CRANFIELD UNIVERSITY

BADIS DJAMAA

Pervasive Service Discovery in Low-power and Lossy Networks

PhD

Academic Year: 2012 - 2015

Supervisor: Professor Mark Richardson

June 2015

CRANFIELD UNIVERSITY

PhD

BADIS DJAMAA

Pervasive Service Discovery in Low-power and Lossy Networks

Academic Year 2012 - 2015

Supervisor: Professor Mark Richardson

June 2015

ABSTRACT

Pervasive Service Discovery (SD) in Low-power and Lossy Networks (LLNs) is expected to play a major role in realising the Internet of Things (IoT) vision. Such a vision aims to expand the current Internet to interconnect billions of miniature smart objects that sense and act on our surroundings in a way that will revolutionise the future. The pervasiveness and heterogeneity of such low-power devices requires robust, automatic, interoperable and scalable deployment and operability solutions. At the same time, the limitations of such constrained devices impose strict challenges regarding complexity, energy consumption, time-efficiency and mobility.

This research contributes new lightweight solutions to facilitate automatic deployment and operability of LLNs. It mainly tackles the aforementioned challenges through the proposition of novel component-based, automatic and efficient SD solutions that ensure extensibility and adaptability to various LLN environments. Building upon such architecture, a first fully-distributed, hybrid push-pull SD solution dubbed EADP (Extensible Adaptable Discovery Protocol) is proposed based on the well-known Trickle algorithm. Motivated by EADPs' achievements, new methods to optimise Trickle are introduced. Such methods allow Trickle to encompass a wide range of algorithms and extend its usage to new application domains. One of the new applications is concretized in the TrickleSD protocol aiming to build automatic, reliable, scalable, and time-efficient SD. To optimise the energy efficiency of TrickleSD, two mechanisms improving broadcast communication in LLNs are proposed. Finally, interoperable standards-based SD in the IoT is demonstrated, and methods combining zero-configuration operations with infrastructure-based solutions are proposed.

Experimental evaluations of the above contributions reveal that it is possible to achieve automatic, cost-effective, time-efficient, lightweight, and interoperable SD in LLNs. These achievements open novel perspectives for zero-configuration capabilities in the IoT and promise to bring the 'things' to all people everywhere.

ACKNOWLEDGMENTS

This PhD could not have been possible without the help and support of many professors, peers and loved people.

Foremost, I would like to express my deepest gratitude to my beloved parents for their support, love, prayers, patience, and sacrifices. I owe them all the success I make in my life. I would like also to extend my sincere thanks to the members of my family for their support and encouragements.

A big thank you is given to my supervisor Professor Mark Richardson for supervising this research. I am particularly very grateful to the freedom of exploration, support, and encouragements he gave. I learned from him that enthusiasm, perseverance, and patience are keys to success.

I would like to thank all the people who served on my thesis committees over the three-year period at Cranfield. Special thanks are due to Dr. Peter Barker and Dr. Bob Walters for offering their continuous support and help.

I am also thankful to my friends and colleagues who supported me throughout the up and down moments. I want to single out Piers Maclean for his constant support, encouragements and the Arabic lessons we had. Piers: Thank you.

My thanks are also extended to my office mates: Krasin, Jaime, Nasyitah, Alex and Luc, who were been fantastic. Special thanks are expressed to Krasin for being always there willing to discuss my ideas and provide help. Additionally, I would like to thank the lunch-group: Ioannis Vagias, Drs. Adam Zagorecki and Alessio Balleri, and the team at Barrington library: Mandy Smith, Lynne Seddon and Jessica Goff, who were always there ready to help. I am also grateful to the support I received from Amanda Marlow, my English teacher Keith Price and his wife Lorraine.

I wish also to extend my heartfelt thanks to my colleagues and friends back in Algeria, particularly Drs. Mohamed Aissani, Ali Yashir, Saliha Aouat and future Dr. Walid Charifi for their encouragements and support. My sincere thanks are also expressed to my friends Hillal, Ali and Rahim for the best moments we shared.

Finally, I want to thank all anonymous journal and conference reviewers for their insightful comments, and the people at the Contiki and IETF mailing lists for continuous discussions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

6Lo	IPv6 over Networks of Resource-constrained Nodes
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
6TiSCH	IPv6 over the TSCH mode of IEEE 802.15.4e
ACE	Authentication and Authorization for Constrained Environments
BLE	Bluetooth Low Energy
BLIP	Berkeley Low-power IP
CBOR	Concise Binary Object Representation
CBR	Constant Bit Rate
CCA	Clear Channel Assessment
CCI	Channel Check Interval
CNN	Constrained-Node Network
CoRE	Constrained RESTful Environments
CSL	Coordinated Sampled Listening
DA	Directory Agent
DAD	Duplicate Address Detection
DECT-ULE	Digital Enhanced Cordless Telecommunications-Ultra Low Energy
DHT	Distributed Hash Table
DICE	DTLS In Constrained Environments
DNS	Domaine Name System
DNSSD	Extensions for Scalable DNS Service Discovery
DNS-SD	DNS-based Service Discovery
DODAG	Destination Oriented Directed Acyclic Graph
DPA	Directory Proxy Agent
DPWS	Devices Profile for Web Services
DRD	Distributed Resource Directory
DSR	Dynamic Source Routing
DTLS	Datagram Transport Layer Security
EADP	Extensible Adaptable Discovery Protocol
ENUM	Electronic Number Mapping

EUI	Extended Unique Identifier
FFD	Full Function Device
GL	Group Leader
GM	Group Member
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPSO	IP for Smart Objects
JSON	JavaScript Object Notation
LFU	Least Frequently Used
LLN	Low-power and Lossy Networks
LPL	Low-Power Listening
LPP	Low-Power Probing
LQI	Link Quality Indicator
LWIG	Light-Weight Implementation Guidance
MAC	Medium Access Control
MANET	Mobile Ad-hoc Networks
MCU	Micro Controller Unit
mDNS	Multicast DNS
MPL	Multicast Protocol for LLN
MS/TP	Master-Slave/Token-Passing
MTU	Maximum Transmission Unit
NCE	Neighbour Cache Entries
ND	Neighbour Discovery
NFC	Near Field Communication
P2P	Peer to Peer
PAN	Personal Area Network
PLC	Power Line Communication
RA	Reply Agent
RD	Resource Directory

RDC	Radio Duty Cycling
REST	Representational State Transfer
RFD	Reduced Function Device
RFID	Radio Frequency Identification
RIT	Receiver Initiated Transmission
RPL	Routing Protocol for LLN
RSSI	Received Signal Strength Indicator
SA	Service Agent
SAaaS	Sensing Actuating as a Service
SAM	Service Advertisement for MANET
SD	Service Discovery
SDP	Service Discovery Protocol
SenML	Sensor Markup Language
SICS	Swedish Institute of Computer Science
SLIM	Service Location and Invocation Middleware
SLP	Service Location Protocol
SSLP	Simple Service Location Protocol
TCP	Transmission Control Protocol
TSCH	Time Slotted Channel Hoping
TTL	Time to Live
UA	User Agent
UDDI	Universal Description Discovery and Integration
UDGM	Unit Disk Graph Medium
UDP	User Datagram Protocol
uPnP	Universal Plug and Play
URI	Uniform Resource Identifier
WS-DD	Web Service Dynamic Discovery
WSN	Wireless Sensor Network
xmDNS	Extended Multicast DNS

RELEVANT PUBLICATIONS

Below is a list of relevant publications made during this research

Journal Publications:

1. **Badis Djamaa**, Mark Richardson, Nabil Aouf and Bob Walters: Towards Efficient Distributed Service Discovery in Low-power and Lossy Networks. *Springer Wireless Networks, vol. 20, no. 8, pp. 2437–2453, Nov. 2014.*
2. **Badis Djamaa** and Mark Richardson: Optimizing the Trickle Algorithm. *IEEE Communication Letters, vol. 19, no. 5, pp. 819–822, May 2015.*
3. **Badis Djamaa** and Mark Richardson: The Trickle Algorithm: Issues and Solutions. *Elsevier Computer Networks Journal (under review).*

Peer-reviewed Conference, Workshop and Book-chapter Publications:

1. **Badis Djamaa** and Rob Witty: An Efficient Service Discovery Protocol for 6LoWPANs. *IEEE/SAI Science and Information Conference (SAI), London, 2013, pp. 645–652.*
2. **Badis Djamaa**, Nabil Aouf, Mark Richardson, and Bob Walters: Enhancing Delay-based Packet Forwarding Schemes in Wireless Sensor Networks. *4th Annual International Conference on Energy Aware Computing Systems and Applications (ICEAC), 2013, pp. 12–17.*
3. **Badis Djamaa**, Mark Richardson, Nabil Aouf, and Bob Walters: Service Discovery in 6LoWPANs: Classification and Challenges. *8th IEEE International Symposium on Service Oriented System Engineering (SOSE), 2014, pp. 160–161.*
4. **Badis Djamaa**, Mark Richardson, Nabil Aouf, and Bob Walters: Unicast/multicast Performance in Single-hop Duty-cycled 6LoWPAN Networks. *9th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), 2014, pp. 140–145.*
5. **Badis Djamaa** and Mark Richardson: Towards Scalable DNS-Based Service Discovery for the Internet of Things. *8th Ubiquitous Computing and Ambient Intelligence Conference (UCAmI): Personalisation and User Adapted Services, Springer LNCS, 2014, pp. 432–435.*
6. **Badis Djamaa**, Mark Richardson, Peter Barker, and Mohamed Aissani: Multicast Burst Forwarding in Constrained Networks. *81st IEEE Vehicular Technology Conference , 2015 (to appear)*
7. **Badis Djamaa**, Mark Richardson, Peter Barker, and Ian Owens: Discovery of Things: A Fully-Distributed Opportunistic Approach. *81st IEEE Vehicular Technology Conference, 2015 (to appear)*

Reports:

1. **Badis Djamaa** and Mark Richardson: The Trickle Algorithm: Issues and Solutions -Cranfield University. Available Online: https://dspace.lib.cranfield.ac.uk/handle/1826/9116.

Chapter 1

Introduction and Problem Statement

Wireless Sensor Networks (WSNs) have proven of significant use in multiple application domains over the last decade. Such Low-power and Lossy Networks (LLNs), traditionally deployed as isolated proprietary systems, are evolving to be one of the main pillars of ubiquitous computing [1] and an essential building block of the emerging Internet of Things (IoT) [2]. In this vision, low-power WSNs are no longer seen as isolated systems running proprietary protocols. Instead, they are considered as a central part of the IoT architecture, which integrates a multitude of devices including sensors, actuators, computers and smartphones. Indeed, an important portion of the projected billions of devices connected to the future Internet are expected to be low-power [3]. In this context, new applications are envisaged introducing many challenges to the research community. Interoperability, power consumption, mobility support, end-to-end networking and security are among the most prominent ones.

While energy challenges can be addressed via LLN-specific Radio Duty Cycling (RDC) mechanisms, achieving interoperable operations in the IoT requires a technology that allows seamless integration of such heterogeneous systems. To this end, the Internet Protocol (IP) promises to agnostically combine multiple heterogeneous systems with the assurance of stable and proven networking designs that can evolve over time without breaking backward compatibility. These features drove a movement towards all-IP networks where all internetworking should be done via IP (*everything over IP*) and IP should run on low-layer constrained networking technologies such as the IEEE 802.15.4 standard [4] (*IP over everything*). However, IP was not designed with LLN constraints in mind. Thus, while the latest IP standard (IPv6) [5] requires a minimum packet size of 1280 bytes, LLNs operating over the IEEE 802.15.4 standard can only support a maximum frame size of 127 bytes; more than 10 times less than what IPv6 requires. These conflicting requirements led the Internet Engineering Task Force (IETF), the organisation responsible for developing and maintaining Internet standards, to charter a working group in order to investigate IP feasibility over LLNs. The working group delivered its results in 2007 by proposing the IPv6 over Low-power Wireless Personal Area Network (6LoWPAN) standard [6].

6LoWPAN opened doors for new standardisation efforts in the field and expanded the application domains of LLNs even further. Hence, while WSNs were traditionally considered as static networks, leveraging on IP technologies and IoT applications, a large number of sensors are expected to be mobile. Examples include wearable sensors (e.g., health monitoring devices), sensor-enhanced mobile phones (e.g., smartphones), and smart vehicles (vehicles equipped with sensing devices) [7]. These devices together with many others are expected to create mobile IoT applications such as smart-traffic grids, smart healthcare systems, smart logistics and mobile command, control and collaboration systems. Because of the challenges mobility raises for the connectivity of 6LoWPAN networks and hence on their operability, mobility must be handled along with the whole network stack [8] to build robust WSN-in-motion systems.

On account of the above, the IoT market is expected to grow exponentially by internetworking more than 50 billion heterogeneous devices [9] from a multitude of manufacturers providing an abundance of services. However, the trend towards all-IP networks is only a first step providing network-layer interoperability. Service Oriented Architecture (SOA) [10] is expected to provide the application-layer interoperability and hence promise to realise seamless integration of IoT systems. Indeed, SOA makes it easier to develop flexible, reusable and interoperable applications. Nonetheless, because of its resource consumption, SOA has to date remained relatively neglected in WSNs. However, with the emergence of the IoT, lightweight SOA-based solutions for smart object networking are now being investigated. Hence, the concept of Sensing/Actuating as a Service (SAaaS) [11] is introduced. In such a paradigm, sensor and actuator network capabilities are abstracted as services that can be automatically discovered and used via standard interfaces in order to enable interoperability and ease of use.

In this context, automatic Service Discovery (SD), a fundamental requirement of any service-oriented system, emerges as a major challenge to the usability of such networks.

Indeed, SD is the main enabler of loosely-coupled automatic operations in the IoT as it allows clients (service consumers) to automatically locate suitable services that meet their needs. Thus, it provides means to achieve automatic methods for discovering and accessing available services. Failures at the SD stage would compromise the whole array of benefits afforded by SOA. Indeed, because of its significance to the IoT, SD in 6LoWPAN networks is one of the design requirements of two recent IETF working groups [12], [13]. It is also the primary driver of this thesis, which aims to provide interoperable, lightweight, efficient and automatic SD in 6LoWPAN networks.

1.1 Problem statement

This thesis targets automatic service discovery as a key component in achieving automatic interoperable zero-configuration tasks in the IoT. The problem statement of this thesis is illustrated in Figure 1-1.

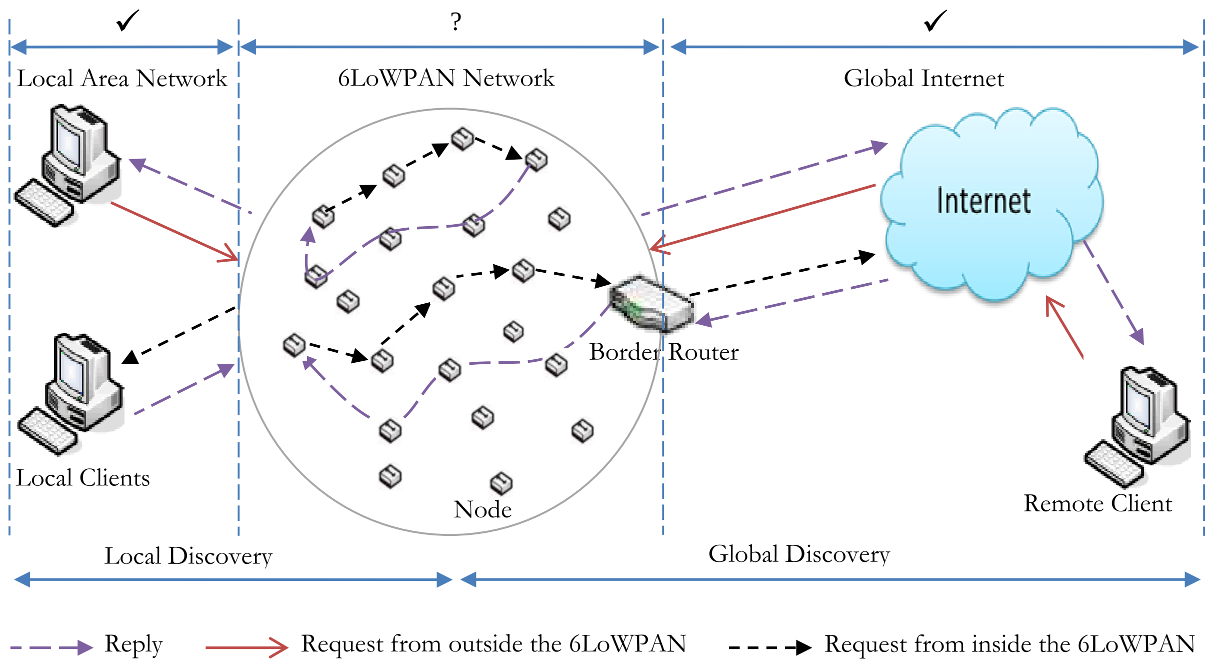


Figure 1-1 Problem statement

This figure shows how SD can allow seamless integration of 6LoWPAN networks with traditional IP networks in the IoT. While service discovery in traditional IP networks has achieved maturity with a plethora of solutions proposed for both local area networks [14]–[17] and the global Internet [18], [19], service discovery in 6LoWPAN networks is still emerging. This is due to the relative newness of the field. Therefore, this research limits itself to investigating SD in 6LoWPAN networks, but without losing sight of interoperability with traditional IP networks for both local and global discovery (Figure 1-1). Such interoperability is discussed in the last chapter.

The 6LoWPAN nodes involved in the generic scenario depicted in Figure 1-1 consist of a set of static and/or mobile devices including sensors, actuators, and gateways towards traditional IP networks. A node in such a system may consist of a physically separated constrained device or be embedded in other devices (laptops, smartphones, etc.), and it may act as a service provider, service consumer or both. Various LLN applications, including ubiquitous healthcare systems (e.g., the global healthcare monitoring system [20]), environmental monitoring (e.g., the farmyard application [21]), smart logistics (e.g., the intelligent container [22]) and home automation systems can fall under the above research scenario.

1.2 Aims and objectives

This thesis aims at providing zero-configuration, plug-and-play capabilities for 6LoWPAN networks in order to succeed the deployment of the above applications. To this end, many challenges have to be addressed including self-configuration, mobility handling, time efficiency, device limitations, scalability and interoperability. To address these challenges, this thesis aims to achieve the following points:

- **Support of automatic, zero-configurable and efficient SD in LLNs**
  To achieve automatic SD, a fully distributed architecture would be preferable. Such architecture can provide self-organisation and self-functioning of the network. At the same time, it can accommodate device constraints in terms of energy consumption, computing and memory demands since it does not require complex algorithms. This, in turn, might enable scalability if it is well managed. However, if such a solution is not well conceived, it may become a burden on the network and the system's energy consumption. Therefore, this thesis aims to provide efficient methods to manage these shortcomings.
- **Response to environmental constraints and user/application requirements**
  Respecting the device and network constraints alone might not provide the quality of service required by the user and the applications. Thus, an SD solution should

be time-efficient and support mobility of nodes while provides reliability. To achieve these characteristics, a fully distributed system might be again preferred as it does not require building and managing any topologies. However, to achieve time efficiency and reliability requirements, new methods have to be designed. Such mechanisms are investigated in this research.

- **Support for seamless integration with other IoT systems**
  Standards-based service descriptions are a critical point in achieving seamless integration of LLNs in the IoT. This research investigates, as a last point, interoperable operations in the IoT based on the adoption and adaptation of standards-based service descriptions in LLNs.

By addressing the above points, this research aims at achieving automatic, cost-effective, time-efficient, reliable, lightweight, and interoperable SD in LLNs. To quantify these aims, the measurable objectives are: low amount of traffic generated in the network, fast discovery times, high discovery and hit success rates as proxy of reliability, low consumed energy and low network radio duty cycle as indicators of efficient energy consumption, and finally discussions about the size of the implementations to demonstrate the lightweight aspect of the proposed solutions.

1.3 Research methodology

Based on the above aims and requirements, an iterative research methodology was adopted in this study. Such a research methodology builds upon 4 main steps namely: (i) identifying the requirements; (ii) researching and developing the solutions; (iii) designing and implementing the outcomes; and (iv) finally, testing and validating the system. Iterative research steps were made during this process.

Guided by the above steps, a comprehensive literature review was carried out in order to identify and extract the requirements of pervasive SD in LLNs. The requirements were formulated, analysed and discussed in Chapter 3. The outcome came to the conclusion that new approaches were needed in order to address the problem. Based on this, new solutions for SD were proposed, relying on the well-known Trickle algorithm [23], [24], which was optimised in the process. These solutions were iteratively improved and evaluated with the guidance of the above requirements.

To validate the work, the proposed solutions were implemented in major WSN operating systems, analysed, and evaluated in both time-accurate simulations and testbed experiments. To this end, different scenarios were developed to gain insights into the performance of the proposed algorithms and how they answer the research question. For instance, three simulation scenarios have been considered in the evaluations throughout the thesis. The first scenario considers a relatively large network of 100 nodes randomly distributed in an area of interest. Such a scenario assesses the performance of the proposed solutions in mobile IoT applications such as emergency response and similar applications. A second scenario considers a medium-sized publicly available network of 31 nodes in order to evaluate the performance of the proposed solutions in home automation systems and similar environments. Finally, a third simulation scenario considers other categories of interesting IoT applications such as street lighting and vehicular networks where node deployment generally follows a line topology pattern.

In addition to the above simulation scenarios, two testbeds are also used as part of the evaluation methodology. For instance a local single-hop testbed was used along with simulations to evaluate the RDC related contributions while a large scale publicly available testbed was used to evaluate the proposed discovery solutions. This evaluation methodology is adopted in order to assess the performance of the proposed solutions in addressing different IoT application needs as suggested in benchmarking for LLN protocols [25]. Details of the operating systems, simulators and testbed tools used in this research are presented in section 2.6 and specific configurations of the simulation and testbed scenarios are discussed at their places in corresponding chapters.

1.4 Contributions: Towards zero-configuration IoT

This thesis demonstrates the feasibility of fully distributed SD as a main enabler for zero-configuration networking in LLNs. To this end, and in addition to the literature review, analysis, and identification of requirements, the main contributions of this research can be summarised as follows:

- **Component-based SD:** The first contribution of this thesis introduces an extensible adaptable discovery solution tailored to 6LoWPAN requirements. An architectural novelty of this solution is the proposition of a component-based architecture for SD in 6LoWPAN networks, which allows for the extensibility and adaptability of the solution depending on a particular IoT scenario. Thus, components can be substituted, added or removed with minimum effects on the architecture and the operability of the solution.

- **EADP:** An Extensible Adaptable Discovery Protocol is then proposed based on the above design (Chapter 4). Besides leveraging a component-based architecture, EADP introduces a new variant of the Trickle algorithm to be used in hybrid SD solutions. EADP aims to minimise network traffic while providing support for LLN requirements such as support for sleepy nodes and group communication.
- **TrickleSD:** Based on the achievements of EADP's Trickle algorithm and building on its component-based architecture, Chapter 5 proposes replacing some of EADP's components using other Trickle variants. To this end, three main contributions are introduced in this chapter. The first proposes a simple, yet powerful optimisation to Trickle. The power of such an optimisation resides in its simplicity and achievements, allowing Trickle to reach new applications and usages in LLNs. Subsequently, other methods to enhance Trickle are proposed (section 5.5.3). Building on these contributions, the remainder of Chapter 5 introduces new algorithms to replace some EADP components using optimised Trickle and thereby contributing a new discovery solution dubbed TrickleSD.
- **Radio duty cycling:** The above contributions take advantage of the broadcast nature of the wireless channel to achieve efficient cooperative SD tasks. Indeed, without broadcast, zero-configuration operations would have been impossible. However, broadcast in duty-cycled 6LoWPAN networks might be resource expensive. Chapter 6, therefore, provides a comprehensive analysis of broadcast and unicast communication patterns in duty-cycled networks. Subsequently two contributions are introduced to enhance the performance of broadcast. The first addresses the problem of transmitting broadcast bulk data over RDCs while the second introduces a generic solution to an inherent problem encountered in one of the most commonly deployed class of RDC protocols.
- **Interoperability:** Another main contribution of this thesis is the integration of EADP and TrickleSD with two standards-based service description formats

designed to foster integration of LLNs in the IoT. Based on the particularities of each format, Chapter 7 proposes proof-of-concept integrations and takes advantage of such integrations to substitute some of EADP and TrickleSD mechanisms using unicast to remedy broadcast inefficiencies in duty-cycled networks. This contribution aims to complete the picture depicted in Figure 1-1.

Finally, it should be noted that the mechanisms and protocols developed in this thesis are generic enough to be applied separately or collectively to different problems in the field.

1.5 Thesis outline

The structure of this thesis is depicted in Figure 1-2. Our journey begins in Chapter 2 with a description of ubiquitous sensor networks, their characteristics and limitations before tackling the emerging trend towards all-IP networking in WSNs. The 6LoWPAN standard along with the resulting standardisation efforts, relevant to this thesis, are then presented. This chapter ends by describing the tools used in this research project. Chapter 3 assess the feasibility of service-oriented architectures in 6LoWPAN networks through service discovery challenges. In this context, a systematic review of state-of-the-art Service Discovery Protocols (SDPs) in the IoT is carried out, with a particular focus on SD in 6LoWPAN networks. Requirements of efficient SD in 6LoWPAN are identified at the end of the chapter.

Following the conclusions of Chapter 3, Chapter 4 proposes EADP; an Extensible, Adaptable Discovery Protocol for 6LoWPAN networks. EADP contributes a new Trickle variant along with many interesting mechanisms detailed and discussed in Chapter 4. Its performance is formally analysed and extensively evaluated in the same chapter. Such evaluations showed many attractive features but also revealed some drawbacks that are addressed in the subsequent chapter. Building on the earlier work, Chapter 5 introduces three main contributions. It primarily proposes two methods for optimising the well-known Trickle algorithm to expand its reach and allow it to remedy EADP's drawbacks. Subsequently, a new SDP (TrickleSD) is proposed. The optimisations along with TrickleSD are thoroughly analysed, evaluated and discussed in the same chapter.

Being based on broadcast communications, the above solutions could suffer inefficiencies in radio duty-cycled networks. To respond to this, Chapter 6 is set apart to investigate the performance of broadcast communication in such networks. In the process, two mechanisms are proposed to enhance broadcast communication under radio duty-cycling. Both cycle-accurate and local testbed experiments are carried out to assess their performance. These mechanisms showed important improvements in the latency and energy consumption of broadcast communication, which further improved the performance of EADP and TrickleSD. However, these energy and latency achievements are still far from those of unicast.

Looking to remedy broadcast inefficiencies, and trying to complete the proposed solutions with interoperable service descriptions, Chapter 7 proposes integrations of EADP and TrickleSD with two widely-deployed description formats that foster seamless integration of 6LoWPAN networks in the IoT and allow completion of the picture depicted in Figure 1-1. Based on such integrations, some mechanisms of the proposed SD solutions could be substituted using unicast. The thesis ends by presenting conclusions drawn from the current research and outlining avenues for future research.

