its local cache to perform address resolution, i.e. *host_name* to address translation, for all host-address bindings in the network. This is used to evaluate the effect of the ASM enhancement on the cache hit ratio as presented in Subsection 4.5.5.2.

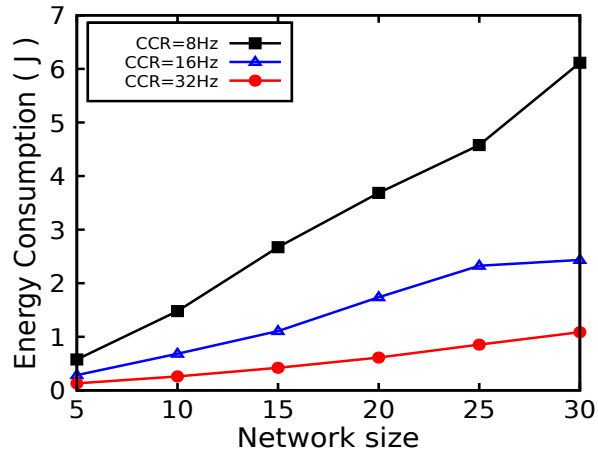
4.5.2 Impact of RDC on mDNS Performance

RDC parameters are expected to have a direct impact on overall network performance [125]. Therefore, this study first evaluates the effect of using RDC on the original mDNS protocol and attempts to select a CCR value to be used as a base for the rest of the performance evaluation. Figure 4.4 shows the obtained results for a) the convergence time, b) the energy consumption for the transmission state Tx , and c) the energy consumption for the CPU ON state CPU_{on} as a function of the network size η for CCR values of 8Hz, 16Hz, and 32Hz.

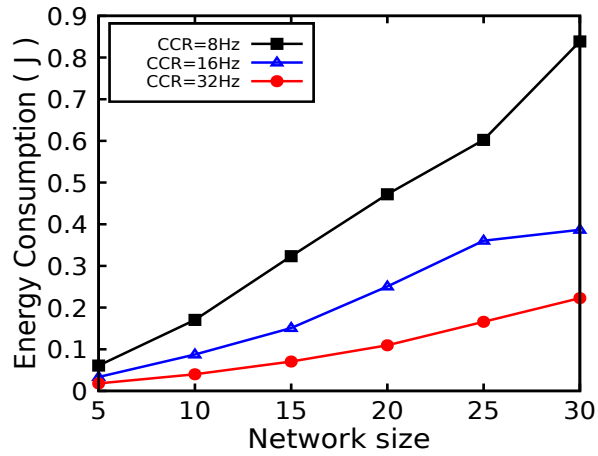
From Fig. 4.4, one can observe that the mDNS performance is inversely proportional to the CCR value. The network convergence time and the energy usage increase



(a) Network convergence time



(b) Energy usage at *Tx* mode



(c) Energy usage at *CPU* mode

Figure 4.4: Impact of different CCRs on mDNS performance as a function of network size

with decreased CCR for a given number of nodes. The results shown in Fig. 4.4(b) and Fig. 4.4(c) are counter intuitive since lower CCR may be thought to result in lower energy consumption and in turn better performance. The depicted trend is due to the ContikiMAC implementation of the broadcast method. The ContikiMAC approach forces the broadcasting nodes to transmit their packets for the whole wake-up interval resulting in a large number of transmitted packets. In addition, during these prolonged broadcast periods, other neighboring nodes are not permitted to send and have to postpone their transmissions until the channel is clear. These adverse effects are larger for smaller values of CCR. Increasing CCR for high values will have diminishing benefits as the energy used in the listening state also increases due to more frequent wake-up times [126].

As it is clear from Fig. 4.4, the CCR value of 32 Hz gives the best performance compared to 8 Hz and 16 Hz. For the rest of evaluations and comparisons for both mDNS and EmDNS the value of 32 Hz will be used for the CCR parameter.

4.5.3 Implementation Overhead

Figure 4.5 shows the storage requirements for mDNS and EmDNS for three types of motes. The requirement is characterized for both RAM and ROM, and the total available memory is also depicted. It can be observed that memory requirements in terms of ROM and RAM for EmDNS are slightly larger than those required for mDNS with an average of 4.2% and 3.5% increase across the considered motes, for RAM and ROM requirements, respectively.

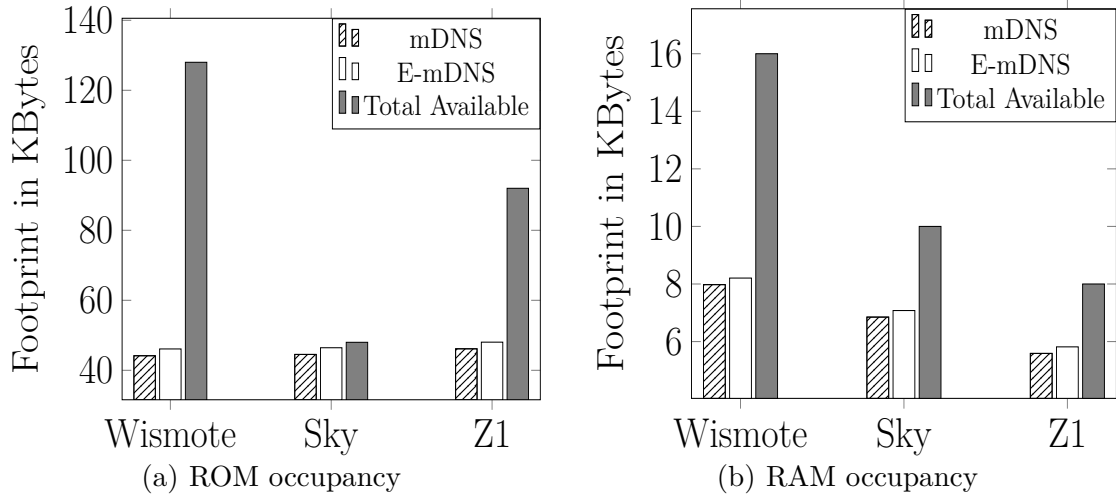
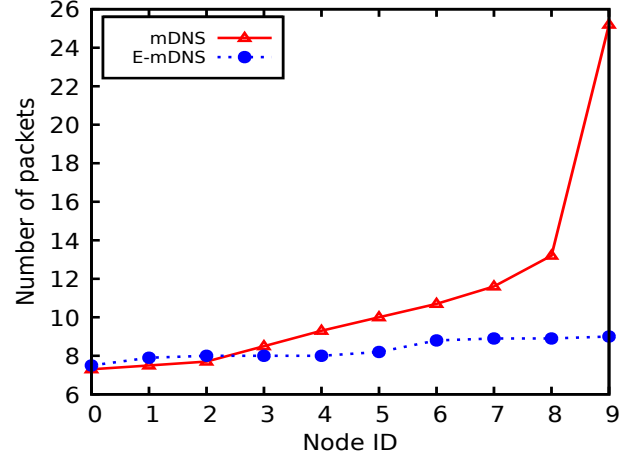


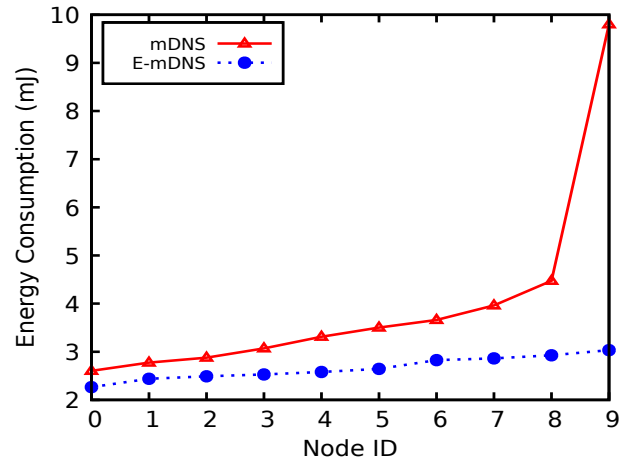
Figure 4.5: Memory footprint overhead

4.5.4 Load Balancing and Fairness

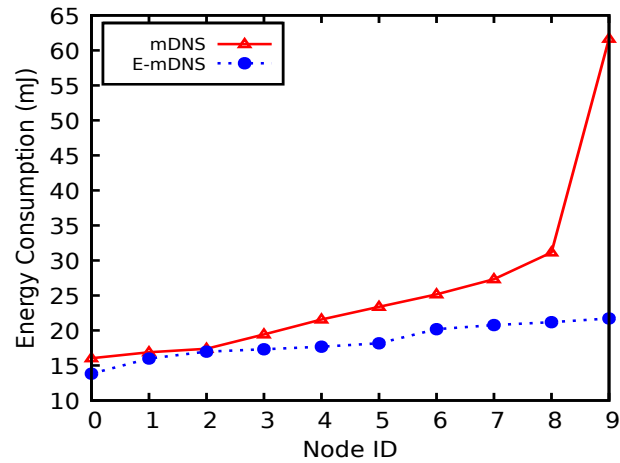
Distributing the load evenly among the participating nodes in LLNs is essential to increase the overall network lifetime. Uneven-load distribution among LLNs nodes causes faster depletion of batteries for the overloaded ones. This subsection aims to evaluate the load balancing aspect for both mDNS and EmDNS. Figure 4.6 shows the per-node evaluation of both mDNS and EmDNS in terms of three factors: the number of transmitted control packets, the consumed energy for the CPU_{on} state, and the consumed energy for the Tx state. The evaluation is performed for $\eta = 10$ nodes, and the results are plotted versus the node identity. It can be seen that all factors are higher for nodes with higher identities. This is because a higher identity node completes its auto-configuration before lower identity nodes and has, as per mDNS and EmDNS, to respond to all received probes while it is in the PS or to all RAs while it is in the AS as well. Therefore, this higher identity node ends up sending more control packets and consuming more energy for its CPU and radio



(a) Transmitted control packets



(b) Consumed energy at CPU_{on} mode



(c) Consumed energy at Tx mode

Figure 4.6: Per-node performance evaluation comparing mDNS and EmDNS for $\eta = 10$

transceiver, relative to a lower identity node. It can also be noted that differences between performance figures for mDNS and EmDNS for a given node identity are greater for greater identities. This means that EmDNS can potentially produce a greater reduction in control packets and energy consumption for larger network sizes. Lastly, it can be observed that results for EmDNS do not vary significantly across nodes compared to that for mDNS. This means that nodes are roughly equally loaded relative to the nodes for the case of mDNS. This balance is a direct consequence of the fact that EmDNS with its SSE and ASM, eliminates numerous transmissions that would have otherwise congested the network.

To examine the fairness issue further, we evaluate the same three factors used in Fig. 4.6 but for different values of η . We then utilize Jain FI as defined in [127] using Eq. 4.4:

$$FI = \left(\sum_{i=0}^{\eta-1} x_i \right)^2 / \left(\eta \cdot \sum_{i=0}^{\eta-1} x_i^2 \right) \quad (4.4)$$

where FI ranges between 0 & 1, x_i is the average performance figure for node i , and η is the network size under consideration. A value of 1 for FI corresponds to 100% fairness, i.e. all x_i 's are equal, while a value of 0 corresponds to no fairness at all, i.e. great variation in x_i 's. Figure 4.7 shows the FI as a function of the number of nodes for mDNS and EmDNS protocols. It can be seen again that EmDNS clearly provides a greater balancing, relative to mDNS, between nodes in terms of number of control packet transmission and energy consumption, for network sizes ranging from 5 to 30 nodes.

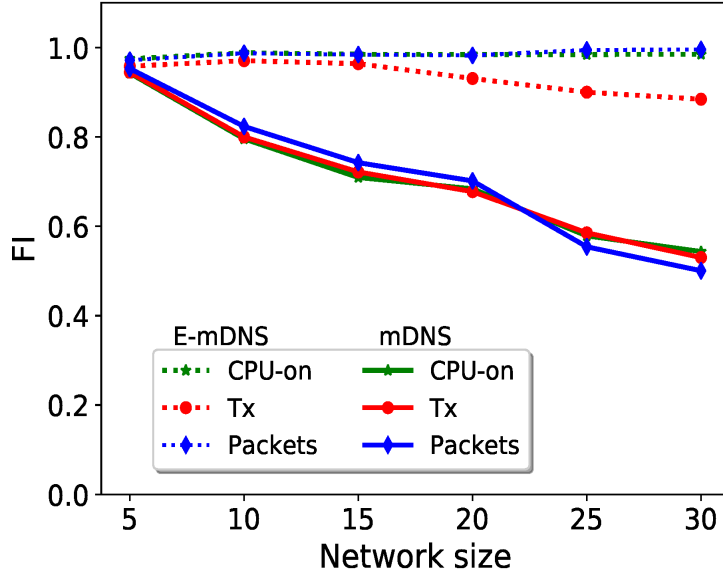


Figure 4.7: Fairness index using different performance metrics as a function of network size η

4.5.5 Results on Protocol Performance

The key performance metrics for auto-configuration protocols is giving unique configurations with the least number of control packets, or equivalently, with the least consumption of energy. This subsection discusses the obtained results for these metrics using both mDNS and EmDNS.

4.5.5.1 Identity Uniqueness Percentage (IUP)

The ultimate goal of any auto-configuration protocol is to allocate a unique identification for each node joining the network. If the protocol fails in this task, then multiple nodes will be identified by the same identification, and this will lead to network resolution conflicts. We define IUP as the ratio of unique identifications, i.e. unique *hosts_names*, in the network to the total number of nodes η . Therefore, a

perfect auto-configuration protocol will always produce a unique identification for every node, and thus IUP will be equal to 100%. Figure 4.8 compares the obtained IUP for mDNS and EmDNS protocols as a function of the network size. As can be

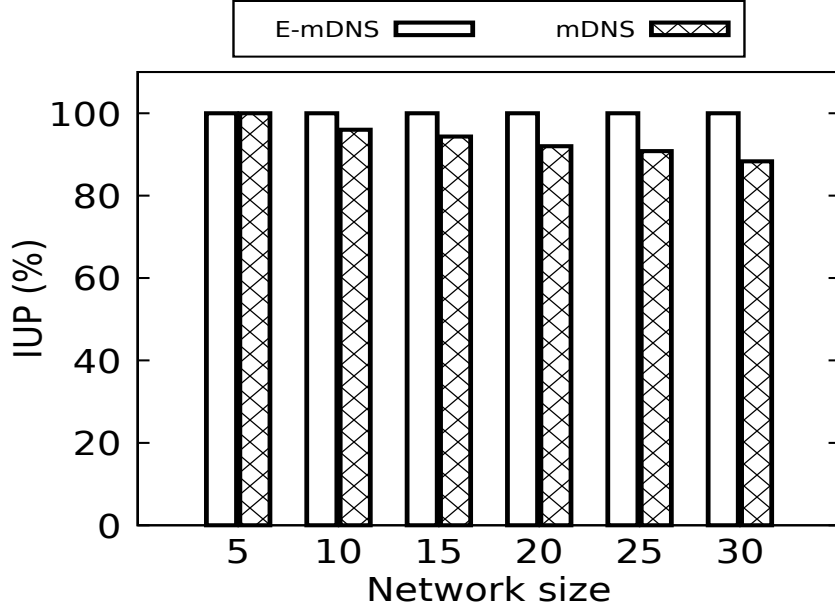


Figure 4.8: IUP as a function of network size η

observed from Fig. 4.8, the IUP achieved by mDNS degrades gradually as the number of simultaneous start-ups increases. This decrease is a direct consequence of the high traffic experienced by the nodes participating in the simultaneous start-up event. As the offered load to the network increases, the packet loss also increases due to link congestion and buffer overflow. The high number of the lost probe and/or announcement packets for mDNS, causes nodes not to respond to conflicting configurations, and thus committing to identifications that are owned by other nodes. On the other hand, IUP is equal to 100% for EmDNS for all considered network sizes as shown in Fig. 4.8. EmDNS reduces the operations that contribute to generating considerable

traffic in the network. Hence, link congestion and buffer overflows are less likely, relative to the case of mDNS, leading to a higher percentage of successfully completed configurations.

4.5.5.2 Effectiveness of the ASM

One way to check the correctness and functionality of the proposed ASM is to measure the Local Hit Ratio (LHR) for address resolution. The LHR indicator is the percentage of address resolution requests that are fulfilled by consulting the local cache, relative to the total number of requests [18] and it is calculated using Eq. 4.5.

$$LHR\% = \frac{\text{Number of requests fulfilled locally}}{\text{Total number of request}} \times 100 \quad (4.5)$$

High LHR percentages imply minimum or zero adverse effects of the implemented ASM enhancement. Nodes are still capable of storing identifications in their respective cache, despite the elimination of the announcement messages, and the used TTL attribute for cached entries manages to delete stale and uncompleted identifications entries from the local cache.

Figure 4.9 depicts the LHR as a function of the cache size parameter *MAX_ENTRIES* for the network of size equal to 20 nodes. The LHR is averaged over all the participating nodes in the network. As expected, the larger the *MAX_ENTRIES*, the higher the LHR percentage, however, both protocols mDNS and EmDNS do not reach 100% even for large cache sizes. The previous set of results for the *IUP* figure showed that mDNS does not achieve a perfect IUP as some nodes

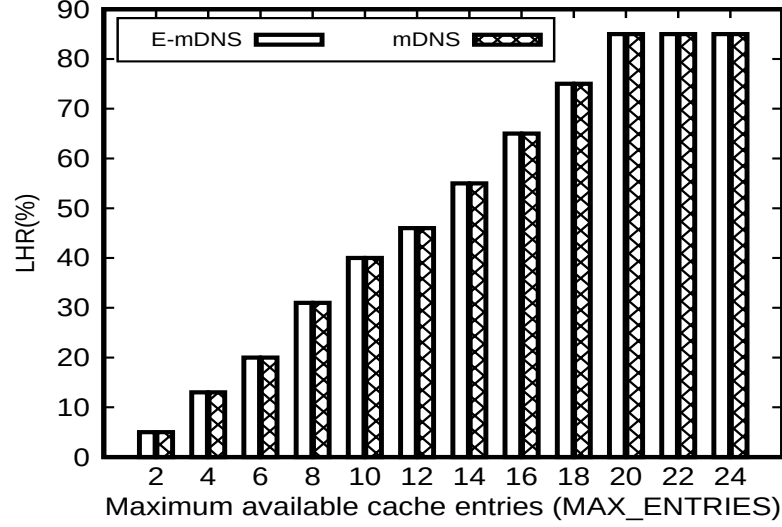


Figure 4.9: LHR as a function of $MAX_ENTRIES$ when $\eta = 20$

end up with repeated identifications even after the network converges. These repeated identifications prevent the cache from being filled with the proper identifications even if $MAX_ENTRIES$ is large enough to keep all these configurations. Furthermore, the EmDNS, also, does not achieve an LHR equal to 100%, even for large values of $MAX_ENTRIES$. This shortcoming is due to missing some probe messages in some rare cases causing the premature removal, i.e. before receiving MAX_PROB_MSGs identical probes, of cache entries with the *TENTATIVE* attribute. As a summary, it can be concluded that the ASM eliminates the need for the AS resulting in less communication overhead while approximately maintaining the same LHR.

4.5.5.3 Network Convergence Time (NCT)

NCT is the time required by all nodes, participating in the simultaneous start-up event, to complete their configurations. It is literally, the time till the last node completes its auto-configuration procedure. Ideally, the NCT should be minimized at

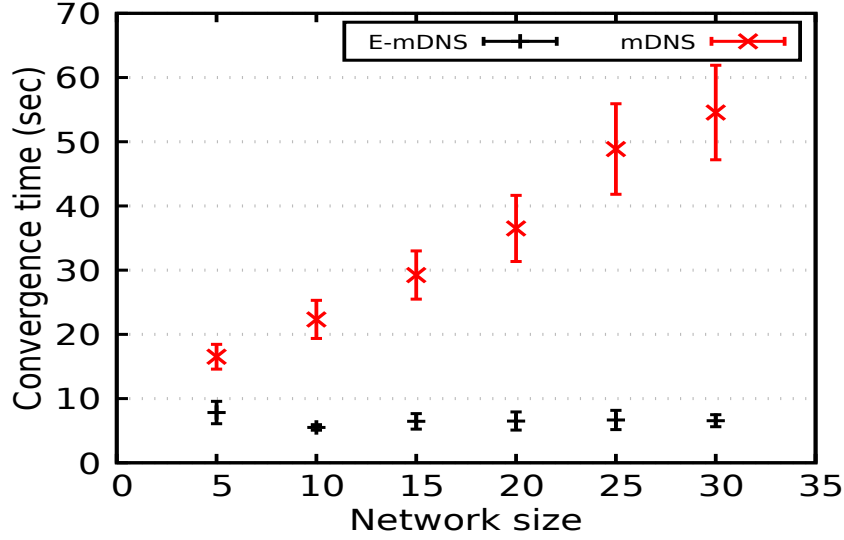


Figure 4.10: NCT as a function of network size η

no extra cost in energy consumption. Lower NCT means less delay for getting the unique configuration in the network. Figure 4.10 depicts the network convergence time using EmDNS compared to the case when mDNS is used for different sizes of the network. The figure shows that the NCT scales almost linearly with the size of the network for the mDNS protocol, while it increases very slowly for the EmDNS protocol. This reduction and also the tight variation in the confidence interval are attributed to eliminating the 1000 ms period in case of EmDNS and to changing the conflicting configuration right after the conflict is detected. EmDNS achieves a significant 88% reduction in NCT for $\eta = 30$ nodes, while the average reduction is equal to 77% for the considered range number of nodes.

4.5.6 Control Traffic Overhead

This subsection evaluates the number of control packets as a function of network size for the proposed EmDNS protocol. It attempts to assess the scalability of the protocol as the network size increases. To this end, the corresponding number of control packets is one of the critical performance metrics for low-power IoT networks and it is imperative to bound this overhead as much as possible. Figure 4.11 shows a comparison between mDNS and EmDNS for the average number of control packets as a function of η . Results show that EmDNS easily outperforms mDNS and has significantly lower number of control packets, especially for large network sizes. EmDNS

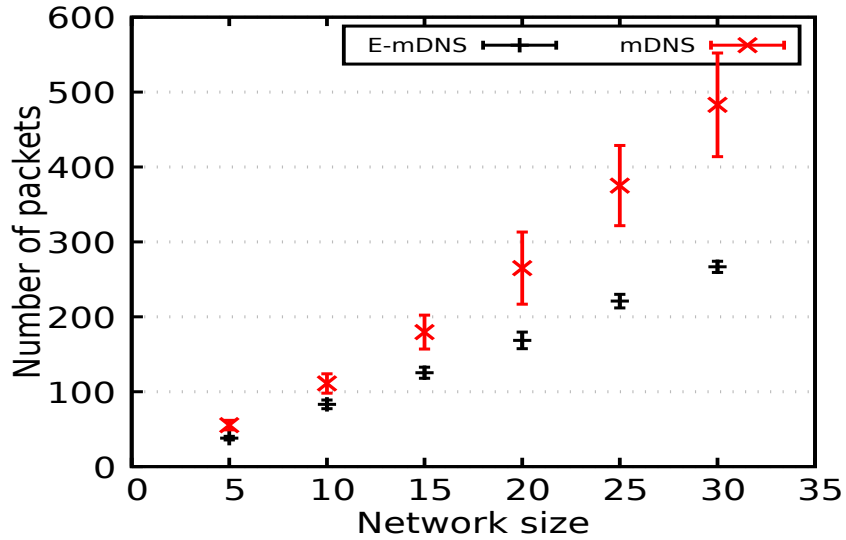


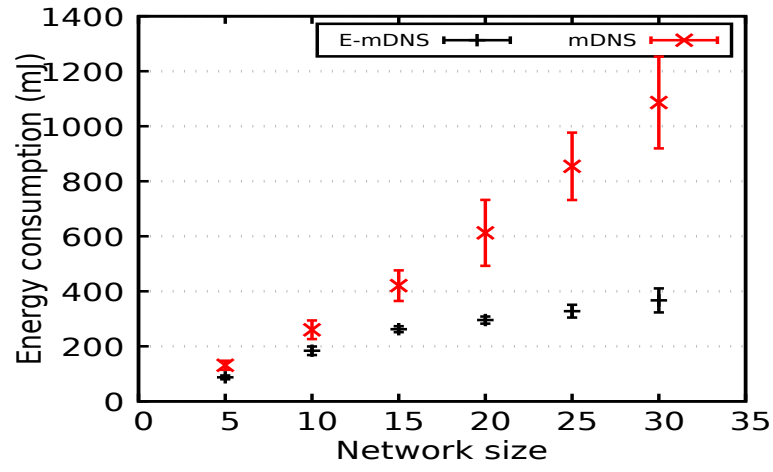
Figure 4.11: Overall number of control packets as a function of network size η

reduces the average number of packets for the configuration phase of mDNS protocol by an average of 34.8% over the considered number of nodes since it requires fewer packets for the configuration phase, relative to mDNS. This is mainly due to the PSS enhancement that allows nodes to reuse previously reached and committed configu-

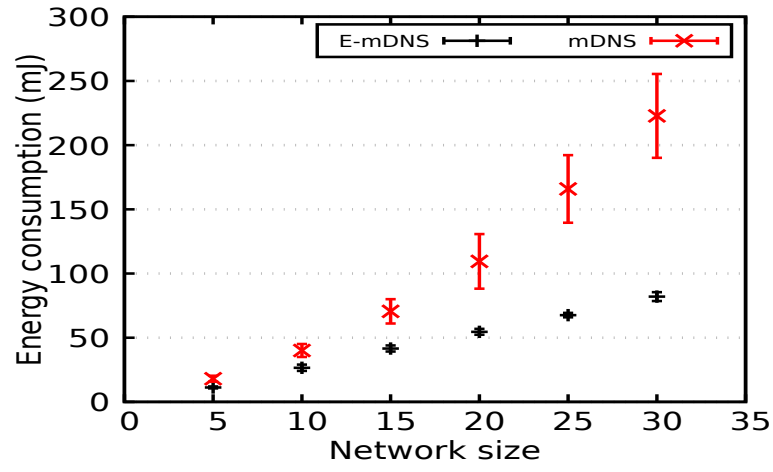
rations, whereas mDNS nodes have to always use the default configuration which is typically shared by all nodes. A fraction of the observed reduction in control packets is also attributed to the ASM. It can be seen that the reduction is greater for greater network sizes, with a maximum reduction of 45% for $\eta = 30$ nodes.

4.5.7 Energy Consumption

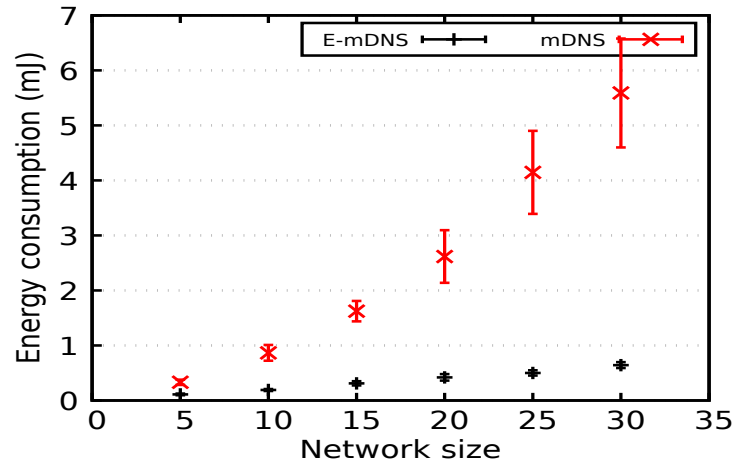
In this subsection, we characterize the energy consumption for the CPU and the transceiver components. The comparisons shown in Fig. 4.12 consider the energy needed by the Tx mode of the radio transceiver, and the micro-controller for both modes of CPU_{on} and CPU_{lpm} . The energy usage at the listening ' Rx ' mode is excluded from our comparison as it highly depends on the used RDC mechanism. Again, the consumed energy in milli Joules is plotted as a function of the network size for mDNS and EmDNS. It is noted from Fig. 4.12 that nodes using the proposed EmDNS consume significantly lower amounts of energy for both the radio transceiver and the CPU components. In specific, averaging over the range of number of nodes, EmDNS decreases the energy depletion at Tx mode by an average of 46.6% relative to mDNS as shown in Fig. 4.12(a). This reduction in energy consumption is attributed to the following reasons. When the mDNS is used, the nodes have to run through the operations between $[t_1, t_4]$, corresponding to node N_i shown in Fig. 4.3(a), for multiple rounds resulting in a wasteful increase in energy consumption. The number of iterations executed by each node depends on the time when the node decides to change its conflicting probed configuration. Changing the configuration at the CC stage comes after the node has tried to probe with the same configuration already used in



(a) Tx mode



(b) CPU_{On} mode



(c) CPU_{lpm} mode

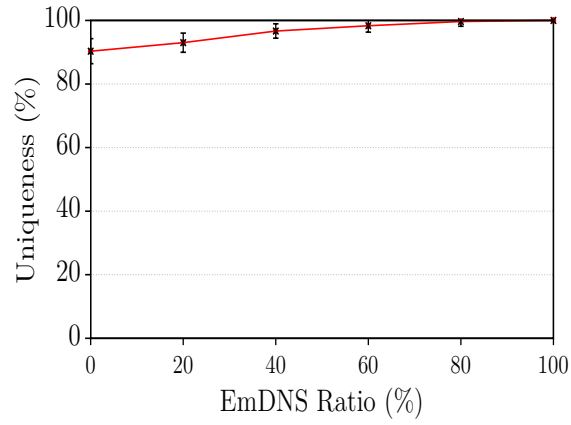
Figure 4.12: Total network energy usage by different components as a function of network size η

the network. Additionally, the PSS and the ASM mechanisms prevent sending out unnecessary packets which further leads to reducing energy usage. Each operation executed by the CPU consumes an amount of energy, and due to the frequent operations executed, the CPU will stay on most of the time. As depicted in Fig.4.12(b) and Fig. 4.12(c), EmDNS decreases the energy consumption by an average of 47.5% and 81.0% for CPU_{on} and CPU_{lpm} modes, respectively. The curves show that the maximum reduction occurs for the maximum network size of 30 nodes. The maximum reduction for energy consumption is equal to 66%, 63%, and 89%, for Tx , CPU_{on} , and CPU_{lpm} modes, respectively.

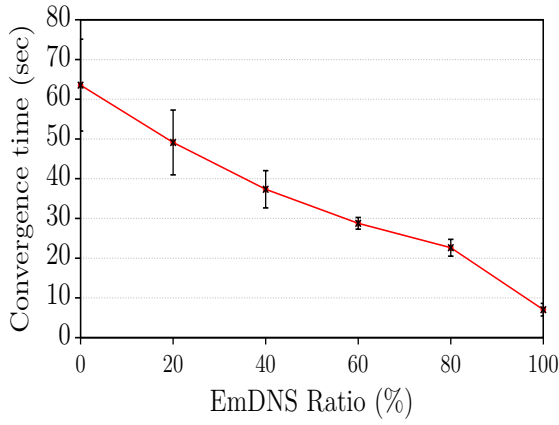
4.5.8 EmDNS Backward Compatibility Support

Considering the consistency and compatibility of the proposed EmDNS extension is utmost of importance. This is to guarantee seamless connections between EmDNS and mDNS nodes. For the sake of ensuring backward compatibility, EmDNS exploits the same messaging structures and operating semantics defined in mDNS without any modification.

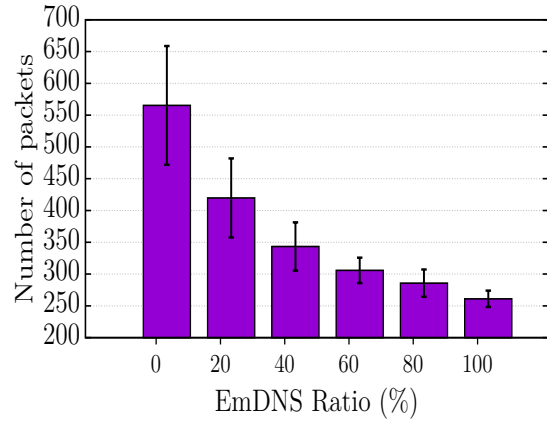
To assess the EmDNS compatibility issue, this subsection shows the performance results conducted specially for this purpose. In this evaluation part, we first set-up the network by making all the nodes running mDNS protocol. Then 20 nodes are randomly selected and replaced with nodes that support the proposed EmDNS. The corresponding results are gathered, and then the same procedure is repeated by selecting 20 more nodes to support EmDNS until all nodes run EmDNS extension. From the results obtained, the following conclusions can be inferred. First, nodes



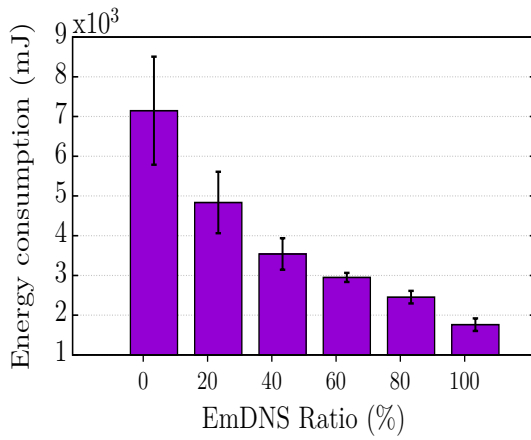
(a) IUP



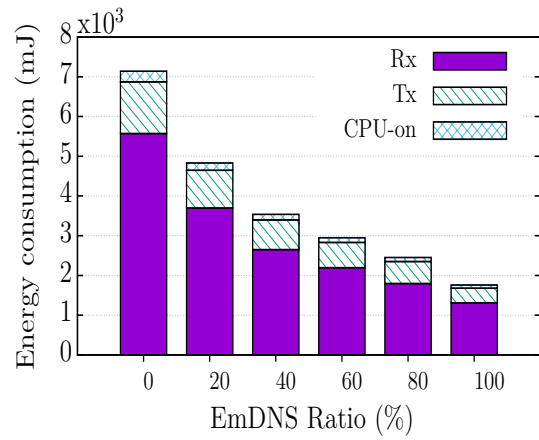
(b) NCT



(c) Control Packets



(d) Total Energy



(e) The Stacked Energy

Figure 4.13: The hybrid network performance with different EmDNS node deployments.

operating EmDNS or mDNS work together seamlessly in the same hybrid network, and therefore EmDNS supporting nodes can coexist with conventional mDNS nodes. Therefore, the interoperability for control packets between EmDNS and mDNS nodes is verified. Second, as more EmDNS nodes are added to the network, better system performance is observed as depicted from the results reported in Fig. 4.13.

Figure 4.13(a) shows that the uniqueness percentage (i.e. IUP) increases as the HRPL nodes to NSRPL nodes ratio increases from 0% (i.e. all nodes run mDNS) to 100% (i.e. all nodes run EmDNS). Furthermore, It can be seen from Figs. 4.13(b), 4.13(c), that the NCT and control packets transmitted, decrease gradually as the EmDNS nodes to mDNS nodes ratio increases. Accordingly, consumed energy reduction is achieved proportionally to the EmDNS nodes ratio increases as depicted in Fig. 4.13(d). Figure 4.13(e) breaks down the total energy shown in Fig. 4.13(d) into its constituent component. This figure shows that the energy, consumed by the radio transceiver in its both transmitting and listening modes and also by the micro-controller, is decreased gradually as the EmDNS ratio increases.

4.6 Chapter Summary

The successful deployment of LLN-based applications, especially in highly dynamic scenarios, requires LLN nodes with the capability to adapt and configure themselves automatically to the surrounding environment without any human intervention. Because of the expected high number and the low capabilities of LLN nodes, the manual or centralized naming systems are not practical. Adapting the traditional protocols

to work with these constrained nodes with no traffic reduction causes inefficient utilization of their limited resources. In this chapter, we present EmDNS as an extension for the mDNS protocol, which is considered to be the default distributed protocol for configuring IoT nodes. Specifically, the proposed EmDNS includes three traffic reduction mechanisms, namely; the PSS, the SSE, and the ASM. Both protocols are evaluated using the Cooja emulator under a wide range of parameters. This study investigates initially the impact of the RDC mechanism on the performance of the original mDNS and decides on the RDC parameter to be used as a base for the rest of the performance evaluations. Moreover, the chapter evaluates the memory footprint requirement for both mDNS and EmDNS and determines that EmDNS requires only 4.2% more RAM storage than mDNS. Additionally, we evaluate the load balancing aspect for both protocols and demonstrate that with EmDNS there is no significant disparity between nodes, relative to the case of mDNS, in terms of number of transmitted control packets or the amount of consumed energy. The evaluation then focuses on the performance of EmDNS relative to mDNS in terms of IUP, the LHR, the NCT, the number of control packet transmissions, and the energy consumption for both the CPU and the radio transceiver components. The evaluations of IUP and the LHR aim to verify the correctness and efficiency of the EmDNS protocol functions in terms of granting unique identifications for simultaneously starting nodes and the correct address resolution of hosts when the execution of the protocol completes. Results show that EmDNS achieves higher IUP, especially for a larger number of nodes, while both protocols provide roughly the same LHR figure. In terms of NCT, the EmDNS protocol reduces the network convergence time by a factor ranging between 2 and

8, relative to that for mDNS, for a network of size ranging between 5 to 30 nodes. Furthermore, EmDNS is found to generate an average of 34.8 % fewer control packets for the range considered for the number of nodes. Correspondingly, the energy consumption is also reduced by an average of 46.7% for the radio Tx mode, while it was reduced by averages of 47.5% and 81% for the CPU_{on} and the CPU_{lpm} modes, respectively. The percent reduction whether in terms of the number of control packets or energy consumption always increases with the size of the network. For our simulation parameters, the maximum reduction always occurred for a network with 30 nodes.

CHAPTER 5

HYBRID RPL

5.1 Chapter Motivations

Cyber-Physical Systems (CPSs) are engineered solutions aimed at monitoring and controlling the physical environment through seamless integration of computational algorithms and physical components [42]. CPSs come in many application forms such as power networks [3], traffic control, safety systems [5], building automation [4], industrial monitoring [6], autonomous vehicles, and distributed robotics [128]. CPSs further employ standards-based architectures to be connected with the Internet to realize the IoT vision [42]. Unlike pure sensing that focuses on traditional data gathering, several CPS-based applications require distributed decision making, control, and actuation [129]. This requires nodes to communicate with each other inside a network in what is called N2N communications. However, based on RPL's current specifications, it does not cater well to fulfill N2N-related routing requirements and that limits its adoption in several LLN-based applications [36]–[39]. The following points summarize some of the relevant limitations that motivate the work of this

chapter:

- *Reliability*: restricting N2N-communications to use only the pre-established DODAG routes causes severe traffic congestion near the Border Router (BR) or the root. As a consequence, packet loss rate would increase which negatively impacts protocol reliability [34].
- *Latency*: the up and down routing along the DODAG topology may end up with a path significantly worse than the shortest path available between the source and destination pair. Moreover, packet drops caused by the congestion trigger more retransmissions which increase end-to-end delivery delays.
- *Links asymmetry*: the LLN wireless links are asymmetric in nature [130]. This is due to the effect of some factors such as the transmitter power and receiver sensitivity variation, reflectors, and absorbers [131]. Not accounting for this asymmetry produces less than optimal routing paths.
- *Load distribution*: nodes placed near to the BR may become congested at high traffic loads. The congested nodes would drain their energy faster than other nodes leading to the energy hole problem [132]. Evenly distributing the load over the participating nodes is one of the primary goals of any routing algorithm mainly to avoid disconnected regions.

To alleviate the aforementioned limitations, we propose a hybrid RPL-based extension referred to by HRPL. With HRPL, we retain the benefits of both the optimized RPL-based forwarding structure and the potential capability of the BR for adopting

link-state routing. To do so, the work in this chapter proposes that the BR node runs a centralized algorithm to maintain up-to-date routing information and construct the best-quality path, if it exists, between any source-destination pair within the LLN domain.

The idea of delegating the control of the network to the BR has been studied by many literature studies. For example, the routing mechanisms presented in pTunes [133], Hydro [33] and WirelessHART [134] build a partial network model populated at the root. Hydro maintains a centralized routing mechanism for LLNs in which the BR collects information about the participating nodes to build a global view of the network. The collected knowledge is carried using dedicated predefined control plane packets or piggybacked in the transmitted data packets. The network operated by the WirelessHART is managed by a centralized manager which is in charge of scheduling, creating routes, and optimizing the network. Recently, Corbalan *et al.* in [135] presented a mechanism referred to by SMOG to build a fully centralized LLNs topology model using a probabilistic data structure (i.e. Bloom filter). SMOG aims to evaluate the model accuracy and analyze the possible inconsistencies occurring between the model and the underlying physical topology. Finally, Tripathi and Oliveira in [136] propose a centralized algorithm managed by the BR to control DAO emissions in the NSRPL mode.

5.2 Chapter Contributions

The major contributions included in this chapter are as follows:

- We develop HRPL which integrates the concept of link-state based routing with the existing RPL forwarding structure to discover more efficient N2N routes. This is achieved using two mechanisms: 1) An adaptive reporting mechanism that is implemented at the non-BR nodes to provide the BR with their neighboring information. This mechanism also prevents nodes from sending their neighboring information for too little changes to avoid the excessive traffic in the network. Each node maintains both a current and advertised updated information for all their neighbors and does not send a new update until the positive or negative change reaches a predefined threshold, and 2) A single source shortest path mechanism that is implemented at the BR to construct, upon request, the shortest path between any two nodes in the network. The shortest path is determined by considering the accumulated ETX needed to successfully deliver the packet along the path. Subsequently, the constructed path is used to route the data packets towards their targets by making use of the source routing approach [56].
- HRPL does not define new control messages, but rather utilizes existing ones. This guarantees interoperability when both HRPL and non-HRPL nodes operate seamlessly together in a hybrid network with no backward compatibility issues. Such backward compatibility is crucial in practice because some routing layers in non-programmable chips such as Zigbee cannot be updated [40].
- HRPL is targeted for IoT nodes enabled by ContikiOS version 3.0. Being based on an RPL-oriented approach, HRPL is platform-independent and can be easily

adapted to work with other IoT-enabled OSs such as TinyOS [65], RIOT [137].

HRPL source code is made publicly available [138] to allow further improvements and evaluations by the research community.

- An experimental performance evaluation is carried out using the Cooja emulator to evaluate HRPL feasibility and efficiency. The evaluation is achieved through; 1) Evaluating the HRPL backward compatibility when HRPL and non-HRPL nodes work together in a hybrid network, and 2) Evaluating the HRPL performance against original RPL modes and an RPL-based opportunistic routing approach [101] under various network conditions.

5.3 Network Model

This section discusses the network model used in this chapter. This model considers a static multi-hop LLN. The studied network comprises of a set of LLN nodes $\mathcal{N}$ rooted at the BR and addressed by $1, 2, 3, \dots, \eta$ where $\eta = |\mathcal{N}|$. The network could be connected, through the BR, to the public Internet or to a private network to form a wide area network [139]. The network is modeled as a time-varying non-negative weighted graph $\mathcal{G}(\mathcal{N}, \mathcal{L}, \mathcal{Q}(t))$ where $\mathcal{L}$ represents the set of all possible wireless links connecting all node pairs in $\mathcal{N}$, and $\mathcal{Q}(t)$ holds the ETX of all links in $\mathcal{L}$. For every node $n \in \mathcal{N}$, we define a neighborhood set $\Omega_n^t \subset \mathcal{N}$, which contains all nodes in the one-hop distance from the node n . Let $\mathcal{E}_{n,m}^t$ denotes to the ETX of a particular link starting at node n and ending at node m at time t where $\mathcal{E}_{n,m}^t \in \mathcal{Q}(t) \forall m \in \Omega_n^t$. Due to the dynamic nature of wireless links, the value of $\mathcal{E}_{n,m}$ is measured and updated

continuously utilizing the link-layer feedback. That is, $\mathcal{E}_{n,m}$ is updated whenever a traffic flow takes place between n and m reflecting some topological change. Due to the links asymmetric characteristic, $\mathcal{E}_{n,m}^t$ is not necessarily equal to $\mathcal{E}_{m,n}^t$. To make $\mathcal{E}_{n,m}^t$ robust in the case of excessive or rapid link condition changes, it is smoothened using an Exponentially Weighted Moving Average (EWMA) filter as shown in (5.1).

$$\mathcal{E}_{n,m}^t = \delta \times \mathcal{E}_{n,m}^{t-1} + (1 - \delta) \times \mathcal{E}_{n,m}^t \quad (5.1)$$

where the parameter $0 \leq \delta \leq 1$ defines the EWMA smoothness level. The default value for δ is 0.3. Table 5.1 summarizes the notations that are used throughout this chapter.

Table 5.1: The summary of the notations used in this chapter

$\mathcal{N}$	The set of all LLNs nodes
$\mathcal{L}$	The set of all LLNs wireless links
(n, m)	The wireless link starting at node n and ending at m
$\mathcal{Q}(t)$	The set of all LLNs wireless link metric qualities at time t
$\mathcal{E}_{n,m}^t$	The quality indicator for the link from node n to node m at time t
Ω_n^t	The set of all single-hop neighbors of node n at time t
δ	The smoothness level of the exponentially weighted moving average filter
β	Reporting hysteresis threshold parameter
$\Delta\mathcal{E}_{n,m}$	The difference between $\mathcal{E}_{n,m}^t$ and $\mathcal{E}_{n,m}^{t-1}$
$\Delta t(\mathcal{E}_{n,m})$	The elapsed time from last reporting
DIO_n^t	DIO sent by node n at time t
DAO_n^t	DAO sent by node n at time t
$\mathcal{T}_{max}$	Maximum interval to report new update of $\mathcal{E}_{n,m}$

5.4 Overview of Hybrid RPL (HRPL)

This section describes the proposed algorithmic elements and highlights the method of utilizing the link-state routing approach.

5.4.1 HRPL Core Phases

Typically, HRPL contains two core phases; an initialization phase and a route discovery phase. These are explained below.