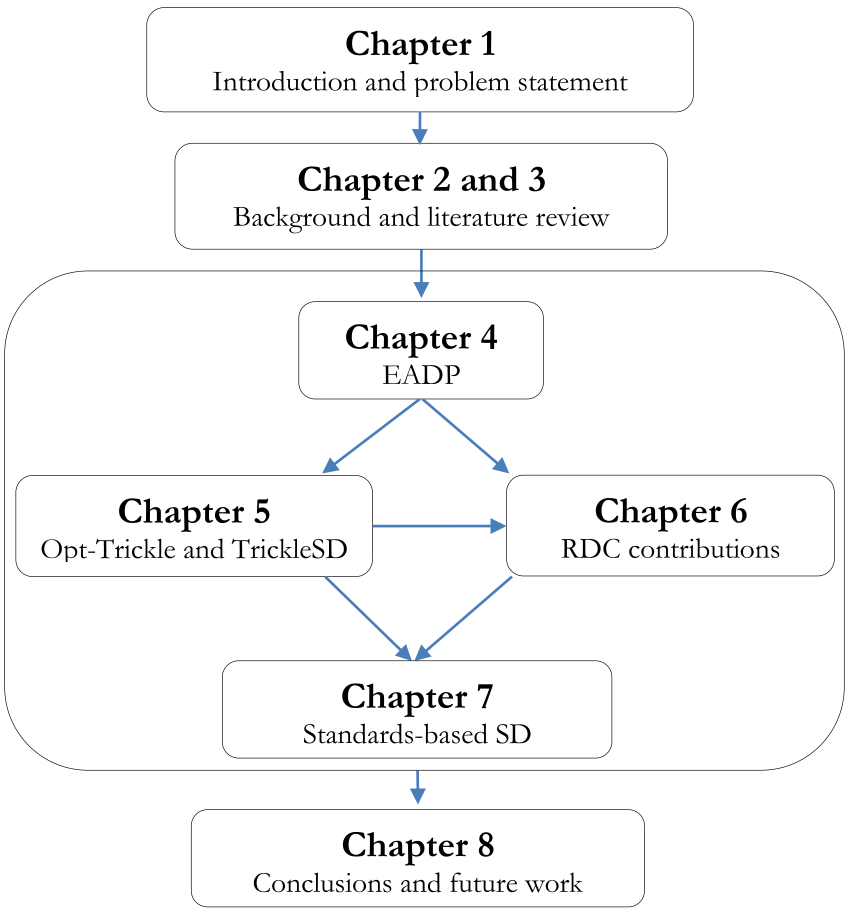

Chapter 1
Introduction and problem statement
Chapter 2 and 3
Background and literature review
Chapter 4
EADP
Chapter 5
Opt-Trickle and TrickleSD
Chapter 6
RDC contributions
Chapter 7
Standards-based SD
Chapter 8
Conclusions and future work

Figure 1-2 Thesis structure

Chapter 2

Low-power and Lossy Networks: Past, Present and Future

Having described the research context, aims and methodology, this chapter introduces the state-of-the-art technologies that allow WSN and IoT realisation. It begins by introducing terminology, definitions and the characteristics and limitations of WSN. Next, it describes the technology that brought IP to WSN along with on-going standardisation, industry and research efforts in the field and shows how this research project is situated among these efforts. Lastly, it introduces the research tools used in this project.

2.1 Terminology

For more than a decade of wireless sensor network research, many terms have been coined such as sensornets, wireless sensor and actuator networks, ubiquitous sensor networks, etc. In addition, because of the wide range of applications of such networks, domain-specific terms have emerged to describe the devices and their operations in specific contexts. From a business perspective, other terminology is being increasingly deployed including: smart object networks, Internet-connected objects, Internet of things and its variants, (constrained) machine to machine (M2M), wireless embedded Internet, Internet of the physical world, the sensor-actuator Internet, etc. Recent standardisation efforts in the field have introduced new terminology that tries to encompass many of the characteristics of such networks. Examples include Low-power and Lossy Networks (LLNs) [26] and Constrained-Node Networks (CNNs) [27]. These standards-related terms are the main terminology adopted in this document.

An LLN has been defined as being *"typically composed of many embedded devices with limited power, memory, and processing resources interconnected by a variety of links, such as IEEE 802.15.4 or low-power Wi-Fi"* [26]. Recently the term Constrained-Node Network has been introduced in [27] to describe networks running on devices with severe constraints on power, memory and processing resources. In this sense, the network is already constrained by the devices but it might also be constrained in terms of the communication technology [27]. Since LLNs are typically composed of constrained nodes [26], an LLN is defined as *"a constrained-node network with certain network characteristics, which include constraints on the network as well"* [27]. As discussed in the previous chapter, the use of IP over such networks is made possible via the 6LoWPAN standard [6]. 6LoWPAN, which is the underlying technology assumed in this research project is the primary driver of both networks and it is being used as LLN and CNN [27]. In the remainder of this document, the term 6LoWPAN is used when the focus is on IP networking. LLN and CNN are used interchangeably.

2.2 Constrained-node networks: definition and characteristics

In 1999, a revolutionary new technology was considered one of the 21 ideas for the 21st century [28] and in 2003 it was said to be one of 10 new technologies that will change the world [29]. This technology is none other than wireless sensor networks. The development of such networks is made possible through technical and technological advances in the fields of micro-electro-mechanical systems and wireless communication technologies. As a result, it becomes possible to mass produce smart and small devices combining sensing/actuating units, computing capabilities and communication capacities at a reduced cost.

When interconnected, these smart devices can cover a broad range of application areas, including industrial monitoring, smart grid and transportation systems, home and building automation, smart healthcare monitoring, environmental and urban monitoring, (e.g., parking and road monitoring), energy management, assets tracking, and mobile command, control and collaboration systems [26], [27]. Such applications are, however, constrained by both device and communication characteristics.

2.2.1 Constrained devices

While sensors/actuators have been around for a long time, it is only recently that it has been possible to produce integrated objects combining sensing/actuating capabilities with processing power and communication capabilities. These objects, also known as motes, sensors/actuators or smart objects are constrained in many aspects, hence the name constrained devices (alternatively constrained nodes, when the properties as network nodes are in focus) [27].

Constrained devices are often characterised by limited memory and computing capacities, short communication ranges, low data rates, and limited power resources as they are generally powered by non-rechargeable batteries or energy harvesters. Motes can perform three complementary tasks: reading/actuating on a physical quantity, processing, and communication. Several types of sensors embedded in constrained nodes can be distinguished such as seismic, thermal, visual, infrared and acoustic. They can monitor a broad range of ambient phenomena, including: temperature, humidity, pressure, noise, movement, presence or absence of some types of objects, and the speed, direction and volume of a given object. Depending on the sensed data, an actuator could be called on to modulate the flow of a fluid (e.g., water, gas), control electricity distribution (e.g., turn a light on/off), perform a mechanical operation (e.g., open/close a window), and so on.

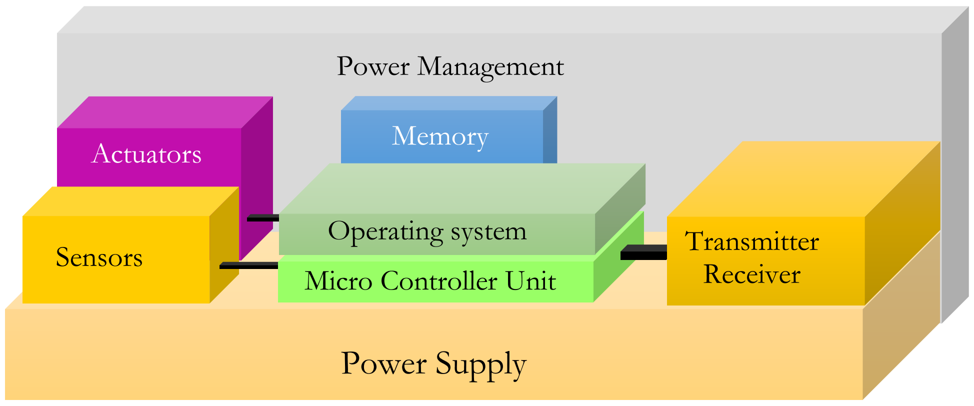


Figure 2-1 Architecture of constrained devices (reproduced from [30])

Constrained devices generally follow the same architecture based on a central core around which the various input/output, communication and power interfaces are articulated. Figure 2-1 shows the main components of a constrained device, namely the sensing/actuating unit, central processing unit, communication unit and the power unit.

The latter is generally represented by a non-rechargeable, non-replaceable battery and is the key constraint in the design of CNN applications. Because of the energy constraints, power management should be done at all levels.

2.2.2 Examples and classes of constrained devices

While constrained devices generally follow the same architecture (Figure 2-1), some differences exist depending on their capabilities. Thus, various types of constrained devices are available on the market. Examples presented in Figure 2-2 include TelosB developed by Crossbow, Econotag developed by Redwire and Waspmote designed by Libilum. These platforms adopt different MCU (Micro Controller Unit) architectures that shape their characteristics. Thus, while old platforms (e.g., TelosB) run on 8/16-bit MCUs, recent platforms (e.g., Econotag) run on 32-bit MCUs.

Although these improvements in performance are expected to continue, such devices will probably continue to be considered as constrained [26], [27]. This is because of a desire to scale down the characteristics of the nodes, and hence their space occupancy and cost, in order to scale up the connectivity to the larger number of nodes expected in the IoT [27]. In this context, [27] classifies current constrained devices into three classes: Class 0 (RAM: << 10 KB , Flash: << 100 KB); Class 1 (RAM: ~ 10 KB, Flash: ~ 100 KB); and Class 2 (RAM: ~ 50 KB, Flash: ~ 250 KB). A representative set of constrained device characteristics along with their classes is depicted in Table 2-1. Finally, it should be noted that sensors can also be integrated into other devices such as smartphones and laptops.

Table 2-1 Characteristics of representative constrained devices

architecture	**Model**	**MCU**	**RAM**	**Flash**	**radio chip**	**Class** [27]
MSP430	TelosB	MSP430F1611	10 KB	48 KB	CC2420	Class 0, 1
	XM1000	MSP430F268	8 KB	116 KB	CC2420	Class 1
AVR	MicaZ	ATmega128L	4KB	128 KB	CC2420	Class 0, 1
	Waspmote	ATmega 1281	8 KB	128 KB	8 radios	Class 1
ARM	Econotag	ARM7MC13224	96 KB	128 KB	integrated 802.15.4 radio	Class 2
	CC2538	ARM Cortex M3	16,32 KB	128, 256 512 KB	integrated 802.15.4 radio	Class 2

Econotag[1]	**Wismote[2]**	**TelosB[3]**	**Shimmer[4]**	**Waspmote[5]**

Figure 2-2 Representative constrained devices

2.2.3 Constrained networks

Many technologies have emerged to realise CNN applications. These technologies are constrained in many aspects including low-throughput, short communication ranges, high and unpredictable packet losses, limitations on packet sizes, and limitations on reachability, as the radio generally enters long sleep periods [27]. For illustrative purposes, some representative standards-based technologies are presented below:

- **Dash7** [31]: A wireless technology targeting RFID (Radio Frequency Identification) applications. DASH7 technology is standardised under the ISO/IEC 18000-7 standard.
- **Z-wave** [32]: A wireless technology designed for low-bandwidth data communication targeting embedded applications such as security sensors and home automation systems. It operates on sub 1 GHz frequency bands. Recently, Z-wave's lower layers have been standardised as the ITU G.9959 standard.
- **Bluetooth Low Energy (BLE)**: BLE is a wireless personal area network technology targeting low-power consumption, which is introduced as part of Bluetooth 4.0 specification [33]. Because of its pervasiveness in consumer electronics, BLE is an attractive technology for CNN applications.

[1] http://store.redwirellc.com/
[2] http://www.aragosystems.com/en/wisnet-item/wisnet-wismote-item.html
[3] http://www.memsic.com/wireless-sensor-networks/
[4] http://www.shimmersensing.com
[5] http://www.libelium.com/products/waspmote/

- **Low-power Wi-Fi**: With the evolution of wireless systems-on-chips, many low-power Wi-Fi sensor platforms have been developed. Wi-Fi-based CNNs are made possible by combining Wi-Fi mesh networking and WSNs.
- **EnOcean** [34]: EnOcean is an energy harvesting wireless technology ratified as the international ISO/IEC 14543-3-10 standard in 2012. It targets mainly CNN applications in building automation systems.
- **IEEE 1901.2** [35]: A standard for narrowband Power Line Communication (PLC) targeting smart grid applications. Apart from using PLC, IEEE 1901.2 shares similar constraints as its wireless counterparts. Indeed IEEE 1901.2 uses the same frame as the widespread IEEE 802.15.4 standard [4].
- **IEEE 802.15.4** [4]: IEEE 802.15 is of particular interest to CNNs thanks to its low-power, open-stack, robustness, and flexibility. It is widely anticipated that IEEE 802.15.4 will play a significant role in CNNs. This standard is the subject of the next section.

Clearly, each technology has its characteristics, targets, forces and limits as can be seen from Table 2-2. However because of its attractive features, the majority of today's constrained devices rely on the IEEE 802.15.4 standard.

Table 2-2 Representative wireless technologies for CNNs

	BLE	**Wi-Fi**	**ZigBee, etc.**	**EnOcean**	**DASH7**	**Z-wave**
Standard	Bluetooth Ver. 4.1	IEEE 802.11	IEEE 802.15.4	ISO/IEC 14543-3-10	ISO 18000-7	ITU-T G.9959
Frequency	2400 MHz	2400 MHz	868/915/ 2400 MHz	315/868/90 2 MHz	433 MHz	Around 900 MHz
Modulation	GFSK	CCK/QA M64 (b/g)	QPSK	ASK or FSK	FSK or GFSK	FSK or GFSK
Data-rate	1 Mbps	54 Mbps	250 Kbps	125 Kbps	200 Kbps	100 Kbps
MTU	27 bytes [36]	2304 bytes	127 bytes	14 bytes	-	64/158 bytes [37]
Range	30m	300m	300m	30m	1,000m	30m
Channels	40	11-14	1, 10, 16	-	-	-
Network size	-	30	65535	~ 20	-	232
Lifetime	multi-year	days	multi-year	-	multi-year	-

2.3 The IEEE 802.15.4 standard

The IEEE 802.15.4 task group was chartered to *"investigate a low data rate solution with multi-month to multi-year battery life and very low complexity"* [38]. The group published the first edition of the standard in 2003 (IEEE 802.15.4-2003). The standard offers basic lower-layer networking primitives (mainly the Physical layer (PHY) and the Medium Access Control (MAC)) for low-rate wireless personal area networks. IEEE 802.15.4 focuses on low-power, low-complexity, low-data-rates, low-cost, and short-range wireless communication between devices with minimum human interactions. Thus, unlike the standards designed for human to machine interactions such as IEEE 802.11 (Wi-Fi), the IEEE 802.15.4 standard is mainly designed for M2M communication. The standard was revised and enhanced in 2006 (IEEE 802.15.4-2006) and 2011 (IEEE 802.15.4-2011) [4].

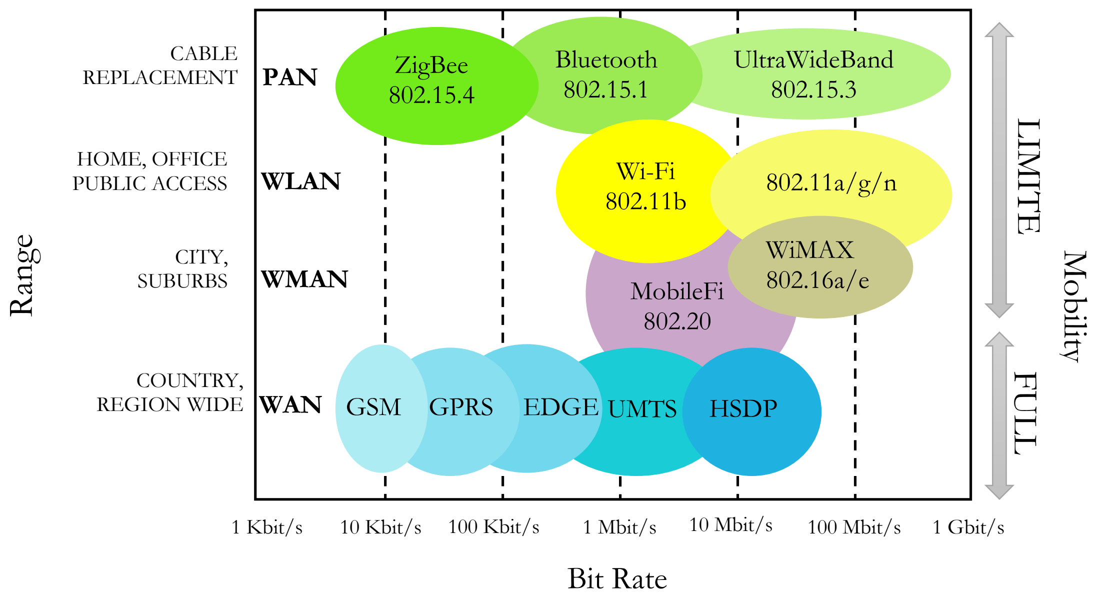
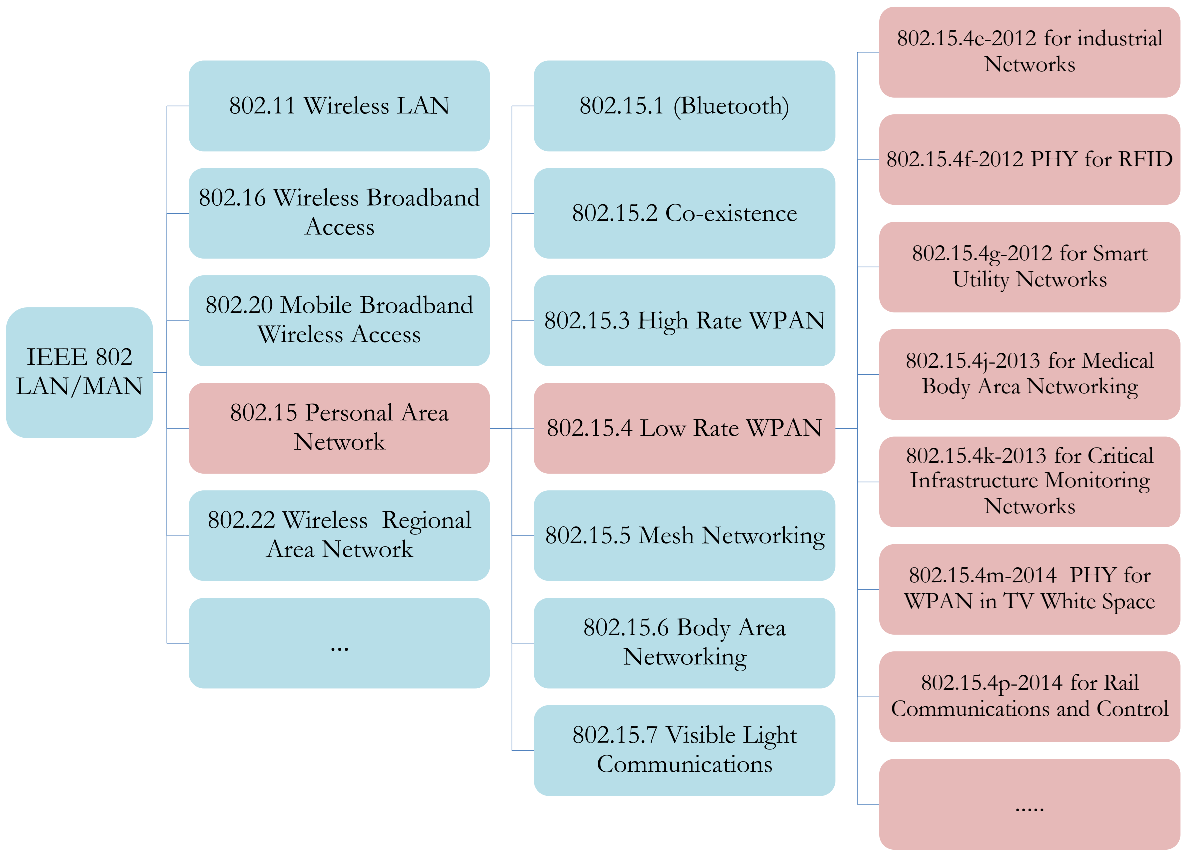


Figure 2-3 Wireless technologies and their characteristics (reproduced from [39])

Figure 2-3 situates the IEEE 802.15.4 standard in the wireless space with respect to aims, data rates and mobility support. As can be seen from this figure, IEEE 802.15.4 is designed to fill the gap in low-power and low-data-rate wireless communication. It supports a maximum of 250 kbps throughput for up to a distance of 300 metres. IEEE 802.15.4 also provides good mobility support in its category. The standard's range of features proved attractive and subsequently amendments were added resulting in: IEEE 802.15.4e for industrial applications; IEEE 802.15.4f targeting active RFID applications; IEEE 802.15.4g for smart metering utility networks; 802.15.4k for critical infrastructure monitoring; and 802.15.4j for medical body area networks, etc. These amendments along with their place in the IEEE wireless standards are depicted in Figure 2-4. Characteristics relevant to this research, introduced by such amendments, are described in the corresponding places.



IEEE 802 LAN/MAN
802.11 Wireless LAN
802.16 Wireless Broadband Access
802.20 Mobile Broadband Wireless Access
802.15 Personal Area Network
802.22 Wireless Regional Area Network
...
802.15.1 (Bluetooth)
802.15.2 Co-existence
802.15.3 High Rate WPAN
802.15.4 Low Rate WPAN
802.15.5 Mesh Networking
802.15.6 Body Area Networking
802.15.7 Visible Light Communications
802.15.4e-2012 for industrial Networks
802.15.4f-2012 PHY for RFID
802.15.4g-2012 for Smart Utility Networks
802.15.4j-2013 for Medical Body Area Networking
802.15.4k-2013 for Critical Infrastructure Monitoring Networks
802.15.4m-2014 PHY for WPAN in TV White Space
802.15.4p-2014 for Rail Communications and Control
.....

Figure 2-4 IEEE 802.15.4 position in the IEEE wireless standards

2.3.1 IEEE 802.15.4 characteristics

In this section, the key features and limitations of the IEEE 802.15.4 are presented. The complete IEEE 802.15.4-2011 specifications can be found in [4].

The IEEE 802.15.4 standard specifies the usage of two frequency bands at 868/915 MHz (continent dependent) and 2400 MHz (worldwide) and multiple channels as shown in Figure 2-6. The standard defines two classes of devices namely Full-Function Devices (FFD), in which all the necessary functionalities are supported, and Reduced-Function Devices (RFD) that only implement a limited number of functionalities. An RFD can only communicate with an FFD, as shown in Figure 2-5. When internetworked, RFDs and FFDs create a Personal Area Network (PAN). A PAN coordinator is responsible for setting up and maintaining the network. Such role can only be taken by an FFD. For addressing, nodes in a PAN might use a 16-bit short or 64-bit Extended Unique Identifier (EUI) link-layer addresses. Depending on the application requirements, an IEEE 802.15.4-based network might adopt one of following basic topologies:

- **Star topology:** In this topology, the communication can only be established via the PAN coordinator. Hence, as shown in Figure 2-5, node A has to pass through the PAN coordinator in order to communicate with node B. Applications that may benefit from this topology include home automation systems, computer peripherals and personal healthcare [4].
- **Peer-to-peer topology:** In this topology, each device can communicate directly or indirectly with any other network element. To do so, FFDs also perform the role of communication relays as shown in Figure 2-5. Most of the applications cited in section 2.2 adopt this topology.

Thanks to its high flexibility, the peer-to-peer topology allows the creation of mesh networks, which can compensate for the short communication range. Thus, it can achieve long-range communications through multi-hop meshing. Mechanisms for creating and managing the mesh are left for upper layers and are not part of the IEEE 802.15.4 standard.

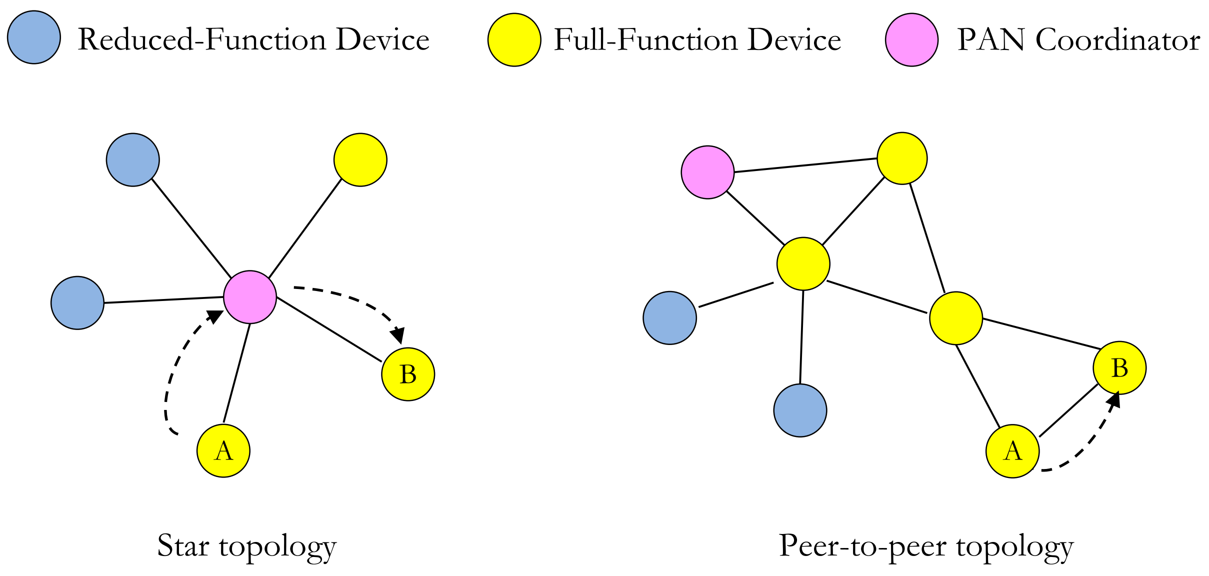


Figure 2-5 IEEE 802.15.4 network topologies

To access the communication channel, the IEEE 802.15.4 MAC layer adopts the Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) strategy. To further avoid collisions, the physical layer performs a Clear Channel Assessment (CCA) when running CSMA/CA in order to transmit MAC frames, of which there are four types (data, acknowledgment, beacon and MAC command frames). Additionally, a slotted CSMA/CA strategy might be envisaged. The maximum size of a frame, known as the Maximum Transmission Unit (MTU), is 127 bytes. This is one of the main limiting factors when developing IEEE 802.15.4-based applications.

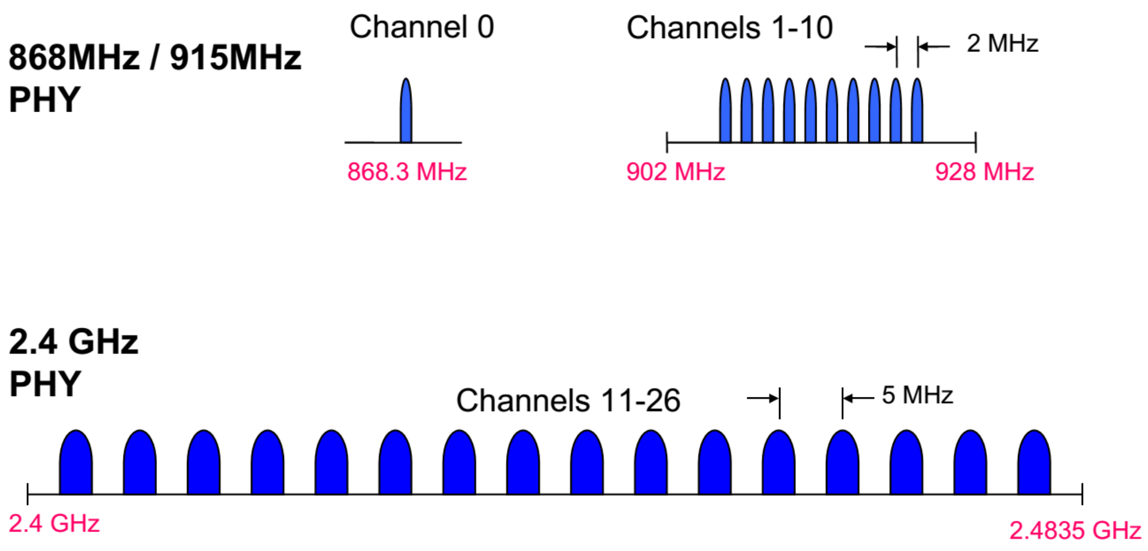


Figure 2-6 IEEE 802.15.4-2003 frequency bands and channels [40]

2.3.2 IEEE 802.15.4 features

The most outstanding features and benefits brought by IEEE 802.15.4 are:

- **Link Quality Indicator (LQI)**: The physical layer of the IEEE 802.15.4 provides a very useful indicator of the quality of a link, extracted from every received packet. This newly introduced feature has attractive use-cases, especially if combined with the Received Signal Strength Indicator (RSSI).
- **Received Signal Strength Indicator (RSSI)**: IEEE 802.15.4 standard provides an RSS value estimated by a receiver during data reception. Such an indicator conveys useful link information including an estimate of the distance between a sender-receiver pair. RSSI and LQI are briefly discussed in the following subsection.

- **Channel hopping**: This is an interesting feature provided by the IEEE 802.15.4 standard which supports up to 16 channels to switch between in the 2.4 GHz band (Figure 2-6). Channel switching is further explored in IEEE 802.15.4e amendment [41] by defining robust channel hopping mechanisms.
- **Powering on/off the radio transceiver**: The physical layer allows the turning on/off of the radio transceiver in order to save energy; a precious resource for IEEE 802.15.4 devices. This feature allows developing Radio Duty Cycling (RDC) mechanisms which are briefly described in section 2.3.4.
- **Increased MTU**: One of the main constraints of the IEEE 802.15.4 standard is its limited MTU of 127 bytes. Thanks to the IEEE 802.15.4g amendment [42], up to 2047-byte MTU is possible. This may require different hardware.

2.3.3 RSSI and LQI

RSSI is a metric widely deployed in wireless standards including IEEE 802.15.4. It measures the strength of a received signal delivered in dBm. RSSI has a relation to many parameters including the distance d between a sender-receiver pair. The most widely adopted model of RSS is the log-normal model given by equation 2-1:

$$RSS = PL\,(d_0) + 10\eta \log_{10}\left(\frac{d}{d_0}\right) + X_\sigma \,, \qquad 2\text{-}1$$

where $PL\,(d0)$ is a constant measured at a reference distance d_0, η is the path-loss exponent in a specific environment and X_σ is a random normal variable modelling other environmental artefacts. RSSI can potentially be used as an indicator of the relative distance between a sender and a receiver. However, since RSS is affected by all above parameters, LQI might be used to enhance RSSI-based estimations.

LQI is a metric introduced in the IEEE 802.15.4 to measure the errors in the modulation of a successfully received frame, and hence the quality of the link between a sender-receiver pair. It is a unit-less metric delivered as an integer between 0 and 255 indicating the lowest and highest link qualities, respectively. Unlike RSSI, LQI is implemented differently by radio-chips. For instance, the wide-spread CC2420 chip [43] delivers, for every received frame, a correlation value: CORR in the interval 50 to 110. To be compliant with the standard, CORR values must be converted to LQI range using equation 2-2, for example. a and b are found empirically.

$$LQI = (CORR - a) \times b. \quad \text{2-2}$$

From a distance point of view, if the LQI is high, more confidence might be given to the RSSI value as a potentially good estimate of the distance. Since RSSI and LQI are directly extracted from every received frame and because of the precious information they incorporate, they can be used to enhance some of the mechanisms proposed in this thesis. Such a usage is introduced in Chapter 5.

2.3.4 Energy conservation through radio duty cycling

Power consumption is a significant concern for IEEE 802.15.4 since in many of the intended applications devices are battery powered [4]. In such systems, the radio has been shown to be the dominating energy consumer [44], especially as it generally consumes as much energy when it is idle as when it is transmitting or receiving [43]. Thus, RDC mechanisms are employed to reduce power consumption. RDC enables the nodes to spend most of their operational time sleeping while waking up periodically to check for activity. This way, RDC protocols can provide an Always-On Illusion: *"always on but mostly off"* [45].

The IEEE 802.15.4e amendment specifies three RDC techniques, namely TSCH: Time Slotted Channel Hoping; CSL: Coordinated Sampled Listening; and RIT: Receiver Initiated Transmission. Such techniques are implemented at the MAC layer right above the PHY as shown in Figure 2-7 and can achieve up to 99% sleep time [45], [46]. Each of the above techniques represents, respectively, one the following RDC classes: synchronous RDCs; Low-Power Listening (LPL); and Low-Power Probing (LPP).

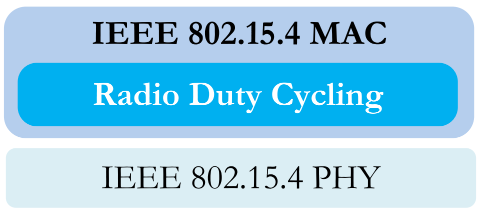


Figure 2-7 Radio duty cycling

2.3.4.1 Synchronous RDCs

In synchronous RDC methods, nodes synchronize their sleep/wakeup schedules such that communication can take place. Early protocols in this category include S-MAC [47] and T-MAC [48]. Recent protocols include the IEEE 802.15.4e's TSCH. While synchronous protocols can ensure low radio duty cycles, the synchronisation process introduces extra overhead and complexity. To remove this complexity, asynchronous approaches allow nodes to work independently by choosing their own sleep schedules. To enable communication, these protocols rely on two main techniques namely: LPP and LPL described below.

2.3.4.2 Low-power probing

In LPP-based approaches (Figure 2-8), instead of a sender initiating the communication, a potential receiver alerts potential senders, via broadcast *probe*s, of its availability to receive data. Upon reception of a probe, a node with pending data examines whether the probe's initiator is its intended recipient and sends an acknowledgment warning the receiver to stay awake. If no acknowledgment is received, the potential receiver goes back to sleep [49]. This technique was first implemented in the Koala system [49]. A more efficient protocol has been introduced in RI-MAC [50]. RIT is the standard-based protocol in this category.

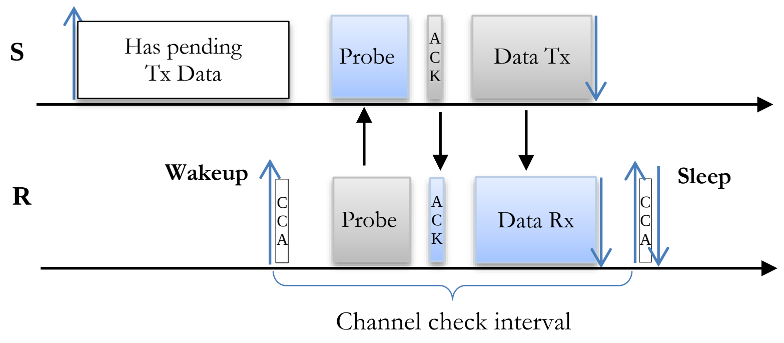


Figure 2-8 Low-power probing

2.3.4.3 Low-power listening

In the basic B-MAC's LPL mechanism [51], senders start by transmitting a "wakeup signal" called a preamble that is long enough to ensure that the receiver's periodic Clear Channel Assessment (CCA, Figure 2-9) catches it up. Consequently, the receiver stays awake to receive the data. This long preamble transmission can cause interference with other nodes and may prevent packet reception and affect throughput [52]. In addition, a node implementing the B-MAC's LPL mechanism may wakeup and remains awake only to receive a packet targeting other nodes. To address these issues, X-MAC [53] replaces the long preamble with a strobed preamble readable by packetised radios. The strobed preamble (Figure 2-9) consists of a sequence of short, repeated preambles each embedding the destination address. This will not only allow irrelevant nodes to go immediately back to sleep, but also let the intended receiver inform the sender, via the acknowledgment packet, to stop transmitting the preamble and start sending the data. On the other hand, WiseMAC [54] tries to address B-MAC issues and shortens the preambles, by learning the schedules of neighbours from the received acknowledgments. Finally, BoX-MACs [55] substitute the strobes by the data itself.

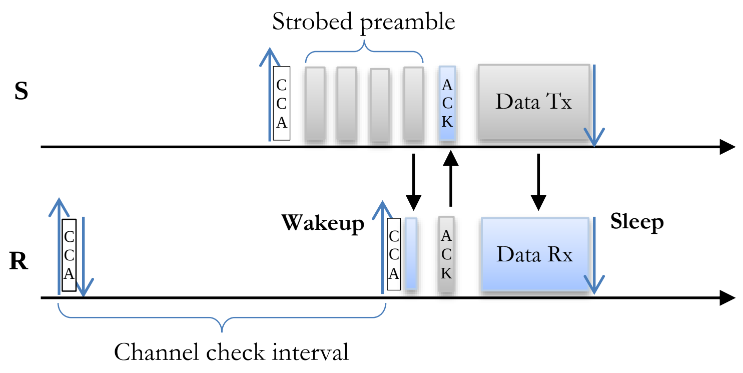


Figure 2-9 Low-power listening

Recent protocols in this category, namely ContikiMAC and CSL (CSL is directly influenced by Emnet [45]), build on previous achievements. Thus, in a similar approach to BoX-MACs, ContikiMAC uses the data as strobes and provides a better wakeup mechanism achieved via precise timing constraints. This way, ContikiMAC decouples the receiver's consumed energy from the length of the wakeup signal. On the other hand, CSL retains a strobed preamble, called a chirp, similar to X-MAC, and provides techniques to decouple the receiver's energy from the length of the wakeup signal. For instance, CSL embeds in every chirp a *rendezvous* time indicating the remaining time until the sender starts data transmission. This allows a receiver hearing a chirp to learn when the sender data transmission will begin and thereby it can sleep until before then.

Therefore, the receivers' energy consumption is decoupled from the length of the sleep interval as in ContikiMAC.

Both ContikiMAC and CSL deploy a *phase-lock* mechanism similar to that of WiseMAC. For instance, in ContikiMAC upon reception of an acknowledgment, a sender records the time and stops its transmission (*Stop Tx* & *Learn Rx,* Figure 2-10). Since wakeups are periodic and assuming that the same period is deployed, the sender can synchronize its subsequent transmissions with the receiver's wakeup (*Phase-lock*, Figure 2-10). However, because of clock-drifts, the *phase-lock* mechanism needs to be updated periodically.

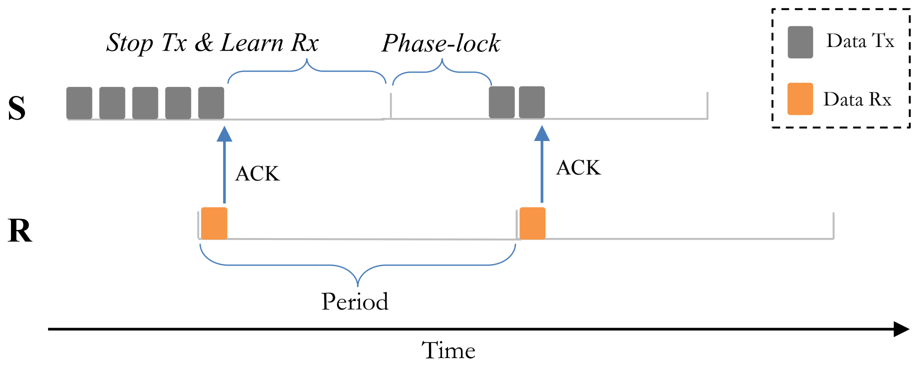


Figure 2-10 Unicast in ContikiMAC

2.3.4.4 Streaming over RDCs

RDC protocols save noticeable energy at the expense of increased transmission delays and decreased throughput since devices can only receive when they are awake. This becomes aggravated in multi-hop networks where the process is accumulated at each hop. To address this issue, [45] introduced a streaming functionality over RDCs. Doing so, a sender can indicate to a receiver that more data are pending in order to remain awake and receive them. To this end, the sender sets the pending bit of the IEEE 802.15.4 frame header to 1. The pending bit is reset to zero by the sender for the last frame thereby informing the receiver that the stream has ended. This basic idea was elaborated in [56] where new techniques and optimisations for burst forwarding were designed for ContikiMAC. However, such mechanisms mainly target unicast transmissions and exploit the acknowledgments to realise burst forwarding. Indeed the authors of [56] state: "*the bursts provide a rapid retransmission mechanism: every packet is repeatedly sent until reception of a link layer acknowledgment or the end of the wakeup transmission period*".

Because of the importance of RDC, it should be considered when developing applications for LLNs. Contributions of this research project regarding RDC will be the subject of Chapter 6.

2.3.5 Representative IEEE 802.15.4-compatible radio chips

Based on the above features, IEEE 802.15.4-complaint low-power radio-chips are coming on to the market at an accelerated pace. Only a few radio-chips are available for the continent specific sub 1 GHz bands (Texas Instruments' CC1200 is an example). Hence, the worldwide available 2.4 GHz band has attracted most of the IEEE 802.15.4-compliant radio chips. Table 2-3 from [45] presents a representative set of widely used IEEE 802.15.4-based radio chips and their properties.

Table 2-3 Properties of representative IEEE 802.15.4 radios [45]

Make	Model	VCC	Transmit		receive		Sleep (uA)	Wake (ms)
			(mA)	(dBm)	(mA)	(dBm)		
Atmel	RF230	1.8-3.6	16.5	+3	15.5	-101	0.02	1.1
Freescale	MC13192	2-3.6	30	+4	37	-92	1.0	7-20
Jennic	JN5121	2.2-3.6	50	+1	45	-90	5.0	2.5
	JN5139	2.2-3.6	34	+0.5	34	-97	2.8	2.5
Texas Instruments	CC2420	2.1-3.6	17.4	0	18.8	-95	1.0	1.0
	CC2430	2.0-3.6	17.4	0	17.2	-92	0.5	1.0
	CC2520	1.8-3.8	25.8	+5	18.5	-98	0.03	0.3

The above characteristics made IEEE 802.15.4 very attractive for many upper layer wireless technologies such as ZigBee [57], WirelessHART [58] and ISA100.11a [59]. Conversely, IEEE 802.15.4 also provides the basis for the wireless embedded Internet through the 6LoWPAN standard which is the subject of the following section. Finally, it should be noted that while this section has focused on the IEEE 802.15.4 standard, the overall focus of this research is link-layer independent and targets IP-enabled LLNs as an enabler for the IoT. Hence, other similar LLN technologies adopting IP could benefit from the techniques developed in this research.

2.4 Constrained-node networks: the future is IP

Until recently, IP was thought to be too complicated and very power consuming for CNNs [45]. However, with the current trend towards all-IP networks, IP applicability in CNNs is reinvestigated. As a result, the IETF had chartered the 6LoWPAN working group to investigate the feasibility of transmitting IP packets over IEEE 802.15.4 links.

2.4.1 IPv4 or IPv6

As a starting point in its investigation, the working group had to choose which version of IP should be adopted and adapted to constrained-node networks. Today, (2015), IP version 4 (IPv4) is still the most widely deployed version. It has successfully allowed the establishment of a global network of millions of nodes internetworking billions of users [45]. However, the success of IPv4 is catching up with its address space limitations. In response, the IETF developed IP version 6 (IPv6). The first version of this new specification was published in December 1998 as RFC 2460 [5]. IPv6 now incorporates the learning gained over 30-year usages of IPv4 and uses 128-bit addresses instead of 32-bit ones. This expands the available address space to uniquely address 3.4×10^{38} objects, which is about 2.8×10^{14} times larger than that of IPv4. This makes IPv6 very attractive when it comes to design IP for the IoT; expected to interconnect more than 50 billion devices by 2020 [9].

Having adopted the IP version to be adapted, the next main challenge for the working group was addressing the conflicting MTU sizes between IEEE 802.15.4 and IPv6 standards. Thus, while the former specifies 127-byte MTU, the latter requires a minimum of 1280 bytes. This is among other issues addressed by the 6LoWPAN standard.

2.4.2 The 6LoWPAN network stack and features

The first specification of the 6LoWPAN standard was published in 2007 as RFC 4944 [6] and updated in 2011 by RFC 6282 [60]. The standard mainly proposes an adaptation layer (layer 2.5) between the IP network layer (layer 3) and the IEEE 802.15.4 link layer (layer 2), as shown in Figure 2-11.

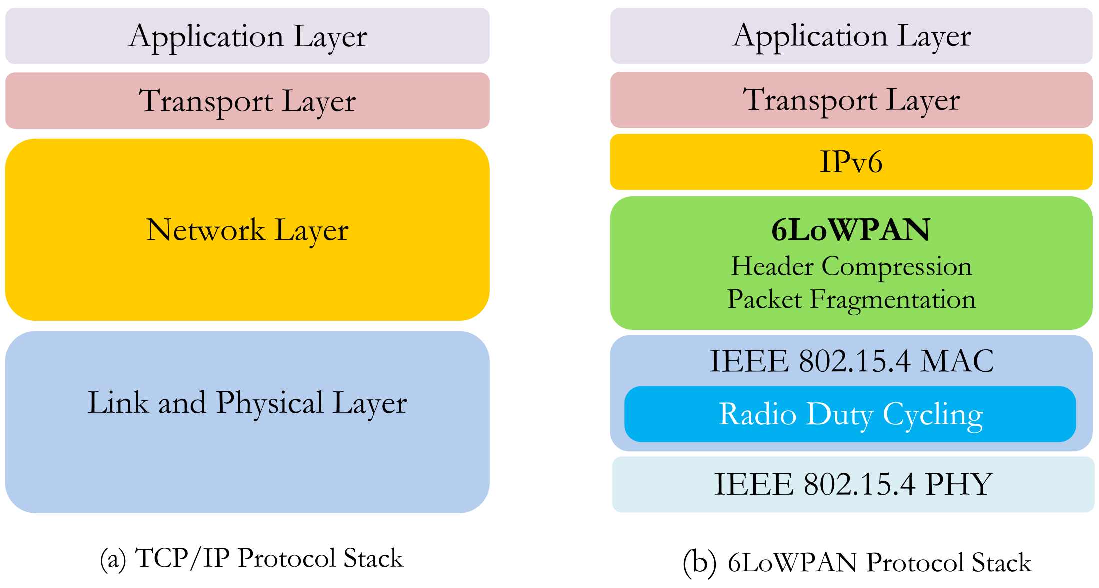


Figure 2-11 6LoWPAN protocol stack

The 6LoWPAN format defines how IPv6 packets are carried in IEEE 802.15.4 frames and specifies the adaptation layer's key elements, presented in the following points:

- **Header compression**: To compress IPv6 packets, the header fields that can be extracted from the information carried in the IEEE 802.15.4 frames, such as link-local addresses and payload length, are eliminated (stateless compression). Other header fields such as global addresses are compressed based on shared contexts within a 6LoWPAN (stateful compression) [60]. In addition, 6LoWPAN can also compress standard protocol headers like TCP, UDP and ICMP.
- **Packet fragmentation/reassembly**: In response to the conflicting MTU sizes between the IPv6 and the IEEE 802.15.4 standards, the 6LoWPAN adaptation layer provides mechanisms to fragment an IPv6 packet into multiple IEEE 802.15.4 frames to accommodate their transmission and reassemble them to form the IPv6 packet upon reception.
- **Layer 2 forwarding:** Unlike traditional IP networks where routing tasks are performed only at layer 3, 6LoWPAN makes it possible to support layer 2 forwarding of IPv6 packets; known as mesh-under routing. Thus, the adaptation layer can carry link-layer addresses of the endpoints of an IP hop (Figure 2-12 (a)). Alternatively, the IP stack might accomplish traditional IP routing via layer 3 (route-over routing) where each IEEE 802.15.4 link is an IP hop (Figure 2-12 (b)). The implication of such a distinction on a network can be seen in Figure 2-13.

As shown in Figure 2-13, route-over configurations take every link as an IP hop and hence can work using IP addresses agnostically of the underlying technology. This allows coexistence of different link-layer technologies to form a single IP network. On the other hand, mesh-under approaches make use of link-layer addresses to deliver the network as a single IP hop in a fashion similar to Ethernet. Thanks to its attractive features, route-over is most preferred; however, a door is always open for mesh-under in 6LoWPAN related standards. Finally, it should be noted that while 6LoWPAN defines and provides basics on how to perform routing in 6LoWPAN networks, the routing itself is not a part of the standard and is left for investigation by other works.

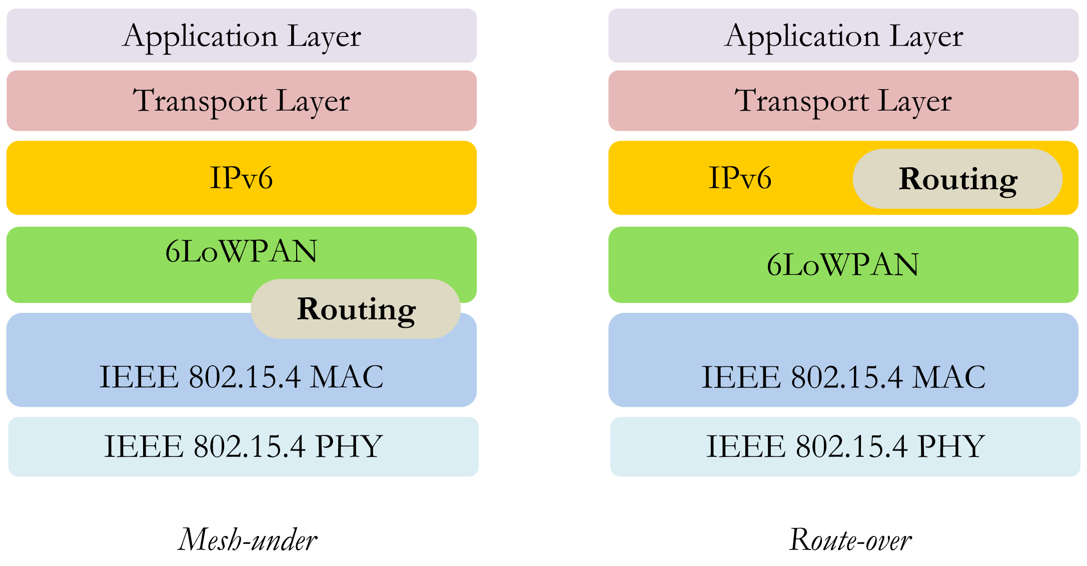


Figure 2-12 Mesh-under vs. route-over routing (reproduced from [61])

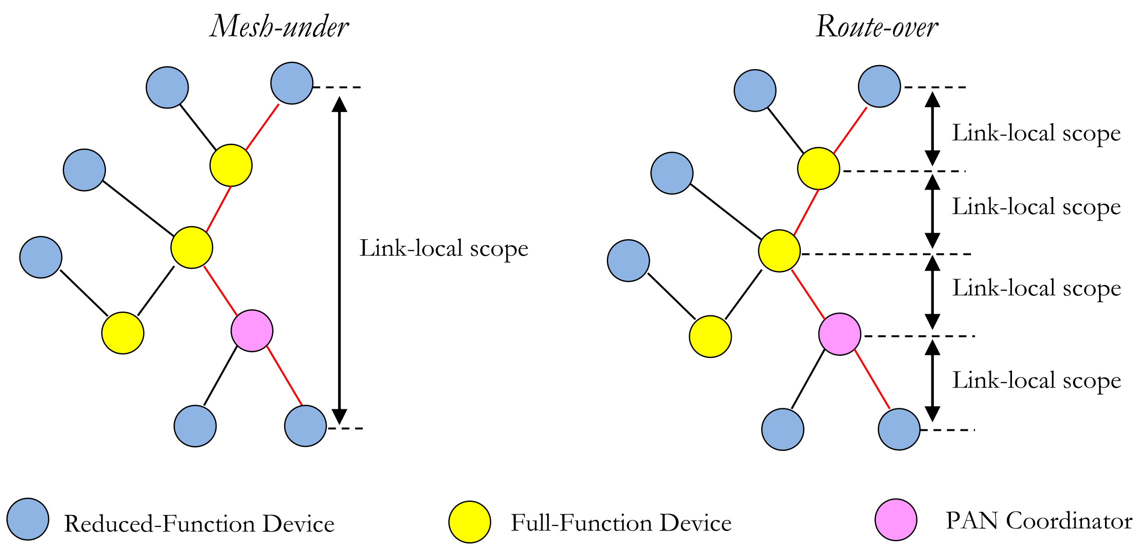


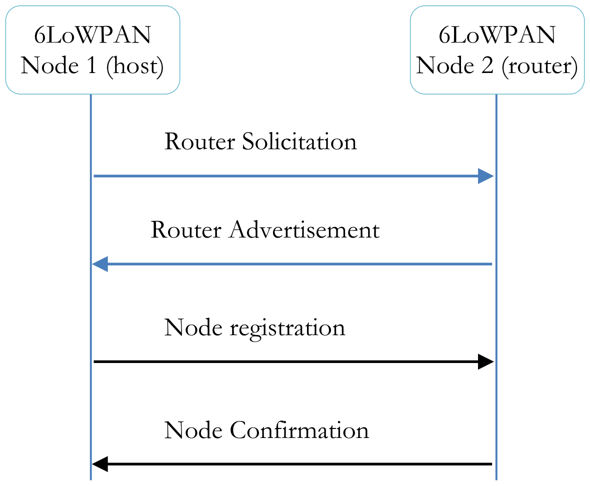
Figure 2-13 Mesh-under vs. route-over implication

2.4.3 Neighbour discovery optimisation for 6LoWPAN networks

Neighbour Discovery (ND) [62] is a crucial protocol in IPv6 networks. It provides many important mechanisms used by nodes to discover each other's presence and maintain reachability information. ND is also used for address auto-configuration, address resolution, neighbour unreachability detection and duplicated address detection along with prefix and parameter distribution. Being the core of IPv6 functionality, ND should also be deployed in 6LoWPANs. However, ND is not designed to address 6LoWPANs' constraints. In addition, most of its tasks rely on broadcast communication which is less efficient for sleepy nodes. To address these issues, an ND optimisation for 6LoWPANs (6LoWPAN-ND) has been introduced in RFC 6775 [63].

In 6LoWPAN terminology, nodes are either hosts which transmit/receive packets but do not route them or routers which route information on behalf of others. Particular types of routers called edge routers are deployed to connect 6LoWPANs with other IP networks. By analogy to Figure 2-5, hosts can be either RFDs or FFDs, routers are FFDs and edge routers are PAN coordinators. 6LoWPAN-ND mainly optimizes the host-router interactions by avoiding multicast as much as possible. In addition, it provides substitutable mechanisms to: (i) perform multi-hop Duplicate Address Detection (DAD) and; (ii) distribute multi-hop prefixes and context information. DAD is used to ensure uniqueness of IPv6 addresses derived from the 16-bit short link-layer addresses for IPv6 stateless address auto-configuration. In this case, a delay is expected since the address has to be forwarded to the border routers, which perform DAD against all registered addresses. However, if IPv6 addresses are derived from the 64-bit EUI addresses, no need to perform DAD as 64-bit EUI addresses are assumed to be globally unique [63]. In the second substitutable mechanism, the former is used to establish and join a network while the latter is required by stateful header compression.

The importance of ND tasks in bootstrapping and maintaining a network is illustrated in the following example: when a host joins a network, it assigns itself a link-local IPv6 address and broadcast a router solicitation message to find default routers. A router will respond by a unicast router advertisement to the node. The node then assigns a global address and tries to register it with its default router using a unicast neighbour solicitation message containing in addition to the address, a registration lifetime. If the registration was successful, a neighbour advertisement message is received. The node then performs maintenance by sending neighbour solicitation messages with new address registration before the expiry of the lifetime. This process is depicted in Figure 2-14.



Router's Neighbour Cache

Neighbour	Lifetime	Type	State
Node 1	10h	Registered	Reachable

Figure 2-14 6LoWPAN-ND message exchanges and neighbour cache

The 6LoWPAN-ND process described above implies that nodes keep and manage Neighbour Cache Entries (NCEs). Thus, when a node interacts with a router by sending router solicitation messages, the router creates a tentative NCE to be kept for a short lifetime in its neighbour cache. When the registration is confirmed, the tentative NCE is converted to a registered NCE for the specified lifetime. When routers send router advertisements to hosts, and when they receive router advertisements or multicast neighbour solicitations from other routers, routers insert a garbage-collectible NCE in their caches [63]. Such NCEs are managed by the rules defined in IPv6 ND [62].

Many implementations of the 6LoWPAN standard have already emerged for different operating systems and platforms. For instance, 6LoWPAN is being implemented in traditional operating systems such as Linux and in many CNN-specific operating systems such as TinyOS [64] and Contiki [65], described later on in this chapter.

2.4.4 6LoWPAN implications

The advantages of an IP adaptation layer for CNNs are among others:

- **Interoperability**: Bringing IP to CNNs enables interoperability at the network layer with other IP networks. This constitutes a common ground for building higher layer interoperability solutions through e.g., discovery mechanisms of available capabilities and services.

- **Manageability**: IP has established robust tools for network management. Bringing IP into CNNs allows such tools to be used in easing administration and management of a vast number of smart devices. For instance, one can use ping to check if a constrained node is connected.
- **Established security**: IP provides trusted security solutions (e.g., data encryption, firewalls, and access control). Such proved solutions will motivate wide development and spread of CNN applications.
- **Productivity and easy learning curve**: Most network developers today are familiar with IP networking. Thus, 6LoWPAN will significantly reduce the development time and cost of CNNs and increase the productivity. In addition, 6LoWPAN together with other standardisation efforts minimise the vast number of arbitrary and proprietary solutions encountered when developing CNN applications [45].
- **Web-based interfaces**: IP has established many web-based interfaces which hide the complexity from the end-user and thereby enable non-expert users to easily access and use provided services. Thus, with the subsequent works resulting from 6LoWPAN (section 2.5.3); constrained devices are being simply accessed through a browser.

In addition to the above benefits brought by IP to CNNs, 6LoWPAN presents other attractive features when compared with both IP-based (e.g., Wi-Fi) and non-IP based (e.g., ZigBee) solutions.