5.4.1.1 The initialization phase

HRPL inherits the operations of RPL to pre-establish the upward and downward routing during the DODAG construction process. Each HRPL node reports its neighbors information and their link qualities to the BR as explained in Subsection. 5.4.2. HRPL nodes utilize DAOs messages of RPL to piggyback their neighboring information as detailed in Subsection. 5.4.3. The BR then builds a complete view of nodes' interconnections so that it can construct, upon request, the shortest path between any two nodes in the network.

5.4.1.2 The route discovery phase

In the discovery phase, the source node that wants to send an N2N data packet should fetch a path, from its local cache, to the intended destination. If the required path is not cached, the source node should initiate a route discovery process by sending an HRPL Route Request (HRREQ) message to the BR asking about a path to the

destination of interest. The HRREQ is a UDP packet that is depicted in Fig. 5.1.

Type	Seq	Dest
------	-----	------

Figure 5.1: HRREQ payload format

The HRREQ is then forwarded to the BR via RPL-based upward routing. Upon receiving the HRREQ, the BR constructs a shortest path, if any, to the destination stated in the HRREQ message. To construct such a path, the BR applies the mechanism detailed in Subsection. 5.4.4. The constructed path is then sent back to the source node via RPL-based downward routing by means of the HRPL Route Reply (HRREP) message. Figure. 5.2 depicts the UDP payload format of the HRREP where “*CmprN*” indicates the compression level applied to the addresses in the vector “*Path[1...Hops]*”. In the case of the loss of an HRREQ or HRREP, the routing falls back to the default NSRPL mode

Type	Seq	Dest	Hops	ETX	CmprN
Path[1...Hops]					

Figure 5.2: HRREP payload format

Referring to the scenario shown in Fig. 2.4, the constructed path in HRPL (i.e. between nodes 9 and 11) would be expected to pass through ⑨ → ⑩ → ⑪. Upon receiving the HRREP, the source node locally caches the path and then utilizes the source routing approach to send the data packet to the intended destination over the provided path.

5.4.2 Reporting of Neighboring Information

Recalling the RPL behavior outlined in Subsection 2.2.3.1, each node $n \in \mathcal{N}$ periodically unicasts its reachability information (i.e. prefix) upwards to the BR using the target and transit options of DAO messages. HRPL utilizes this fact by making nodes piggyback their neighboring information in the transmitted DAO as well. In other words, at the reporting time, each node $n \in \mathcal{N}$ appends its neighboring information (i.e. in the form of $(m, \mathcal{E}_{n,m}^t)$, $\forall m \in \Omega_n^t$). Since the appended information grows in size as the neighborhood list increases, this results in high traffic particularly at the nodes located along the path towards the BR. Accordingly, this leads to high network congestion and high energy depletion. However, it is observed experimentally that sometimes, at the reporting time, there is an insignificant difference denoted by $\Delta \mathcal{E}_{n,m}$ between the current updated metric $\mathcal{E}_{n,m}^t$ and the one previously advertised $\mathcal{E}_{n,m}^{t-1}$. This newly reported update should be considered redundant as it would not contribute towards the shortest path construction from the $\mathcal{BR}$ point of view. In addition, insignificant updates make the already constrained network situation even worse. To deal with this problem, we propose an adaptive reporting mechanism named an updated metric with the hysteresis reporting function referred to by Updated Metric with the Hysteresis Reporting Function (UMHRF) and described in Algorithm 2. The proposed mechanism adaptively determines whether a particular $(m, \mathcal{E}_{n,m}^t)$, should be appended to the transmitted DAO based on $\Delta \mathcal{E}_{n,m}$ at the time of the reporting time. The adaptive updating has the benefits of being energy efficient and making the network more scalable since control traffic is no longer linearly propor-

Algorithm 2 UMHRF

Input: Ω_n^t , $\mathcal{Q}(t)$, β and $\mathcal{T}_{max}$ **Output:** DAO_n^t

```
1: initialize  $DAO_n^t$ 
2: for each  $m \in \Omega_n^t$  do
3:    $\Delta\mathcal{E}_{n,m} \leftarrow \mathcal{E}_{n,m}^t - \mathcal{E}_{n,m}^{t-1}$ 
4:   if ( $|\Delta\mathcal{E}_{n,m}| > \beta \parallel \Delta t(\mathcal{E}_{n,m}) \geq \mathcal{T}_{max}$ ) then
5:      $DAO_n^t \leftarrow DAO_n^t \cup \{(m, \mathcal{E}_{n,m}^t)\}$ 
6:   end if
7: end for
8: return  $DAO_n^t$ 
```

tional to the reporting times and the size of the Ω_n^t . UMHRF is designed to provide the BR with the node's neighboring information while preventing excessive churn in the network. UMHRF does so by employing two mechanics which are: 1) The change in $\mathcal{E}_{n,m}^t$ is detected utilizing the link-layer feedback, and 2) UMHRF appends the $\mathcal{E}_{n,m}^t$ to the DAO_n^t only if one or both of the following conditions are satisfied; i) The absolute value of $\mathcal{E}_{n,m}$, $|\Delta\mathcal{E}_{n,m}|$, is higher than a predefined threshold β , and ii) The period since the last reporting time, $\Delta t(\mathcal{E}_{n,m})$, exceeds the maximum reporting interval $\mathcal{T}_{max}$. The latter condition aims to avoid the indefinite suppression of a particular update due to its insignificant value. The node-related neighboring information, stored in the BR, is assigned an expiration time and then it is removed if no more updates are received. The second part of UMHRF is called "hysteresis".

5.4.3 HRPL Neighboring Option

To ensure compatibility with RPL, HRPL uses the existing control message types of RPL. HRPL introduces a new standard ICMP option added to the original DAO base object. The proposed option is named as HRPL Neighboring Option (HNO) and is

illustrated in Fig. 5.3. The payload of the HNO option has a variable length, specified by *Option Length* field and it corresponds to the size of Ω_n^t at the reporting time slot.



Figure 5.3: HRPL neighboring information option

In Fig. 5.3, the *CntN* field specifies the number of neighbors which have their information appended to the aggregated DAO_n^t message. The *Metric*[1...*CntN*] vector stores the quality of the links between the node n and its neighbors, $\mathcal{E}_{n,m}^t \forall m \in \Omega_n^t$. In the configuration of RPL-based domains, it is common that all nodes within the domain share the same prefix of their addresses. The *CmprN* field indicates the number of prefix octets (i.e. bytes) shared amongst the addresses of participating nodes and the BR. In other words, the *CmprN* defines the compression level applied to the neighbor's address which consequently allows compaction of the *Neighbor*[1...*CntN*] vector. The shared prefix octets are not carried within the HNO and each address in *Neighbor*[1...*CntN*] has a length of $(16 - \text{CmprN})$ octets since the full address has a length of 16 octets. Finally, padding is utilized such that the length of the DAO_n^t message is made a multiple of 4 bytes. Let N_{max} be the maximum number of neigh-

boring information entries that can be contained in one aggregated DAO_n^t . N_{max} can be easily calculated using (5.2).

$$N_{max} = \left\lfloor \frac{MTU - DAO_{over} - HNO_{over}}{(16 - CmprN) + 1} \right\rfloor \quad (5.2)$$

where MTU is the Maximum Transmission Unit available at the link-layer, DAO_{over} is the overhead associated with the DAO packet which is needed for the DAO base object and the RPL-related options, and HNO_{over} is the overhead associated with HNO option. HNO_{over} comes in terms of the fixed octets assigned to *Type*, *Option Length*, *CntN*, and *CmprN* fields.

The remaining part of this subsection explains (5.2) using a numerical example. This illustration considers the communication technology IEEE 802.15.4 used in this study. For IEEE 802.15.4 links, an MTU of 127 octets is supported. However, 46 octets out of these 127 octets are reserved for the higher-layers headers which are divided into 40 octets for IPv6 header, 4 octets for ICMP, and 2 octets for the compression header. Therefore, only 81 octets available for the payload. DAO_{over} consumes 46 octets divided into 4 octets for the DAO base, 20 octets for the target option, and 22 octets for the transit option. If a full IPv6 address of 16 octets is included in the *Neighbor* entry, the aggregation scope would be very challenging and limited. However, it is common to use either one or both the following compression strategies 1) A header compression technique as in [140], and 2) An address field compression by using 2 octets, i.e. $CmprN=14$. Using a $CmprN=14$ and uncompressed header, a maximum of $\left\lfloor \frac{81-46-3}{3} \right\rfloor$ or 10 neighbors can be contained in one aggregated DAO

packet. From an implementation point of view, the option codes in the range “0x01-0x09” are reserved for RPL option types according to the RPL specifications. To avoid interfering with existing RPL option codes, the HNO code in DAO messages is set to “0x0A”.

5.4.4 Shortest Path Construction

Given the BR has a global view of the network connections, it can always provide the shortest path, if such one exists, in terms of the additive $\mathcal{E}_{n,m}^t$ between any source and destination pair. Finding a path, upon request from node s , between s itself and node d can be solved as a Single Source Shortest Path (SSSP) problem. Let $\xi(n, m) = 1$ if the link (n, m) is included in the constructed shortest path, then the HRPL SSSP can be formulated as follows:

$$\begin{aligned}
& \min \sum_{(n,m) \in \mathcal{L}} \mathcal{E}_{n,m} \xi(n, m) \\
& s.t. \quad \sum_{\{m:(n,m)\} \in \mathcal{L}} \xi(n, m) - \sum_{\{m:(m,n)\} \in \mathcal{L}} \xi(m, n) = \lambda_n, \quad n = 1, 2, \dots, \eta \\
& \quad \xi(n, m) \in \{0, 1\}, \quad \forall (n, m) \in \mathcal{L} \\
& \quad \text{where} \quad \lambda_n = \begin{cases} 1 & \text{if } n = s \\ -1 & \text{if } n = d \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

In the literature, several algorithms have been developed for solving the *SSSP* problem.

All SSSP algorithms are based on the iterative approach and essentially rely on the same important concept of distance labels. Since Dijkstra's algorithm [141] is used over directed graphs with non-negative weights, this makes it a perfect solution for the graph under study. The Dijkstra's algorithm used to solve SSSP of HRPL is described in Algorithm 3.

Algorithm 3 Dijkstra's algorithm for SSSP of HRPL

Input: $\mathcal{G}(\mathcal{N}, \mathcal{L}, \mathcal{Q}(t))$, s , d

Output: Vectors Π and Γ to extract the shortest path between s and d

```

1:  $\mathcal{C} \leftarrow \mathcal{N} \setminus \{s\}; \Gamma(s) \leftarrow 0; \Pi(s) \leftarrow s;$ 
    $\Gamma(n) \leftarrow \infty; \Pi(n) \leftarrow nil; \forall n \in \mathcal{N}, n \neq s$ 
2: while  $\mathcal{C} \neq \emptyset$  do
3:    $n \leftarrow \arg \min_{m \in \mathcal{C}} \Gamma(m); \mathcal{C} \leftarrow \mathcal{C} \setminus \{n\}$ 
4:   if  $n == d$  then
5:     break
6:   end if
7:   for each  $m \in \Omega_n^t$  do
8:     if  $\Gamma(m) > \Gamma(n) + \mathcal{E}_{n,m}$  then
9:        $\Gamma(m) \leftarrow \Gamma(n) + \mathcal{E}_{n,m}; \Pi(m) \leftarrow n$ 
10:    end if
11:  end for
12: end while

```

During the execution of this algorithm, the nodes set $\mathcal{N}$ is divided into two groups; temporary and permanent nodes that are stored in a candidate list $\mathcal{C}$. The algorithm initially sets the source node as a permanent node. The algorithm maintains and adjusts a vector $\{\Gamma(1), \Gamma(2), \dots, \Gamma(\eta)\}$ of labels either scalars or ∞ . Once the algorithm is terminated, the labels $\Gamma(n)$ can be intuitively viewed as the length of the paths from s to node n . Additionally, the algorithm maintains a vector $\Pi(m), m = 1, 2, \dots, \eta$ to store the predecessor indices of the nodes which are labeled as permanents. At each iteration, the algorithm removes from $\mathcal{C}$ the node n corresponding to the minimum temporary label and adds it to the list of permanent nodes. Then the outgoing links

from n are scanned to update the temporary labels associated with the nodes adjacent to n . The algorithm iterates until all nodes are set to permanents (i.e. $\mathcal{C} = \emptyset$).

There are different ways to implement this algorithm which depend on the particular data structure used to represent the sets $\mathcal{N}$ and $\mathcal{L}$. These representations are mainly utilized to facilitate nodes addition and removal, as well as to find the node with the minimum label. In this work, an adjacency list is used at the BR to represent the nodes and their neighboring information. This representation is efficient for many applications due to the following reasons i) it is easier to find the links connected to any particular node, and ii) its space complexity is small in the order of $O(|\mathcal{N}| + |\mathcal{L}|)$ since the total length of all the lists is $2 * |\mathcal{L}|$. This is so since each link appears twice, once for each of its end-nodes. The slowest operation that might commonly be used is testing whether a pair of nodes are connected by a link; this has to be done by scanning through one of the lists, but it may be speeded up through sorting the adjacency lists and using binary search. The direct implementation of this algorithm requires $O(|\mathcal{N}|^2)$ steps because of the two nested loops.

5.5 Performance Evaluation

This section reports and analyzes the simulation results to assess the feasibility and efficiency of HRPL. The first and second subsections list the performance metrics and their definitions, and describe the simulation model and the used configuration, respectively. The third subsection shows the memory footprint for the used HRPL implementation, while the forth subsection analyzes the backward compatibility (i.e.

interoperability) aspect when HRPL works within a hybrid network. The last subsection provides a detailed performance comparison of HRPL against both NSRPL, SRPL, and ORPL modes under different traffic scenarios.

5.5.1 Performance Metrics

In this simulation, we mainly concentrate on the following performance metrics [60].

Please note that $n \in \mathcal{N}$ and $t \in \{1, 2, \dots\}$ in all subsequent equations of this subsection.

- Routing Control Traffic (RCT): Refers to the total number of routing-related control messages, in terms of DIOs and DAOs, transmitted by all participating nodes in the network. RCT is calculated by (5.3) .

$$RCT = \sum_{\forall n} \sum_{\forall t} \left(DIO_n^t + DAO_n^t \right) \quad (5.3)$$

- Packet Delivery Ratio (PDR): Is the fraction of data packets successfully received over those sent in end-to-end communication. Let $pkt_n^{snd}(t)$ and $pkt_n^{rcv}(t)$ define the sent or received packets by node n , respectively, at time t . Then PDR can be calculated using (5.4).

$$PDR(\%) = \left(\sum_{\forall n} \sum_{\forall t} pkt_n^{rcv}(t) \right) / \left(\sum_{\forall n} \sum_{\forall t} pkt_n^{snd}(t) \right) * 100 \quad (5.4)$$

- End-to-end delay or simply delay: Is the average time for all transmitted data packets from sources to successfully being received at respective destinations.

This delay accounts for all possible delays during the transmission such as trans-

mission, re-transmission, and queuing time.

- Energy consumption (in Joules): Is the ratio of total consumed energy to the number of data plane packets successfully received. Let E_n be the total energy consumption by node n during the entire simulation period. The total energy consumption E_{total} and the average consumed energy per received packet E_{pkt}^{rcv} are calculated as in (5.5) and (5.6), respectively.

$$E_{total} = \sum_{\forall n} E_n \quad (5.5)$$

$$E_{pkt}^{rcv} = E_{total} \Bigg/ \sum_{\forall n} \sum_{\forall t} pkt_n^{rcv}(t) \quad (5.6)$$

- Load distribution fairness: We evaluate the load distribution in regard to three metrics: transmitted DIOs, transmitted DAOs, and energy consumption over the participating nodes. For those, we utilize Jain FI to calculate the respective distribution fairness as defined in [127] using (5.7). The F_n parameter represents the average performance figure of interest for the node n .

$$FI(F_n) = \left(\sum_{\forall n} F_n \right)^2 \Bigg/ \left(\eta \cdot \sum_{\forall n} F_n^2 \right) \quad (5.7)$$

- End-to-end hop count or simply hops: Refers to the average number of hops the packet traverses between the source to its final destination.
- End-to-end transmission count or simply transmissions: Refers to the average number of MAC-layer transmissions needed to successfully deliver a data packet

along the used path. This metric is tightly coupled with other crucial metrics of interest such as reliability, delay, and energy consumption [142]. A high number of transmissions indicates that packets are routed either along paths with a high number of hops or along congested paths.

5.5.2 Simulation Model and Configurations Set-up

HRPL is implemented on ContikiOS version 3.0 and evaluated through Cooja emulator. We consider the uniform 2D-grid topology shown in Fig. 5.4 as a physical topology for the network under study. The topology used in the experiments consists of 50 nodes that are deployed in an area of size 450 m * 200 m. The BR is marked with ① and placed in the top left corner of the network to maximize the network diameter.

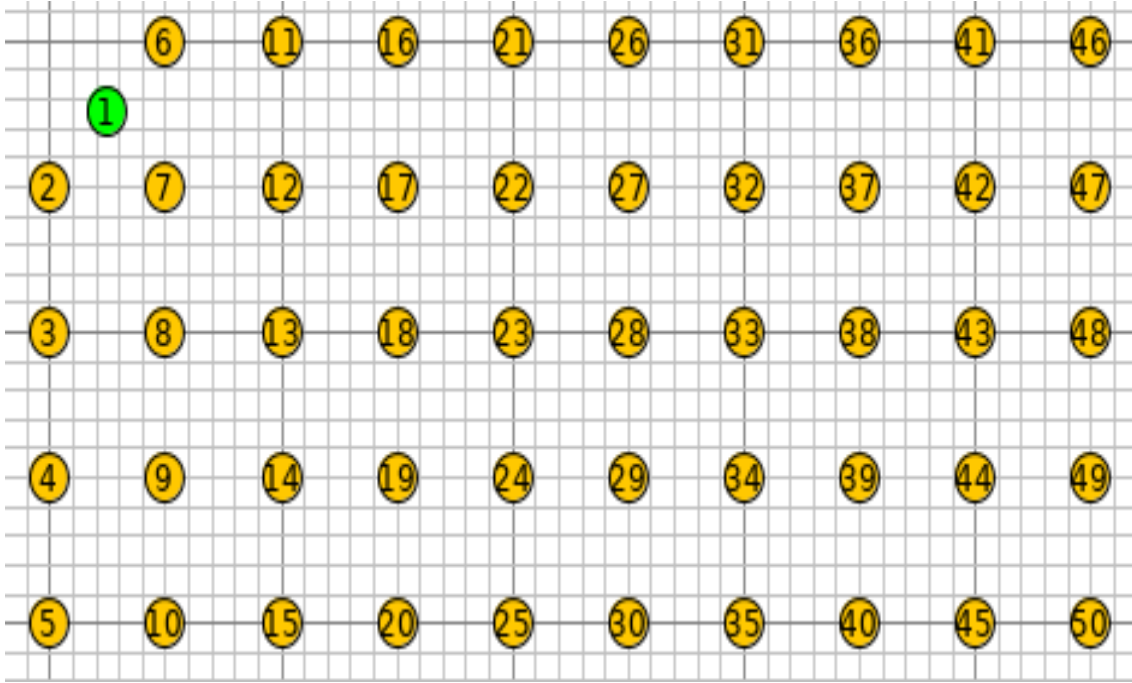


Figure 5.4: The topology of the network under study

To model the lossy environment, we opt a UDGM model commonly used in the literature [143]. The UDGM is used with a transmission range of 60 meters and an interference range of 80 meters. For the sake of simplicity, the transmission range is unified for all nodes including the BR in the studied network. The Wismote platform is used for experimenting with the implementation. The simulation uses the IEEE 802.15.4 protocol at the MAC layer accompanied by unslotted CSMA/CA in the beacon-less mode. For the RDC, ContikiMAC protocol [124] is used. The default CCR value used in all experiments is 8 Hz. The UDP, which does not involve re-transmissions, is used as a default transport layer. The routing protocol utilizes a routing table that consists of entries where each entry has a next-hop node, path cost associated with the destination node, and expiry time. To bring fairness into the subsequent comparisons, we set the routing table for both HRPL and SRPL to 20 entries and the expiry time for the inserted route entry is set to 10 minutes.

Using the aforementioned hardware and software configuration, all the nodes except the BR are divided into two mutually exclusive lists of nodes: the sending nodes list, and the receiving nodes list. The sending nodes generate UDP data packets regularly according to a specified traffic intensity parameter. The traffic intensity parameter is defined by the number of Packets Per Minute (PPM) generated by each node. For each packet sent, a random destination is chosen from the receiving list. The sending list has 24 nodes that randomly chosen nodes not belonging receiving list. The receiving list is set to 20 nodes and these nodes are placed in the last two rows of the network shown in Fig. 5.4. The aforementioned topology for the sending and receiving nodes is mainly to avoid having upward or downward communications

in the simulation as we focus only on N2N communications. The application layer of the sending nodes passes the generated packets to the IPv6 layer. The packet size is 127 bytes in which the payload consumes 15 bytes and the rest are used for the networking and framing overhead. Each experiment lasts 5.5 hours which corresponds to more than 32000 IPv6 packets, for the traffic intensity of 4PPM, exchanged for each experiment. Results are averaged over 10 experiment runs with different seed used for each run. The seed value affects, among other things, the order of packet transmissions, the timers of the TA and MAC back-offs mechanism, and the nodes start-up times. A Javascript code is used to extract the results and log them to text files for subsequent analysis. The generated log files of all runs are fed into a bash shell script developed to compute the required statistics. To automate the graphs generation from the obtained results, in-house developed GNUPLOT-based scripts [144] are used. Finally, some of the presented charts show error bars that correspond to a 95% confidence interval around the mean. Table 5.2 summarizes the parameters used throughout this evaluation.

5.5.3 Implementation Overhead

Since BR is assumed to be a resourceful node, we limit the calculation of memory footprint to all nodes other than BR. Fig. 5.5 reports the total available memory for the Wismote mote along with ROM and RAM requirements for HRPL, SRPL, and NSRPL implementations. The memory footprint depicted in Fig. 5.5 includes the ContikiOS code and all the underlay IoT stack modules besides the routing protocol ContikiRPL. In terms of ROM requirement shown in Fig. 5.5(a), HRPL has a slightly

Table 5.2: General configurations for experiments

Property	Configuration
Platform	ContikiOS v3.0 / Cooja
Topology	Regular 2D-Grid topology
Emulated mote	Wismote with 16KB RAM and 128KB ROM
Network scale	50 nodes
Network dimensions	450m * 200m
Position of the BR	Upper-left corner of the studied network
Traffic intensity	Variable (i.e. 1, 2, 3, and 4PPM per node)
Transport layer	UDP protocol
Network layer	RPL mode (variable) + uIPv6 + 6LoWPAN
UMHRF threshold (β)	1.5
Radio Environment	UDGM
Transmission range	TX: 60m, Interference: 80m
MAC & PHY	CSMA/CA & IEEE 802.15.4
RDC/CCR	ContikiMAC/8Hz
Experiment iterations	10 iterations
Simulation period	5.5 hours

larger memory requirement than that required for SRPL and NSRPL by 7.2% and 9.01%, respectively, for the considered Wisemote mote. On the other hand, Fig. 5.5(b) depicts that HRPL has a RAM requirement that is 5.6% and 16.7% more than that

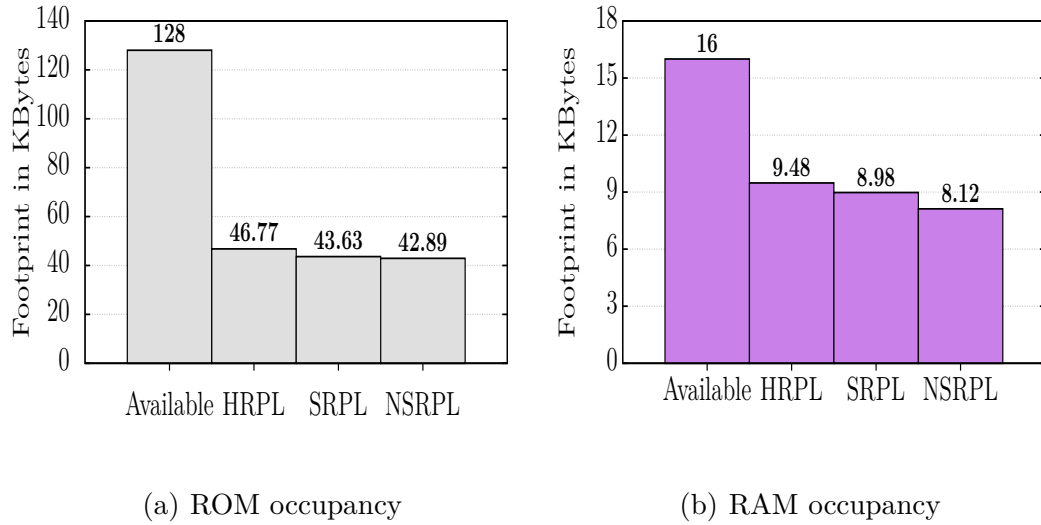
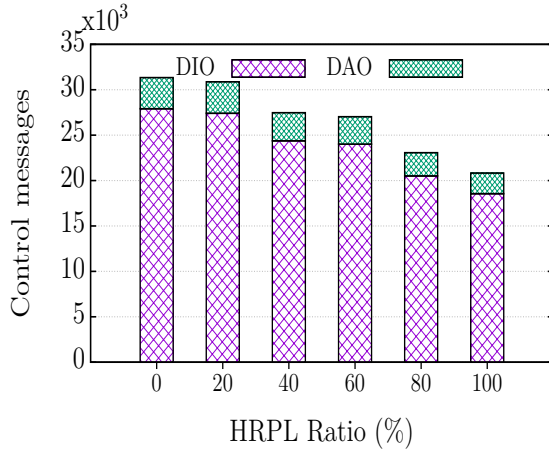


Figure 5.5: Total available memory with Wismote platform and memory footprint overhead for HRPL, SRPL, and NSRPL implementations

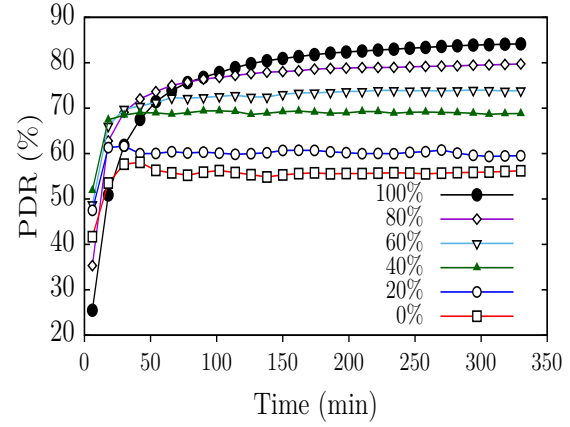
required for SRPL and NSRPL, respectively. The results show that NSRPL mode consumes least resources. The small RAM requirement with NSRPL is due to the fact that NSRPL does not store any routing entry in any node other than the BR. This is unlike HRPL and SRPL where more RAM is needed to store the routes provided by HRREQ and DAOs. Regarding the ROM requirement, SRPL needs space to store the functions of DAO-related operations and route entries maintenance. While the HRPL implementation produces an insignificant increase in memory footprint, the code can be optimized further to yield less memory usage.

5.5.4 Support for Hybrid Networks

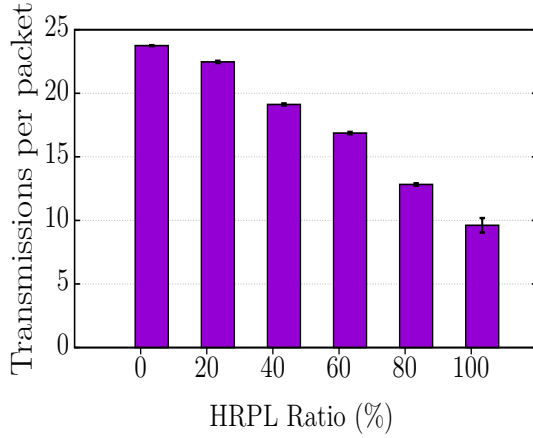
Ensuring interoperability between HRPL and non-HRPL nodes is an essential requirement for supporting hybrid networks. When a node running non-HRPL processes receives DAO packet with the new HNO option, it considers the HNO option to be unknown. The unknown ICMP option then would be ignored and the rest of DAO message would be processed safely. This section assesses the ability of HRPL to work in a hybrid network without backward-compatibility problems. First we set-up the network by making all the nodes running NSRPL mode as our work is built over this mode. Then 10 randomly selected nodes are replaced with nodes that support the proposed HRPL. The corresponding results are collected, and then the same procedure is repeated by selecting 20 more nodes to support HRPL until all nodes run HRPL mode. The following observations can be made from these experiments. First, nodes operating NSRPL or HRPL work together seamlessly in the same hybrid network, and therefore HRPL supporting nodes can coexist with conventional



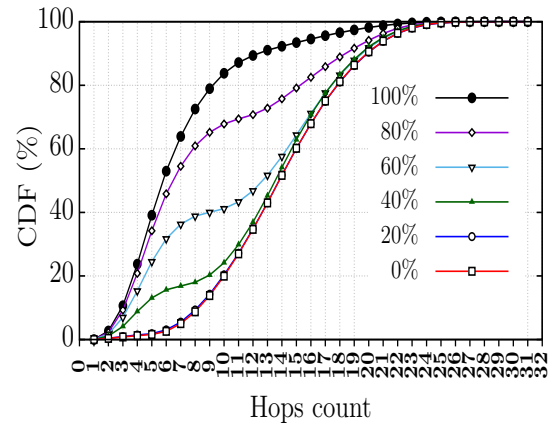
(a) RCT



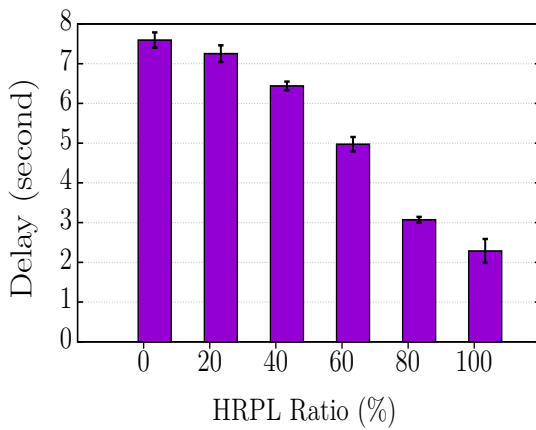
(b) PDR



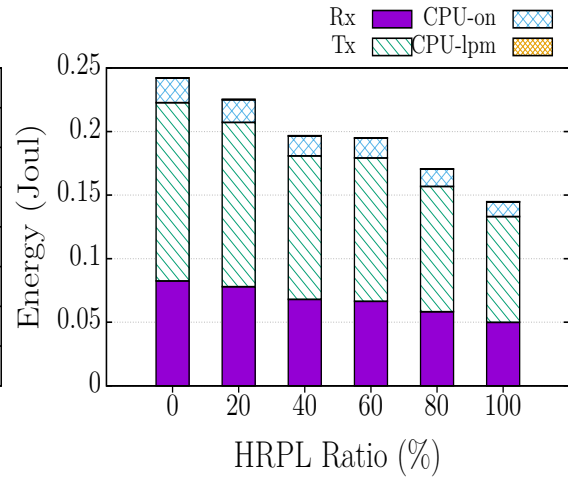
(c) MAC-transmissions



(d) Hops CDF



(e) Packet delay



(f) The energy consumption per received packet

Figure 5.6: The hybrid network performance with different HRPL node deployments at traffic intensity of 4 PPM

NSRPL nodes. Furthermore, the BR serves both HRPL and NSRPL nodes at the same time. In a sense, the BR forwards the packets downward to their destination when the source nodes is an NSRPL node and responds by HRREP to the HRREQ initiated by an HRPL node. Thus, the interoperability for data and control plane packets between NSRPL and HRPL nodes on one side, and the BR on the other side is verified. Second, as more HRPL nodes are added to the network, better system performance is observed as depicted from the results reported in Fig. 5.6. One can see from Figs. 5.6(a), 5.6(c), and 5.6(e) that the RCT, transmission count, and delay, respectively, decrease gradually as the percentage of HRPL nodes increases from 0%, i.e. all nodes run NSRPL, to 100% when all nodes run HRPL. Figure 5.6(b) shows that PDR increases from 60% to 90% when the ratio of HRPL nodes changes from 0% to 100%. Moreover, Fig. 5.6(d) depicts the CDF of hop count for the received packets. The figure shows that while about 80% of the received packets need 8 hops or less for 100% HRPL nodes, this increases to 18 hops when no nodes support HRPL. Accordingly, consumed energy decreases as the HRPL nodes percentage increases as depicted in Fig. 5.6(f). The aforementioned behaviors are attributed to the reasons illustrated in the next subsection.

5.5.5 Comparison with NSRPL and SRPL Modes

The comparisons conducted in this subsection show the effects of HRPL-related enhancements on the considered performance metrics compared to NSRPL and SRPL modes. In the first series of these comparison figures, we show the effects of HRPL-related enhancement on the RCT required to maintain the RPL-based DODAG topol-

ogy and the load fairness with respect to the RCT for the participating nodes. The effects of those enhancements on the network performance-related metrics are discussed subsequently. The second set of comparison figures study the energy-saving and fairness with respect to the energy depletion for all nodes in the network. For all comparisons, we consider four traffic intensities 1, 2, 3, and 4PPM, which correspond to inter-sending intervals of 60, 45, 30, and 15 seconds, respectively.

5.5.5.1 Results on RCT

As mentioned earlier in Subsection 2.2.3, the DIO and DAO messages are used for the DODAG construction process. To maintain the DODAG consistency, some events such as the Trickle’s timer resetting, parent switching, and loop warning, are taking place while RPL is running. These events require control messages to be transmitted upward to a preferred parent or propagated downward to the whole sub-DODAG according to the policy of the mode in use as detailed in Subsection 2.2.3.3. Figure 5.7 reports the occurrence frequency of these events throughout the entire simulation period. On the other hand, Fig. 5.8 shows the corresponding RCT generated during the simulation using the studied modes under different traffic intensities. For HRPL, the HRREQ and HRREP messages are considered part of the RCT overhead as well. The RCT, specially DIOs and DAOs, is highly affected by the previously mentioned events as it is evident from the high correlation between Fig. 5.7 and Fig. 5.8. In a sense, more parent switches, warnings, and resets contribute to more DIOs and DAOs generation. Also, one can notice that, with SRPL, DAO packets contribute significantly more to the overall RCT relative to the other two modes and the variation

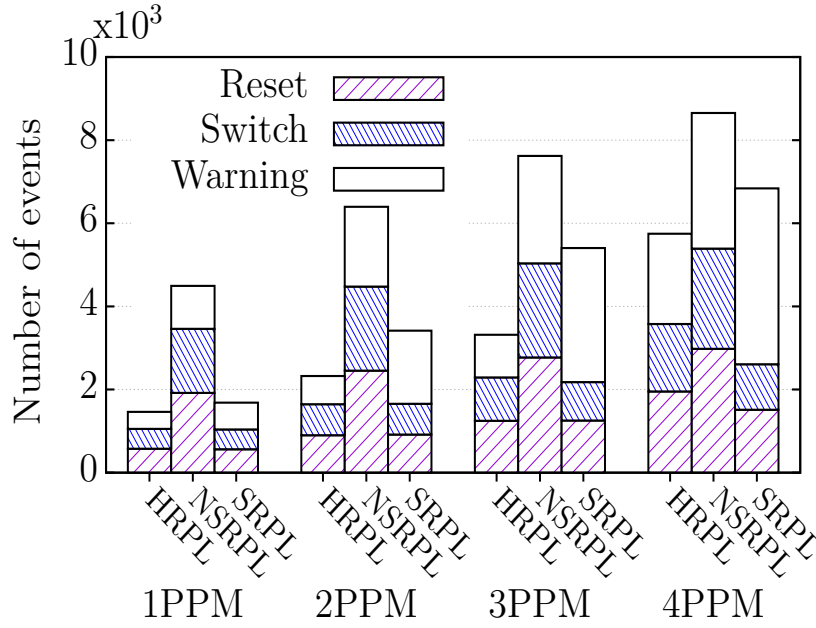


Figure 5.7: RPL events frequency under the studied modes as a function of the traffic intensity.

gets higher as the traffic increases. However, this variation is less for HRPL and NSRPL when considering total RCT. Frequent parent switch leads to frequent DIO and DAO transmissions which increases the control cost but it keeps the forwarding structure up-to-date. When the parent switch event takes place, the node sends out the DIO packet to inform other nodes with this change. Additionally, the node sends out a DAO message to the new preferred parent in the case of SRPL or to the BR in the case of both HRPL and NSRPL modes. Moreover, in SRPL, the transmitted DIO triggers all nodes in the sub-DODAG to send DAO messages upward to fix the respective routing entries of the sending nodes in their preferred parents' caches. In contrast, for both HRPL and NSRPL, only the one-hop distance nodes and the BR need to know about the switching event. On the other hand, resetting the Trickle timer to the minimum value also increases the DIO transmissions since subsequent DIO messages now have to be sent more frequently. For all cases, Fig. 5.8 demonstrates that HRPL

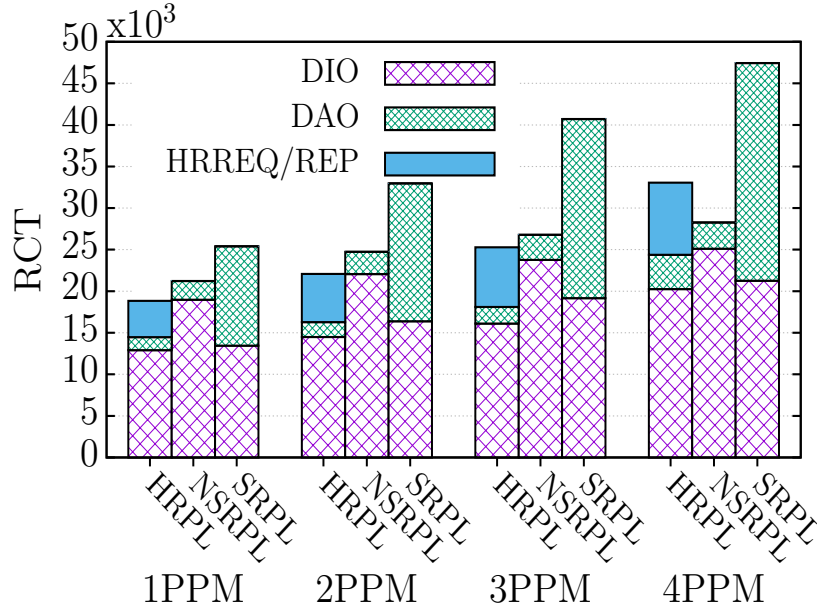
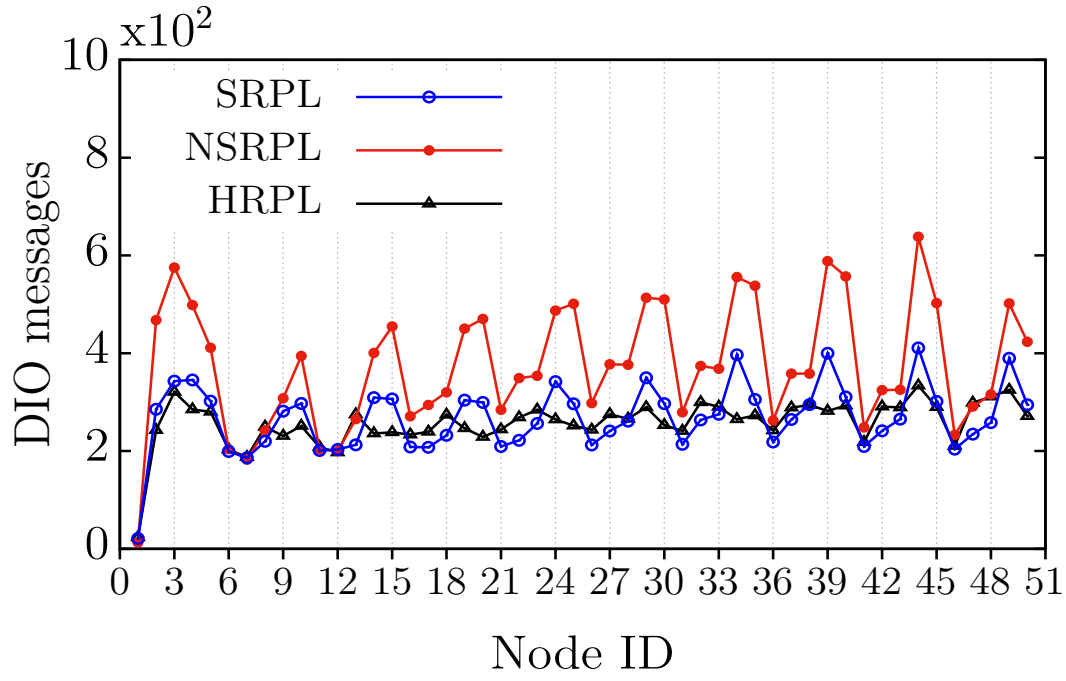


Figure 5.8: Routing control traffic generated using the studied modes as a function of the traffic intensity.

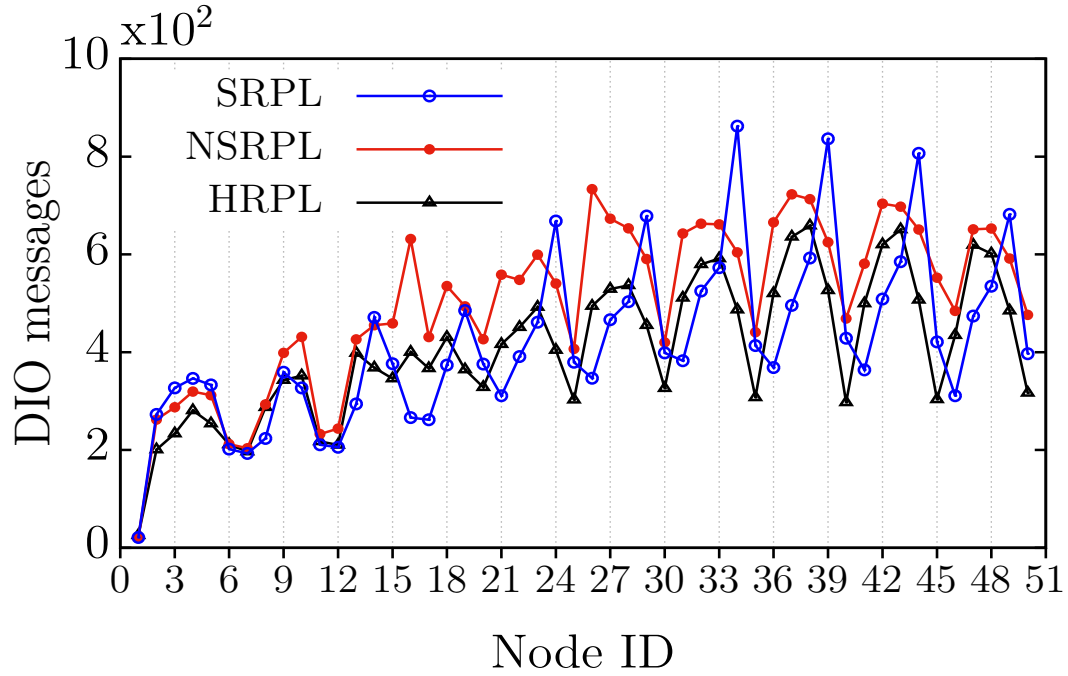
contributes to fewer event generations which translates to less RCT compared to NSRPL and SRPL modes.

5.5.5.2 Results on DIOs per node

Figures 5.9(a) and 5.9(b) report the amount of transmitted DIOs per node for the studied modes for traffic intensities of 1PPM and 4PPM, respectively. These plots have been obtained by averaging the transmitted DIOs for each node across all experiment runs. Results indicate that the $\mathcal{BR}$ emits the least number of DIOs for all considered modes. This is because $\mathcal{BR}$ sends out DIOs only in two cases; 1) at the initialization stage of the DODAG construction, and 2) at the global DODAG repair. The nodes other than the $\mathcal{BR}$ emit DIOs to maintain the DODAG consistency. The DIOs emission interval is highly affected by the aforementioned events. Generally,



(a) DIOs per node at traffic of 1PPM



(b) DIOs per node at traffic of 4PPM

Figure 5.9: Amount of DIO packets transmitted per node at traffic intensities of 1PPM and 4PPM

the farther the nodes from the $\mathcal{BR}$ are, the higher the number of DIOs emissions are as these nodes are more susceptible to changes in their DODAG depth. It should be pointed out that the number of DIOs does not depend on the size of the sub-DODAG tree of a particular node, but rather on the node level in the sub-DODAG tree. The further the node from the sub-DODAG tree root is, the more DIOs are sent. One can also observe that, at the traffic intensity of 1PPM, HRPL exhibits less fluctuation in the frequency of transmitted DIOs relative to the other modes. This behavior is a byproduct of the load balancing provided by HRPL because it forwards the data packets horizontally to their destinations. When the traffic intensity increases to 4PPM, HRPL has almost the same fluctuation as NSRPL. However, HRPL experiences less fluctuation compared to SRPL for all considered traffic intensities. These observations are also confirmed by FI(DIOs) results presented in Fig. 5.12.