Figure 2-15 from [66] compares the three technologies (Wi-Fi, ZigBee and 6LoWPAN) on a scale from 0 to 5 with respect to 7 parameters that allow realisation of the Internet of things, namely: mobility and multicast supports, mesh networking and scalability, low cost and low power consumption, and support for two-way communication between devices. In all the 7 features, 6LoWPAN had the top score. Indeed, it hits the highest grades in 5 out of the 7 parameters.

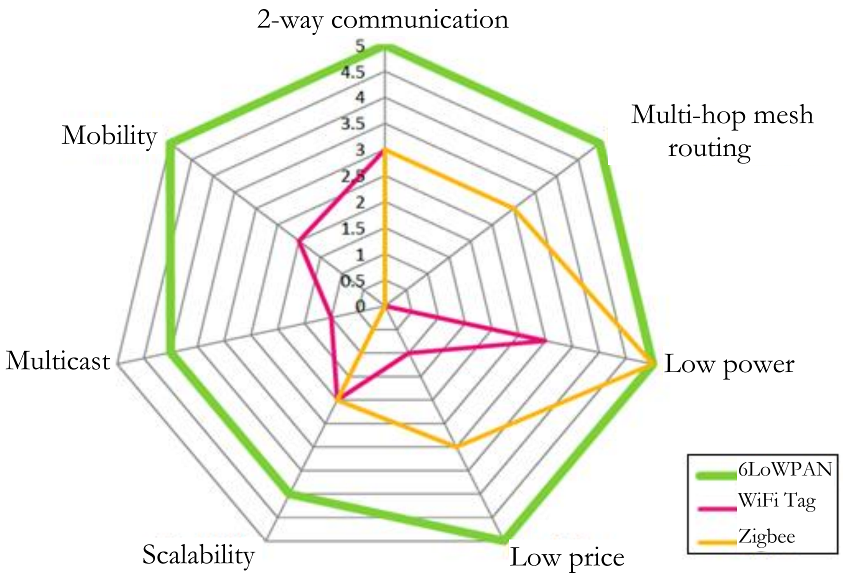


Figure 2-15 Comparison between 6LoWPAN, Wi-Fi and ZigBee [66]

2.4.5 Typical 6LoWPAN network architectures

Built upon IEEE 802.15.4-supported network topologies (section 2.3.1), 6LoWPAN provides three main architectures namely: the simple LoWPAN architecture, the extended LoWPAN architecture and the ad-hoc LoWPAN architecture. The two former architectures describe 6LoWPAN usage in infrastructure based environments (e.g., homes, enterprise buildings, etc.) while the latter introduces its usage in infrastructure-less, ad-hoc environments. The three architectures are presented in Figure 2-16 where nodes can be hosts (H), routers (R) or edge routers. A simple LoWPAN is defined by the set of nodes sharing the same IPv6 prefix, usually delivered by the edge router. An extended LoWPAN architecture with more than one edge router and their nodes can be also envisaged. In this case, extended LoWPANs use a backbone link, e.g., Ethernet, to coordinate information about the network. Finally, an ad-hoc LoWPAN which can operate without established infrastructures is also shown in Figure 2-16.

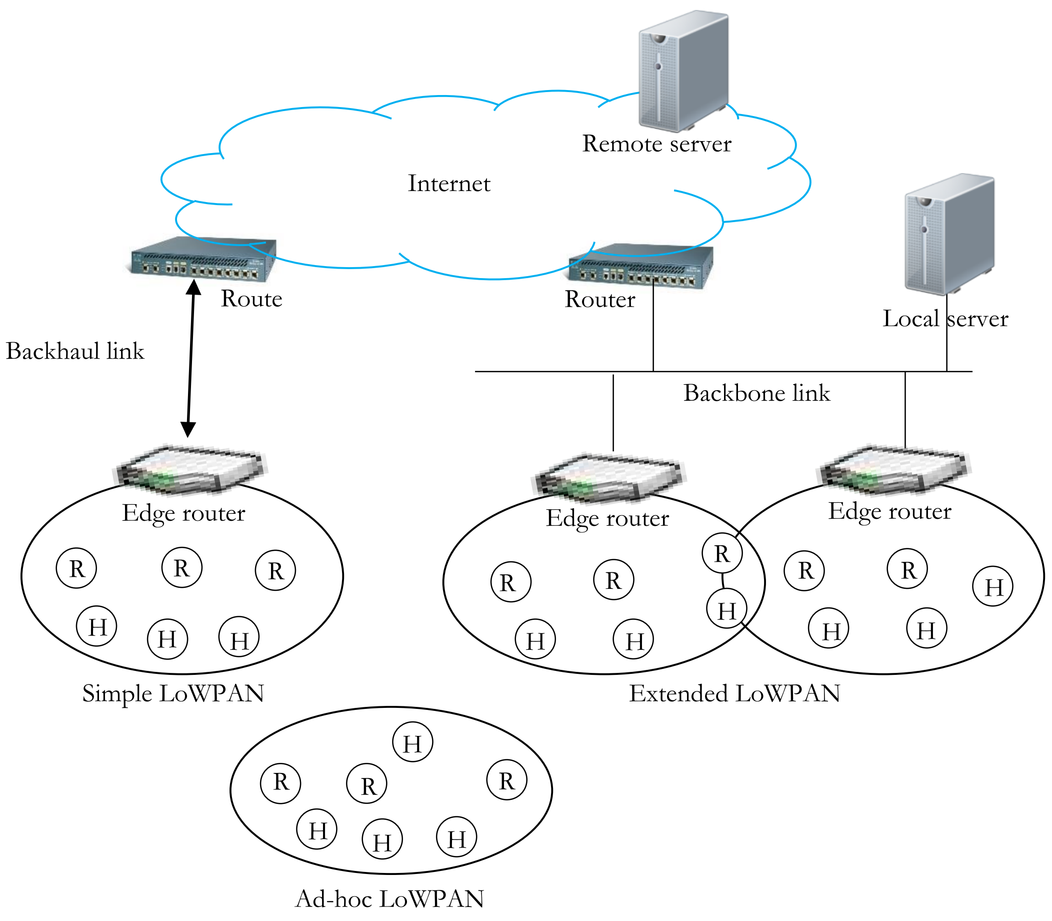


Figure 2-16 Typical 6LoWPAN network architectures (reproduced from [8])

2.4.6 Mobility in 6LoWPAN networks

Anticipating the potential of CNNs, a draft for mobility consideration in such networks [67] has been presented to the 6LoWPAN working group. Indeed, mobile scenarios of 6LoWPAN networks are growing fast. Such scenarios include smart transportation grid, smart healthcare, smart logistics and mobile command, control and collaboration systems. While some applications such as smart logistics may require *network mobility*, where nodes together with the edge router are moving and roaming between multiple access points, others, like smart transportation systems, require node mobility where nodes are independently and randomly moving. In the latter, nodes may move within the same IP network controlled by an edge router, known as *micro-mobility*, or roam between 6LoWPANs and hence change their point of attachments from an edge router to another, called *macro-mobility*, as shown in Figure 2-17. While this physical mobility is fully understood, 6LoWPAN networks might appear moving although they are not. This is because of frequent topology changes caused by environmental parameters affecting radio connectivity. Moreover, network dynamics may occur as nodes run out of power, fail or are removed from the network.

Because of the challenges mobility imposes on the connectivity of 6LoWPAN networks and hence on their operability, it should be handled along with the whole network stack [8]. Thus, routing protocols in LLNs are expected to provide alternative links, the IP layer should provide techniques for dynamic address assignments, and upper layers should opt for techniques with minimum dependency on fixed infrastructures. In essence, it is desirable to have a distributed opportunistic architecture where nodes can dynamically discover each other and cooperate without the need for central servers or human administration. Finally, it should be noted that traditional mobile IPv6 protocols such as Mobile IPv6 [68] and NEMO [69] are not directly applicable to 6LoWPAN networks and need to be adapted.

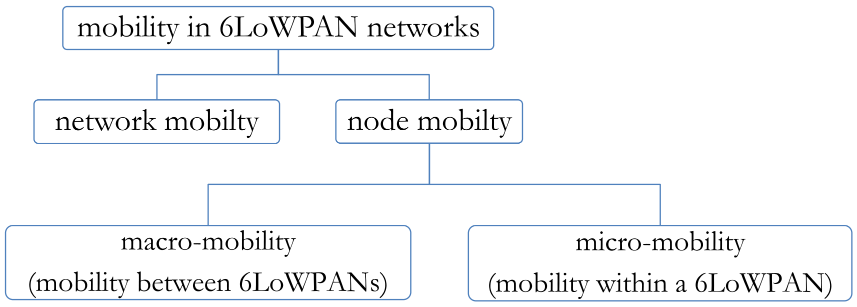

mobility in 6LoWPAN networks
network mobilty
node mobilty
macro-mobility
(mobility between 6LoWPANs)
micro-mobility
(mobility within a 6LoWPAN)

Figure 2-17 Mobility in 6LoWPAN networks

2.4.7 6LoWPAN as a technology

Despite being heavily related to IEEE 802.15.4 standard, 6LoWPAN is now being referred to as a technology and its techniques are being adopted by an important range of LLN technologies. For instance, the 6LoWPAN working group is substituted by 6Lo [70] which is expanded to investigate IPv6 packet transmission over BLE links [36], Z-wave technology [37], and many other constrained network technologies shown in Figure 2-18. In addition, a new working group has recently been chartered by the IETF to investigate transmission of IPv6 packets over the TSCH mode of the IEEE 802.15.4e amendment. This group is named 6TiSCH and is standardising IPv6 for industrial applications to realise the so-called industrial IoT.

With 6LoWPAN, it becomes possible to build low-power IP networks over constrained wireless links and, therefore, expand the reach of IP even further. However, while 6LoWPAN specified how to use IPv6 over constrained links, it lets it open for other work to define upper-layer protocols. The following section introduces the main subsequent standardisation efforts relevant to this thesis.

2.5 Other standardisation efforts

The standardisation of IPv6 over low-power networks has opened doors into realising the IoT vision. Thus, many international standardisation bodies are working together to push this vision from academia to industry. At the IETF, for instance, many working groups have been chartered, as can be seen from Figure 2-18. Two of these efforts are of interest to this thesis and are discussed below.

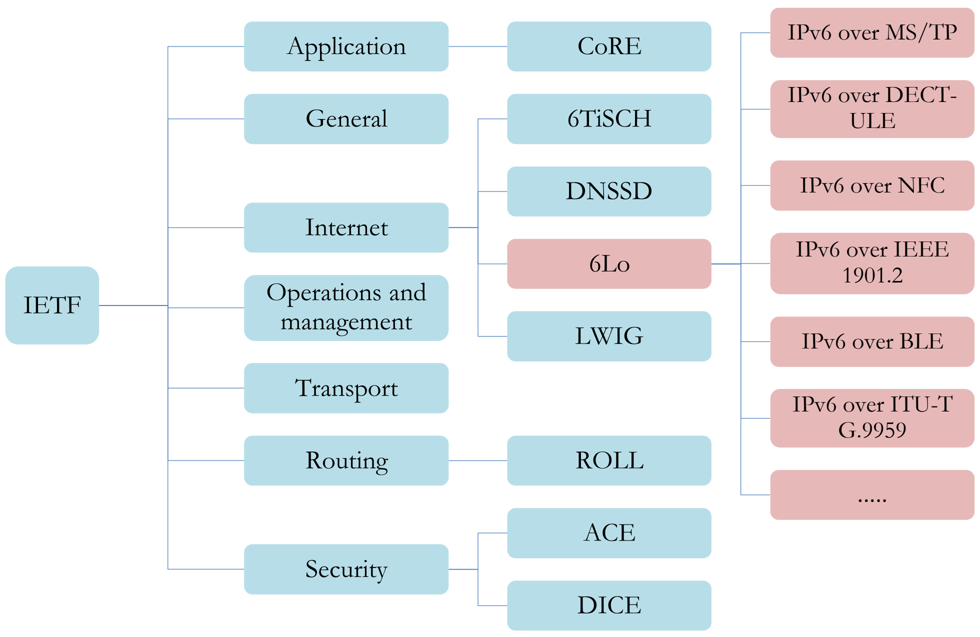


Figure 2-18 CNN-related standardisation at the IETF

2.5.1 IETF ROLL and the RPL routing protocol

Foreseeing the importance of standardised routing protocols in providing interoperable solutions for LLNs, the IETF chartered Routing Over Low-power and Lossy networks (ROLL) working group in 2008 to investigate adaptation or design of a routing protocol for LLNs. The working group concluded that there was a requirement for a new routing standard to address LLN challenges. ROLL decided to adopt a route-over approach and hence proposed RPL: the IPv6 Routing Protocol for LLNs. RPL's main specification, RFC 6550, was published in 2012 [71]. By adopting the route-over approach, RPL can bridge together multiple subnets that might be composed of one or more constrained-network technologies with other wired or wireless technologies in one network.

RPL functions by building and maintaining a Destination Oriented Directed Acyclic Graph (DODAG) based on an objective function. By building the DODAG, RPL can support three types of traffic: multi-point-to-point, from the nodes to the DODAG root, as shown in (Figure 2-19 left), point-to-multi-point from the DODAG root to the nodes, and a non-optimised point-to-point pattern whereby traffic passes up by the DODAG root and down to the nodes in the case of a non-storing mode (Figure 2-19 centre) or in the case of a storing mode, by a common parent (Figure 2-19 right). Note that an optimised reactive point-to-point RPL specification is proposed in RFC 6997 [72]. In order to detect and avoid routing loops, RPL uses a data path validation mechanism that ensures avoiding loops when forwarding the data packet. To do so, every data packet transports a RPL packet information including the rank of the transmitter in the DODAG. A receiver observing a rank inconsistency concludes a potential routing loop and initiates a local repair [71].

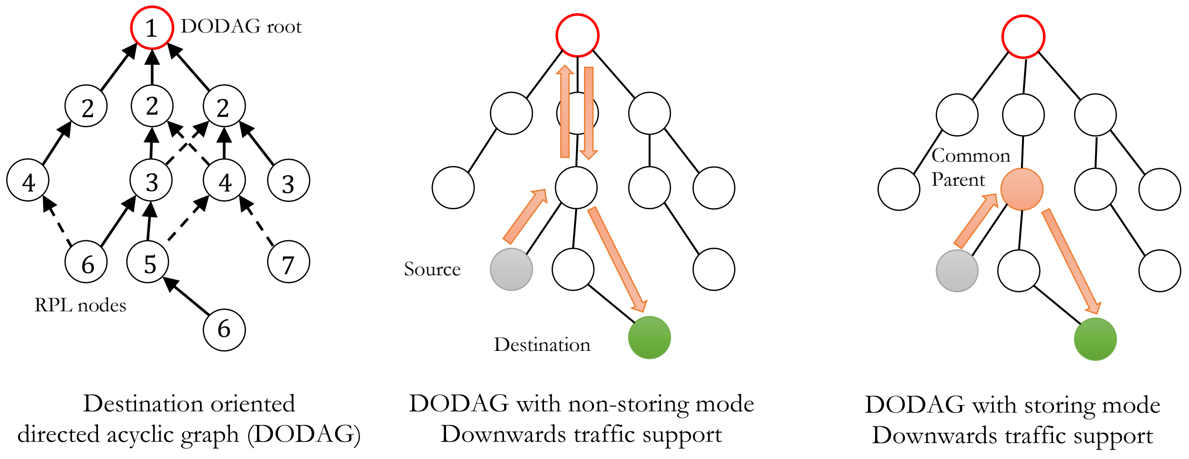

1
DODAG root
2
2
2
4
3
4
3
6
5
7
RPL nodes
6
Destination oriented directed acyclic graph (DODAG)
Source
Destination
DODAG with non-storing mode Downwards traffic support
Common Parent
DODAG with storing mode Downwards traffic support

Figure 2-19 RPL topology and architecture (reproduced from [73])

Thanks to its flexibility, RPL has become the de-facto routing protocol in LLNs. Thus, it proposes many objective functions using several metrics to tune its usage for specific LLN applications. However, by building and maintaining the DODAG, RPL is not particularly optimised to support mobile 6LoWPAN networks. Being the only routing protocol currently standardised and implemented for LLNs, RPL is the underlying routing protocol used in the evaluations of this research.

2.5.2 The Trickle algorithm

To minimize routing control traffic, RPL relies on the well-known Trickle algorithm [23], which has emerged as a basic networking primitive that can ensure fast and reliable resolution of data inconsistencies with low maintenance cost, while scaling well with network density [74]. For its usefulness as a generic algorithm in LLNs, ROLL also standardised Trickle as an Internet standard in a separate RFC 6206 [24].

Beside its deployment to manage routing control traffic frequency in RPL and CTP [75], Trickle is used in many applications including reliable broadcast/dissemination [76]–[80]. For instance, the IPv6 Multicast Protocol for LLNs (MPL) [80], being currently standardized by the IETF, heavily rely on Trickle to achieve cost-effective reliable multicast in LLNs. Furthermore, Trickle is the state-of-the-art algorithm used in dissemination and over-the-air programming protocols in WSNs. It is the heart of Deluge [76], Dip [77], Drip [78] and DHV [79]. Moreover, Trickle is delivered as a standard library in major WSN operating systems such as TinyOS and Contiki.

A node using Trickle periodically broadcasts its data unless it has recently heard identical ones. As long as nodes agree on what data they have, Trickle exponentially increases the transmission window and enters a maintenance mode with infrequent transmissions (for the sake of detecting inconsistencies). When data disagreements are detected, Trickle enters a propagation mode and starts transmitting more quickly. To realise this behaviour, and as by [24]'s notations, Trickle maintains three variables namely:

- a consistency counter c,
- an interval I,
- and a transmission time t within I.

In addition, it defines three configuration parameters namely:

- the minimum interval size $Imin$,
- the maximum interval size $Imax$, and
- a redundancy constant k.

When Trickle starts, it sets c to zero, I to a random value between $[Imin;\ Imin \times 2^{Imax}]$ and picks t from $[I/2;\ I)$. Picking t from the second half of the I interval allows for a listen-only period which avoids the short-listen problem [23]. Whenever a node hears the same data (dotted lines in Figure 2-20), it increments c. At time t, a node transmits (dark box in Figure 2-20) if and only if c is less than k. Otherwise, the transmission is suppressed (grey box in Figure 2-20). When I expires, Trickle doubles the interval length up to the time specified by $Imax$. Finally, if a node hears an inconsistent data and I is greater than $Imin$, I is set to $Imin$. Otherwise, Trickle does nothing. Whenever I is set (a new interval begins), c is reset to zero and t to a random value in $[I/2;\ I)$. This behaviour can be expressed by the following 6-step algorithm introduced in RFC 6206 [24].

- $\boldsymbol{Step}\ \mathbf{1}$: When Trickle starts execution, it picks I uniformly at random from $[Imin;\ Imin \times 2^{Imax}]$ and begins the first interval.
- $\boldsymbol{Step}\ \mathbf{2}$: At the start of an interval, Trickle resets c to 0 and picks t uniformly at random from $[I/2; I)$.
- $\boldsymbol{Step}\ \mathbf{3}$: Whenever a node hears a consistent transmission, Trickle increments c.
- $\boldsymbol{Step}\ \mathbf{4}$: At time t, Trickle transmits if and only if c is less than k ($c < k$). Otherwise, the transmission is suppressed.
- $\boldsymbol{Step}\ \mathbf{5}$: At the expiration of an interval, Trickle doubles the current interval size I up to the time specified by $Imax$. Trickle then starts a new interval as in $\boldsymbol{Step}\ \mathbf{2}$.
- $\boldsymbol{Step}\ \mathbf{6}$: If an inconsistent transmission is received while I is greater than $Imin$, the receiver resets the Trickle timer. To do so, Trickle sets I to $Imin$ and starts a new interval as in $\boldsymbol{Step}\ \mathbf{2}$. Otherwise, i.e. I was equal to $Imin$ when detecting the inconsistency, Trickle does nothing. Note that the timer can also be reset by application defined events external to Trickle.

For its simplicity, reliability, scalability and robustness, Trickle is adapted and adopted as the basis of the automatic SD solutions developed in this research. Moreover, this research introduces a generic optimisation of Trickle that addresses its main weakness concerning latency while preserving its strengths (see Chapter 5).

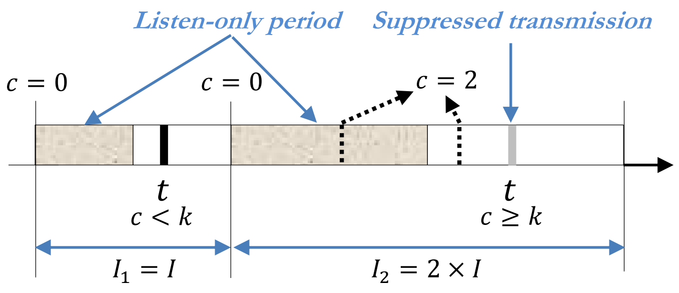


Figure 2-20 Trickle over two intervals with k = 1

2.5.3 IETF CoRE and the CoAP application protocol

In continuation of providing upper layer standardisation for constrained-node networks, the IETF has chartered the CoRE (Constrained RESTful Environments) working group in 2010 [12] to investigate the feasibility of the Representational State Transfer (REST) architecture [81] in constrained environments. REST is an architectural design style that reposes on the concept of resource that can be accessed via a Uniform Resource Identifier (URI). It provides a limited set of methods to manipulate resources in a stateless way. RESTful web services run many of today's web applications. Adapting the REST architecture to CNNs subscribes to a trend towards a web of things where constrained-device services can be simply accessed through a web browser.

CoRE proposed the Constrained Application Protocol (CoAP) which was standardised in June 2014 as RFC 7252 [82]. CoAP methods (standard GET, POST, PUT and DELETE methods) provide RESTful interactions while CoAP transactions ensure reliability. CoAP was designed such that messages can be easily translated from/to HTTP in order to foster integration of CNNs with the web. However, unlike HTTP which depends on the heavy TCP protocol to ensure reliability, CoAP operates, by default, over the lightweight UDP protocol and provides specific reliability mechanisms (CoAP transactions) as shown in Figure 2-21. As can be seen from this figure, CoAP can also be deployed over other transport layers and even over other link technologies. The flexibility and interoperability features afforded by CoAP have attracted extensive interest from the research community, and it is because of these same features that CoAP is being considered in this thesis. Indeed, the mechanisms developed in this work might be used to allow discovery of services working over CoAP. An attempt at such integration is described in Chapter 7.

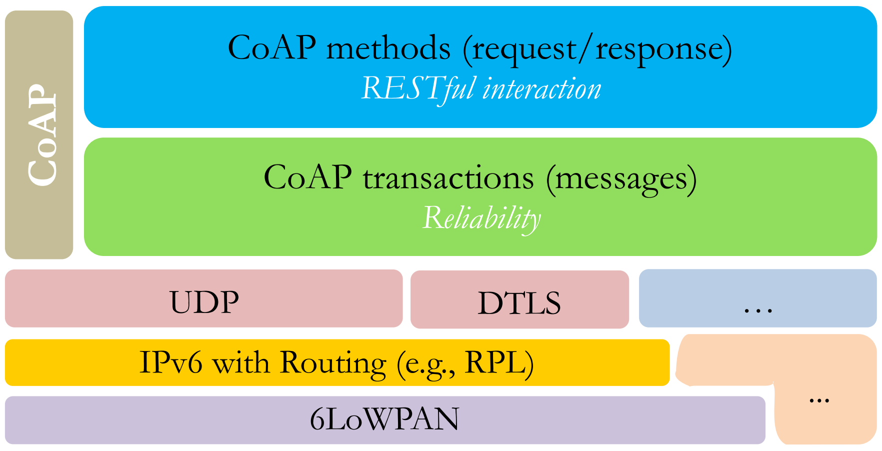

CoAP
CoAP methods (request/response)
RESTful interaction
CoAP transactions (messages)
Reliability
UDP
DTLS
…
IPv6 with Routing (e.g., RPL)
…
6LoWPAN

Figure 2-21 Constrained Application Protocol (CoAP)

2.5.4 Service discovery at the IETF

Service discovery is one of the main components enabling CNN pervasiveness. It allows automatic discovery, control, and maintenance of services provided by constrained devices. For instance, both CoRE and a recently chartered IETF working group called DNSSD (Extensions for Scalable DNS Service Discovery) [13] specify service discovery in CNNs as a main goal. These works, however, are still in early stages. Pervasive SD in CNNs is also the primary focus of this research. The relationships of the techniques and contributions of this research to such standardisation efforts are highlighted in the following chapter. Furthermore, chapter 7 is devoted to showing integrations of the contributions of this research with some of the techniques being developed by CoRE and DNSSD in order to provide interoperable services in the IoT.

Having described the emerging standardisation efforts relevant to this work, the following section briefly introduces some other international activities in the field along with marketing alliances promoting the use of IP in CNNs.

2.6 International activities and research tools

Following the success of 6LoWPAN and related standards, many international activities, marketing alliances and research tools are being developed [83]. This section introduces some marketing organisations created to promote the use of IP-based CNNs commercially and then moves on to describes the main tools provided by the research community to design, develop and evaluate new contributions.

2.6.1 International activities and marketing organisations

Two well-known marketing alliances are promoting the potential of IEEE 802.15.4-based Internet of thing applications, namely the ZigBee and the IP for Smart Objects (IPSO) alliances, are introduced below.

- **ZigBee Alliance** [57]: ZigBee is a well-known alliance that accompanied the IEEE 802.15.4 standard from its introduction in 2003. For a long time, ZigBee was the only provider and maintainer of IEEE 802.15.4-based solutions. Recently, ZigBee created the ZigBee-IP stack to incorporate 6LoWPAN.
- **IPSO Alliance** [84]: IP for Smart Object Alliance was founded in 2008 to advocate the adoption of IP in devices and networks used in energy, healthcare and industrial applications.

Recently a new alliance called the Thread Group [85] has been created with the aim of providing reliable, secure and compatible connectivity to the products used in home automation systems. Thread relies on the 6LoWPAN technology to achieve such aims.

2.6.2 Research tools

This section focuses on the two most popular open-source platforms which are used in this project; namely TinyOS and Contiki OS. The simulators related to such systems are briefly discussed. Finally, the testbed platforms used to validate the contributions of this thesis are also described.

2.6.2.1 TinyOS, BLIP and TOSSIM

TinyOS [64] is a popular operating system designed for CNNs. It has a component-based architecture aimed at reducing code size to fit constrained nodes. TinyOS has a rich library of tools including network protocols. The event-driven execution model of TinyOS allows energy saving since executions are triggered by incoming events. A simplified architecture of TinyOS is shown in Figure 2-22 which depicts three key features: sensing, actuating and communication primitives. TinyOS is programmed in NesC (Network embedded systems C) language [87]. A 6LoWPAN implementation called BLIP (Berkeley Low-power IP) is also available for TinyOS.

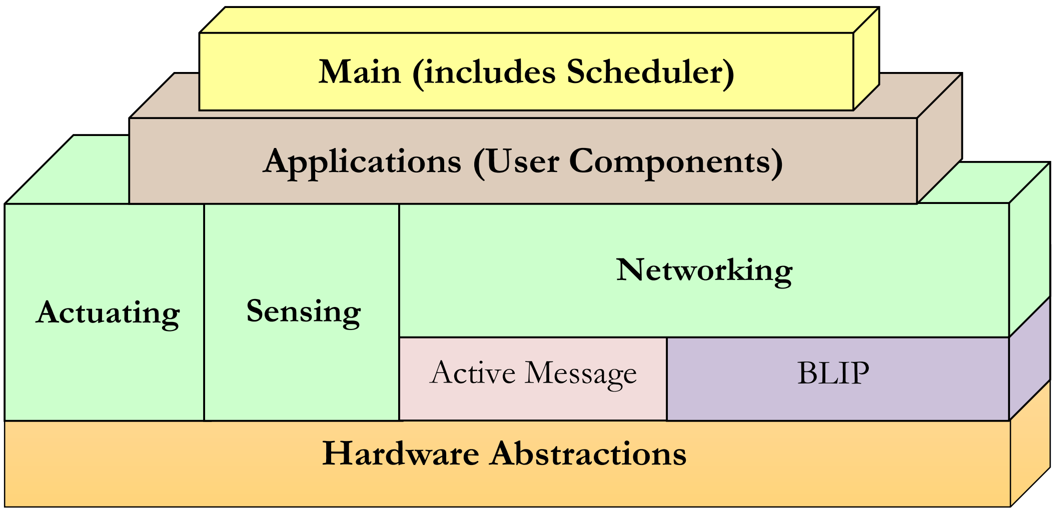


Figure 2-22 Simplified architecture of TinyOS extended with BLIP

In order to develop and evaluate new designs before deployment, network simulators are indispensable. Indeed, network simulators offer full visibility and control, allow bugs to be discovered early, and save time and cost.