5.5.5.3 Results on DAOs per node

Transmitted DAOs per node are reported in Fig. 5.10. The results show that SRPL exhibits higher oscillation for transmitted DAOs per node compared to the other two modes. With SRPL, the larger sub-DODAG the node has, the larger the number of transmitted DAOs. This is because nodes with larger sub-DODAG have to forward a large number of DAOs transmitted by their children. Figure 5.11 depicts same results shown in Fig. 5.10 but the nodes ID in the x axes have been ordered according to the distances from the BR. In general, as this figure shows, the nodes closer to the BR generate higher DAOs messages compared to the nodes that are farther from the BR. Also the figure shows that the fluctuation in the transmitted DAOs range

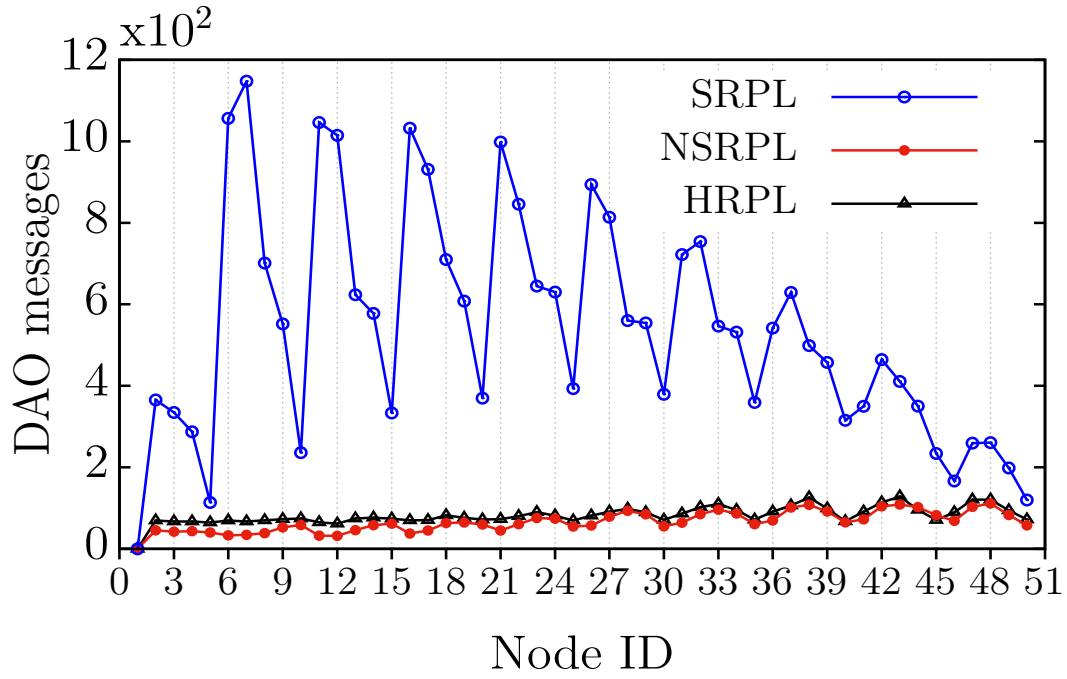


Figure 5.10: Amount of DAO packets transmitted per node at the traffic of 4PPM.

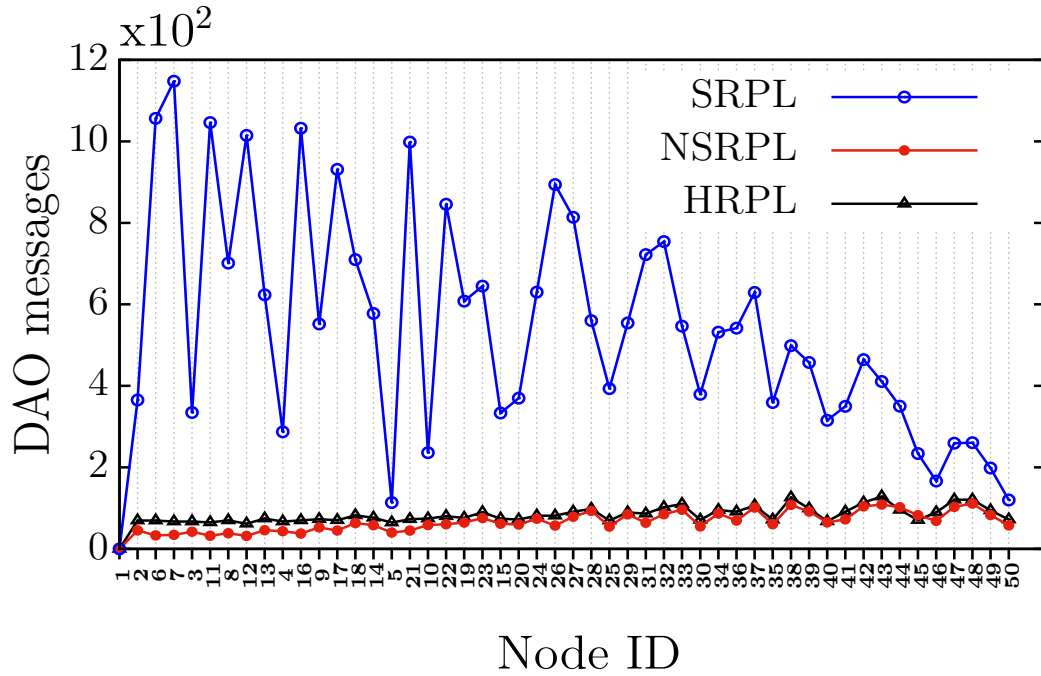


Figure 5.11: Amount of DAO packets transmitted per node at the traffic of 4PPM. The node IDs on the x-axis are ordered based on the node's distance from the BR

is insignificant in both NSRPL and HRPL cases. This is a direct the consequences of the fact that nodes in both of these modes do not save any DAOs-based routing entries in their caches, and thus there is no need to solicit DAOs from the nodes in the sub-DODAG tree when parent switch events occur.

5.5.5.4 Results on RCT fairness index

To scrutinize the RCT fairness aspect further, we evaluate both FI(DIOs) and FI(DAOs) for different traffic intensity as depicted in Fig. 5.12(a) and Fig. 5.12(b), respectively. Here, again it is clear that HRPL has the highest FI figures and thus the maximum load distribution fairness across all nodes and for all considered traffic intensities, except for minor case of FI(DIOs) at 4PPM where it is slightly lower than the corresponding fairness index for NSRPL. This balance has a significant effect on many other metrics such as energy consumption and network congestion.

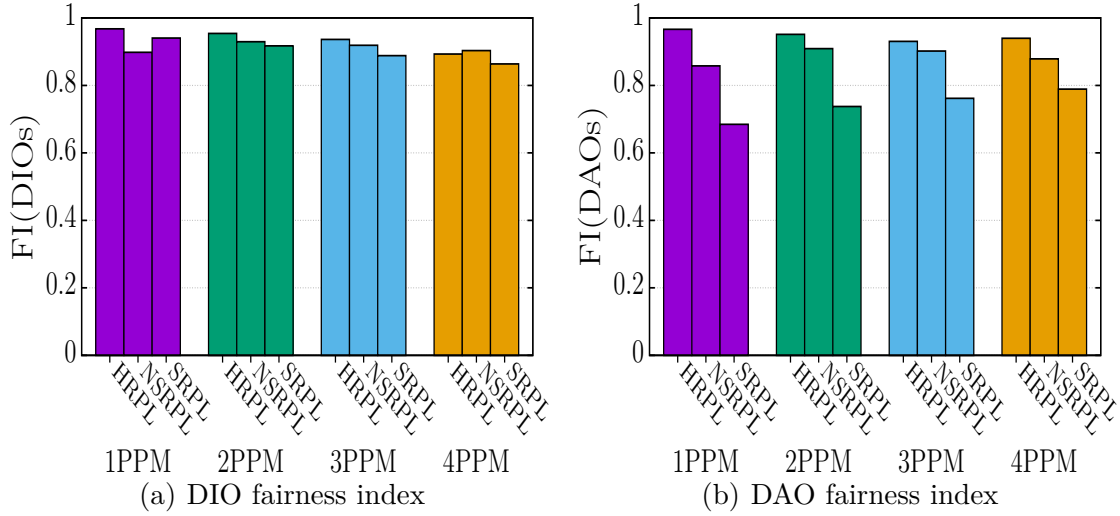


Figure 5.12: Fairness index for control messages with varying traffic intensity

5.5.5.5 Results on PDR

Figure 5.13 shows the performance of HRPL, NSRPL, and SRPL modes in terms of the PDR percentage for traffic intensities of 1PPM and 4PPM. The PDR percentage is reported periodically every 12 minutes for the duration of the simulation. Figure ?? shows that HRPL provides considerably higher PDR percentages that are about 10% and 20% more compared to that for NSRPL and SRPL, respectively. The results indicate that PDR decreases from 97% to 86% with HRPL when the traffic intensity increases from 1PPM to 4PPM. However, PDR decreases from 90% to 63.7% and from 60.5% to 49% for NSRPL and SRPL, respectively. As expected, it can be observed

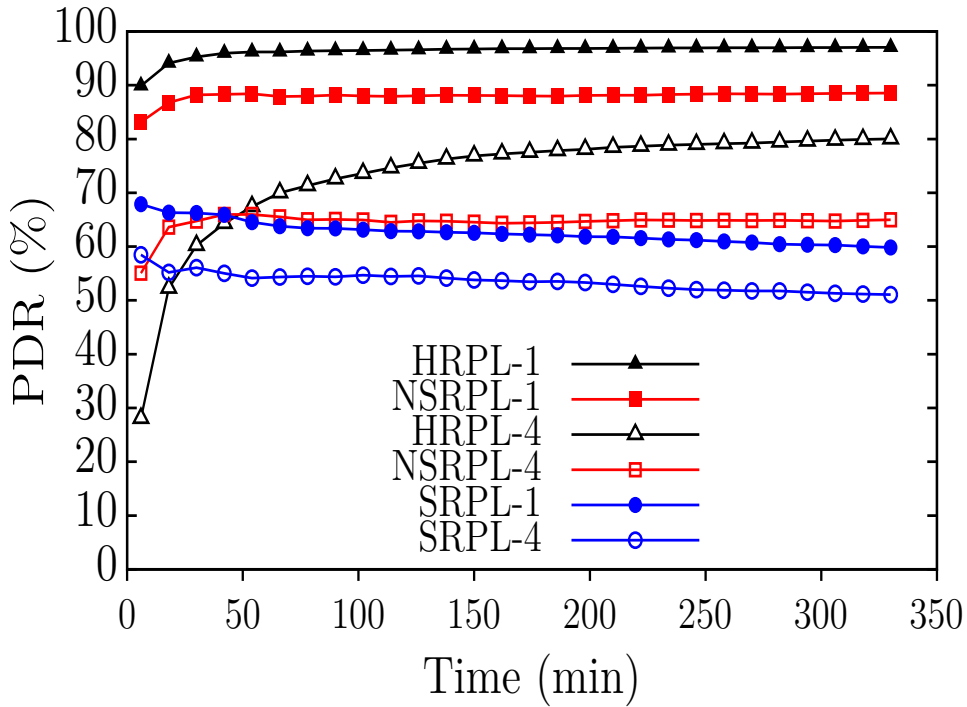


Figure 5.13: PDR using the studied modes as a function of the traffic intensity.

from Fig. 5.13 that higher traffic intensity leads to less PDR percentage across all considered modes. This is because heavy traffic increases the network congestion and buffer overflow which lead to high packet loss rate [143].

5.5.5.6 Results on packets loss

Data packets are lost due to various reasons such as routing errors, MAC collisions, or/and buffer overflows. Routing errors occur when the BR can not find a downward path toward the destination. The MAC contention causes a packet to be dropped if it exceeds the maximum allowed re-transmission limit. Similarly, buffer overflow causes packets to be dropped when the node has no memory capacity to store arriving packets before processing them. Figure 5.14 depicts the breakdown of the data packet loss percentage concerning the three aforementioned loss causes for the considered traffic intensity.

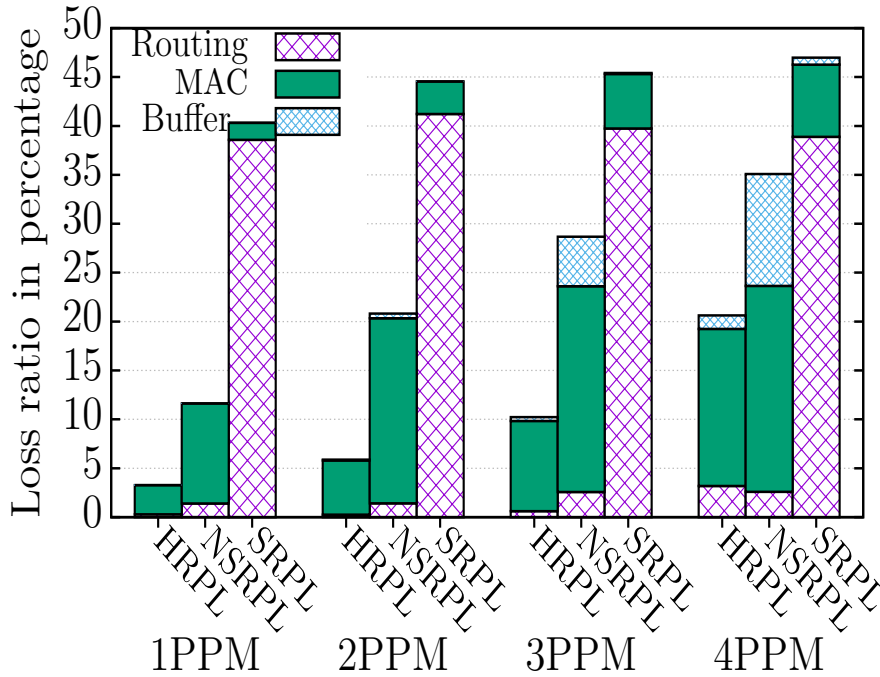


Figure 5.14: The packet loss percentage caused by routing errors, MAC collisions, and buffer overflow

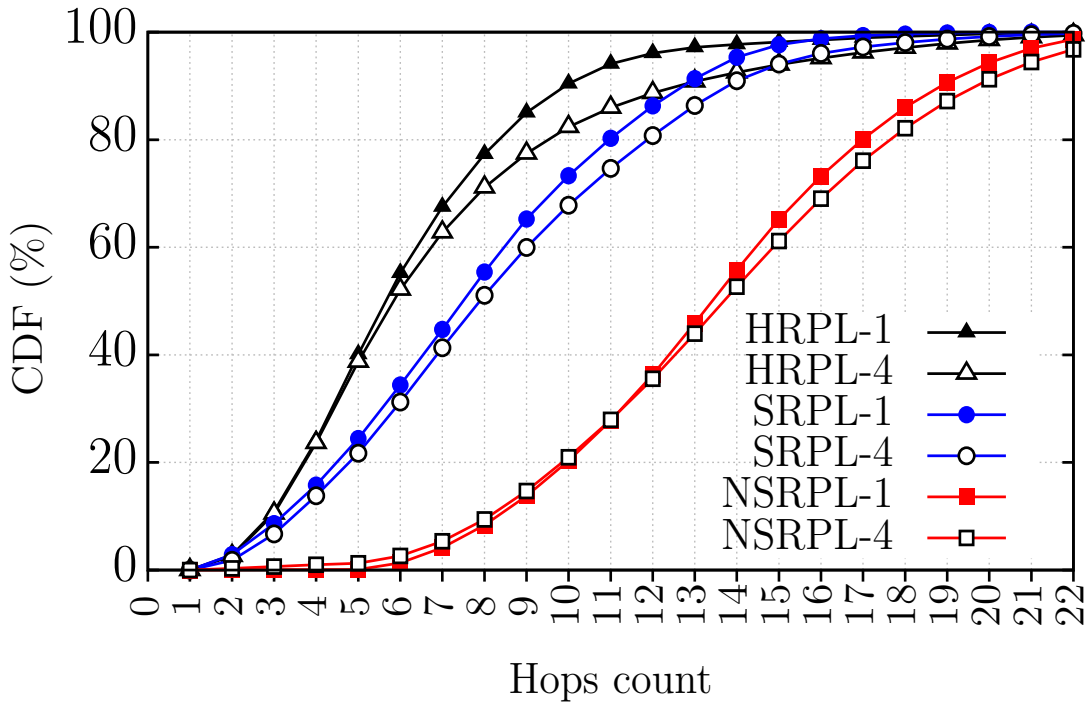
It is observed that the loss caused by the MAC contention increases when the traffic intensity increases. This intuitive proportional relationship is attributed to the

fact that multiple nodes share the same radio channel to send and receive data and control packets. Increasing the traffic intensity would increase the channel congestion which is translated to more re-transmissions and that leads to exceeding the allowed re-transmission limit and packet loss. For NSRPL, loss caused by MAC collision increases significantly as the traffic intensity increases. This increase is attributed to the increase of the contention for nodes near the BR as packets transit through the BR and then are source-routed to the destination. In addition, these nodes for the case of NSRPL suffer from buffer overflow especially when the traffic intensity increases as they are responsible for forwarding heavy load upward to the BR. On the other hand, SRPL suffers from high routing errors due to the lack of memory to store all the routing entries for all targets in the node's associated sub-DODAG. The packets are forwarded upward to the common ancestor which is eventually the BR itself to hopefully find a path to the destination. If the BR has no route to reach the corresponding destination, the BR drops the packet. This unreachability problem results in higher packet loss which degrades the protocol reliability. However, it is noted that HRPL does not suffer from the routing errors as it uses the source routing strategy to route the packets horizontally with no need to forward them upward to the BR. Also, a small number of packets are lost due to the buffer overflow in HRPL. This is because the used paths are evenly distributed in the network, and traffic is not congested for nodes near the BR as in the case for NSRPL. In summary, it can be concluded that HRPL shows robust reliability as the traffic intensity increases.

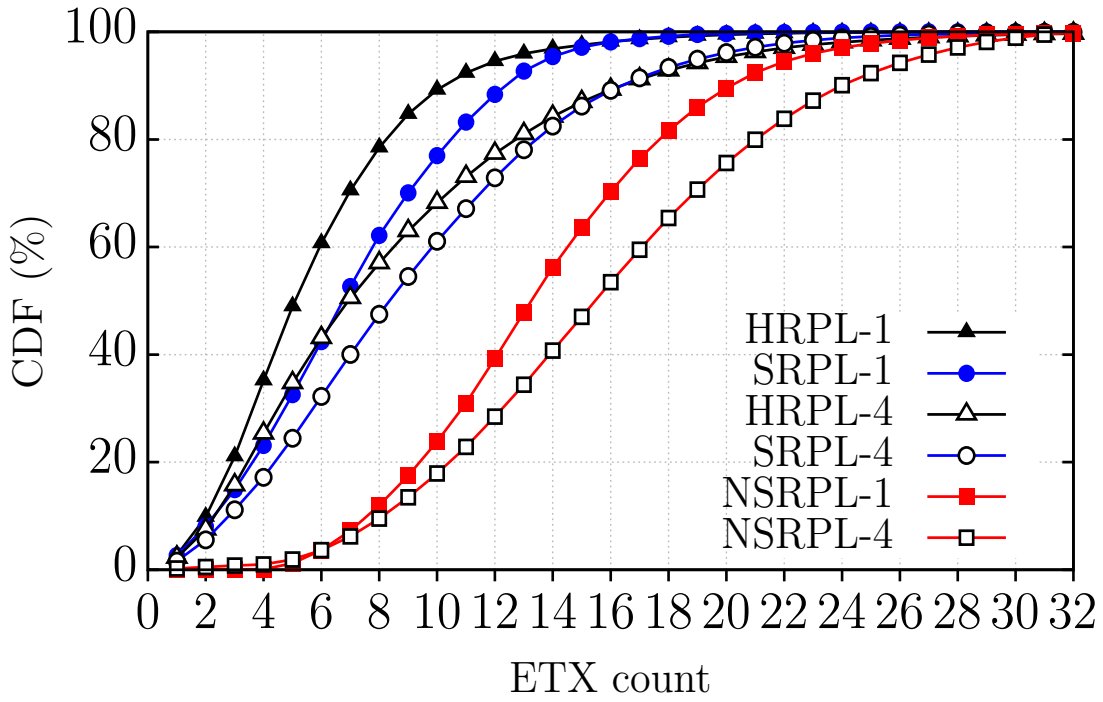
5.5.5.7 Results on hops and transmissions count

To assess the quality of paths which are constructed by HRPL, we show in Fig. 5.15 the CDF for both hops count and transmissions count for all delivered packets using the considered modes for the two extreme traffic intensities 1PPM and 4PPM.

Figure 5.15(a) demonstrates that with HRPL mode 80% of the delivered packets traverse 9 hops or less to reach their final destinations for the traffic of 4PPM. However, the same threshold leads to 11, and 17 hops or less for SRPL, and NSRPL modes, respectively. The figure also shows that CDF for the hop count for all studied modes does not vary significantly across the considered traffic intensities. On the other hand, Fig. 5.15(b) depicts the CDF for the transmissions count for all delivered packets again for the two extreme traffic intensities of 1PPM and 4PPM. The results reveal the following observations. First, unlike the hops count, the transmissions count increases with the increase of the traffic intensity for all studied modes. Second, HRPL exhibits the best performance in terms of the transmissions count compared to the other two modes. With HRPL, about 80% of the delivered packets require 8 transmissions or less traffic intensity of 1PPM and this value increases to 10 transmissions or less for the traffic intensity of 4PPM. On the other hand, for SRPL and NSRPL, about 80% of the delivered packets require around 11 and 18 transmissions or less, respectively, for the traffic of 1PPM. When the traffic increases to 4PPM, the delivered packets by SRPL and NSRPL require 14 and 21 transmissions or less, respectively. Finally, for the no congestion case, i.e. low traffic, a packet needs roughly one transmission to be successfully received by the next-hop as can be seen from Figs. 5.15(a) & 5.15(b) for



(a) CDF of hop count



(b) CDF of transmission count

Figure 5.15: CDF for hops and MAC transmissions

the traffic of 1PPM for all studied modes.

To sum up the above observations, we plot in Figs. 5.16(a) and 5.16(b) the average number of hops and the corresponding number of transmissions needed per delivered data packet as a function of the traffic intensity for the studied modes. Again one can observe that the hop count is not very sensitive to the traffic intensity level. For all cases, the results indicate that the HRPL mode delivers packets with the least number of hops and transmission count compared to the other routing modes. The transmission count for delivered packets using the HRPL mode is approximately 16.6% and 55.4% less than that produced by the SRPL and NSRPL modes when averaged across the considered traffic intensities.