TinyOS provides a discrete simulator called TOSSIM [88]. TOSSIM simulates the behaviour of the MicaZ platform at the bit-level. It uses an empirical signal-to-noise curve to decide on the success of a transmission. TOSSIM also simulates the noise and interferences present in the evaluation environments, which greatly improve the quality of the simulation. However, for scalability reasons, TOSSIM does not emulate real hardware executions, which might diminish its accuracy. Thus, it might not reflect the exact behaviour of time-sensitive operations such as RDC. Finally, interaction with TOSSIM is done via the command line interface.

Given its code maturity, TinyOS is used to evaluate some of the contributions of this thesis, particularly parts of Chapter 5. However, having a recent 6LoWPAN implementation, TinyOS is not yet mature enough to evaluate the 6LoWPAN-based contributions and therefore Contiki OS is used as the main development platform.

2.6.2.2 Contiki OS, uIP and Cooja

Contiki [65] is an open-source operating system targeting CNNs, designed by the Swedish Institute of Computer Science (SICS). Contiki includes a relatively mature micro IP implementation (uIP), along with a 6LoWPAN implementation (SICSlowpan). Unlike TinyOS, which has developed its own language, Contiki uses C. The architecture of

Contiki is shown in Figure 2-23. The networking stack provides support for non-IP networking via the Rime stack and for IP networking using either uIPv4 or uIPv6 over 6LoWPAN. Upper layer protocols in Contiki include UDP and CoAP. Lower layer stacks contain numerous MAC and RDC implementations.

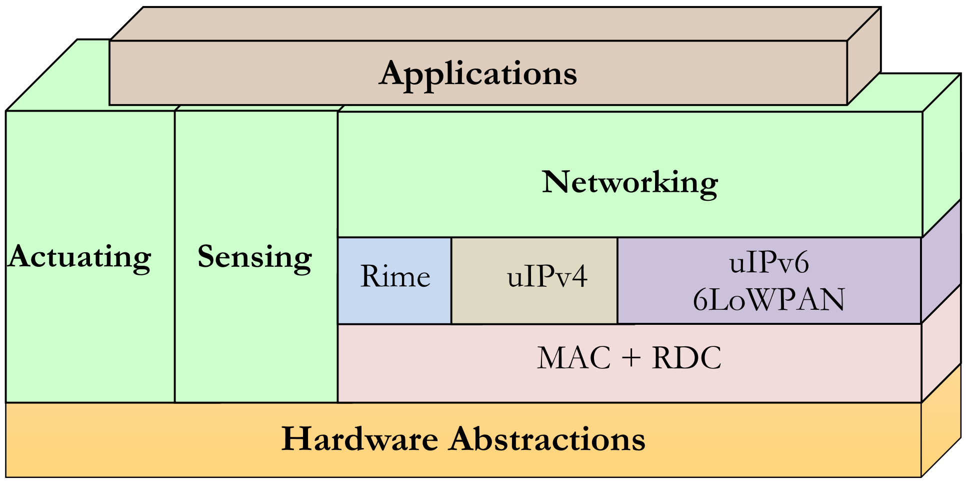


Figure 2-23 Simplified architecture of Contiki OS

Contiki provides a flexible simulator called Cooja [89] which combines instruction-level emulation of the Tmote Sky mote [90] components with the network simulator to provide accurate simulations. Cooja currently provides three models to simulate radio connectivity, namely: the unit disk graph medium, the directed graph radio medium, and the multi-path ray-tracer medium. In addition, it offers various tools to debug and analyse a network via both graphical and command line interfaces. By emulating real-hardware executions, Cooja can provide accurate timing for time-sensitive operations such as RDC, but at the expense of scalability. Thus, compared with TOSSIM, Cooja simulations take much more time and are very dependent on the number of nodes being emulated.

2.6.2.3 Local and public testbeds

Because there might be huge differences between real networks and simplified simulation models, testbed evaluations are a critical part of many experimental methodologies. In this study, a local testbed, as well as public large-scale testbeds such as Indryia [69], were used in conjunction with network simulations. Indriya is a public large-scale testbed provided by the national university of Singapore. It currently contains around 127 active motes irregularly deployed in a three-floor building as shown in Figure 2-24. Details of the configurations of each testbed are introduced in corresponding chapters.

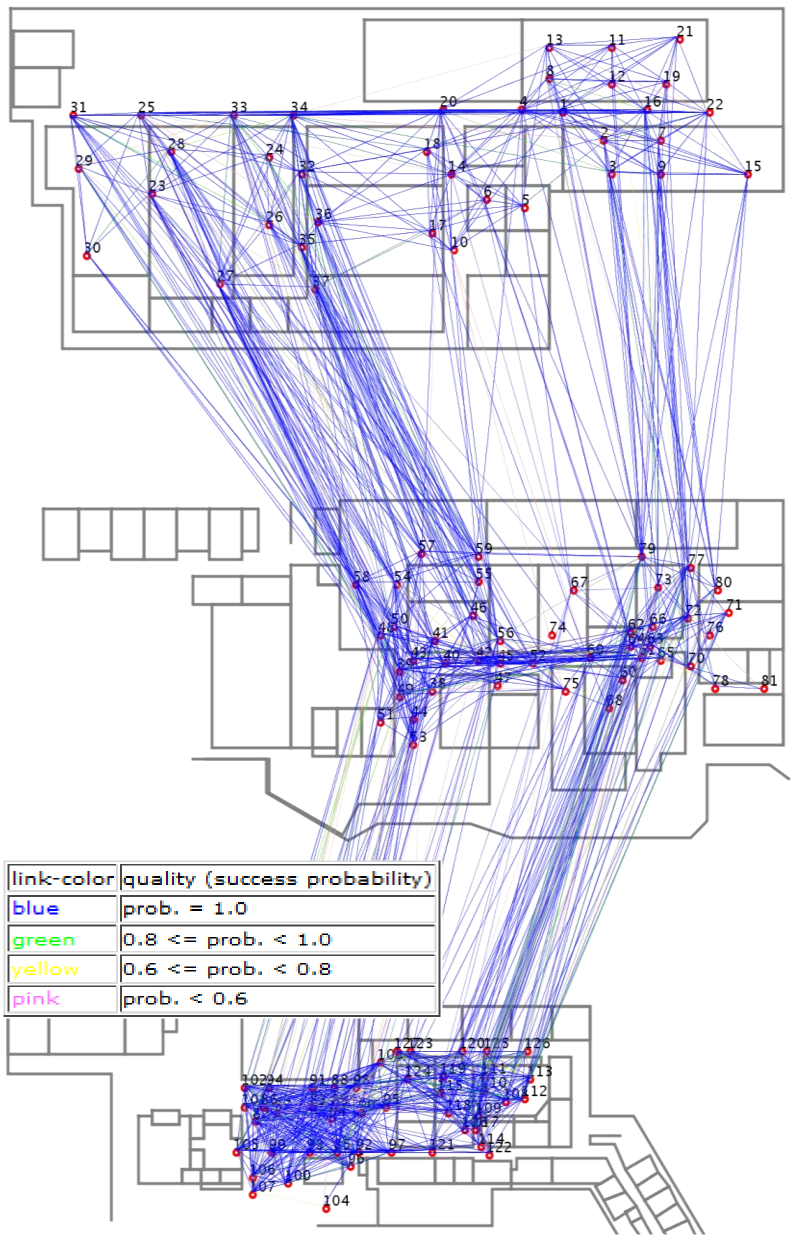


Figure 2-24 Layout of the Indriya testbed [91]

2.7 Summary

This chapter has focused on CNN technologies, standardisation efforts and the emerging Internet of things concept. One of the features that will make the IoT a reality is providing higher layer interoperable solutions. As mentioned earlier, service oriented architectures promise to offer such interoperability. The following chapter briefly presents efforts applying service oriented architectures to constrained-node networks and then tackles the problem of service discovery in CNNs as a fundamental element in making successful interoperable interactions in the IoT.

Chapter 3

Service Discovery in Low-power and Lossy Networks

The trend towards all-IP networks discussed in the previous chapter provides network-layer interoperability for the IoT. SOA is expected to provide the application-layer interoperability and hence promises to achieve IoT objectives. In a service-oriented system, the available capabilities are modelled as services. A service is provided by a service provider and used by entities called service consumers. To bring providers and consumers together, a service discovery protocol is required. This chapter presents the concepts of service and service discovery; scrutinises, classifies and discusses existing 6LoWPAN SDPs; and then demonstrates the need for new approaches to deal with service discovery in pervasive CNNs. Finally, it discusses the requirements and challenges of designing a new solution.

3.1 Services in CNNs

Non-interoperable ways of developing CNN applications in the first decade of the 21st century have prevented their wide adoption. To address this issue, All-IP networking, discussed in the previous chapter, and service oriented architectures are identified as key paradigms. Indeed, SOA makes it easier to develop flexible, reusable and interoperable applications based on the concept of service.

3.1.1 Service

A service is *"a piece of software that can essentially act as a container of related capabilities. It is comprised of a body of logic designed to carry out these capabilities and a service contract that expresses which of its capabilities are made available for public invocation"* [10]. By this definition, a service is mainly defined by a service contract (commonly referred to as service description) and the underlying logic which implements the provided capabilities.

Within this philosophy, a SOA application is composed of a number of services integrated in a loosely coupled manner that allows, on the one hand, higher flexibility and response to the changes in the environment, and provides, on the other hand, reusability of available services to create new applications. However, because of its resource consumption, SOA has been relatively neglected in CNNs so far [22]. With the IoT, SOA applicability in CNNs has been reinvestigated and hence the concept of Sensing/Actuating as a Service (SAaaS) has emerged.

3.1.2 Sensing/Actuating as a Service

In SAaaS, sensor and actuator network capabilities are modelled as services that can be discovered and invoked using standard methods. Figure 3-1 shows an example of typical services provided by a constrained device.

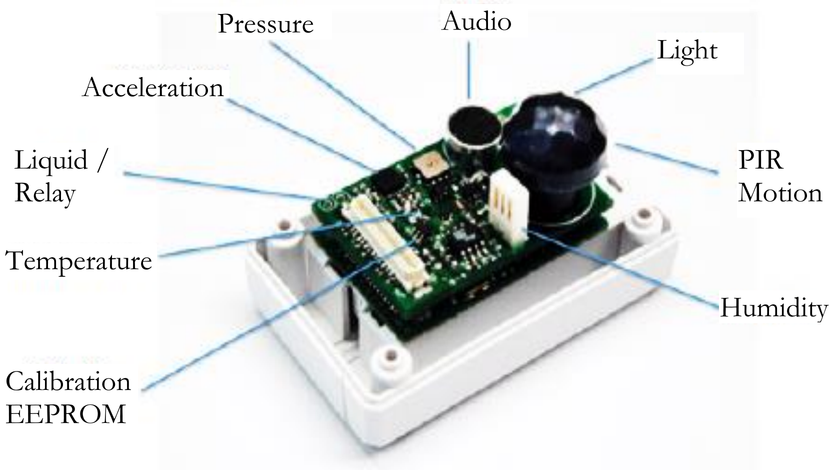


Figure 3-1 Sensing/Actuating as a Service [11]

Many frameworks have been proposed to realise this paradigm in CNNs [92], [93]. However, such efforts develop partial proprietary solutions, which have prevented their growth. Recently, CoRE, discussed in the previous chapter, has been chartered to provide a standard way of adopting SOA principals in CNNs using the REST style. To realise loose-coupled SOA applications in LLNs, service discovery is mandatory.

3.2 Service discovery

The following subsections define service discovery and then present its objectives, importance, and main entities.

3.2.1 Service discovery process

Service discovery is the process of automatically locating suitable services that can meet the requesters' needs. It involves locating requested services, retrieving service descriptions and executing a matchmaking algorithm between those descriptions and the requests, and finally selecting the most relevant services. The result of a service discovery process is the address of potential service providers that can offer the requested service. When the address is retrieved, the client may further access the service (service delivery). Depending on the requests and available services, there might be two distinguishable conceptual types of discovery:

- **One request, one response:** A client is looking for at least one service that matches its needs. The service discovery process might terminate when a service is found.
- **One request, all responses:** the client is interested to know all available instances that fulfil its needs. In this case, the service discovery process should continue until all available instances are visited.

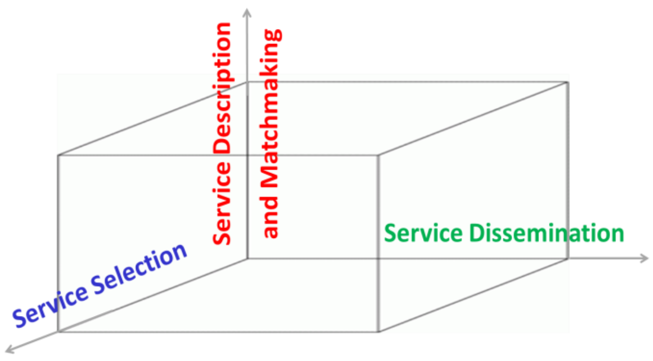


Figure 3-2 A simplified service discovery framework

SD is truly a multi-dimensional issue. Figure 3-2 presents a simplified framework to decompose the complexity of SD in order to help understanding, categorizing and comparing SD protocols. As can be seen in this figure, SD has at least three components namely:

- **Service dissemination:** Responsible for defining node interactions, discovery architecture, and request forwarding rules…etc. It can be rather complex so that a large number of SDPs only address this component [94]. This component is the primary focus of this chapter and subsequent work.
- **Service description and matchmaking:** This component is considered as the foundation of SD [94] as it shapes service information. It is the container describing available services, and it is responsible for finding the (syntactic or semantic) match between requests and service descriptions.
- **Service selection:** Service selection is responsible for selecting the most relevant services that match the client request. This component decides on the best fitting services depending on service, network, and user contexts.

This decomposition might also provide a philosophy for designing new solutions [94]. For instance, a new SDP for mobile 6LoWPANs may focus on the dissemination part while taking the description and matchmaking component from an existing protocol.

3.2.2 SD perspective of CNN services

This section categorizes service interactions in LNNs from an SD perspective. The resulting types below are for illustrative purposes to infer the requirements of SD in LLNs. Throughout this document, the concept of service is abstracted, and its definition will depend on the context, as will be detailed in Chapter 7. In all cases, the types of service interactions can be categorized into:

- **Simple services:** Simple services provide support for simple service interactions where a single data is required at the invocation time (e.g., sensor readings: temperature, humidity, pressure). In this case, the requested value might be piggybacked in the discovery reply.
- **Alert services:** This enables sending alerts when an abnormal situation occurs. The discovery process is needed for locating gateway services (e.g., Internet access gateways) in order to announce the event.

- **Complex services:** This case handles complex service interactions such as the history of a specific measure (e.g., temperature monitoring over the last hour) or complex sensor readings (e.g., multimedia services). This case requires the establishment of a communication session between the provider and the consumer. Thus, it engages in addition to discovery, a delivery phase.
- **Broadcast services:** Broadcast services allow sending a command to a group of nodes, configuring the network, etc. In this case, a reply may not be required. A request containing necessary attributes might be sufficient.

Thus, unlike traditional client-server unicast discovery approaches, SD in 6LoWPANs should cover the typical service interactions above, by providing both unicast and multicast discovery support. For instance, the former responds to the request for finding a temperature sensor in a room (simple service) while the latter fulfils the need to switch on/off all lights in a room (broadcast service).

3.2.3 Service discovery entities

Having outlined the importance of SD and types of service interactions in LLNs, this section presents the main entities involved in an SDP, namely:

- **The client** (or user, service consumer): The entity (application/person) that is looking for a service. In service discovery protocols, the client's role is typically represented by a User Agent (UA) that issues requests on behalf of the user.
- **The provider** (or server, service provider): The entity that offers the service. In SDPs, typically, the Service Agent (SA) is the process acting on behalf of the provider. It is responsible for publishing service information and issuing responses to the client when a matching service has been identified.

In order to facilitate discovery, a third entity called service directory is generally deployed.

- **The directory**: A network element dedicated to host, partially or entirely, descriptions of available services on behalf of the nodes. This role is generally realised by a Directory Agent (DA) which is responsible, in addition to managing the directory database, for advertising its presence to the network. It offers registration interfaces for providers and lookup services for clients.

These entities cooperatively participate in achieving SD objectives. However, the directory is employed differently in different SOA-based environments depending on their requirements. Thus, in infrastructure-based networks (Figure 3-3) the directory is an independent entity. In pervasive environments, however, devoting a fixed entity is hardly applicable and hence other approaches, explored below, can be adopted.

Independently of the adopted discovery approach, two main functions are performed by SDPs namely: the registration of service descriptions and the lookup process. While, in traditional SD approaches these two functions are mandatory, in pervasive environments an SDP may have either one or both.

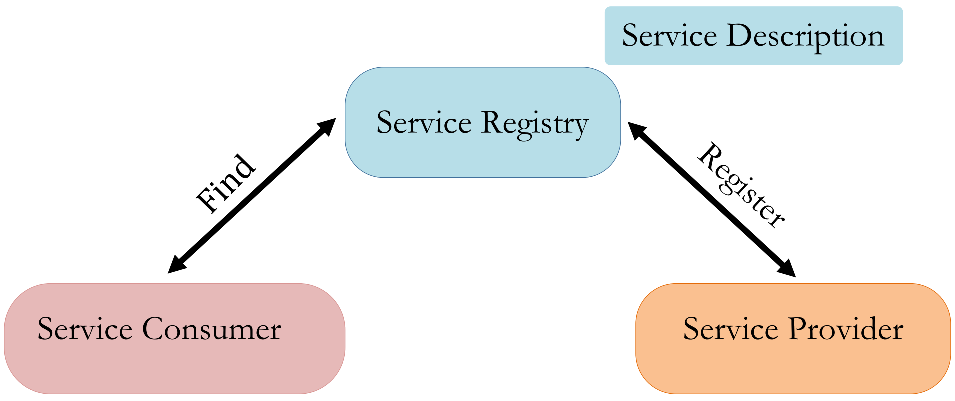


Figure 3-3 Traditional service discovery architecture

3.3 Service discovery in CNNs: review and classification

Service discovery has attracted a significant amount of research in both industry and academia; thereby many SDPs have been proposed. For instance, [95] surveys about 200 discovery frameworks. It is outside the scope of this chapter to discuss them all; it suffices to mention that none of these protocols are related to or have properties that make them suitable for use in 6LoWPANs. The remainder of this chapter focuses on protocols related to 6LoWPANs. Since 6LoWPAN SDPs are still immature [96]–[99], and for the sake of completeness, some representative SDPs deployed in other systems are considered, as depicted in Figure 3-4 (a).

3.3.1 Classification

Numerous SD classifications have been proposed in both traditional IP Networks (e.g., Internet, LAN) and ad-hoc networks (e.g., WSN, MANET, P2P networks) [95], [100]–

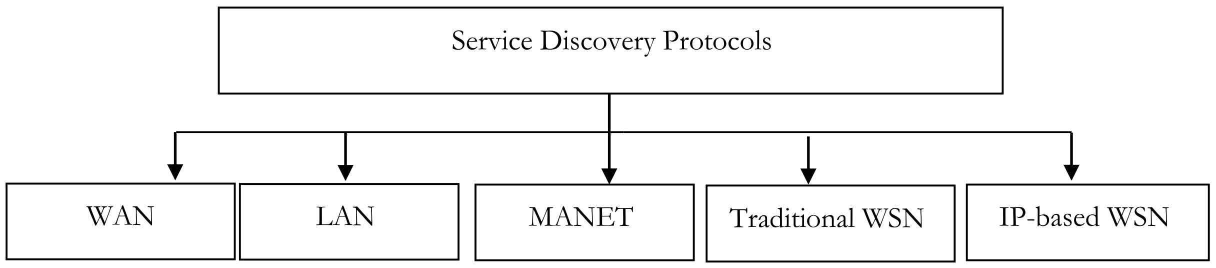
[103]. Those classifications are based on different aspects of SD including network structure, protocol design, service descriptions, service matchmaking and discovery scope. This chapter presents a classification of 6LoWPAN SDPs, which is discovery architecture and service dissemination driven. Later in this chapter, relevant matchmaking and description format are considered.

With a focus on the service dissemination component (section 3.2.1), SDPs classes and sub-classes are distinguished. Figure 3-4 (b) shows the principal classes of SDPs from a discovery architecture perspective that can be: centralised-directory based; distributed-directories based; or fully-distributed direct approaches. Direct approaches can be further classified as pull-based, push-based and hybrid push-pull models. Note that the sub-classes distinguished in Figure 3-4 (c) are intended to provide a more comprehensive categorisation of the proposed principal classification. Those sub-classes could be further detailed for specific needs. For instance, integrated protocols can be specified into:

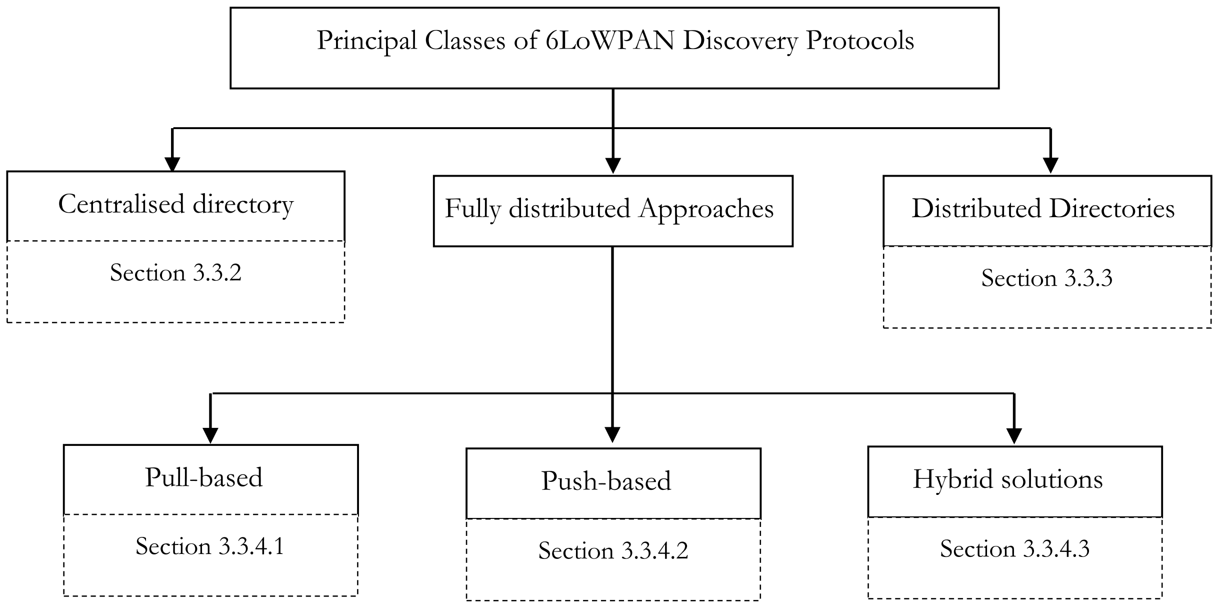
- dissemination-description integration;
- discovery-delivery integration and;
- routing-discovery integration (cross-layer design)

While the dissemination-description and discovery-delivery integration can still fit the layered design of 6LoWPAN networks, the routing-discovery integration which is heavily investigated in ad-hoc environments [104]–[107] is hardly applicable. This is justified by the fact that binding an SDP to a specific routing protocol violates SOA, in general, where the interoperability is preferred over optimisation [108]. It also violates the 6LoWPAN architecture, in particular, where the layered design is the main characteristic allowing seamless integration with traditional IP networks [109]. To avoid breaking the layered design and hence be applicable in heterogeneous networks, this chapter focuses on application-layer SDPs.

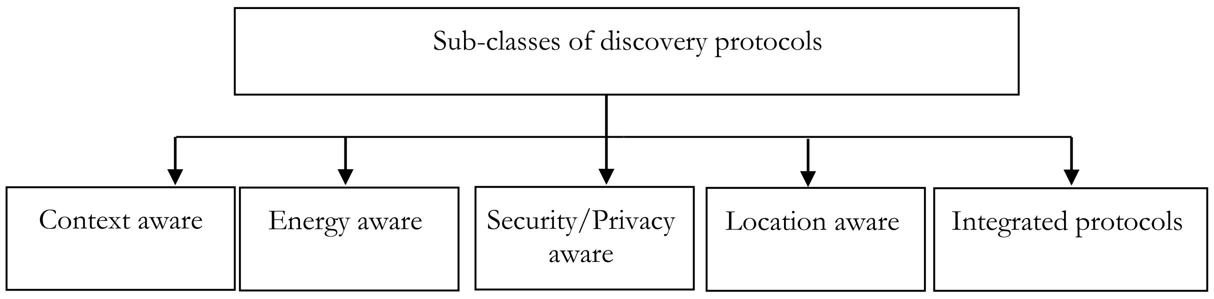
The following sections review and discuss state-of-the-art SDP protocols. They are organised according to the main classification (Figure 3-4 (b)). The sub-classes will be referred to in the text describing each protocol. Note that this classification is non-exclusive. Thus, some SDPs such as SSLP [110] can fall under more than one category by supporting both directory-based and directory-less approaches.



(a) Service discovery protocols



(b) Principal classes of service discovery protocols in LLNs



(c) Sub-classes of service discovery protocols for LLNs

Figure 3-4 The proposed service discovery classification

3.3.2 Centralised directory-based protocols

The centralised discovery approach is mainly deployed in wired, large-scale traditional IP networks. Thus, various industry-based SDPs adopt this approach to ensure proficient SD. Examples include Service Location Protocol (SLP) [18] and Universal Description Discovery and Integration standard (UDDI) [19].

Some of the recently introduced 6LoWPAN SDPs [110]–[112] also adopt the centralized approach as an architectural design. These protocols are intended to operate in infrastructure-based environments such as home automation systems, intelligent buildings, and smart cities. However, it is argued that even in these environments directory-based approaches suffer from single-point of failure and hence backup schemes should be supported.

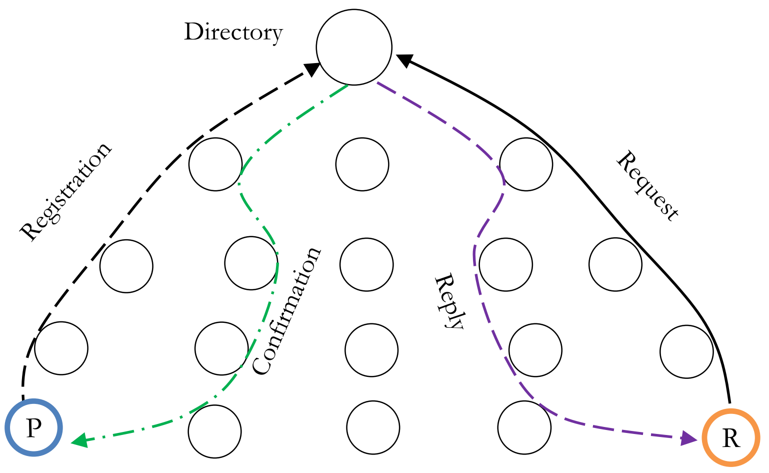


Figure 3-5 Centralised-directory-based service discovery

- **SSLP** [110]**:** SSLP is a lightweight version of the traditional SLP protocol. SSLP mainly relies on a central directory in order to store available service information, although it proposes a basic fully-distributed mechanism for small-size networks. In addition to the user agent, service agent and the directory agent, SSLP introduces the concept of translation agent to perform translations between SSLP and SLP services. This latter can allow seamless integration between 6LoWPANs and traditional IP networks operating over SLP. However, it introduces complexity and delays [96].
- **TRENDY** [111]: TRENDY is a centralized SDP designed to discover services working over CoAP. TRENDY chooses the 6LoWPAN border router as DA 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). This mechanism is used by TRENDY in responding to the new requirements introduced by CNNs such as group discovery (broadcast services, section 3.2.2). However, in addition to 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.

- **Resource Directory (RD)** [112]: the CoRE working group has proposed the RD to deal with service discovery in CoAP-based networks. Like TRENDY, RD (work in progress) stores all resources offered by CoAP servers, so, requesters can discover any required resource just by querying the RD. However, to be able to use the RD functionalities, CoAP nodes must first discover its presence in the network.

Most of the SDPs mentioned above assume the presence of a resource-rich central directory able to store all available services and are mainly targeting static networks. While this approach is vital for an infrastructure based environment, it is hardly used in distributed, dynamic environments. Not only because central directories are demanding in terms of processing, storage resources, energy consumption and bandwidth utilization, but also because they may not support topology changes that can be frequent in many 6LoWPAN applications as result of node mobility, faults or dead-nodes (discharged battery). Thus, distributed directories and directory-less (direct) approaches were explored [21], [97], [113], [114].

3.3.3 Distributed-directories-based protocols

Protocols in this category employ clustering and overlay techniques to construct hierarchical structures holding the distributed backbone of directories. Some of these protocols, e.g., [115], [116] exploit the underlying clustering mechanisms deployed at the routing layer. However, besides relying on routing mechanisms, such protocols build complex clustering structures requiring more resources and high maintenance efforts that do not fit the CNNs limitations [117]. To deal with this, specific lightweight clustering algorithms have been developed such as Cluster-based SD [117].

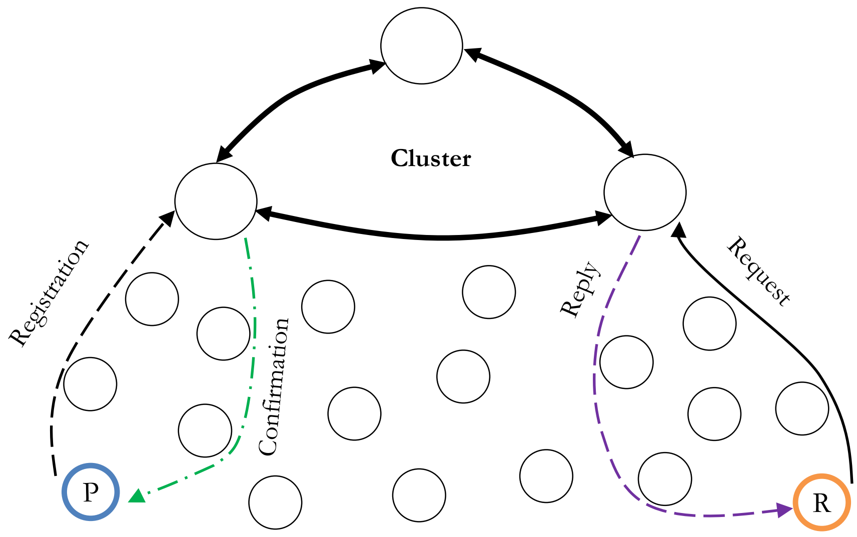


Figure 3-6 Distributed-directories-based service discovery

- **Cluster-based SD** [117]**:** Cluster-based SD is designed for heterogeneous non-IP-based WSNs. It proposes a lightweight clustering algorithm for building and maintaining a distributed backbone of directories. Cluster-based SD uses a two-step algorithm to support service discovery, namely cluster building and discovery process. The former assigns grades to nodes depending on their capacities and constructs independent sets of cluster heads. At this point, the latter can start. To this end, each node registers its services and the ones received from its children with its parent. When resolving a request, a node forwards it to its parent if no match has been found locally. When the request arrives at the cluster head with no match, it is forwarded to the head of adjacent clusters. Maintenance is triggered when an anomaly is detected. In addition to the cost of building and maintaining clusters, Cluster-based SD assumes the presence of resource-rich nodes to play directories' roles.
- **Context-aware SD**: The authors of [96] use vicinity information to enhance SD in 6LoWPANs. The proposed protocol supports the same architecture as SSLP and introduces the concept of Directory Proxy Agent (DPA) to manage the user and service contexts. Network elements are organized in a hierarchical manner, and multiple DPAs are considered to cache service information and contexts in

their vicinity. The information of DPAs is exchanged periodically. Clients are connected to the nearest DPA to find the closest services. Besides requiring the deployment of DPAs, this approach is fragile to network dynamics and generates a lot of traffic. In addition, the proposed scheme has complex mechanisms for accessing services outside the 6LoWPAN, which results in reduced performance [97].

- **ENUM-based SD** [97]: ENUM-based SD aims to discover services from inside as well as outside a 6LoWPAN. It uses a distributed directory approach based on the idea of resource-rich master nodes holding information about available services in their vicinity. Only master nodes are assigned Electronic Number Mapping (ENUM) [118] that allows them to discover and be discovered from outside the 6LoWPAN. Translation between incoming queries and the ENUM-based description is done at the gateway that also performs domain name conversion. Added to the introduced complexity, delays, and maintenance overhead, ENUM-based SD assumes the availability of powerful nodes to play the role of masters.
- **CoAP with RELOAD** [119]**:** The authors of [119] proposed the use of the REsource LOcation And Discovery (RELOAD) protocol [120] to discover services working over CoAP. RELOAD forms an overlay network to provide storage and messaging services in a P2P network. RELOAD lets it open for applications to define new use-cases. Thus, [119] describes a use-case on how to use CoAP with RELOAD in order to discover CoAP services internetworked over a wide-area geographical coverage. By relying on the RELOAD infrastructure, this approach is not applicable for the targeted scenarios.
- **Distributed Resource Directory (DRD)** [121]**:** a DRD is proposed in [121] to realise SD in CoAP-based networks. Alternatively to the RD, the DRD defines an overlay to play the role of RD. DRD is constructed using a Distributed-Hash-Table-based P2P (DHT-Based P2P) overlay offering discovery, registration and proxy services for CoAP nodes. However, the authors do not specify how to construct and maintain the overlay.

In summary, this approach generally assumes the availability of resource-rich nodes to play the role of distributed directories and needs synchronisation between them to keep service information updated, which might imply high maintenance overhead. Furthermore, responding to network dynamics incurs high discovery overhead as a result of re-clustering and service re-registration, making this approach less suitable for dynamic environments [97].

3.3.4 Fully distributed protocols

In the absence of any directory to store service information, nodes make use of multicast/broadcast of service requests/advertisements in order to realise service discovery. Three possible ways, shown in Figure 3-7, are considered in the literature to accomplish fully distributed SD, namely push mechanisms, pull mechanisms and hybrid mechanisms. This class handles mobility better [122].

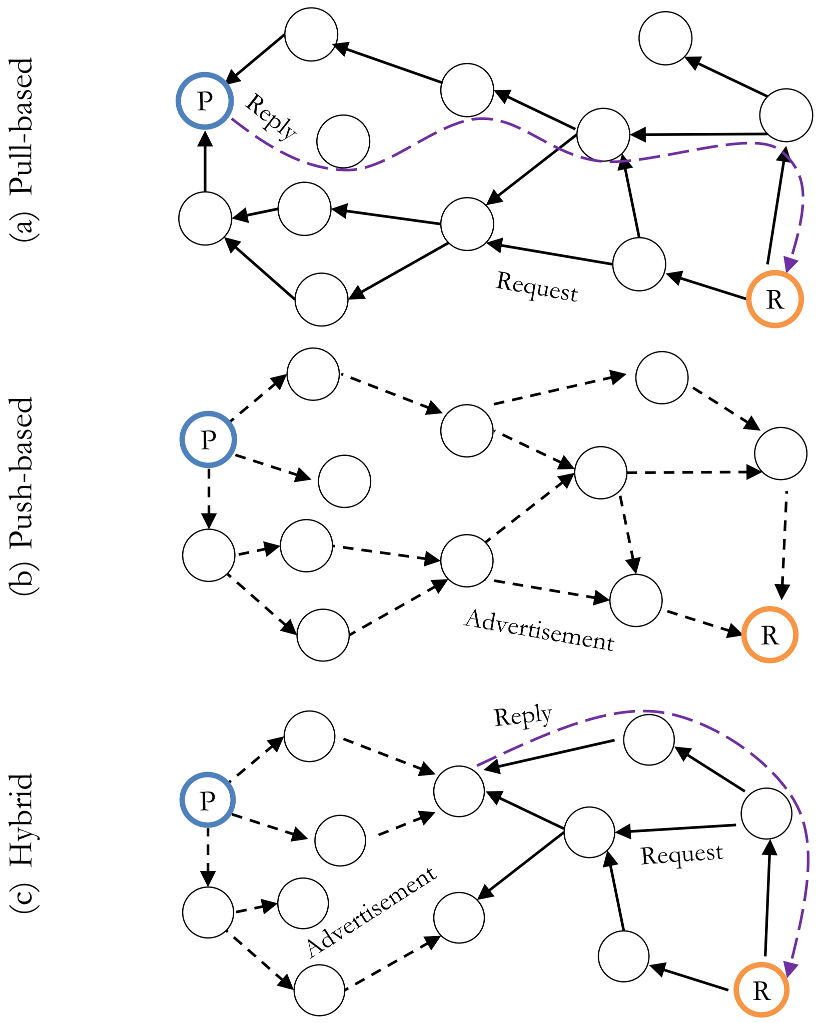


Figure 3-7 Fully distributed service discovery

Many research-based SDPs for ubiquitous environments fall under this category. Also numerous industry-based SDPs developed to operate in (W)LAN adopt this approach such as Apple Bonjour [14], OASIS Web Service Dynamic Discovery standard (WS-DD) [15] generally deployed in the Devices Profile for Web Services (DPWS) framework, and Microsoft UPnP [16] which incorporate the SSDP [17] protocol. These industry-based protocols were firstly developed for wired networks and then adapted to wireless environments. These protocols are mainly intended for unconstrained small resource-rich networks with limited dynamics. Thereby, they cannot be directly applied to CNNs. An attempt to use UPnP in CNNs operating over CoAP is proposed in [123]. Another attempt to use WS-DD in CNNs is described in the uDPWS framework[6].

Note that fully-distributed SD can be also realised via unicast using random walks variants. A typical main drawback of such approaches is that they require a node to have an accurate knowledge of its neighbourhood. Such a requirement cannot always be assumed in LLNs and trying to build one incurs more traffic, especially under mobility. For these reasons, such approaches are not detailed in this chapter.

3.3.4.1 Pull-based protocols

In pull approaches, also known as reactive or passive approaches, requests are issued on demand of a service and propagated across the network. Upon the reception of a matching request, a provider generates a reply containing information about the service and how to access it. Representative protocols in this category include:

- **SLIM** [113]: The Service Location and Invocation Middleware for Mobile wireless sensor and actuator networks (SLIM) incorporates a fully distributed pull mode SDP to discover services provided in mobile WSNs. SLIM relies on link-layer broadcast to forward service requests and/or replies. To minimise generated traffic, SLIM uses a delay-and-cancel algorithm based on RSSI of received packets. Such a mechanism might not guarantee discoverability.
- **NanoSLP** [114]: NanoSLP is a discovery protocol for non-6LoWPAN IP-based CNNs developed within the nanoIP stack [114]. Despite the apparent similarities

[6] http://ws4d.e-technik.uni-rostock.de/udpws/

with the original SLP protocol, NanoSLP does not support DA, which makes it a fully-distributed SDP. Communication between UAs and SAs is carried out directly, via unicasting or broadcasting. NanoSLP allows the service discovery and delivery integration. Thus, it allows piggybacking requested values in the reply message thereby optimising the latency by saving one round-trip time. However, NanoSLP develops its query language and uses a modified service/attribute SLP scheme for service descriptions that make it hard to integrate with other systems.

- **CoAP resource discovery** [124]: Resource discovery provides a pull-based SD mechanism to discover resources available in CoAP networks. Thus, as by [124] specification, a GET request to the appropriate multicast address might be made for $/.well-known/core$. Matching nodes reply with a payload in the CoRE link format (section 3.4.1). Note that in order to limit the number and size of responses, the request has to specify known attributes.

While, from a latency perspective, the pull mode is less efficient, it is suitable for high dynamic environments as it generates less traffic. However, having a big latency to discover services makes this class of protocols less reactive to highly dynamic environments. Take the example when a service is present in the network for a period of time, and towards the end of this period a requester asks to use this service. In pull mode, upon getting the information about the service, it may be unavailable for invocation. As a conclusion, and beside the latency problem, the pull-based solutions might be less efficient in multi-hop networks as the throughput decreases with increasing hop-count resulting in high delays and low discoverability rates.

3.3.4.2 Push-based protocols

In push mode, also known as proactive or active search, nodes use link-local broadcast to send unsolicited advertisement of services. DEAPspace [125] and service advertisement for mobile ad hoc networks (MANET) [109] are examples of such approaches. Bluetooth SDP [33] is another example of push mode based SDP intended to work over Bluetooth radios.

- **DEAPspace** [125]: DEAPspace provides a pure push model to realise service discovery in ad-hoc single-hop networks. DEAPspace divides the time into slots and advertises service information once in each slot.
- **Service Advertisement for MANET (SAM)** [109]: In multi-hop networks, SAM [109] employs a push model to realise discovery tasks in small size mobile ad hoc networks. SAM uses a fixed interval push to advertise periodically available service information to the whole network using IP multicast.
- **Push-only SD for 6LoWPANs** [22]: A very recent PhD thesis [22], carried out in parallel to this research, proposed a push-only SDP for 6LoWPAN networks. The initial version of the proposed scheme simply relies on a periodic push with fixed interval to advertise services in the network. An enhanced version based on Trickle was lately proposed and compared with the fixed interval periodic push. However, the discussion in [22] only considers one provided service in the network and mainly results in applying Trickle as-is (section 2.5.2). Furthermore, the push-only approach might suffer great scalability and performance issues when the number of nodes and services increases.

Applying the push mode makes the network aware of new services as soon as a device joins the network. Thus, allowing a node to find information about available service by only performing local lockup. This allows more time for invoking the service and making use of it, which increases service utilization. However, while the push mode reduces the latency considerably, it might introduce a large amount of traffic in order to keep the network updated about available services. In addition, it requires the nodes to cache information about all available services which consumes large memory resources, not available for CNNs.

Having presented the pros and cons of the pull- and push-based SDPs and having seen the completive relationship between them; a hybrid approach seems to be a promising candidate solution to investigate for the problem of SD in 6LoWPAN networks.

3.3.4.3 Hybrid push-pull-based protocols

Many ad hoc SDPs using hybrid mechanisms exist in the literature targeting mainly MANETs such as PDP [126], ADDER [127], Konark [128], GSD [129] and traditional WSNs such as NanoSD [21] and Imesh [130].

- **PDP** [126]: In PDP the pull mode is preferred; hence the nodes advertise their services only when other devices request them and/or when they issue service replies. While this mechanism allows the protocol to minimise push mode traffic, it neither ensures optimising the latency nor the accuracy [131]. Another point that makes PDP unsuitable for CNNs is its service description method based on using heavy URL schemes similar to the one used in SLP.
- **ADDER** [127]: ADDER adopts a hybrid approach in which nodes periodically advertise services in their vicinities and clients issue requests to locate services which are not available locally. While this method exploits the latency benefits of the push mode, it generates high overhead. In addition, ADDER has scalability issues regarding the number of services included in one advertisement. To deal with this, ADDER uses a fixed probability to decide which services should be included, which may compromise the performance realised in terms of latency. Thus, while ADDER uses a service description that suits CNNs, it introduces high overhead by using a fixed-period push.
- **NanoSD** [21]: NanoSD is designed to discover services in dynamic, mobile and heterogeneous WSNs. It borrows ideas from NanoSLP in developing its service description. NanoSD focuses on techniques to use compact packet and service descriptions sizes. However, like ADDER and NanoSLP, NanoSD does not propose advanced dissemination techniques to minimise traffic. Instead, it lets the users customize it depending on their needs. Thus, it supports the possibility of customising the advertisement period; the query attributes and uses delta advertisements reporting only recent changes. However, the NanoSD traffic minimisation mechanism is partial and only works in particular cases. In addition, developing a customised service description limits its interoperability with other systems.

- **mDNS/DNS-SD** [132], [133]: Domain Name System based Service Discovery (DNS-SD) [133] when employed with multicast DNS (mDNS) [132] provides fully distributed hybrid SD that constitutes the basis of zero-configuration networking. mDNS relies mainly on a pull mode approach to retrieve information and proposes a push mode via unsolicited responses enabling nodes to advertise their resources at start-up, wakeup from sleep, and when detecting network changes. Relying on multicast allows nodes to see responses to DNS queries and hence enables them to detect conflicting responses and keep their caches updated.
- **mDNS/DNS-SD in 6LoWPANs**: The authors of [134] have investigated the feasibility of mDNS/DNS-SD in 6LoWPANs and proposed a lightweight implementation in Contiki OS [65]. Another work was carried out in [135] where the suitability of mDNS/DNS-SD for constrained networks was again demonstrated. However, the authors reported overhead issues that prevent mDNS/DNS-SD from being directly applied in 6LoWPANs. To address such issues, they proposed compression of DNS records in [136]. The proposed optimisations include: (i) regrouping a set of records in a single DNS message; (ii) using adjustable DNS message compression; and (iii) proposing an enhanced DNS message compression.

Note, that the ZigBee-IP stack specifies mDNS/DNS-SD as a protocol for realizing SD [137]. Finally, it should be noted that while mDNS is intended for link-local scope [132], the above works do not specify how to use it in multi-hop 6LoWPANs. An attempt to specify such a usage is proposed in extended multicast DNS (xmDNS) [138]. The new IETF DNSSD working group also sets scalable DNS-SD in 6LoWPANs as one of its objectives.

The above hybrid protocols developed for multi-hop networks suffer from high generated overhead and/or latency issues. To address these problems, location-based SDPs have been investigated. For instance, Imesh [130], a distance-sensitive location-based SDP intended for static wireless sensor and actor networks, adopts a hybrid mode. In Imesh, service providers advertise their location information in four geographic directions. This information is further published by subsequent receivers. On the other hand, service consumers conduct a limited lookup process to discover nearby services. However, relying on location information limits Imesh applicability.

From the above analysis, and to the best of the author's knowledge, no hybrid fully-distributed SDPs have been considered for multi-hop 6LoWPAN Networks. Having seen the benefits of such an approach, this research investigated the applicability of such an approach to 6LoWPANs. The proposed solution will be described in Chapter 4. The following sections tackle the service description component and point out the design goals and the challenges for SDPs in 6LoWPANs. Table 3-1 presents a comparison of discovery architectures with regards to scope, support of sleepy nodes, overhead, the need for dedicated servers, and communication primitives used to realise discovery tasks.

Table 3-1 Comparison of service discovery architectures

		Zero-configuration	**Require Infrastructure support**	**Communication Primitive**	**Overhead**	**Local/Global**	**Sleepy nodes**
Central Directory		No	Yes	Broadcast and mainly unicast	Advertisement/ discovery of the registry Registration and re-registration	Both	Yes
Distributed Directory		No	Yes	Broadcast and mainly unicast	Building the Cluster or the overlays Registration and re-registration	Both	Yes
Fully distributed	**Pull**	Yes	No	Broadcast	None	Local only	No
	Push	Yes	No	Broadcast	Advertisement	Local only	Yes
	Hybrid	Yes	No	Broadcast	Advertisement	Local only	Yes

3.4 Service description and matchmaking

Service description and matchmaking is a central aspect of SD. Indeed, it is the main component ensuring seamless integration and zero-configuration. Traditionally, the Web Service Description Language (WSDL) is the de-facto standard to ensure interoperability and growth of web services. Such a standard is, however, very resource consuming and cannot fit CNN requirements. To address this issue, many descriptions optimised for CNNs have been developed. Examples include NanoSLP, NanoSD and ENUM-based SD. While, such descriptions can fit CNNs, they create the reverse problem of no longer being interoperable. Recently, new service description formats have been proposed for CNNs such as CoRE link format [124]. Also, the use of DNS-SD for the same is being considered by the DNSSD working group. These standards-based service descriptions, described in the following subsections, are of interest to this research as to provide interoperable operations between the proposed solutions and wider Internet services in the IoT. Integrations of such standards-based descriptions with the protocols and mechanisms developed in this work are demonstrated in Chapter 7.

3.4.1 CoRE link format

So far, the term service is used to refer to the capabilities offered in a service-oriented system. CoAP makes a distinction between service and resource. A resource in CoAP is any capability provided by a (constrained) node which accepts RESTful interactions that is identified by a URI and accessed via CoAP methods (section 2.5.3). A service in CoAP represent an endpoint (protocol, hostname/IP address, port) [99]. The CoRE link format is used to describe the resources hosted by constrained devices and their relationships. It is carried as a payload in the compact CoAP message format shown in Figure 3-8. A resource description in the link format has many resource attributes including resource type (rt), interface description (if) and path.

- **Resource type (rt):** The resource type is a string describing the resource hosted by a constrained server. Examples of resource types include 'kitchen-humidity' and 'outdoor-temperature'. One of the main usage of the rt attribute is to allow resource discovery [139].

- **Interface description (if):** This attribute is represented by a string describing the REST interface of a type of resources, and may include a link to a service description document. An if attribute might be shared by many resource types. For instance, the above resource types might be accessed via a well-known interface 'sensor'. Additional details on well-known interfaces and their descriptions can be found in [139].

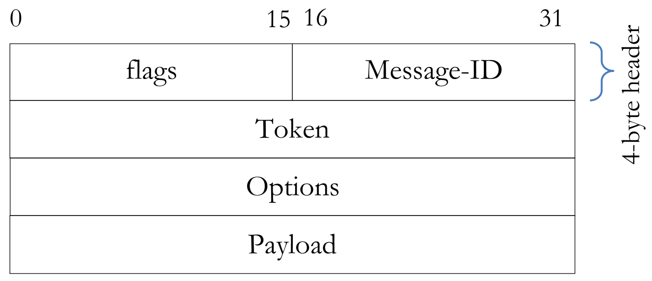


Figure 3-8 CoAP message format

To achieve resource discovery, each CoAP node provides a $/.well-known/core$ resource as the main access point to its offered resources. In order to discover provided resources, a client generates a Token (Figure 3-8) and issues a request to the $/.well-known/core$ in the following format $/.well-known/core\{?search*\}$, to be sent to the address of a specific node, if known, or to an appropriate multicast address. The provider(s) return(s) the list of matching resources in the CoRE link format in a response message containing the same Token. The client consults the Token contained in the received response in order to map it to the request.

Enabling multicast-based resource discovery allows achieving zero-configuration-like pull-only discovery over CoAP. However, this option may cause severe performance degradations since it can generate an abundant amount of traffic as a result of needless reply storms. To avoid such issues, RFC 6690 [124] recommends including known attributes such as rt and if in the $\{?search*\}$ argument to filter requested resources. If filtering is not present in a multicast request, a server should not respond to such requests [124]. However, RFC 6690 does not specify how such attributes are made known to the nodes.

3.4.2 Services over DNS

DNS-SD standardises the use of DNS Resource Records (RRs) [140] for the purpose of SD. When coupled with mDNS, DNS-SD provides the basis for zero-configuration networking. Although not specifically designed for CNNs, mDNS/DNS-SD features have attracted much of research in the field [134]–[138]. The following subsections briefly describe the DNS message format used to transport DNS-SD messages and then discusses the DNS-SD description.

3.4.2.1 DNS message format and features

A DNS message (Figure 3-9) is composed of a header of 12-byte size containing information about the message, the number and type of variable-size sections of the message (question, answer, authority, additional) together with flags and the message identification number. The same message is used for both requests and responses. A request message mainly contains the header accompanied with the question section while a DNS response message may contain, in addition to the header, the answer, authority and/or the additional sections [140].

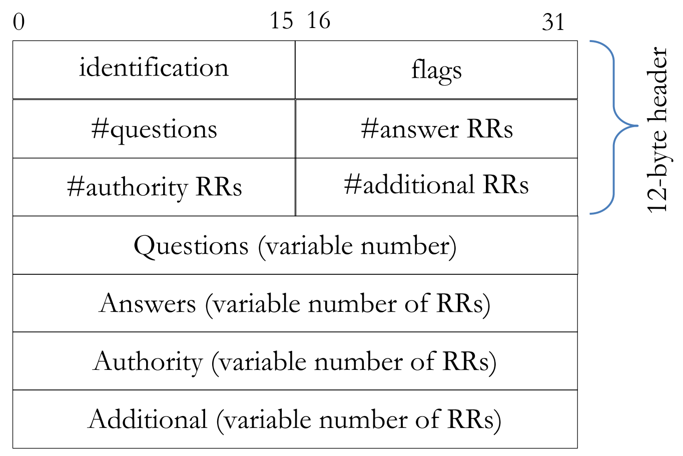

0
15
16
31
identification
flags
#questions
#answer RRs
#authority RRs
#additional RRs
Questions (variable number)
Answers (variable number of RRs)
Authority (variable number of RRs)
Additional (variable number of RRs)
12-byte header

Figure 3-9 DNS message format

The identification number in the DNS message header is a 16-bit integer that identifies a query. The following 16 bits represent flags that indicate whether the message is a request or response, whether there were errors or not, etc. The other parts of the header indicate the number of questions, answers, authority and additional records contained in the message. If a number is zero, the corresponding section is not present in the message. Otherwise, the answer section contains the RR of the responses. The two other sections are optional. The general format of an RR mainly contains the name, type as an integer, class, time to live (TTL), and resource data. Specific RR formats can be defined separately when required. Examples of RRs include IPv4/IPv6 address records (A/AAAA), pointer record (PTR), service locator record (SRV) and text record (TXT).

DNS allows the inclusion of resource records, believed to be useful for the client, in the additional section of a reply message to enable efficient network usage. Doing so, a lot of information might be redundant. To respond to this, DNS offers another feature that allows shortening names included a DNS message by using pointers to their prior occurrences [140]. This stateless name compression allows representing a repetitive occurrence using just 2 bytes. This way, it allows more records in one message, and it becomes more useful when more records are included in a single message.

3.4.2.1 DNS-SD

DNS-SD defines conventional usage of DNS messages and resource records to facilitate the discovery of services available in the network. It mainly specifies how a particular service instance can be described and accessed using PTR, SRV, TXT and A/AAAA records. The role of each record is represented in Table 7-1.

Table 3-2 Roles of DNS records in DNS-SD

Record	Role
PTR	Assigns instances to a service
SRV	gives the target host and port of a service instance
TXT	user defined text to convey additional information on using the service
AAAA	maps a hostname to an IPv6 address

Clients search for DNS-SD services by requesting the PTR records of a $< service >. < domain >$. The result is a DNS response message containing a set of zero or more PTR records listing matching $srvice\ instance\ names$ of the format:

$$service\ instance\ name\ = < instance >. < service >. < domain >$$

The $instance$ part is intended to give a user-readable descriptive string for a $service$ instance. The $service$ part can be of the form $[\_subtype.\_sub.]\_type.proto$. The use of $\_subtype$ allows clients to request for a narrower set of results. The $domain$

part can be $.local$ when used with mDNS in a local-scope or the configured domain if a DNS infrastructure is available.

When a client chooses to contact a particular $service\ instance\ name$, it asks for its SRV and TXT records. The SRV record gives the port number, service type, and hostname where the service resides. It also contains a priority and weight parameters to give preference when the same service is hosted in multiple places. Additional information about the service is conveyed via the TXT record in a *key-value* pair format. The exact *key-value* pairs are protocol dependent. For instance, a URI path of a CoAP resource might be included in the TXT record if DNS-SD is used to discover CoAP services. To resolve the hostname to an IP address, a query for the A/AAAA record is made.

A detailed comparison between these standards-based descriptions including aims, differences and completeness can be found in [99]. Table 3-3 presents a comparison of CoRE link format and DNS-SD with representative service descriptions used in CNNs with respect to compactness, applicability, interoperability and the capacity to filter queries for narrowing discovery results. Finally, it should be noted that while these descriptions provide the basic syntactical matchmaking to realise SD, semantic-rich service descriptions are also being considered for CNNs [141].

Table 3-3 Comparison of service descriptions for CNNs

	Compactness	**Query filtering**	**Applicability**	**Interoperability**
CoRE Link format	Yes	Yes	CNNs	Yes
DNS-SD	No	No	Generic	Yes
NanoSD	Yes	Yes	CNNs	No
NanoSLP	Yes	Yes	CNNs	No
SSLP	No	No	Generic	Yes
ENUM	No	No	Generic	No

3.5 CNN SDPs: requirements and challenges

From the above analysis and discussions, the following requirements should be met by an SDP for pervasive 6LoWPANs. It should be noted that the requirements might differ depending on the usage.