5.5.5.8 Results on delay

According to RFC 5673 [25], most of the industrial-enabled CPSs are very delay-sensitive. Therefore, data transmission and routing should be optimized to be as timely and reliable as possible [42]. Figure 5.17 intends to compare the average delay per delivered data packet for the modes under study as a function of the traffic intensity. Results indicate that HRPL exhibits slightly less delay than SRPL by an average of 7.4% across the considered traffic intensities. This small difference is attributed to the fact that the delay is computed by considering only received packets. Therefore, packets that have near common ancestors with shorter paths are over-represented in the SRPL delay. Additionally, SRPL routes the packets in most cases through the common ancestor of the source and the destination which avoids the high traffic that may occur near the BR. HRPL experiences a delay very similar to that of SRPL par-

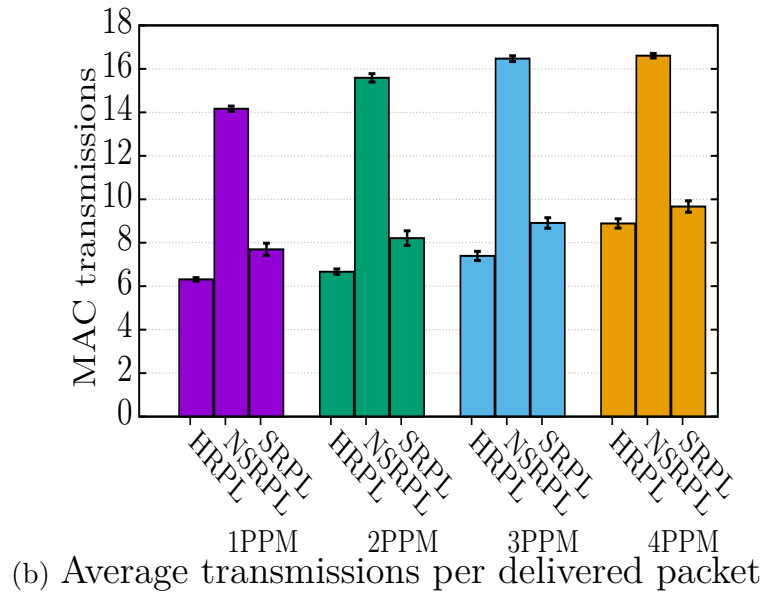
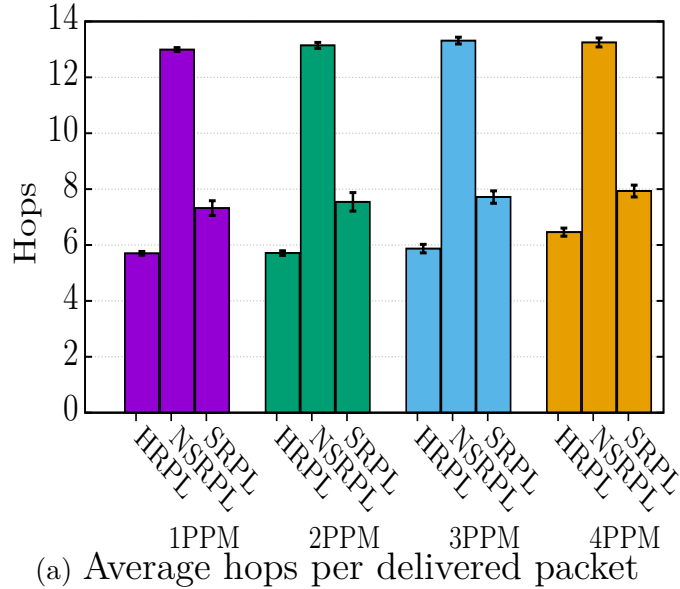


Figure 5.16: The average number of hops and MAC transmissions count per received packet using the studied modes.

ticularly at high traffic intensity. This due to the fact that the SRPL routes packets through the nearest common ancestor of the source and the destination. Therefore, when the traffic increases, the ancestor nodes experience high load leading increased

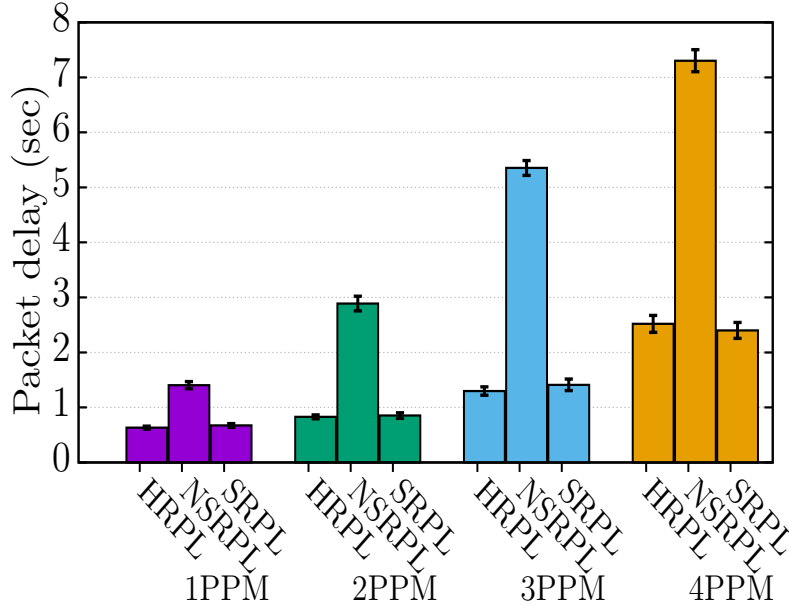


Figure 5.17: The average delay per received packet as a function of traffic intensity.

queuing and MAC contention delays. On the other hand, HRPL reduces the delay by an average of 74% relative to NSRPL. This is mainly because NSRPL routes all packets through the BR, which leads to greater buffer occupancy and increased channel-access delays especially at the nodes closer to the BR. The load balancing achieved by HRPL helps in reducing the MAC contention and queuing delays. Finally, it can be noted that HRPL scales almost linearly with the traffic intensity and this is attributed to the buffer occupancy and MAC delays previously discussed in PDR discussion.

As mentioned earlier in this chapter, if any an HRPL node wants to communicate with another node, first it sends an HRREQ message to the BR asking for a path to the intended destination. The BR then constructs the shortest path, if any, and responses to the requesting node by means of an HRREP message. Let $\mathcal{T}_{req}^{tx}$ refer to the sum of the HRREQ and HRREP transmission times, $\mathcal{T}_{req}^{proc}$ refer to the time needed to process the HRREQ by the BR, and $\mathcal{T}_{pkt}^{tx}$ refer to the packet transmission time over

the path provided by the BR. Then the total delay that the particular packet may experience using HRPL mode is calculated by (5.8).

$$Delay = \mathcal{T}_{req}^{tx} + \mathcal{T}_{req}^{proc} + \mathcal{T}_{pkt}^{tx} \quad (5.8)$$

Table 5.3 breaks-down the HRPL delays reported in Fig. 5.17 into the constituent components specified in (5.8) for all considered traffic intensities. The table contents show the time percentage for each component relative to the total delay. From

Table 5.3: Time percentage for the components in HRPL delay

	$\mathcal{T}_{req}^{tx}$	$\mathcal{T}_{req}^{proc}$	$\mathcal{T}_{pkt}^{tx}$
1PPM	3.07%	0.03%	96.89%
2PPM	2.52%	0.0088%	97.47%
3PPM	1.09%	0.024%	98.9%
4PPM	0.5%	0.0009%	99.49%

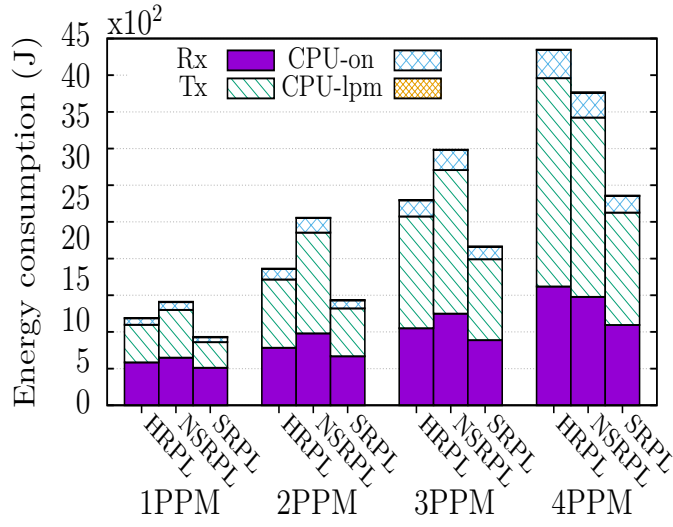
Table 5.3, the following observations can be inferred:

- $\mathcal{T}_{pkt}^{tx}$ represents the highest percentage of the HRPL delay.
- Interestingly, even though HRPL requires the source node to ask the BR for a path to the destination, $\mathcal{T}_{req}^{tx}$ and $\mathcal{T}_{req}^{proc}$ are insignificant compared to $\mathcal{T}_{pkt}^{tx}$. This observation implies that the overhead incurred by HRPL is acceptable.
- $\mathcal{T}_{req}^{tx}$ decreases by increasing traffic intensity. This is attributed to the fact that the sending nodes locally cache the received paths from the BR and do not request the path in subsequent transmissions. As the traffic intensity increases, the fraction of requests fulfilled from the local cache also increases leading to a decrease in the average $\mathcal{T}_{req}^{tx}$ percentage relative to the total delay.

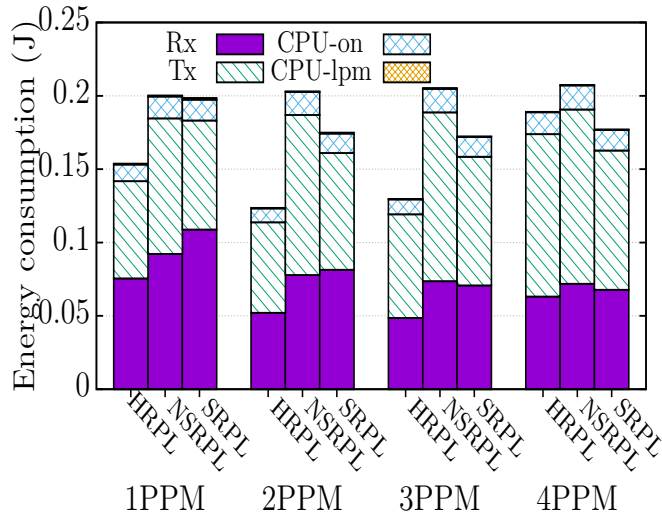
- It seems that there is no correlation between $\mathcal{T}_{req}^{proc}$ and traffic intensity. This is because the intended destinations are randomly selected and $\mathcal{T}_{req}^{proc}$ depends on the path lengths constructed by the BR.

5.5.5.9 Results on Energy Consumption

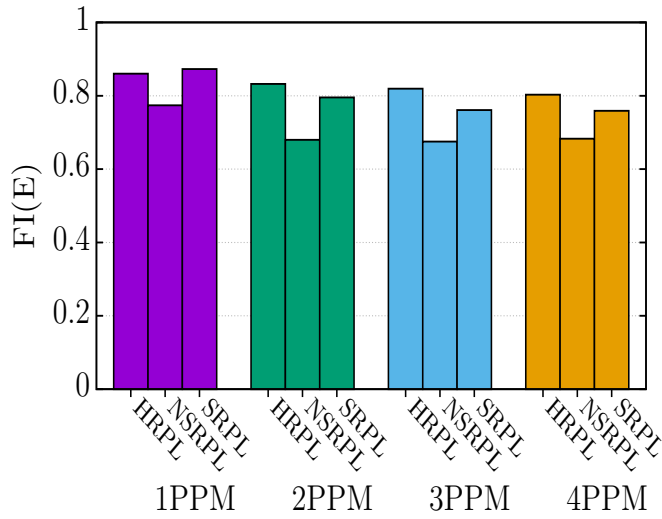
In this subsection, the consumed energy by HRPL, NSRPL, and SRPL modes is evaluated. Figure 5.18(a) depicts the energy consumption for the whole network using the considered modes as a function of the traffic intensity. Results show that the main portion of the consumed energy is due to radio transmissions. The RDC mechanism contributes to saving most of the energy consumed by the radio while in the listening mode. Also, the results indicate that SRPL outperforms both HRPL and NSRPL consuming lower overall energy. However, this low energy consumption comes at the cost of lower PDR for by SRPL as discussed earlier. Specifically, averaging over the considered traffic intensities, the total energy consumed by HRPL is 20.57% higher than that consumed by SRPL. However, HRPL decreases the total energy depletion by an average of 21.3% relative to NSRPL. It is important to point out that if we consider the consumed energy per delivered packet, then HRPL easily outperforms both SRPL and NSRPL as shown in Fig. 5.18(b), except for the traffic intensity of 4PPM where HRPL consumes slightly higher energy compared to SRPL. For this case, HRPL decreases the energy consumption by an average of 34.98% and 21.68% compared to NSRPL and SRPL, respectively, for the considered traffic intensities. At the traffic intensity of 4PPM, SRPL consumes less energy per received packet relative to HRPL. This is only because a small fraction of the sent packets is delivered for the



(a) Total energy consumption



(b) Average energy consumption per delivered packet



(c) Fairness index for the energy consumption

Figure 5.18: Energy consumption using HRPL, NSRPL, and SRPL modes as a function of the traffic intensity

case of SRPL where the delivered packets are routed through near common ancestors with short and less congested paths. Therefore, these packets are over-represented in the results for the consumed energy.

Finally, we examine the fairness index in terms of energy consumption, $FI(E)$, over the participating nodes, where the consumed energy is used as the parameter in (5.7). Figure 5.18(c) shows the $FI(E)$ using the considered modes as a function of the traffic intensity. It can be seen that at the traffic intensity of 1PPM, the balance provided by SRPL is slightly higher than that for HRPL. However, for higher traffic intensities, HRPL provides better balance compared to SRPL.

5.5.6 Comparison with Opportunistic RPL (ORPL)

Referring to the categories of N2N routing studies outlined in Subsection 3.2, it is unfair to compare most of these works with the approach proposed in this work. For example, extra GPS hardware and special spectral sensors are required with the geographical- and spectrum-based approaches, respectively. These additions lead to unfair comparison as our approach which does not use any extra special tools or hardware other than the minimum requirements specified by the standard. Furthermore, most of the other works have no public implementation available to aid in the comparison effort. An opportunistic RPL study, presented in [101] and referred to by ORPL, is the most appropriate work for a fair comparison. This is in addition to the fact that its implementation is publicly available. In this subsection, a brief overview of ORPL is presented and then a performance comparison between HRPL and OPRL is given. ORPL is a compact- and opportunistic-based approach enhancing the scal-

ability of RPL in N2N routing. It utilizes the Expected Duty Cycles (EDC) as the routing metric, which refers to the number of MAC wake-up periods required to reach the root. Like SRPL, a node in ORPL stores routing information for all nodes in its sub-DODAG, however, ORPL utilizes a bloom filter to store the routing information about children nodes instead of the routing table used in SRPL. ORPL handles N2N traffic by first routing packets upwards to any common ancestor, and then downwards to the destination. A node anycasts N2N packets to a set of nodes instead of unicasting to a particular next-hop. The anycast strategy exploits the first awoken node to improve the routing and thus the name the opportunistic routing. This routing exploits more nodes as the next hop to find shortcuts between branches of the topology. When a node receives a packet, it decides whether to forward or ignore it. When the packet is forwarded, a link-layer acknowledgment is sent back.

The RPL-enabling DAO messages are not used to build the upward routing, but ORPL nodes instead broadcast their built routing set. Upon receiving the broadcasted set, the node merges the set with its own through a bitwise OR operation. Since the DAO messages are not used, ORPL is not backward compatible with the standard, and therefore RPL and ORPL nodes cannot work seamlessly in a hybrid network.

The rest of this subsection focuses on the comparing the performance of HRPL to that of ORPL. Firstly in terms of the memory footprint, HRPL has larger ROM and RAM sizes of 8.7% and 12.6%, respectively, relative to that for ORPL. Secondly, Fig. 5.19 compares the performance of both HRPL and ORPL in terms of the PDR for traffic intensities 1, and 4 PPM. It can be noticed that HRPL provides better PDR specifically for higher traffic intensities. This is attributed to the high contention and

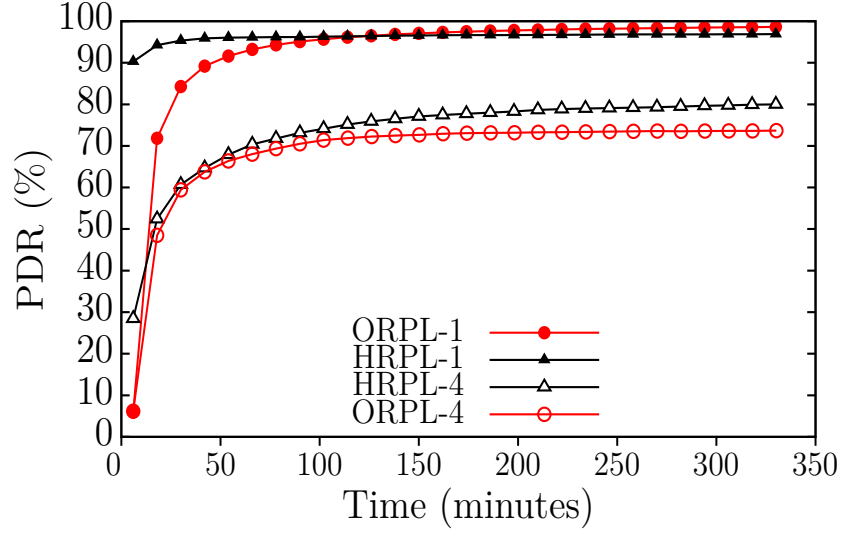


Figure 5.19: PDR using HRPL and ORPL as a function of 1PPM and 4PPM traffic intensities

buffer overflow that occur in the vicinity of common ancestors for the case of ORPL, resulting in higher loss rates relative to HRPL. Additionally, due to the use of bloom filters, a wrong routing decision may occur as a result of false positives during the checking of a node against the routing set.

Thirdly, Fig. 5.20 depicts the average MAC transmissions per delivered packet. As

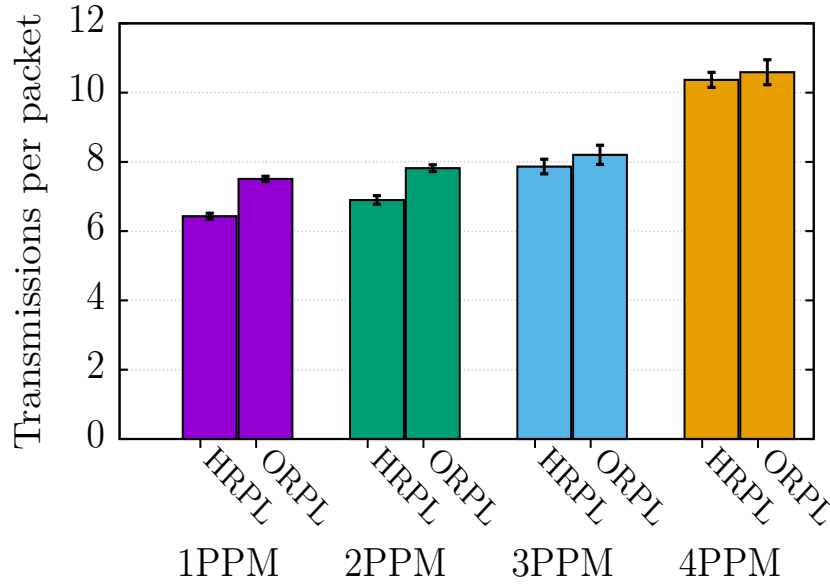


Figure 5.20: The average MAC transmissions per received packet

the figure shows, HRPL gives a comparable transmissions count per received packet relative to ORPL. This is because for some source-destination pairs, ORPL cannot find a shortcut over branches and hence the corresponding packets have to be routed through the root.

Figure 5.21 compares the delay per received packet using HRPL and ORPL for the considered traffic intensities. The figure shows that HRPL reduces the delay by up to 40% relative to ORPL for traffic intensity of 4PPM. Routing packets through the root and through common ancestors in ORPL increases the buffering and processing delays. In addition, the false positive problem mentioned above triggers a recovery action to bring a packet back to the right path. This recovery adds an extra delay for packets to reach their destinations. Furthermore, because ORPL cannot distinguish between an actual false positive and a loss due to bad link conditions or topology changes, a recovery procedure is triggered in all these cases. All the above reasons contribute to the excess delays for packets in the case of ORPL.

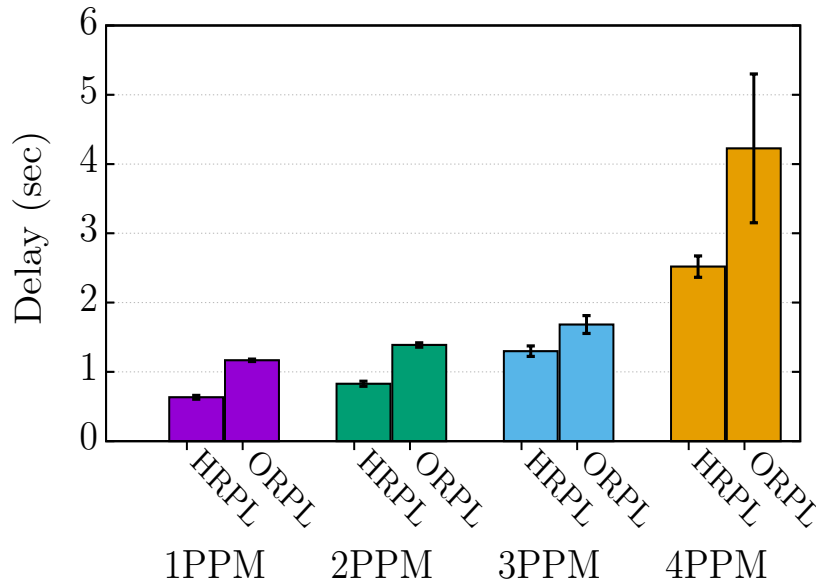


Figure 5.21: The average delay per received packet as a function of traffic intensity

Finally, Fig. 5.22 shows the energy consumption per received packet consumed by the CPU and radio transceiver using HRPL and ORPL for the considered traffic intensities. The figure shows that for high traffic intensity, ORPL consumes more energy compared to HRPL. The CPU energy consumed for ORPL is higher than that for HRPL. This is due to the bloom filter-operations which ORPL nodes have to execute. The notable operations, among others, are node insertion-based hashing to the bloom filter, merging of routing sets, and checking of element membership. Also, the figure shows that the higher the traffic is, the higher the energy consumed by the radio transceiver is for ORPL. This is because the number of received packets gets smaller in ORPL as the traffic goes higher, which in turn is translated to higher energy consumption per received packet.

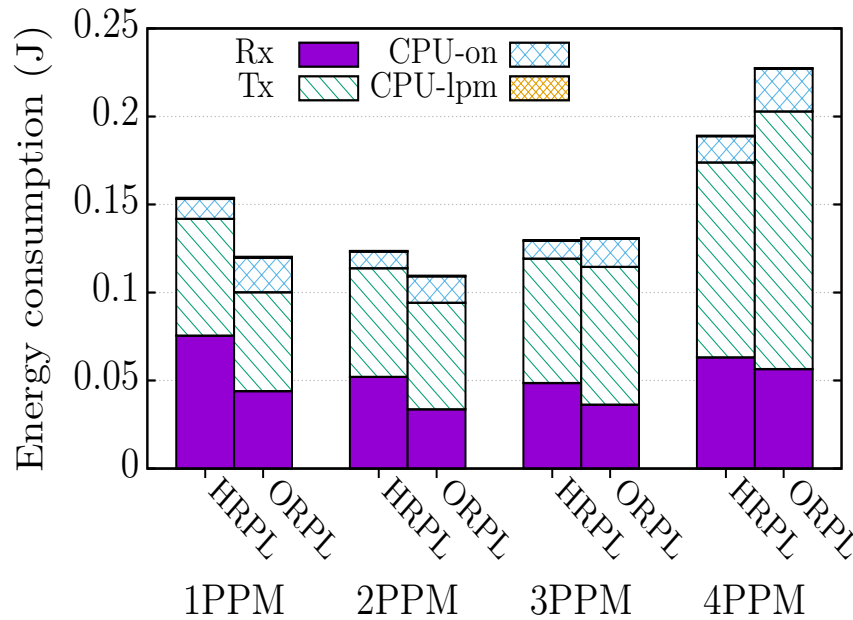


Figure 5.22: The Energy consumption per received packet as a function of traffic intensity

Figure 5.23 compares the fairness index for energy consumption across nodes for both HRPL and ORPL as a function of traffic intensity. As the figure shows, the ORPL fairness index gradually decreases with the increase in the traffic. This is due to the uneven distribution of the load particularly for the common ancestors and the nodes positioned close to the root.