- **Scalable:** An SDP should ensure scalability with respect to the number of nodes and services. Thus, it has to minimise generated traffic and should proposes techniques to manage scalability, especially as the bandwidth is very limited (20 to 250 kbps). Another scalability issue that can occur in 6LoWPANs is the limited packet size, which, in the best case, only counts for about 80 bytes at the application layer. Thus, a 6LoWPAN SDP has to use descriptions that, at the same time, fulfil completeness and compactness.
- **Lightweight**: A 6LoWPAN SDP should be lightweight enough to be implemented on constrained devices. Thus, the protocol should minimise consumption of memory and computational resources and should not assume the availability of powerful nodes to be able to operate. Moreover, contributing to the protocol should not be a burden for both resource-lean and resource-rich nodes. In essence, if a device is able to run IP, it should also be able to implement the SDP [133].
- **Self-configuration and ease of deployment**: A 6LoWPAN SDP should allow nodes to start providing and consuming services with minimal configuration (preferably zero-configuration). Thus, new nodes should be able to auto-configure and participate in the system without the need for human intervention. In addition, a 6LoWPAN SDP should provide mechanisms for auto re-configuration and recovery after faults or mobility. In this context, a distributed protocol seems to offer such an ease of configuration.
- **Interoperable:** In the IoT, heterogeneous devices, networks and systems should seamlessly interoperate. Since SD is the first phase in locating existing resources, it must provide interoperability mechanisms that allow seamless integration regardless of the underlying standard and hardware. To do so, an SDP should use standardized service descriptions that can fit the requirement of different hardware as much as possible, e.g., DNS-SD.
- **Energy-aware**: An SD solution should optimize its operations to save energy. Since communication is the most energy consuming part in CNNs [44], an SDP should minimise traffic and take advantage of the distributed nature of the environment as much as possible. For instance, using caching techniques to

mitigate energy consumption to the less consuming parts (computation and storage) is a very important feature a 6LoWPAN SDP should have [126]. Finally, an SDP must also be aware of sleeping mechanisms used by the underlying layers (e.g., RDC).

- **Context-aware and adaptable:** An SDP for 6LoWPANs should be adaptive to the nature of tasks performed at a specific time. Thus, it should benefit from the network context to change its parameters (e.g., advertisement frequency). It should also provide provisioning mechanisms to adapt to the user needs, e.g., allow the most used services to be prioritised. Moreover, the SDP should be flexible to benefit from available resources. Thus, while an SDP should not assume the availability of resource-rich devices in order to operate, it should be opportunistic enough to exploit such devices when they are available.
- **Extensible:** A 6LoWPAN SDP should be extensible to the various 6LoWPAN applications envisaged in the IoT. Thus, it should be flexible enough to be adapted to different application scenarios envisaged in home or office environments as well as to those deployed in larger smart cities applications and unintended deployments. To achieve this, an SDP design should consider the need to add and remove functionalities as required by a specific use-case while still able to operate seamlessly.
- **Time-efficient**: An SDP for 6LoWPANs should be time-efficient, especially in multi-hop and mobile networks. In this context, the fully distributed hybrid push/pull can reduce the discovery time as services will be proactively pushed into the network. This helps to ensure a good user experience as it speeds up the invocation (delivery) process and makes efficient use of available services.
- **Mobility support:** An efficient 6LoWPAN SDP should be able to offer robust discovery services even when network nodes are mobile. In this context, the centralised directory approach was shown to be inefficient [113]. Cluster strategies are also affected by mobility as electing cluster heads, announcing and discovering them are costly [126]. The distributed approach is preferred by research concerning mobility support since fewer configurations are needed and hence the

system is minimally affected by network churn[7]. Therefore, a fully distributed hybrid SDP can provide better mobility support.

- **Reliability:** 6LoWPANs are prone to faults, packet losses and environmental noise. Hence, an SDP should provide reliability guarantees. This can be handled by: storing service information in different locations; providing multipath requests and replies, and handling the case of packet loss. Thus, if a packet is lost, retransmissions can ensure its delivery.
- **Cooperation**: Exploiting the cooperative potential of 6LoWPAN nodes can enhance SD. For instance, taking advantage of multicast communication used in fully distributed approaches to optimise service lookup, service registration, and cache consistency can enhance the discovery without incurring additional cost. Thus, devices can cooperate to delete redundant responses, ignore/stop already solved requests and/or detect inconsistencies, as in [132]. Using some low layer parameters such as RSSI and LQI within an SDP can also be very beneficial.
- **Security and privacy assurance:** security and privacy are key issues in the IoT. Many of the surveyed SDPs do not explicitly take these issues into account. To be successful, an SDP must provide security and privacy assurance. This is especially important when dealing with critical data as in the case of healthcare applications.
- **Other optimisations**: in order to make use of discovered services, a client needs to select the most appropriate ones (service selection). Thus, if an SDP can provide an efficient selection mechanism, it saves network resources as only the most relevant services are sent back to the client. Integrating the delivery of requested data in discovery messages can save time and cost especially for simple services. Grouping similar services that are generally discovered and used together is an efficient optimisation. Many other optimisations can be done depending on the specific use-case; however, an SDP for 6LoWPAN should always prefer interoperability over optimisations. Thus, if an optimisation compromises interoperability, it should be discarded.

[7] Churn: in networking, the frequency of nodes joining and leaving a network is known as the churn of the system.

3.6 Summary

Having surveyed existing service discovery protocols for 6LoWPAN, identified the gap in the 6LoWPAN SDP literature and summarized SDP design challenges, this chapter concludes that there is a need for a novel approach to deal with service discovery in 6LoWPANs which is still immature [96]–[99].

The gap identified in this chapter concerns the lack of efficient, adaptable and extensible hybrid push-pull SDPs to support zero-configuration interactions in 6LoWPAN networks. In response, the author designed EADP: an Extensible, Adaptable Discovery Protocol for 6LoWPANs. EADP will be the subject of the next chapter.

Chapter 4

EADP: an Extensible Adaptable Discovery Protocol for Low-power and Lossy Networks

Having shown the limitations of existing SDPs to pervasive LLNs, the author designed EADP: an Extensible, Adaptable Discovery Protocol. EADP works at the application layer, over UDP-IPv6, and is intended to adapt to the whole span of 6LoWPAN networks, ranging from static ones such as those used in home automation to the most dynamic ones where nodes are carried in vehicles. This chapter presents EADP, its design and architecture, and performance evaluation.

4.1 EADP Design

The design of EADP follows a loose-coupled component-based philosophy. Thus, EADP is made up of many components with minimum dependency in order to achieve adaptability and extensibility. The following subsections introduce EADP's architecture and give an overview of its functioning.

4.1.1 EADP architecture

EADP provides a service discovery mechanism targeting 6LoWPAN networks. Particularly, EADP targets local discovery in 6LoWPAN networks having limited or no infrastructure support. In addition to aiming at a timely reaction to network dynamics, EADP has been designed to provide high discovery rates and fast discovery times with low network overhead and low energy consumption. To ensure these qualities, EADP adopts a fully-distributed approach based on adaptive hybrid push-pull architecture

(Table 3-1). The generic EADP architecture depicted in Figure 4-1 is made up of five main components, namely:

- A User Agent (UA) responsible for discovering services in the pull mode;
- A Service Agent (SA) responsible for registering and advertising services' information in the push mode;
- A State Maintenance (SM) mechanism responsible for managing nodes local directories and making the protocol react seamlessly to network dynamics;
- A Reply Agent (RA) responsible for delivering service replies along with avoiding reply storms;
- A matchmaking component which implements service logic and matches client requests with provided service descriptions.

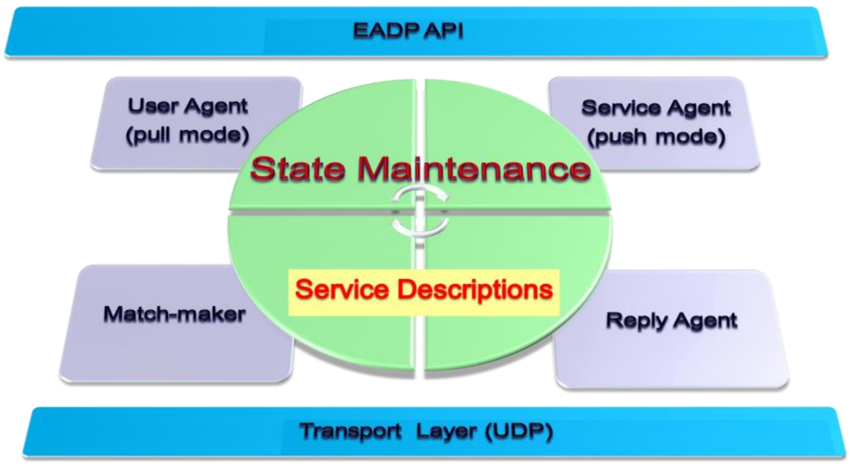


Figure 4-1 Generic EADP architecture

This component-based architecture allows EADP to be flexible, extensible and adaptable to different environments. Thus, one can add/remove functionalities/components depending on specific application needs while keeping the protocol functional. The main focus of this chapter is on the push mode operations and hence it details the contributions regarding the SA and SM. Techniques proposed by the RA component are also introduced and discussed. The other components of the protocols are either left generic (e.g., the matchmaker) or are borrowed from the literature (e.g., the pull mode algorithm). Contributions regarding such components will be detailed in following chapters.

4.1.2 EADP overview

In EADP, when a new node joins a network, it starts by advertising its available services. Upon reception of such a packet, the receiving node's SA decides on the utility of each contained entry and, consequently, adjusts the push mode transmission window using a Trickle algorithm. At the transmission time, the SA includes in its outgoing advertisement useful local and remote stored services (section 4.4). To ensure the liveness of stored services, EADP provides a state maintenance mechanism that deletes any stored service entry at the expiration of its TTL. In addition, the SM provides an algorithm to forward explicit delete-messages initiated by service providers as will be detailed in section 4.5.

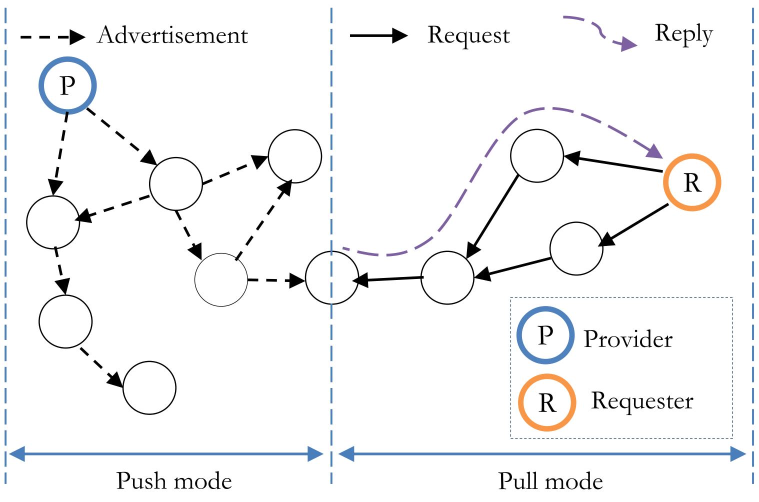


Figure 4-2 EADP overview

On the other hand, once a node needs to discover and use a service; it calls its UA (section 4.3). The UA starts by inspecting the local directory in order to find the requested service; if found the discovery process finishes. Otherwise, a service request message is generated and propagated across the network. Upon finding a service matching the requested criteria (matchmaking), corresponding node's RA generates a service reply message to be sent to the requester, as shown in Figure 4-2. RA (section 4.6) provides techniques to avoid multitude replies if there are numerous nodes to respond, and proposes two mechanisms to deliver the replies: 1) use the underlying routing protocol or 2) exploit a reverse-path constructed when forwarding requests. The service description and matchmaking component of EADP is left generic. This way EADP is not forced to a single service description and query language but can adapt to multiple descriptions and languages. Propositions to integrate EADP with DNS-SD and CoRE link format descriptions will be the subject of Chapter 7. Finally, it should be noted that while EADP provides an adaptive mechanism for managing the push mode, a possibility is always given to a user/node to disable it. For instance, a node having limited battery resource can disable the push mode. This overview is depicted in Figure 4-2.

4.2 Message formats and configuration parameters

In realising the above process, EADP defines three mandatory message types namely: advertisement, request, and reply plus an optional type: delete.

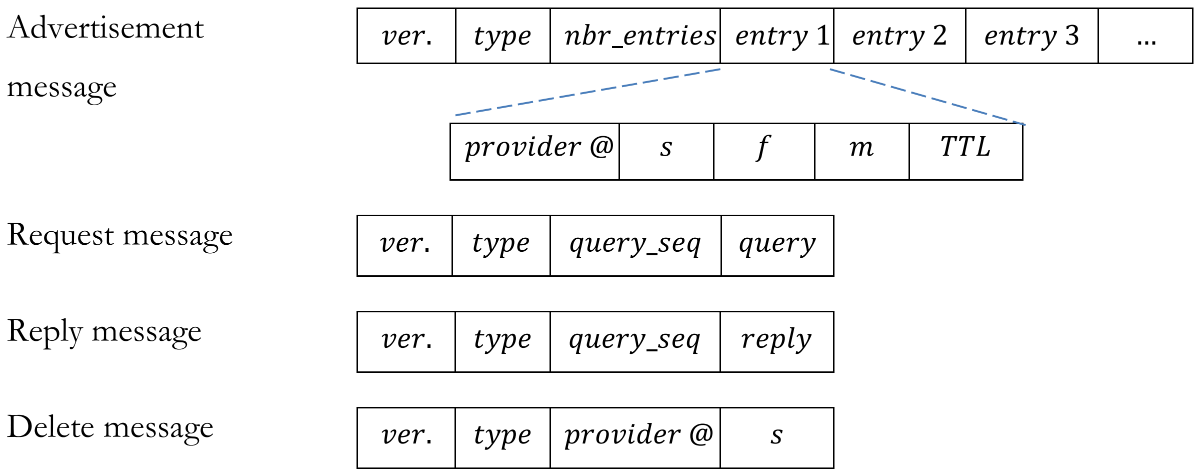


Figure 4-3 Generic message formats of EADP

Table 4-1 EADP configuration parameters

Configuration parameters	**Meaning**
REQUEST_DISK	The maximum number of hops a request is allowed to propagate. After this distance, the request is aborted.
ADVERTISEMENT_DISK	The maximum number of hops, from the provider, a service description is allowed to propagate.
WAIT_RESPONSE_TIME	The time a requester waits for a reply. At its expiration, the requester may resend or abort its request.
REQUEST_RETRANSMISSION_COUNTER	The maximum retries to resend a request. When it reaches zero, the request is aborted.
TTL (Time to Live)	The period of time a service entry is kept in a node's local directory. It should be a multitude of the push period (e.g., 2−5 times).

All the messages share the same header, containing information about the protocol version, message type and other flags. However, for the payload, while the request and delete messages can have fixed payload sizes containing, respectively, the necessary criteria for the requested service and the necessary information to uniquely identify a service, the advertisement and reply messages have variable payload sizes depending on the number and size of the service entries included in the message. The generic format of such messages is depicted in Figure 4-3. The *provider @* of a service entry is a unique identifier that identifies a provider (e.g., a 6LoWPAN compressed IPv6 address). The other fields are described in section 4.4. The *query_seq* is a 16-bit integer used to avoid duplicate transmissions of requests and to match a reply with a specific request. Note that this format is only given for illustrative purposes and can be changed depending on the service types, flags and messages required by a particular service description format employing EADP. Finally, and in addition to Trickle-specific parameters (section 2.5.2), EADP introduces the configuration parameters defined in Table 4-1.

4.3 The user agent algorithm

When a node needs to use a service, it calls its UA. The UA, firstly, checks the node's local directory. If the service is found, the discovery is accomplished in a purely push mode. Otherwise, the UA initiates a service request and propagates it out over the network using a limited flooding algorithm in order to ensure fast and 100% discovery. Simultaneously, the UA sets its request timer for WAIT_RESPONSE_TIME to wait for replies. At the expiration of the timer, if a response has not been received, the UA retransmits the same packet and decrements the REQUEST_RETRANSMISSION_COUNTER. When the counter reaches zero, the UA aborts the request and concludes that the service is either non-existent in the vicinity or unreachable. On the other hand, the UA is always listening for service requests, processing them and deciding whether to forward, abort or generate reply messages when necessary. Thus, upon reception of a request message, the UA asks the matchmaker to match it with the node's local directory entries. If a service matches the request, a reply is generated by the RA. Otherwise, the UA investigates the distance travelled by the request, which can be extracted from the hop count field (the IP TTL field) carried in the IP header and compares it with the REQUEST_DISK. Depending on the results, it decides whether to abort or forward the request. Note that specifying the REQUEST_DISK value as the maximum hop limit filed of the IP header allows saving one byte in every EADP request message and enables automatic aborting of requests at the IP layer when the hop count reaches zero. In addition, this allows the EADP generic request message format (Figure 4-3) to be directly mapped to standard protocol formats such as CoAP and DNS, which might employ EADP for service discovery.

By adopting a limited flooding approach which only requires nodes to re-broadcast received packets to their neighbours, the UA ensures that a request can visit all the nodes and that it will get forwarded at most once by an intermediate node. This is achieved thanks to loop-free primitives based on the use of *query_seq* and a request cache table. Using a limited flooding algorithm is a split-horizon common practice in constrained wireless ad-hoc networks because of its stateless nature allowing it to be implemented in even very constrained nodes. However, flooding might suffer performance degradations in dense networks. This will be investigated and discussed in the following chapter.

4.4 The service agent algorithm

The SA is responsible for controlling EADP push mode where nodes periodically advertise their own and remote services stored in their local directories. To do so, the SA proposes and implements a new variant of the Trickle algorithm in order to minimise the number and size of advertisements.

The Trickle algorithm (section 2.5.2) was originally designed to handle single data dissemination [77], [142]. To enable its use for multiple data items, protocols, in general, use two approaches[8]: the first establishes many parallel Trickles while the second uses a single Trickle that serially manages all data items. The two approaches have different characteristics and performance when compared with the original Trickle. Parallel approaches, applied in [78], [142], introduce a control cost that increases linearly with the number of data items. Serial approaches, applied in [143], keep a fixed control cost but make the latency increase linearly with the number of data items. However, the total cost

[8] Other approaches combining Trickle with other techniques are proposed in [77] and [79].

of the two approaches scales linearly with the number of data items [79]. Another main criticism that can be addressed to the serial approach is its scalability with respect to the size of a control packet. Thus, in serial approaches a packet may exceed the MTU. Finally, it should be noted that the mentioned usage of these approaches assumes a small number of data items [77]. Since, neither the serial nor the parallel approach responds to the requirements of EADP's push mode, the following section introduces a new variant to use Trickle in hybrid push-pull SDPs.

4.4.1 A new variant of Trickle

This section proposes another variant to use Trickle in hybrid push-pull protocols (Figure 4-4). Unlike other Trickle variants, the proposed approach attaches the consistency counter to the data items (services in EADP). Thus, every service in the node's local directory has a consistency counter which is updated following the rules defined below. In addition, and since every node is responsible only for its services, a node only resets the consistency counters of its services in order to minimise the traffic generated when in maintenance mode of Trickle (section 2.5.2). While this approach might not ensure a strict consistency, it fits hybrid push-pull protocols well and gives the proposed variant very attractive proprieties. Such properties include zero control signalling overhead, bounded maximum advertisement size and density independent inconsistency detection time. These benefits are realised thanks to the algorithms below.

To provide a strict consistency for a few crucial services, this new Trickle variant allows 'gossiping politely' about them using Trickle features. Thus, a flag in a service entry can indicate that this is an important service, and hence nodes keep gossiping about it (as if it is one of their own services) in order to ensure that it reaches all network entities. Finally, it should be noted that when a timer is reset by any event, it causes a node to advertise its services and important ones more quickly.

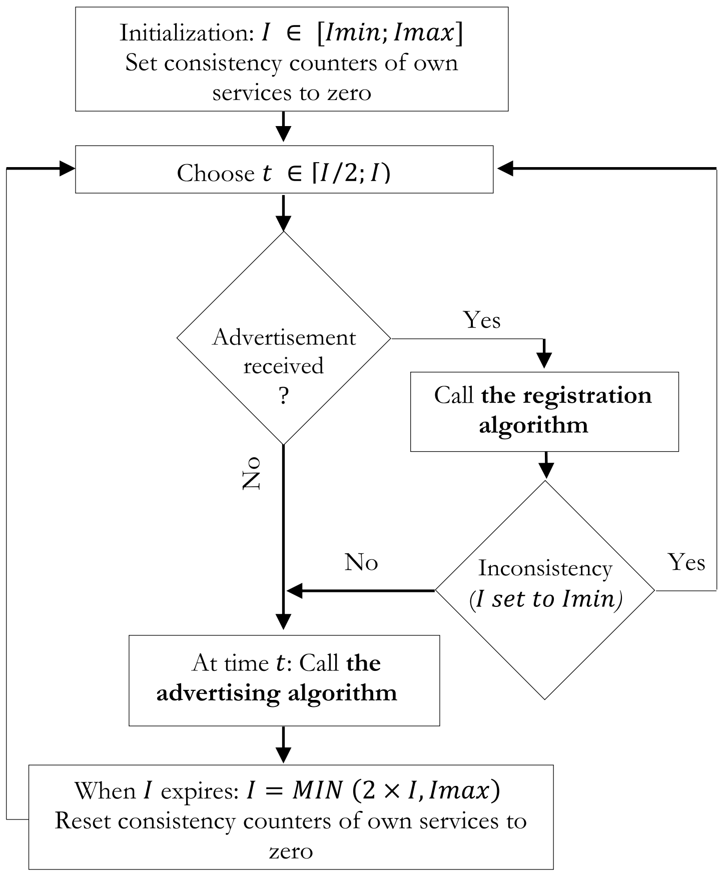

Initialization: $I \in [Imin; Imax]$
Set consistency counters of own services to zero
Choose $t \in [I/2; I)$
Advertisement received ?
Yes
Call **the registration algorithm**
No
Inconsistency (I set to $Imin$)
Yes
No
At time t: Call **the advertising algorithm**
When I expires: $I = MIN\ (2 \times I, Imax)$
Reset consistency counters of own services to zero

Figure 4-4 The proposed SA algorithm

4.4.2 Trickle to control service registrations

Upon the reception of an advertisement adv_msg, the receiver's SA starts the registration algorithm.

4.4.2.1 Service registration algorithm

Each entry in adv_msg represents a service s appended with a metric m (distance in hops) and a sequence number f. The former of the two parameters is used to limit the entry's propagation; it is incremented by each forwarder, whilst the latter is used to ensure loop-free transmissions and it is set and incremented only by the provider. Thus, an entry in adv_msg can essentially be identified by the vector (s, f, m) [127].

The registration algorithm investigates the consistency of each entry (s, f, m) in adv_msg. A received service entry is considered as *consistent* when the corresponding service information is already in the node's local directory and considered as older, or announced as being far. Formally, a *consistent* entry verifies:

- The received entry (s, f, m) is already in the node's local directory, referred to as (s, f', m'), and has a lesser value of f $(f < f')$ or;
- The received entry (s, f, m) is already in the node's local directory, referred to as (s, f', m') and has the same value of f $(f = f')$ and a greater or equal value of m $(m \geq m')$.

If an entry is identified as *consistent*, the registration algorithm only increments its consistency counter c in the node's local directory. Otherwise, the entry is *inconsistent* and hence feasible for registration (either the entry is new or the node received an update for an existing entry). The SA proceeds to the registration of such an entry for a TTL period, increments and updates its distance m and reinitialise its consistency counter c to zero.

4.4.2.2 Resetting the Trickle timer

For the *first inconsistent* entry in adv_msg, if the interval I is greater than $Imin$, the Trickle timer I is set to $Imin$. This is to allow quick updates of the network about inconsistent services by quickly transmitting the next advertisement. It should be noted that applying the *first-inconsistency* approach to reset I ensures a quick inconsistency detection time and allows for timely reaction to network changes.

However, if minimising the cost is, further, more important, a general *n-inconsistency* approach is proposed. In such an approach, once receiving an adv_msg, the registration algorithm resets I if and only if n *inconsistent* entries are reached. To do so, the node keeps an inconsistency counter ic which is incremented every time an inconsistency appears. When ic reaches n $(ic = n)$ and I is greater than $Imin$, I is set to $Imin$.

The *n-inconsistency* approach provides very attractive features. Thus, even if the number of inconsistent services does not reach n, the push algorithm still advertises them with relatively larger periods. On the other hand, if inconsistent services exceed n, remaining services will be advertised in the following intervals. Note also that if an implementation decides to limit the size of an advertisement, the services fitting the size-limit will be sent; the others are not lost, they will be sent in following intervals. The generic version of the registration algorithm is depicted in Figure 4-5.

Registration Algorithm

```
FOR each entry $(s, m, f)$ in the received $adv\_msg$ DO
 IF($m$ <= $ADVERTISMENT\_DISK$) THEN
  $inconsistency \leftarrow false$
  IF $service\_exist$ ($s$) THEN //dubbed $(s, m', f')$
    IF ($f' < f$ || ($f' = f$ && $m < m'$)) THEN
      $inconsistency \leftarrow true$
      $inconsistency\_counter \leftarrow inconsistency\_counter + 1$
    ELSE
      $update\_inconsistency\_counter()$
      $entry\_consistency\_counter \leftarrow entry\_consistency\_counter + 1$
    ENDIF
  ELSE
      $inconsistency \leftarrow true$
      $inconsistency\_counter \leftarrow inconsistency\_counter + 1$
  ENDIF
  IF $inconsistency$ THEN
    $entry\_consistency\_counter \leftarrow 0$
    $update\_service\_cache$ $(s, f, m)$
    IF $inconsistency\_counter = n$ THEN
      $reset\_trickle\_timer$ $(Imin)$
    ENDIF
  ENDIF
 ENDIF
ENDFOR
```

Figure 4-5 The registration algorithm

4.4.3 Advertising rules and protocol scalability

At time t (Figure 4-4), the SA calls the advertising algorithm to form outgoing advertisements. The advertising algorithm includes in the outgoing message all entries whose distances are lesser than or equal ADVERTISEMENT_DISK and flagged as inconsistent in the node's local directory (with a consistency counter c equal to zero). These entries have not yet been made known to the network, thus their announcement is of interest and hence they are prioritised. However, this could be insufficient to optimise discovery times in cases where the wireless transmission is unreliable (interferences, noise…etc.) and if the network is sparse or contains holes as can be Figure 4-6. In this figure, if node N2 hears a consistent entry from N1 and decides to suppress its transmission, a part of the network may not be updated quickly which may delay subsequent replies. Therefore, a redundancy constant k greater than one can be used to include other estimated less useful entries. Hence, the outgoing message will be filled by other entries which verify, when sorted, the condition c is less than k ($c < k$). Notice that this advertising algorithm not only minimises the number of advertisements, but also ensures protocol scalability by controlling and minimising the number of services included in a single advertisement. The advertising algorithm is depicted in Figure 4-7.