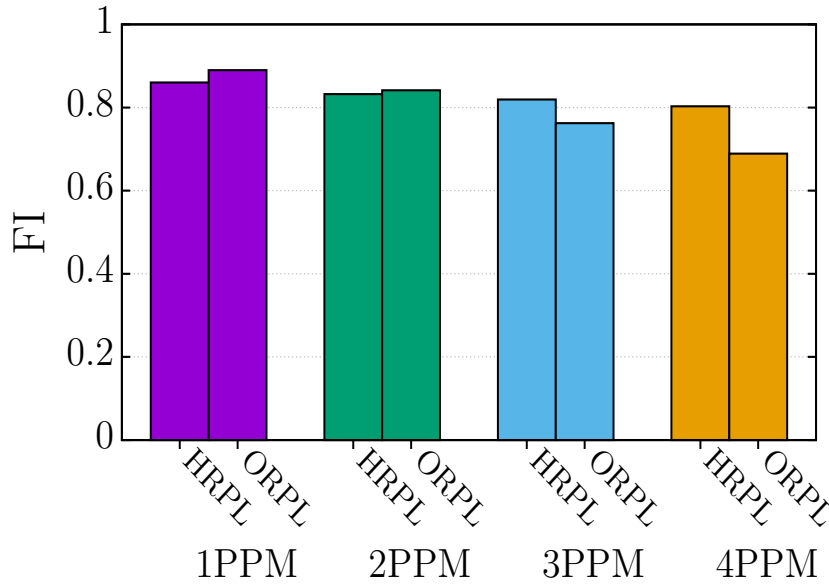


Figure 5.23: The FI of the Energy consumption per received packet as a function of traffic intensity

5.6 Chapter Summary

RPL was mainly designed with an assumption of gathering data towards a single destination, typically the root node, is the predominant traffic flow. On the other hand, the N2N communication, where the root is neither source nor destination, is a prime requirement in most of LLN-based actuating, decision making, and controlling applications. However, RPL does not cater well to such applications where there is a

good number of inward N2N traffic flows. In this chapter, we have proposed a hybrid routing mechanism based on RPL for high N2N communications in LLNs referred to by HRPL. HRPL is implemented on Contiki and evaluated through extensive simulations on Cooja. HRPL is fully backward compatible with RPL where HRPL and RPL nodes can work together seamlessly in a hybrid network. Results show that HRPL provides a considerable improvement in PDR relative to the standard. While HRPL requires a slightly larger memory footprint more than that for RPL, HRPL manages to significantly reduce the control plane packets, number of hops, and MAC transmissions needed to successfully deliver N2N data packets. Accordingly, this is translated to lower packet delay and less energy consumption.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

The Internet of Things (IoT) paradigm has gained broad-spectrum attention due to its high potential influence on many daily-life aspects. LLNs are essential building blocks to realize the vision of IoT. In this thesis, IoT has been studied in general and essential discovery mechanisms in IoT are discussed. In particular, service and routing discovery mechanisms have been studied in this thesis. The Service Discovery (SD) mechanism along with auto-configure naming is required to uniquely identify IoT nodes and their provided services in many areas such as monitoring and remote control. On the other hand, the Routing Discovery (RD) mechanism is required to enable efficient communication amongst these nodes. Novel extensions enhancing the performance of these mechanisms have been proposed, implemented, and evaluated. In this final chapter, the relevant contributions are summarized, some of the limitations of the proposed mechanisms are discussed, an outlook on potential future work is

sketched out, and the publications emanating from this thesis are listed.

6.1 Thesis Main Contributions

This thesis makes the following main contributions to foster the adoption of existing discovery protocols in IoT-based networks.

6.1.1 Enhanced mDNS (EmDNS)

Many diversified services can be offered by Low-power and Lossy Network (LLN) nodes contributing to the IoT vision. A distributed naming and registration system, complying with the Internet Protocol (IP) standard, is needed for the unique identification of these nodes and their provided services in many areas such as monitoring and remote control. Due to the high expected number of IoT nodes, manual configuration is not practical, and an efficient auto-configuration mechanism is essential. Although the multicast Domain Name System (mDNS) is the most common distributed naming protocol nowadays, it is still immature when it comes to working with constrained LLN nodes in IoT. This thesis proposes an Enhanced mDNS (EmDNS) which augments mDNS with three proposed enhancements: the Persistent-based Selection Stage (PSS), the Simultaneous Start-up Enhancement (SSE), and the Announcement Suppression Mechanism (ASM). Using the Cooja emulation platform, we evaluate both the mDNS and EmDNS for a wide range of conditions and network sizes, and produce a large array of performance figures that includes the number of control packets, energy consumption for CPU and radio transmissions, memory footprint, and others.

Through the evaluation of the fairness index for a wide range of number of nodes, we demonstrate that EmDNS provides better load balancing for the collection of nodes in the network relative to mDNS. Furthermore, results indicate that while EmDNS produces a memory footprint in RAM that is only 4.2% more than that for EmDNS, mDNS manages to significantly reduce the number of control packets by an average of 34.8% for the considered range of network size. Accordingly, the corresponding reduction in energy consumption for radio transmissions and CPU is 46.7% and 47.5%, respectively. The evaluation shows that these reductions are greater for a larger number of nodes.

6.1.2 Hybrid RPL (HRPL)

Routing discovery is a pivotal component of the communication stack in LLN. An IPv6 Routing Protocol for LLNs (RPL) has been recently standardized to support the routing discovery in a wide range of LLN-based applications. In RPL, the calculated paths from an arbitrary source to an arbitrary destination, commonly called Node-to-Node (N2N) communication, must go through a near common ancestor or the LLN's root even if shorter paths exist. In this chapter, we introduce Hybrid RPL (HRPL) as an extension for RPL that aims to discover these shorter paths to exploit them utilizing the link-state shortest path approach. HRPL is particularly suited for high N2N traffic between constrained or duty-cycle LLN nodes forming IEEE 802.15.4-based multi-hop networks. The salient features of HRPL include i) an optimized data forwarding for N2N communication pattern, ii) the utilization of the available RPL protocol control messages to avoid the redundant transmissions, and iii) the backward

compatibility as HRPL can seamlessly coexist with conventional RPL implementations in a hybrid network. This work implements HRPL in ContikiOS and provides a thorough performance evaluation using the Cooja emulator for different network conditions. Results show the following main observations: 1) HRPL provides a higher PDR relative to the standard, 2) HRPL manages to significantly decrease the control plane packets required to maintain the underlined forwarding structure, 3) HRPL needs less hop and MAC re-transmission counts to successfully deliver the N2N traffic, and 4) HRPL contributes to reducing the energy consumption leading to an increased overall network lifetime.

6.2 Limitations and Future work

This subsection attempts to identify some limitations of the proposals presented in this thesis and the way that it is planned to deal with them.

6.2.1 Expansion of EmDNS for Multi-hop Networks

In this thesis, the performance of mDNS is evaluated using a network of a one-hop distance. As future work, the authors plan to evaluate the performance of both mDNS and EmDNS protocols in LLNs with multi-hop communication. In this case, we expect that the underlying multicast protocol will play an important role in the performance of the evaluated protocols. This subsection draws the main guidelines for extending the EmDNS described in this thesis to multi-hop communication networks. Towards this end, the following points should be considered.

First, since the PSS mechanism is node-dependent, no modification is needed for the PSS by nodes operating on the multi-hop network. Second, in the traditional IP multicast protocol, delivery typically relies on topology maintenance mechanisms to discover and maintain routes to all subscribers of a multicast group. However, maintaining such typologies in LLNs is costly and may not be feasible given the available resources [145]. Due to the multicast property of mDNS, an efficient multicast-capable protocol should be chosen carefully to enable mDNS and then EmDNS to work efficiently in a multi-hop communication. Furthermore, the Multicast Protocol for Low-Power and Lossy Networks (MPL) protocol, is designed to work in a multi-hop constrained LLNs. The MPL supports two mechanisms to manage the control-plane message transmissions [145], either by the Trickle Algorithm (TA) [146], or by the classic flooding. These mechanisms allow the trade-off between dissemination latency and transmission efficiency. If the TA will be used with MPL, then the TTL value in the proposed ASM needs to be adapted experimentally to avoid the premature deletion of entries with the TENTATIVE state. This is because the TA disseminates control messages based on its adaptive timing techniques. Therefore, the control packets of mDNS or EmDNS may experience some unexpected delays. To this end, the suggested 1000 *ms* for TTL value may be less than the needed time to receive the subsequent message from the probing node positioned some hops away in the multi-hop network. For mDNS or EmDNS, when some name resolution conflicts are not resolved in the configuration phase due to the effect of multi-hop transmission failure, these conflicts can be resolved in the subsequent discovery phase using unicast instead of multicast responses. To enable the unicast responses, an efficient underlying routing protocol

should be used. Nowadays, RPL is the dominant routing protocol for constrained IoT-based networks [29]. Finally, both MPL and RPL protocol apply strict policies to prevent the looped-back multicast control packets echoed to the source node. This feature of echo occurrence prevention enables the proposed SSE mechanism to work correctly in a multi-hop communication scenario.

6.2.2 Expansion of the Analytical Model for Trickle Algorithm (TA)

The limitations of the analytical model, presented in Subsection 2.2.4, for the DIO sending probability estimated by TA are the following: 1) The model considers synchronous start intervals for all participating nodes which is not the case in the real and the simulated scenarios. This assumption might be one of the reasons which may cause the mismatching between model and simulation result, 2) The model considers a unified node degree (i.e. network density) for all participating nodes opposing the real-world scenarios. Furthermore, the model assumes a unified communication range for all nodes, and 3) The model only considers the stability state of the network (i.e. when the interval reaches its maximum value). However, the generalized model should consider intervals of lengths less than the maximum value.

In the future, it is planned to extend that model to overcome these limitations.

6.2.3 Expansion of HRPL Experimental Analysis

Several experiments have been conducted to evaluate the feasibility and performance of HRPL using different traffic intensities. The following are some recommendations to further extend and broaden the experimental work: 1) Evaluating HRPL under various settings of the Trickle timer, 2) Evaluating the influence of the cache capacity on the HRPL performance, 3) Evaluating the effect of RDC parameter by using different values for the CCR, 4) Evaluating HRPL performance using mobile nodes as opposed to static ones, and 5) Investigating the performance of HRPL using hardware prototyping.

6.2.4 Integrating SD and RD Systems

By recalling the RPL-DODAG construction process discussed in Subsection 2.2.3, control messages are broadcasted through the entire network. Utilizing this fact, the information related to the nodes along with their provided services can be advertised to all participating nodes as well.

6.3 List of Publications

This section presents the publications that have been emanated from the work of this thesis.

6.3.1 Journal Papers

- **Mohammed Mahyoub**, Ashraf Mahmoud, Marwan Abu-Amara, Tarek Sheltami, ‘An Efficient RPL-based Routing Discovery Mechanism for Node-to-Node Communications in IoT’, submitted to *IEEE INTERNET OF THINGS JOURNAL*, (2020).
- A. Mahmoud, **M. Mahyoub**, T. Sheltami, and M. Abu-Amara, ‘Traffic-aware auto-configuration protocol for service oriented low-power and lossy networks in IoT’, *Wireless Networks* 25, 4231-4246 (2019).

6.3.2 Conference Papers

- **Mohammed Mahyoub**, Ashraf Mahmoud, Marwan Abu-Amara, Tarek Sheltami, ‘Reliable and low-latency routing mechanism for IoT-based networks’, *in proceeding of IEEE 14th Malaysia International Conference on Communication (MICC) - Special Track on Future Trends in Internet of Things Technologies* (2019)
- **M. Mahyoub**, A. Mahmoud, T. Sheltami, ‘An optimized discovery mechanism for smart objects in IoT’, *in proceeding of IEEE 8th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, pp. 649-655* (2017).

REFERENCES

- [1] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, “Standardized Protocol Stack for the Internet of (Important) Things,” *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1389–1406, 2013.
- [2] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.
- [3] C. H. Lu, “IoT-enabled adaptive context-aware and playful cyber-physical system for everyday energy savings,” *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 4, pp. 380–391, 2018.
- [4] C.-S. Shih, J.-J. Chou, N. Reijers, and T.-W. Kuo, “Designing CPS/IoT applications for smart buildings and cities,” *IET Cyber-Physical Systems: Theory and Applications*, vol. 1, no. 1, pp. 3–12, 2016.
- [5] G. Mois, T. Sanislav, and S. C. Folea, “A Cyber-Physical System for Environmental Monitoring,” *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 6, pp. 1463–1471, 2016.

- [6] S. Li, S. Zhao, P. Yang, P. Andriotis, L. Xu, and Q. Sun, "Distributed consensus algorithm for events detection in cyber-physical systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2299–2308, 2019.
- [7] Ietf, "Routing Over Low power and Lossy networks (ROLL)," 2008. [Online]. Available: <http://www.ietf.org/html.charters/roll-charter.html>
- [8] J. P. Lynch and K. J. Loh, "A Summary Review of Wireless Sensors and Sensor Networks for Structural Health Monitoring," *The Shock and Vibration Digest*, vol. 38, no. 2, pp. 91–128, 2006.
- [9] W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, "Communication systems for building automation and control," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1178–1203, 2005.
- [10] A. R. Al-Ali, I. Zualkernan, and F. Aloul, "A mobile GPRS-sensors array for air pollution monitoring," *IEEE Sensors Journal*, vol. 10, no. 10, pp. 1666–1671, 2010.
- [11] E. Maria, N. Maisonneuve, M. Stevens, M. E. Niessen, P. Hanappe, L. Steels, and E. Maria, "Citizen Noise Pollution Monitoring," *The Proceedings of the 10th International Digital Government Research Conference*, pp. 96–103, 2009.
- [12] S. Lee, D. Yoon, and A. Ghosh, "Intelligent parking lot application using wireless sensor networks," *2008 International Symposium on Collaborative Technologies and Systems*, pp. 48–57, 2008.

- [13] D. Zeng, S. Guo, and Z. Cheng, “The web of things: A survey,” *Journal of Communications*, vol. 6, no. 6, pp. 424–438, 2011.
- [14] W. Edward and C. Richard, “What Can I Do Here ? IoT Service Discovery in Smart Cities,” 2016.
- [15] K. Lee, S. Kim, J. (Paul) Jeong, S. Lee, H. Kim, and J. S. Park, “A framework for DNS naming services for Internet-of-Things devices,” *Future Generation Computer Systems*, vol. 92, pp. 617–627, 2019.
- [16] C. Sarkar, S. Member, A. U. N. S. N, S. Member, R. V. Prasad, S. Member, A. Rahim, and R. Neisse, “DIAT : A Scalable Distributed Architecture for IoT,” vol. 2, no. 3, p. 230 _239, 2014.
- [17] S. Cheshire and M. Krochmal, “Multicast DNS, RFC,” *RFC 6762*, 2013.
- [18] M. Amoretti, O. Alphand, G. Ferrari, F. Rousseau, and A. Duda, “DINAS: A Lightweight and Efficient Distributed Naming Service for All-IP Wireless Sensor Networks,” *IEEE Internet of Things Journal*, vol. 4, no. 3, p. 670_684, 2017.
- [19] S. Cirani, L. Davoli, G. Ferrari, R. Léone, P. Medagliani, M. Picone, and L. Veltri, “A Scalable and Self-Configuring Architecture for Service Discovery in the Internet of Things,” in *IEEE Internet of Things Journal*, vol. 1, no. 5, p. 508_521, 2014.
- [20] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester, “Facilitating Sensor Deployment, Discovery and Resource Access Using Embed-

- ded Web Services,” *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 717–724, 2012.
- [21] B. Carballido Villaverde, R. D. P. Alberola, A. J. Jara, S. Fedor, S. K. Das, and D. Pesch, “Service discovery protocols for constrained machine-to-machine communications,” *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 41–60, 2014.
 - [22] F. Siddiqui, S. Zeadally, T. Kacem, and S. Fowler, “Zero configuration networking: Implementation, performance, and security,” *Computers and Electrical Engineering*, vol. 38, no. 5, pp. 1129–1145, 2012.
 - [23] K. Lynn, S. Cheshire, M. Blanchet, and D. Migault, “Requirements for Scalable DNS-Based Service Discovery (DNS-SD) / Multicast DNS (mDNS) Extensions, RFC,” *IETF RFC 7558*, pp. 1–13, 2015.
 - [24] T. Watteyne, U. C. Berkeley, T. Winter, and D. Barthel, “Routing Requirements for Urban Low-Power and Lossy Networks,” *RFC 5548*, pp. 1–21, 2009.
 - [25] K. Pister, P. Thubert, and T. Phinney, “Industrial Routing Requirements in Low-Power and Lossy Networks,” *RFC 5673*, pp. 1–27, 2009.
 - [26] J. Buron, “Home Automation Routing Requirements in Low-Power and Lossy Networks,” *RFC 5826*, 2010.
 - [27] J. Martocci and P. D. Mil, “Building Automation Routing Requirements in Low-Power and Lossy Networks,” *RFC 5867*, 2010.

- [28] T. Winter, P. Thubert, A. Brandt, T. H. Clausen, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur, “RPL: IPv6 Routing Protocol for Low Power and Lossy Networks,” *Network Architectures and Services*, no. July, pp. 59–66, 2011.
- [29] T. Winter, P. Thubert, A. R. Corporation, and R. Kelsey, “RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks,” *RFC 6550*, pp. 1–157, 2012.
- [30] J. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, “Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks,” *Request for Comments: 6551*, 2012.
- [31] J. Tripathi, “On Design, Evaluation and Enhancement of IP-Based Routing Solutions for Low Power and Lossy Networks,” Ph.D. dissertation, 2014.
- [32] M. Zhao, I. W. H. Ho, and P. H. J. Chong, “An Energy-Efficient Region-Based RPL Routing Protocol for Low-Power and Lossy Networks,” *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1319–1333, 2016.
- [33] S. Dawson-Haggerty, a. Tavakoli, and D. Culler, “Hydro: A Hybrid Routing Protocol for Low-Power and Lossy Networks,” *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pp. 268–273, 2010.
- [34] K. Kwon, S. H. Kim, M. Ha, and D. Kim, “Traffic-aware stateless multipath routing for fault-tolerance in IEEE 802.15.4 wireless mesh networks,” *Wireless Networks*, vol. 24, no. 5, pp. 1755–1774, 2018.

- [35] W. Xie, M. Goyal, H. Hosseini, J. Martocci, Y. Bashir, E. Baccelli, and A. Durresi, “A performance analysis of Point-to-Point routing along a directed acyclic graph in low power and lossy networks,” *Proceedings - 13th International Conference on Network-Based Information Systems, NBIS 2010*, pp. 111–116, 2010.
- [36] E. Ancillotti, R. Bruno, and M. Conti, “The role of the RPL routing protocol for smart grid communications,” *IEEE Communications Magazine*, vol. 51, no. 1, pp. 75–83, 2013.
- [37] M. Zhao, A. Kumar, P. H. Joo Chong, and R. Lu, “A comprehensive study of RPL and P2P-RPL routing protocols: Implementation, challenges and opportunities,” *Peer-to-Peer Networking and Applications*, vol. 10, no. 5, pp. 1232–1256, 2017.
- [38] P. O. Kamgueu, E. Nataf, T. D. Ndie, P. O. Kamgueu, E. Nataf, T. Djotio, and N. Survey, “Survey on RPL enhancements : a focus on topology , security and mobility,” 2018.
- [39] B. Ghaleb, A. Y. Al-Dubai, E. Ekonomou, A. Alsarhan, Y. Nasser, L. M. Mackenzie, and A. Boukerche, “A Survey of Limitations and Enhancements of the IPv6 Routing Protocol for Low-Power and Lossy Networks: A Focus on Core Operations,” *IEEE Communications Surveys and Tutorials*, vol. 21, no. 2, pp. 1607–1635, 2019.
- [40] Y. Tahir, S. Yang, and J. McCann, “BRPL: Backpressure RPL for High-throughput and Mobile IoTs,” *IEEE Transactions on Mobile Computing*, vol. 17,

no. 1, pp. 1–1, 2017.

- [41] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, p. 2347_2376, 2015.
- [42] H. Xu, W. Yu, D. Griffith, and N. Golmie, “A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective,” *IEEE Access*, vol. 6, pp. 78 238–78 259, 2018.
- [43] S. Cirani, L. Davoli, G. Ferrari, R. Leone, P. Medagliani, M. Picone, and L. Veltri, “A scalable and self-configuring architecture for service discovery in the internet of things,” *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 508–521, 2014.
- [44] A. Dunkels, B. Gronvall, and T. Voigt, “Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors,” *IEEE EMNETS, Tampa, Florida, USA*, 2004.
- [45] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, “Cross-Level Sensor Network Simulation with COOJA,” *Proceedings of the 31st IEEE Conference on Local Computer Networks (LCN), Tampa, FL, USA*, 2006.
- [46] S. Cheshire and M. Krochmal, “DNS-Based Service Discovery, RFC,” 2011.
- [47] S. Thomson, T. Narten, and T. Jinmei, “IPv6 Stateless Address Autoconfiguration, RFC 4862,” 2007.

- [48] The Avahi team, “Avahi-daemon.” [Online]. Available: <https://www.avahi.org/doxygen/html/>
- [49] A. van Hoff, “jmdNS.” [Online]. Available: <http://jmdns.sourceforge.net/>
- [50] G. Kaindl, “Bonjour/Zeroconf with Arduino.” [Online]. Available: <https://developer.apple.com/bonjour/>
- [51] R. Klauck and M. Kirsche, “Bonjour Contiki: A case study of a DNS-based discovery service for the internet of things,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7363 LNCS, pp. 316–329, 2012.
- [52] “Overview of Existing Routing Protocols for Low Power and Lossy Networks,” *IETF Internet-draft*, pp. 1–26, 2009.
- [53] X. Liu, Z. Sheng, C. Yin, F. Ali, and D. Roggen, “Performance Analysis of Routing Protocol for Low Power and Lossy Networks (RPL) in Large Scale Networks,” *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2172–2185, 2017.
- [54] J. Tripathi, J. C. de Oliveira, and J. P. Vasseur, “Applicability Study of RPL with Local Repair in Smart Grid Substation Networks,” *2010 First IEEE International Conference on Smart Grid Communications*, pp. 262–267, 2010.
- [55] E. Polytechnique and A. R. Corporation, “The Trickle Algorithm, RFC,” Tech. Rep., 2011.

- [56] J. P. Vasseur, U. C. Berkeley, and V. Manral, “An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL),” *RFC 6554*, pp. 1–13, 2012.
- [57] B. Peres, B. P. Santos, O. A. Otavio, O. Goussevskaia, M. A. Marcos, L. F. Luiz, and A. A. Antonio, “Matrix: Multihop Address allocation and dynamic any-To-any Routing for 6LoWPAN,” *Computer Networks*, vol. 140, pp. 28–40, 2018.
- [58] H. Kermajani, C. Gomez, and M. H. Arshad, “Modeling the message count of the trickle algorithm in a steady-state, static wireless sensor network,” *IEEE Communications Letters*, vol. 16, no. 12, pp. 1960–1963, 2012.
- [59] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, “Operating Systems for Low-End Devices in the Internet of Things: a Survey,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 1–1, 2015.
- [60] K. Kritsis, G. Z. Papadopoulos, A. Gallais, P. Chatzimisios, and F. Theoleyre, “A tutorial on performance evaluation and validation methodology for low-power and lossy networks,” *IEEE Communications Surveys and Tutorials*, vol. 20, no. 3, pp. 1799–1825, 2018.
- [61] Y. B. Zikria, M. K. Afzal, F. Ishmanov, S. W. Kim, and H. Yu, “A survey on routing protocols supported by the Contiki Internet of things operating system,” *Future Generation Computer Systems*, vol. 82, pp. 200–219, 2018.
- [Online]. Available: <https://doi.org/10.1016/j.future.2017.12.045>

- [62] G. Z. Papadopoulos, A. Gallais, G. Schreiner, and T. Noël, “Importance of Repeatable Setups for Reproducible Experimental Results in IoT,” in *Proceedings of the 13th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, ser. PE-WASUN '16. New York, NY, USA: ACM, 2016, pp. 51–59.
- [63] Thingsquare, “Why Choose Contiki.” [Online]. Available: <http://www.contiki-os.org/>
- [64] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt, “Msp430sim - an extensible simulator for msp430-equipped sensor boards.” Computer Systems Laboratory, SICS, 2007.
- [65] S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, P. Levis, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, “TinyOS: An Operating System for Sensor Networks,” *Ambient Intelligence*, pp. 115–148, 2005.
- [66] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, “Software-based on-line energy estimation for sensor nodes,” *Proceedings of the 4th workshop on Embedded networked sensors - EmNets '07*, p. 28, 2007.
- [67] T. INSTRUMENT, “MSP430F543x and MSP430F541x Mixed-Signal Microcontrollers.” [Online]. Available: <http://www.ti.com/lit/ds/symlink/msp430f5437.pdf>
- [68] T. A. Butt, I. Phillips, L. Guan, and G. Oikonomou, “Adaptive and context-aware service discovery for the internet of things,” *Lecture Notes in Computer*

- Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8121 LNCS, pp. 36–47, 2013.
- [69] T. Özcelebi, J. Lukkien, R. Bosman, and Ö. Uzun, “Discovery, monitoring and management in smart spaces composed of low capacity nodes,” *IEEE Transactions on Consumer Electronics*, vol. 56, no. 2, pp. 570–578, 2010.
- [70] M. Woolley, “Bluetooth Special Interest Group, Core Specification Version 5.2 Feature Overview,” no. January, pp. 1–36, 2019.
- [71] M. Nidd, “Service discovery in DEAPspace,” *IEEE Personal Communications*, vol. 8, no. 4, pp. 39–45, 2001.
- [72] S. Helal, N. Desai, V. Verma, and Choonhwa Lee, “Konark - a service discovery and delivery protocol for ad-hoc networks,” in *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, vol. 3, 2003, pp. 2107–2113 vol.3.
- [73] J. Buford, B. Burg, E. Celebi, and P. Frankl, “Sleeper: A Power-Conserving Service Discovery Protocol,” in *2006 3rd Annual International Conference on Mobile and Ubiquitous Systems - Workshops*, 2006, pp. 1–9.
- [74] B. Djamaa, M. Richardson, N. Aouf, and B. Walters, “Towards efficient distributed service discovery in low-power and lossy networks,” *Wireless Networks*, vol. 20, no. 8, p. 2437_2453, 2014.
- [75] R. Marin Perianu, P. H. Hartel, and J. Scholten, *A Classification of Service Discovery Protocols*, ser. CTIT Technical Report Series. Netherlands: Centre for Telematics and Information Technology (CTIT), 2005, no. 05-25.

- [76] F. Zhu, M. W. Mutka, and L. M. Ni, “Service discovery in pervasive computing environments,” *IEEE Pervasive Computing*, vol. 4, no. 4, pp. 81–90, 2005.
- [77] G. Schiele, C. Becker, and K. Rothermel, “Energy-Efficient Cluster-Based Service Discovery for Ubiquitous Computing,” in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*, ser. EW 11. New York, NY, USA: Association for Computing Machinery, 2004, pp. 14–es.
- [78] F. M. Anwar, M. T. Raza, S. Yoo, and K. Kim, “ENUM Based Service Discovery Architecture for 6LoWPAN,” in *2010 IEEE Wireless Communication and Networking Conference*, 2010, pp. 1–6.
- [79] A. J. Jara, S. Varakliotis, A. F. Skarmeta, and P. Kirstein, “Extending the Internet of things to the future Internet through IPv6 support,” *Mobile Information Systems*, vol. 10, no. 1, pp. 3–17, 2014.
- [80] A. J. Jara, P. Lopez, D. Fernandez, J. F. Castillo, M. A. Zamora, and A. F. Skarmeta, “Mobile digcovery: Discovering and interacting with the world through the Internet of things,” *Personal and Ubiquitous Computing*, vol. 18, no. 2, pp. 323–338, 2014.
- [81] A. Ismail and W. Kastner, “Discovery in SOA-governed industrial middleware with mDNS and DNS-SD,” *IEEE International Conference on Emerging Technologies and Factory Automation*, 2016.

- [82] J. He, Y. Zhang, and X. Yuan, “MDNS based automatic discovery method in optical NMS,” *ICOON 2017 - 16th International Conference on Optical Communications and Networks*, pp. 1–3, 2017.
- [83] R. Kurte, Z. Salcic, and K. I.-K. Wang, “A Distributed Service Framework for the Internet of Things,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4166–4176, 2019.
- [84] S. Cheshire, “Hybrid Unicast/Multicast DNS-Based Service Discovery, IETF draft,” pp. 1–31, 2017.
- [85] R. Klauck, “Seamless integration of smart objects into the internet using XMPP and mDNS/DNS-SD,” Ph.D. dissertation, 2016.
- [86] N. Buchina, “Extending service discovery protocols with support for context information,” Ph.D. dissertation, 2014.
- [87] S. Srinivasan, “Improving Content Delivery and Service Discovery in Networks,” Ph.D. dissertation, 2016.
- [88] A. J. Jara, P. Martinez-Julia, and A. Skarmeta, “Light-weight multicast DNS and DNS-SD (lmDNS-SD): IPv6-based resource and service discovery for the web of things,” *Proceeding of 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 731–738, 2012.
- [89] M. Antonini, S. Cirani, G. Ferrari, P. Medagliani, M. Picone, and L. Veltri, “Lightweight multicast forwarding for service discovery in low-power IoT net-

- works,” *proceeding of 22nd International Conference on Software, Telecommunications and Computer Networks*, pp. 133–138, 2014.
- [90] D. Kaiser and M. Waldvogel, “Efficient privacy preserving multicast DNS service discovery,” *IEEE International Conference on High Performance Computing and Communications*, pp. 1229–1236, 2014.
- [91] L. Kerry and S. Don, “Extended Multicast DNS, Internet-Draft,” 2013.
- [92] R. Klauck and M. Kirsche, “Enhanced DNS Message Compression - Optimizing mDNS / DNS-SD for the Use in 6LoWPANs,” no. March, pp. 1–6, 2013.
- [93] M. Stolikj, P. J. L. Cuijpers, J. J. Lukkien, and N. Buchina, “Context based service discovery in unmanaged networks using mDNS/DNS-SD,” *IEEE International Conference on Consumer Electronics (ICCE)*, pp. 163–165, 2016.
- [94] M. Stolikj, R. Verhoeven, P. J. L. Cuijpers, and J. J. Lukkien, “Proxy support for service discovery using mDNS / DNS-SD in low power networks,” *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, Sydney, NSW*, pp. 1–6, 2014.
- [95] Z. Shelby, “Constrained RESTful Environments (CoRE) Link Format, RFC 6690,” 2012.
- [96] C. H. Barriquello, G. W. Denardin, and A. Campos, “A geographic routing approach for IPv6 in large-scale low-power and lossy networks,” *Computers and Electrical Engineering*, vol. 45, pp. 182–191, 2015.

- [97] F. Kuhn, R. Wattenhofer, and A. Zollinger, “Worst-Case optimal and average-case efficient geometric ad-hoc routing,” p. 267, 2003.
- [98] Y. Mao, F. Wang, L. Qiu, S. S. Lam, and J. M. Smith, “S4: state and small stretch routing protocol for large wireless sensor networks,” *Proceedings of the 4th*, vol. 18, no. 3, pp. 761–774, 2007.
- [99] R. Fonseca, S. Ratnasamy, J. Zhao, and C. T. Ee, “Beacon Vector Routing: Scalable Point-to-Point Routing in Wireless Sensornets,” *Proceedings of the 2nd conference on Networked Systems Design & Implementation*, pp. 329–342, 2005.
- [100] A. Reinhardt, O. Morar, S. Santini, S. Zöller, and R. Steinmetz, “CBFR: Bloom filter routing with gradual forgetting for tree-structured wireless sensor networks with mobile nodes,” *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2012 - Digital Proceedings*, 2012.
- [101] S. Duquennoy, O. Landsiedel, and T. Voigt, “Let the tree bloom: scalable opportunistic routing with orpl,” *SenSys, ACM*, pp. 2–14, 2013.
- [102] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, “Collection Tree Protocol,” in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys’09)*, 2009.
- [103] K. H. Kim, W. Y. Seung, D. P. Soohong, H. L. Jae, and M. Geoff, “Hierarchical Routing over 6LoWPAN (HiLow),” *Internet-Draft , IETF*, 2007.

- [104] K. Kwon, M. Ha, T. Kim, S. H. Kim, and D. Kim, “The stateless point to point routing protocol based on shortcut tree routing algorithm for IP-WSN,” *Proceedings of 2012 International Conference on the Internet of Things, IOT 2012*, pp. 167–174, 2012.
- [105] B. S. Peres and O. Goussevskaja, “IPv6 Multihop Host Configuration for Low-Power Wireless Networks,” *Proceedings - 33rd Brazilian Symposium on Computer Networks and Distributed Systems, SBRC 2015*, pp. 189–198, 2015.
- [106] B. Peres, B. P. Santos, O. A. Otavio, O. Goussevskaja, M. A. Marcos, L. F. Luiz, and A. A. Antonio, “Matrix: Multihop Address allocation and dynamic any-To-any Routing for 6LoWPAN,” *Computer Networks*, vol. 140, pp. 28–40, 2018. [Online]. Available: <https://doi.org/10.1016/j.comnet.2018.04.017>
- [107] B. P. Santos, O. Goussevskaja, L. F. Vieira, M. A. Vieira, and A. A. Loureiro, “Mobile Matrix: Routing under mobility in IoT, IoMT, and Social IoT,” *Ad Hoc Networks*, vol. 78, pp. 84–98, 2018. [Online]. Available: <https://doi.org/10.1016/j.adhoc.2018.05.012>
- [108] S. Debroy, P. Samanta, A. Bashir, and M. Chatterjee, “SpEED-IoT: Spectrum aware energy efficient routing for device-to-device IoT communication,” *Future Generation Computer Systems*, vol. 93, pp. 833–848, 2019.
- [109] E. Baccelli, M. Philipp, and M. Goyal, “The P2P-RPL Routing Protocol for IPv6 Sensor Networks: Testbed Experiments,” *SoftCOM 2011, 19th International*

- Conference on Software, Telecommunications and Computer Networks*, no. April 2016, pp. 1–6, 2011.
- [110] M. O. Farooq and D. Pesch, “ERPL: An Enhanced Peer-to-Peer Routing Mechanism for Low-Power and Lossy Networks,” *2018 11Th Ifip Wireless and Mobile Networking Conference (Wmnc)*, pp. 1–8, 2018.
- [111] M. Farooq and D. Pesch, “Reduced overhead routing in short-range low-power and lossy wireless networks,” *Sensors*, vol. 19, no. 5, pp. 1–19, 2019.
- [112] M. Zhao, G. G. N. Ali, R. Lu, P. Han, and J. Chong, “An Energy-Efficient and Self-regioning based RPL for Low-power and Lossy Networks,” in *Invited Paper*, 2016.
- [113] B. P. Santos, M. A. Vieira, and L. F. Vieira, “EXtend collection tree protocol,” *2015 IEEE Wireless Communications and Networking Conference, WCNC 2015*, pp. 1512–1517, 2015.
- [114] I. Pefkianakis, S. H. Wong, and S. Lu, “SAMER: Spectrum aware mesh routing in cognitive radio networks,” *2008 IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks, DySPAN 2008*, pp. 766–770, 2008.
- [115] D. Rosário, Z. Zhao, A. Santos, T. Braun, and E. Cerqueira, “A beaconless Opportunistic Routing based on a cross-layer approach for efficient video dissemination in mobile multimedia IoT applications,” *Computer Communications*, vol. 45, pp. 21–31, 2014.

- [116] V. Issarny, N. Georgantas, S. Hachem, A. Zarras, P. Vassiliadist, M. Autili, M. A. Gerosa, and A. B. Hamida, "Service-oriented middleware for the Future Internet: State of the art and research directions," *Journal of Internet Services and Applications*, vol. 2, no. 1, pp. 23–45, 2011.
- [117] T. Thiago, S. Hachem, V. Issarny, and N. Georgantas, "Service Oriented Middleware for the Internet of Things : A Perspective," *In Proceedings of the 4th European conference on Towards a service-based internet*, pp. 220–229, 2011.
- [118] G. Bouloukakis, N. Georgantas, B. Billet, G. Bouloukakis, N. Georgantas, and B. B. Revisiting, "Revisiting Service-oriented Architecture for the IoT : A Middleware Perspective," *in proceeding of 14th International Conference on Service Oriented Computing (ICSOC)*,, 2016.
- [119] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-based internet of things: Discovery, query, selection, and on-demand provisioning of web services," *IEEE Transactions on Services Computing*, vol. 3, no. 3, pp. 223–235, 2010.
- [120] K. Kata, Y. Nakayama, and T. Seki, "M2M service platform to support "carrier cloud"," *NEC Technical Journal*, vol. 5, no. 2, pp. 116–121, 2010.
- [121] L. Schor, P. Sommer, and R. Wattenhofer, "Towards a zero-configuration wireless sensor network architecture for smart buildings," *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings - BuildSys '09*, p. 31, 2009.

- [122] H. A. A. Al-Kashoash, H. Kharrufa, Y. Al-Nidawi, and A. H. Kemp, “Congestion control in wireless sensor and 6LoWPAN networks: toward the Internet of Things,” *Wireless Networks*, 2018. [Online]. Available: <https://doi.org/10.1007/s11276-018-1743-y>
- [123] G. Anastasi, M. Conti, M. Di, and A. Passarella, “Energy conservation in wireless sensor networks : A survey,” *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009. [Online]. Available: <http://dx.doi.org/10.1016/j.adhoc.2008.06.003>
- [124] A. Dunkels, “The ContikiMAC Radio Duty Cycling Protocol,” *SICS Technical Report T2011:13*, ISSN 1100-3154, pp. 1–11, 2011.
- [125] L. Davoli, M. Antonini, and G. Ferrari, “DiRPL: A RPL-Based Resource and Service Discovery Algorithm for 6LoWPANs,” *Applied Sciences*, vol. 9, no. 1, p. 33, 2018.
- [126] F. Z. Djiroun and D. Djenouri, “MAC Protocols with Wake-Up Radio for Wireless Sensor Networks: A Review,” *IEEE Communications Surveys and Tutorials*, vol. 19, no. 1, pp. 587–618, 2017.
- [127] R. Jain, “The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling,” *SIGMETRICS Performance Evaluation Review*, 1990.
- [128] W. Kang and J. Kim, “PDDS: Scalable Sensor Data Distribution for Cyber-Physical Systems Using GPGPUs,” *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2025–2036, 2018.

- [129] N. Zhang, “Smart logistics path for cyber-physical systems with internet of things,” *IEEE Access*, vol. 6, pp. 70 808–70 819, 2018.
- [130] F. Ren, T. He, S. K. Das, and C. Lin, “Traffic-aware dynamic routing to alleviate congestion in wireless sensor networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 9, pp. 1585–1599, 2011.
- [131] P. Misra, N. Ahmed, D. Ostry, and S. Jha, “Characterization of Asymmetry in Low-Power Wireless Links: An Empirical Study,” in *Distributed Computing and Networking*, M. K. Aguilera, H. Yu, N. H. Vaidya, V. Srinivasan, and R. R. Choudhury, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 340–351.
- [132] A. Liu, X. Jin, G. Cui, and Z. Chen, “Deployment guidelines for achieving maximum lifetime and avoiding energy holes in sensor network,” *Information Sciences*, vol. 230, pp. 197–226, 2013. [Online]. Available: <http://dx.doi.org/10.1016/j.ins.2012.12.037>
- [133] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele, “pTUNES: Runtime parameter adaptation for low-power MAC protocols,” in *2012 ACM/IEEE 11th International Conference on Information Processing in Sensor Networks (IPSN)*, 2012, pp. 173–184.
- [134] A. N. Kim, F. Hekland, S. Petersen, and P. Doyle, “When HART goes wireless: Understanding and implementing the WirelessHART standard,” *IEEE Interna-*

- tional Conference on Emerging Technologies and Factory Automation, ETFA*, pp. 899–907, 2008.
- [135] P. Corbalán, R. Marfievici, V. Cionca, D. O’Shea, and D. Pesch, “Into the SMOG: The Stepping Stone to Centralized WSN Control,” *Proceedings - 2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2016*, pp. 118–126, 2017.
- [136] J. Tripathi and J. C. De Oliveira, “On adaptive timers for improved RPL operation in low-power and lossy sensor networks,” *2013 5th International Conference on Communication Systems and Networks, COMSNETS 2013*, pp. 1–10, 2013.
- [137] E. Baccelli, C. Gundogan, O. Hahm, P. Kietzmann, M. S. Lenders, H. Petersen, K. Schleiser, T. C. Schmidt, and M. Wahlisch, “RIOT: an Open Source Operating System for Low-end Embedded Devices in the IoT,” *IEEE Internet of Things Journal*, vol. 4662, no. c, pp. 1–12, 2018.
- [138] M. Mahyoub, “HRPL implementation,” 2020. [Online]. Available: <https://github.com/Mahyub/HRPL>
- [139] Y. Zhu, C. Xue, H. Cai, J. Yu, L. Ni, M. Li, and B. Li, “On deploying relays for connected indoor sensor networks,” *Journal of Communications and Networks*, vol. 16, no. 3, pp. 335–343, 2014.
- [140] J. Hui and T. Pascal, “Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks,” *IETF RFC 4944*, 2011.

- [141] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959. [Online]. Available: <https://doi.org/10.1007/BF01386390>
- [142] M. R. Akhavan, A. Aijaz, S. Choobkar, and A.-H. Aghvami, “On the multi-hop performance of receiver based MAC protocol in routing protocol for low-power and lossy networks-based low power and lossy wireless sensor networks,” *IET Wireless Sensor Systems*, vol. 5, no. 1, pp. 42–49, 2015.
- [143] H. A. A. Al-Kashoash, H. M. Amer, L. Mihaylova, and A. H. Kemp, “Optimization-Based Hybrid Congestion Alleviation for 6LoWPAN Networks,” *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2070–2081, 2017.
- [144] L. Phillips, *Gnuplot Cookbook*, 2015.
- [145] R. Kelsey, “Multicast Protocol for Low-Power and Lossy Networks (MPL),” *RFC 7731*, pp. 1–29, 2016.
- [146] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks,” *Proceedings of the First Symposium on Networked Systems Design and Implementation*, pp. 15–28, 2004.

VITAE

- **Name:** Mohammed Ayedh Ahmed Mahyoub
- **Nationality:** Yemeni
- **Date of Birth:** 01.02.1984
- **Email:** *m.mahyoub84@gmail.com*
- **Permanent Address:** Sana'a - Yemen

- **EDUCATION**

2014 - 2020	Ph.D. , Computer Engineering, (April, 2020) Faculty of Computer Science and Engineering King Fahd University of Petroleum and Minerals (KFUPM), KSA Thesis Title : "Efficient Discovery Mechanisms for IoT"
2010 - 2013	M.Sc. , Computer and Control Engineering, (April, 2013) Faculty of Engineering, Assiut University, Egypt Thesis Title: "A Generic Scalable Framework based on P2P for Resource Discovery in the Grid Computing"
2005 - 2009	B.Sc. , Computer Engineering, (March, 2009) Faculty of Engineering, Gezira University, Sudan
2003	Secondary School , Scientific Major Alrawdah School, Taiz, Yemen

- **RESEARCH INTEREST**

My main research interests are networking and distributed computing. My work involves both analysing and enhancing the performance of WSN protocols. Specifically, most of my previous studies targeted wireless networks of constrained resources such as IoT networks. The key question that drives my research is, how to efficiently utilize these constrained resources to meet the application's requirements in robust and reliable manners. The main thrust of my work during my Ph.D. program is developing enhanced versions of resource and routing discovery protocols for IoT-based networks. Improving the performance of such protocols requires a strong knowledge spanning different layers of IoT protocols stack. This fact led me to gain a strong background and great interest in MAC, radio duty cycle, and physical layer protocols.

- **RESEARCH APPROACH**

- **Collaborative Research** is something I am passionate about as it invites different perspectives and ideas, helping to address problems from different angles and resulting in high-quality research.
- **Iterative Design:** Designing efficient protocols for large-scale IoT-based systems are often complex, and it is challenging to predict all the possible design issues. The iterative design methodology is very useful in such cases. With this strategy, I begin with an initial design, and subsequently, I im-

plement, or build, and evaluate the protocol to identify design shortcomings and performance bottlenecks. An implementation often uncovers hidden assumptions and the convention of “plan to throw one away” design principle is often very helpful. This process of design-build-evaluate is iterated in order to refine the built protocol.

- **COMPUTER AND NETWORKING SKILLS**

- **Networking:** Extensive knowledge of data communications/networking, routing protocols, TCP/IP, LANs, WANs, WSN, and IoT.
- **Simulation and emulation tools:** CloudSim, TOSSIM, TinyOS, ContikiOS, and COOJA.
- **Data Analytics and Visualization Tools:** GNU PLOT, Matplotlib, and SPSS.
- **Programming Languages:** Matlab, C/C++, Bash Programming, Python, Java, and VHDL (hardware programming language).
- **IDEs:** WinEdit, Eclipse, NetBeans, JDeveloper, and Visual Studio.NET, and TeXstudio.
- **Operating Systems:** Linux, MS-DOS, and Microsoft Windows, and Virtualization of Windows using VirtualBox, and VMWare
- Excellent command of Latex environment.