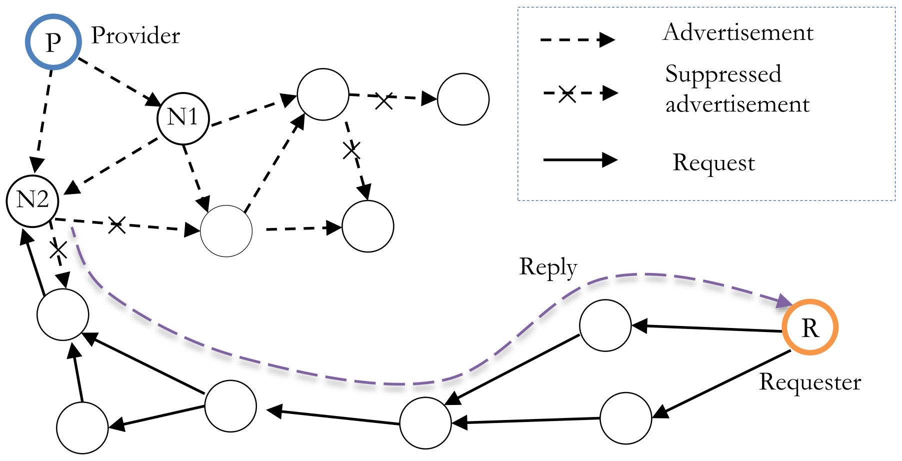


Figure 4-6 The impact of k on the propagation of an advertisement

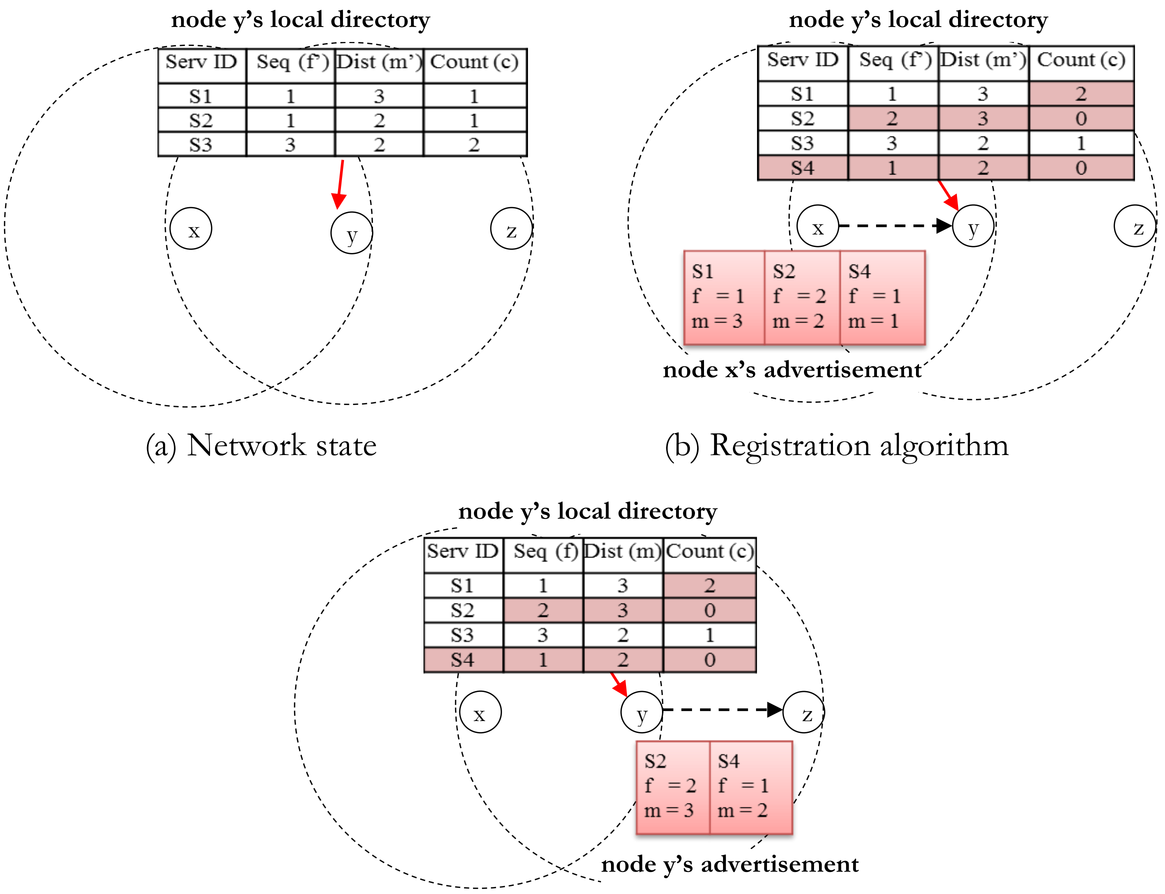
Advertising Algorithm

```
$adv\_msg\_size \leftarrow 0$
$nbr\_inconsistent\_services \leftarrow 0$
FOR each service entry $S$ in my local directory DO
  IF $entry\_consistency\_counter < k$ THEN
    IF $adv\_msg\_size + entry\_size < MTU$ THEN
      $add\_to\ adv\_msg\ (S)$
      $adv\_msg\_size \leftarrow adv\_msg\_size + entry\_size$
      $nbr\_inconsistent\_services \leftarrow nbr\_inconsistent\_services + 1$
    ELSE
      $break$
    ENDIF
  ENDIF
ENDFOR
IF $(nbr\_inconsistent\_services \geq n)$ THEN
  $add\_to\ adv\_msg\ (nbr\_inconsistent\_services)$
  $link\_local\ multicast\ (adv\_msg)$
  $inconsistency\_counter \leftarrow 0$
ELSE
  $inconsistency\_counter \leftarrow nbr\_inconsistent\_services$
  $free\_buffer\ (adv\_msg)$
ENDIF
```

Figure 4-7 The advertising algorithm

4.4.4 An example of execution of the SA

To illustrate the functioning of the above algorithms, let's take the example depicted in Figure 4-8 which depicts a network constructed of three nodes x, y and z.



Serv ID	Seq (f')	Dist (m')	Count (c)
S1	1	3	1
S2	1	2	1
S3	3	2	2

(a) Network state

Serv ID	Seq (f')	Dist (m')	Count (c)
S1	1	3	2
S2	2	3	0
S3	3	2	1
S4	1	2	0

S1	S2	S4
f = 1	f = 2	f = 1
m = 3	m = 2	m = 1

(b) Registration algorithm

Serv ID	Seq (f)	Dist (m)	Count (c)
S1	1	3	2
S2	2	3	0
S3	3	2	1
S4	1	2	0

S2	S4
f = 2	f = 1
m = 3	m = 2

(c) Advertising algorithm

Figure 4-8 An example of execution of the push algorithm

The state of the network, particularly node y, before receiving an adv_msg is depicted in Figure 4-8 (a) which shows node y's local directory containing three service entries S1, S2 and S3 with their respective sequence numbers and distances from their providers. Upon receiving an adv_msg containing three service entries S1, S2 and S4 with their respective values of f and m (Figure 4-8 (b)), the registration algorithm investigates the consistency of each entry. S1 is already present in node y's local directory and received with the same sequence number and distance $(f = f' and\ m = m')$ thus the registration algorithm only increments its c counter. S2 is already present in node y's local directory but received with a new sequence number ($f > f'$), the registration algorithm updates it and resets its c to zero. If the *first-inconsistency* approach is employed, the Trickle timer I is reset to $Imin$. S4 is new; the registration algorithm creates an entry for it with c equal to zero. At time t, the advertising algorithm goes through node y's local directory and includes in the outgoing advertisement entries with c counters less than k. Thus if a constant $k = 1$ and an ADVERTISEMENT_DISK = 4 are used, the outgoing advertisement contains S2 and S4 as illustrated in Figure 4-8 (c).

4.5 The state maintenance mechanism

EADP proposes a state maintenance mechanism aiming to react timely to network dynamics and hence prevent erroneous storage and advertisement of services when they are no longer available. In addition to TTL-based deregistration, the SM provides two primitives to keep the network updated about intended and/or unintended departures of nodes/services.

4.5.1 Explicit service deregistration

The first primitive offers the possibility for a provider to announce service departures via the optional delete-message. In order not to flood the network with delete-messages, EADP uses the same Trickle variant proposed for controlling service advertisements (section 4.4) to manage delete-messages forwarding. Hence, once a provided service become unavailable (e.g., fault of the sensing/actuating components…etc.) or the provider deliberately becomes unavailable (e.g., provider decides to leave the network, device shuts down, etc.); it initiates a delete-message. For optimisation reasons, two considerations can be taken:

- The outgoing message may contain one or more service entries to be deleted
- When a provider is going to leave the network, it can send a delete-message with zero entries to inform the network to delete all its services.

Upon receiving a delete-message, nodes delete corresponding service information from their local directories (all the service entries of a given provider if the delete-message contains zero-entry) then apply the Trickle variant introduced in section 4.4 to manage its forwarding.

While in static networks, exactly the same limited advertising algorithm can ensure deleting services from the network, in mobile networks; however, limiting delete-message forwarding to the ADVERTISEMENT-DISK is impractical since nodes move randomly. Thus, to be able to reach all nodes, network-wide forwarding should be used. In addition, some nodes which are not aware of the service to be deleted may receive the delete-message. In this case, those nodes temporarily cache the delete-message and will be considered in forwarding it. This is to balance the transmission loads and provide the possibility of reaching the nodes storing a copy of the service that can only be reached via non-aware nodes. Note that the speed of clearing the network from unavailable services depends on the Trickle minimum interval $Imin$. Thus, one may use a separate Trickle timer to manage delete-messages forwarding. Also a $k > 1$ might be judged better. However, using a different Trickle timer poses the question of when to stop its execution. The following chapter presents a way to stop a Trickle timer and hence allow using a separate timer for delete-messages.

4.5.2 Enforcing TTL-based deregistration

Since the above mechanism cannot manage the event of unintended departures of nodes, EADP can enforce TTL-based deregistration by exploiting the underlying neighbour discovery protocol to keep EADP updated about the disappearance of nodes. Note that coupling the neighbourhood information with EADP can also save maintenance mode's cost (section 4.4.1). However, such a cost might be used to provide a node with hints about the network dynamics and hence enables it to adjust its TTL values accordingly. Finally, it is worth noting that in a worst case when a node contacts a provider and finds out that a specific service is unavailable, the node may initiate a delete-message.

4.5.3 Local directory management

Each node maintains a local directory of the services available in its vicinity. The size of such directory is bounded by the node's memory constraints. The utility of an entry in a local directory depends on the number of hits. Thus, if the number of hits of a specific entry is high, it is worth keeping it. Otherwise, such an entry can be deleted when low on memory. The aforementioned approach is generally known as Least Frequently Used (LFU) strategy, and it is the one implemented by default in EADP. However, LFU is just one of many strategies that can be adopted. For instance, a node might exploit the external flash memory to store some entries instead of deleting them.

4.6 The reply agent algorithm

The RA is responsible for sending back service replies to the client. Additionally, it provides optional mechanisms to avoid potential reply storms.

4.6.1 Avoiding service reply storms

EADP allows nodes to respond on behalf of others in order to optimize discovery and provide support for sleepy nodes. In the latter, a node might indicate when advertising its services that it is sleeping for the specified TTL and might require an acknowledgment to confirm the node responding on its behalf. However, allowing nodes to reply automatically to matched requests might cause a reply storm, where many nodes caching information about a service, reply simultaneously. This can cause congestion, waste energy and result in redundant responses, especially in dense networks. To avoid such a problem, the RA proposes the following optional solutions.

4.6.1.1 Delaying reply transmissions

This approach simply applies a delay-and-cancel mechanism similar to the one deployed in the Dynamic Source Routing (DSR) protocol [144]. Thus, upon finding a service match, a node alarms a timer for a specific period and chooses a random time to transmit its reply. If a similar reply is heard, the node cancels its transmission. Since replies are sent using unicast in EADP, nodes might not hear them, and the mechanism can be rendered useless. Thus, while in non-duty-cycled networks, an overhearing mechanism can be deployed; in duty-cycled networks overhearing is not applicable. To overcome this limit, the node having a reply may: (i) turn its radio on listening for traffic before unicasting the reply, or (ii) use link-local broadcast. This latter enables potential responders to hear such a message and, at the same time, allows the transmitter to aggregate the replies in one message to be sent to the client via unicast. Note that a node having a reply can bias the random transmission time based on its resources. For instance, a mains-powered node with a bigger cache may decide to reply first. Other mechanisms in this class, e.g., probabilistic replies, can be envisaged.

4.6.1.2 Authoritative nodes

In this strategy, only some authoritative nodes are allowed to respond on behalf of others. For instance, one might only allow nodes, working on behalf of a sleepy node, to respond. Otherwise, cooperative, automatic designations of authoritative nodes might be envisaged. For instance, a node receiving many instances of a service might take the role to respond on behalf of the provider. Such strategies are, however, outside the scope of this work. Notice that in a worst case where nodes are not allowed to respond on behalf of others, the cached entries are still very useful for local needs. Furthermore, they might be used to guide the request towards the provider and hence save requests' propagations in wrong directions.

4.6.2 Optional reverse path routing

EADP provides an alternative reverse-path routing mechanism to reduce resource consumption. This mechanism exploits the path being traversed by the requests, to route back the replies using a route-over, AODV-like routing mechanism [145]. The reverse-path provides three main advantages: (i) it eliminates the overhead generated by the underlying routing protocol; (ii) it avoids delaying the responses when trying to establish routes, especially in the case of reactive routing protocols and; (iii) it uses simple cost-effective primitives. To do so, it just implies the use of a limited-size routing table, having the following structure:

<*destination_addr, next_hop, distance*>

Similarly to [129], when a link in the path is broken, the reverse-path mechanism can detect it when missing the acknowledgment (after a specific number of MAC retries) and can call the routing protocol to continue delivering the packet, as depicted in Figure 4-9. In addition to its importance for resource saving, the reverse-path mechanism can allow the nodes constructing the path to cache information about services contained in the replies, and hence, enhance subsequent requests' latencies. The reverse-path mechanism can always be disabled, especially if the network provides a proactive routing protocol (e.g., RPL). Finally, it should be noted that the reverse-path mechanism might not provide optimal routes.

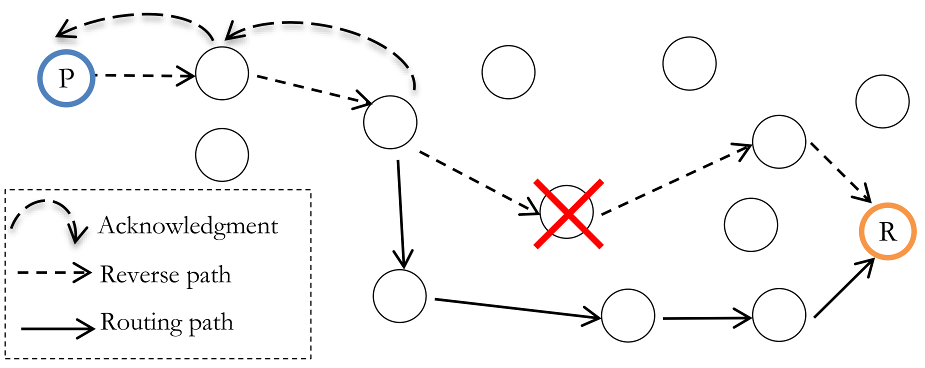


Figure 4-9 The reverse path routing

4.7 Formal analysis of the proposed push algorithm

This section presents a formal worst-case analysis of the proposed push algorithm in the case of single-hop lossless networks. To generalise the conclusions to multi-hop unreliable networks, the methodology presented in [23] can be used. In this latter, it is shown that considering the imperfect nature of wireless links and the multi-hop nature of the network will add a cost that scales logarithmically with the number of nodes. In this study, the performance of the proposed push algorithm is investigated, over one interval, with respect to: the number of advertisements, the size of an advertisement, the total amount of generated traffic and the inconsistency detection latency. These four parameters aim to provide a comprehensive analysis of the time/cost behaviour of the proposed solution.

4.7.1 Assumptions

The following points state the assumptions taken in the analysis. The parameters and notations used in this analysis are presented in Table 4-2.

- The analysis considers the *first-inconsistency* approach which obviously sends more messages than the *n-inconsistency* approach ($n > 1$).
- A provider is assumed to provide one unique service. This assumption is taken to simplify the analysis. It can be generalised when a node provides more than one service.
- For ease of the analysis, the size of a service description is assumed to be the same $service\_size$.

Table 4-2 Parameters used in the analysis

Parameter	Meaning
N	Number of nodes in the network
S	Number of services
Ns	Number of new services
k	Redundancy constant

4.7.2 Worst-case analysis

This section presents a worst case analysis of the number and maximum size of advertisements; the total amount of generated traffic; and the detection latency of inconsistencies for the proposed push approach, serial/parallel Trickle approaches, and fixed-period push algorithms such as the one used in ADDER.

4.7.2.1 Number of advertisements

In the proposed solution, and since nodes are assumed to provide unique services, each node will send its new services even after it hears other nodes' *consistent* data. This makes the proposed solution depend only on the number of new services ($O(Ns)$) occurring in an interval. This gives it a good scalability with the amount of exchanged messages as the number of new services in an interval is generally much lower than the total number of services available in the network. In the serial and parallel approaches of Trickle, the number of messages scale linearly with the number of services $O(S)$ [79]. Finally, in fixed-period push algorithms where each node transmits once per interval, the number of exchanged messages scales linearly with the number of nodes, presenting thereby a worst case message scalability in $O(N)$.

4.7.2.2 Size of an advertisement

With regards to the maximum size of an advertisement, the proposed algorithm presents a very attractive characteristic. Thus, the maximum size of an advertisement generated by the proposed approach is bounded by a constant value equal to $(k+1) \times service\_size$ i.e. the maximum number of entries included in an advertisement message is $k+1$. This gives the proposed approach a good scalability with the size of an advertisement, in $O\ (1)$ which is in the same order of parallel Trickle approaches [77]. However, using the ADDER advertising rule (probability = 1) where each advertisement contains all entries stored in a node's local directory, the size of an advertisement messages might reach S. Thus, the size of an advertisement message in ADDER scales linearly with the number of service ($O(S)$). This creates scalability issues in handling large number of services. For serial Trickle approaches, where the exchanged summary might contains all the information about the data stored in nodes' local directories, the size of a packet also scales linearly with the number of services $(O(S))$ [77].

4.7.2.3 Total amount of traffic

With regards to the total amount of traffic generated in a transmission window, which has a direct impact on the communication's energy consumption, the proposed approach sends in a worst case of approximately $(k + 1) \times Ns$. This comes in $O(Ns)$. In serial approaches where the number of messages is in $O(S)$, each contains in a worst case S entry, the amount of traffic generated in a transmission window scales squarely with the number of services $O(S^2)$. The same analysis gives fixed-period push algorithms a total amount of generated push traffic in $O\ (N \times S)$. Finally, using the same analysis for parallel Trickle approaches shows an amount of generated traffic in $O(S)$. Note that a protocol generating less traffic can save more communication energy which is the main source of energy consumption in LLNs.

4.7.2.4 Inconsistency detection time

For inconsistency detection latency, the proposed approach can detect and react to an inconsistency in a fixed time span $O(1)$ which is in the same order as parallel Trickle approaches and fixed-period push algorithms. However, the detection latency of serial approaches scale linearly with the number of services ($O(S)$) as shown in [79]. A summary of the performance is given in Table 4-3.

Table 4-3 Performance comparison

Metrics/Algorithms	**Number of messages**	**Maximum size**	**Total size**	**Detection latency**
Proposed approach	$O(Ns)$	$O(1)$	$O(Ns)$	$O(1)$
Parallel approach	$O(S)$	$O(1)$	$O(S)$	$O(1)$
Serial approach	$O(S)$	$O(S)$	$O(S^2)$	$O(S)$
Fixed-period push	$O(N)$	$O(S)$	$O(N \times S)$	$O(1)$

4.8 Evaluation of EADP

To consolidate and evaluate the mechanisms proposed in this chapter, EADP was implemented in Contiki OS [65]. In Contiki, a program can be directly run on a device, simulated or emulated. The EADP behaviour was emulated using the Cooja simulator [89] and the emulated Tmote Sky motes [90]. At the link layer, the ContikiMAC radio duty cycling protocol [146] with a channel check rate of 8 Hz was in operation. To accommodate a limitation in the broadcast mechanism of ContikiMAC that will be discussed in Chapter 6, a small fixed delay is added before rebroadcasting requests.

To put EADP results in context, it was compared with ADDER [127]. This choice was driven by the fact that both EADP and ADDER adopt a hybrid push-pull approach working at the application layer over UDP-IPv6. However, while ADDER (Probability = 1) uses for its push mode a fixed transmission window, EADP adjusts the window between $[Imin, Imax]$ and controls advertisement-size using the network context. Having in mind that the discovery time is proportional to the transmission window, EADP was compared with ADDER having a transmission window equal to $Imin$; the best window that EADP can reach (ADDER-b) and with ADDER with a transmission window equal to $Imax$; the worst window that EADP reaches (ADDER-w). Comparing EADP to a fixed-period protocol using the best and the worst values of the interval aims to cover the spectrum of EADP's behaviour. To get further insights into EADP's time/cost performance, EDAP with a transmission window in the range $[Imin/2, Imax]$ is also evaluated. To see the impact of the push mode on EADP discovery times, the push mode was disabled in the EADP-d version. Finally, to assess the benefits of the reverse path routing mechanism, the reverse path was used in the EADP-r version and compared with EADP. The evaluated protocols' variants are summarised in Table 4-4.

4.8.1 Evaluation methodology

A reference network topology depicted in Figure 4-10 was used in this evaluation. Such a scenario can be envisaged, for example, in emergency response applications. 100 Tmote Sky motes were initially uniformly distributed in a square area of 350m×350m. Twenty nodes uniformly distributed throughout the network proactively advertise twenty representative services, S1 to S20. Requests generation follows a CBR (Constant Bit Rate) traffic pattern in which a client generates a service request every two seconds looking for a service provided in the network. By opting for a request every two seconds, this setup used a congested network. To model the mobility of the entities involved in this scenario which might include sensors attached to humans and crew-cars slowly moving in the emergency area and static sensors deployed in and around the area, the random waypoint mobility model was used [113]. Thus, all the nodes were mobile with a maximum speed of 4 m/s (minimum was 0 m/s) and random pauses between 2 and 10 seconds. To account for initialisation biases, the first 1000 seconds of the model were discarded.

Starting from the above reference scenario, the performance of EADP is analysed under different conditions, changing, one by one: the execution time, the number of services, the network density, the maximum speed of nodes, and requests frequency. Table 4-5 summarises the parameters used in the experiments.

Note that by virtue of the push mode, a service might be already in the local directory of a node and can be discovered locally with zero discovery time. In this evaluation, such a best-case is not considered, and the ADVERTISEMENT_DISK was chosen to avoid it.

Table 4-4 Evaluated protocols' variants (scenario #1)

Protocol variant	**Description**
ADDER-b	ADDER using a fixed push period equal to $Imin$ which is the minimum period achieved by EADP.
ADDER-w	ADDER using a fixed push period equal to $Imax$ which is the maximum period achieved by EADP
EADP	Default EADP settings. EADP protocol having an adaptable period between $[Imin, Imax]$, enabling both push and pull modes and using RPL as the underlying routing protocol.
EADP-d	EADP as in default settings having the push mode disabled.
EADP Imin=10s	EADP as in the default setting having the adaptable period in the range $[Imin/2, Imax]$ instead of $[Imin, Imax]$.
EADP-r	EADP as in default settings using the reverse path routing mechanism instead of RPL for routing back the replies.

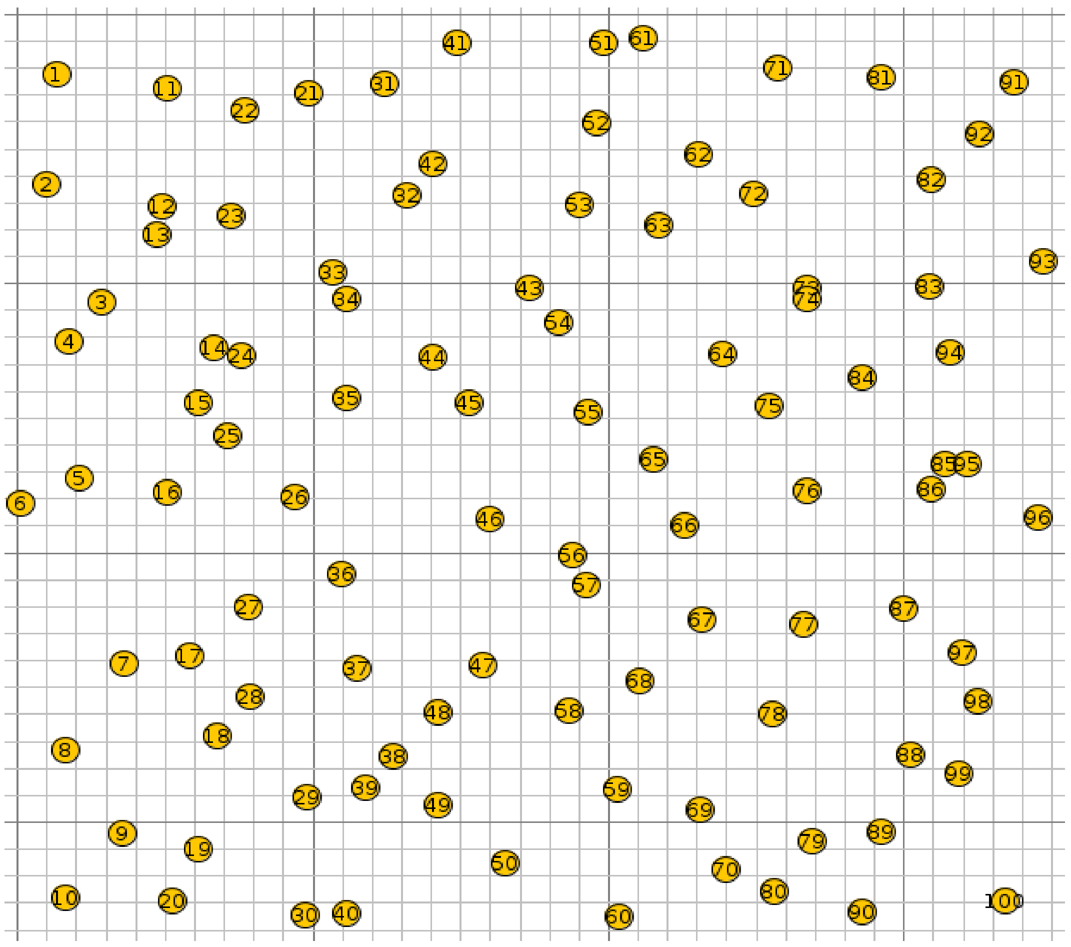


Figure 4-10 Initial topology of scenario #1

Table 4-5 Experimental parameters (scenario #1)

Parameter	**Value**
Duration of one simulation / #iterations / #nodes	350s / 10 / 100
Medium / Transmission range / Throughput	UDGM / 50m / 250kbps
Network area (x, y)	350m x 350m
EADP's $[Imin, Imax]$	[20s; 80s]
ADDER probability/ EADP constant k	1 / 1
Mobility model	Random waypoint mobility
Min-max speed / min-max pause periods	[0m/s; 4m/s] / [2s; 10s]
Traffic pattern / Requests frequency	CBR / 0.5 request/second
REQUEST_RETRANSMISSION_COUNTER	0
REQUEST_DISK / ADVERTISEMENT_DISK	6 / 4
Underlying routing protocol	RPL
RDC/ MAC / Adaptation layer	ContikiMAC / CSMA-CA / 6LoWPAN

4.8.2 Performance metrics

The evaluation focuses mainly on the trade-off between low push overhead and fast discovery times. Thus, the following metrics were measured.

4.8.2.1 Average discovery time

This metric measures the latency realised by an SDP to locate requested services. It is defined as the waiting time in milliseconds, averaged over all the requests, a client waits from transmitting a request to getting its first reply. Note, in the mobile scenarios and because routing registered high packet loss rates, the average hit time is measured instead. This latter is measured as the average time an SDP takes to hit the requested service. Both metrics (hit and discovery times) mirror the time efficiency of an SDP.

4.8.2.2 Average advertisements number per node

This metric measures the capacity of an SDP to minimise advertisement traffic. It is defined as the ratio of the total number of generated advertisements to the number of nodes. This metric quantifies the amount of generated unsolicited push messages. Realising fewer advertisements per node contributes hugely to the cost efficiency and scalability objectives of EADP, hence helps to reduce energy consumption and extends the network lifetime.

4.8.2.3 Average advertisements size per node

In EADP and ADDER, the advertisement size is variable. A protocol which generates big advertisements not only increases the amount of traffic but also suffers from scalability issues when the number of services increases. Thus, the average advertisement size is of great importance in such protocols. It is defined as the ratio between the average size of all the advertisements sent by a node and the number of nodes. Besides its importance to scalability with respect to the size of a single advertisement message, this metric plays an important role in the cost efficiency of EADP. Thus, combining this metric with the previous one gives the amount of traffic generated by EADP's push mode.

4.8.2.4 Average energy consumed per node

This metric measures how much energy, on average, a node consumes in order to realise discovery tasks. It is calculated as the ratio between the total network energy consumed during the simulation time and the number of nodes in the network. To measure this metric, the Contiki power profiler [147] was used. This metric explicitly measures the amount of energy consumed by a network running EADP.

4.8.2.5 Average discovery success rate

This metric measures the capacity of an SDP to respond reliably to client requests. It is measured as the ratio between the number of requests to the number of unique responses received by the clients. Ideally, the discovery success rate would be 1. Practically, it should be as close as possible to 1. The discovery success rate of EADP is an indicator of its reliability. This metric encompasses both the performance of EADP and the underlying routing protocol used to deliver service replies.

These metrics allow us to draw conclusions about latency, generated overhead, energy consumption and scalability of EADP. It is worth noting, that in addition to the above main metrics some secondary metrics such as the rate of false negative discoveries are also discussed. The definition of such metrics is introduced in context.

4.8.3 Results and discussions

The presented results are an average, obtained by running each simulation 10 times, changing each time the seed of the random number generator. The sample mean is plotted in Figure 4-11, Figure 4-12, Figure 4-13 and Figure 4-14.

4.8.3.1 EADP time/cost performance

To see the impact of the push mode on EADP's time/cost performance, the mobility was disabled and the discovery times, the number and size of generated push overhead and the consumed energy were measured for EADP, ADDER-b, ADDER-w and EADP-d (the EADP pure pull version) when varying the execution time (proxied by the number of requests in Figure 4-11).

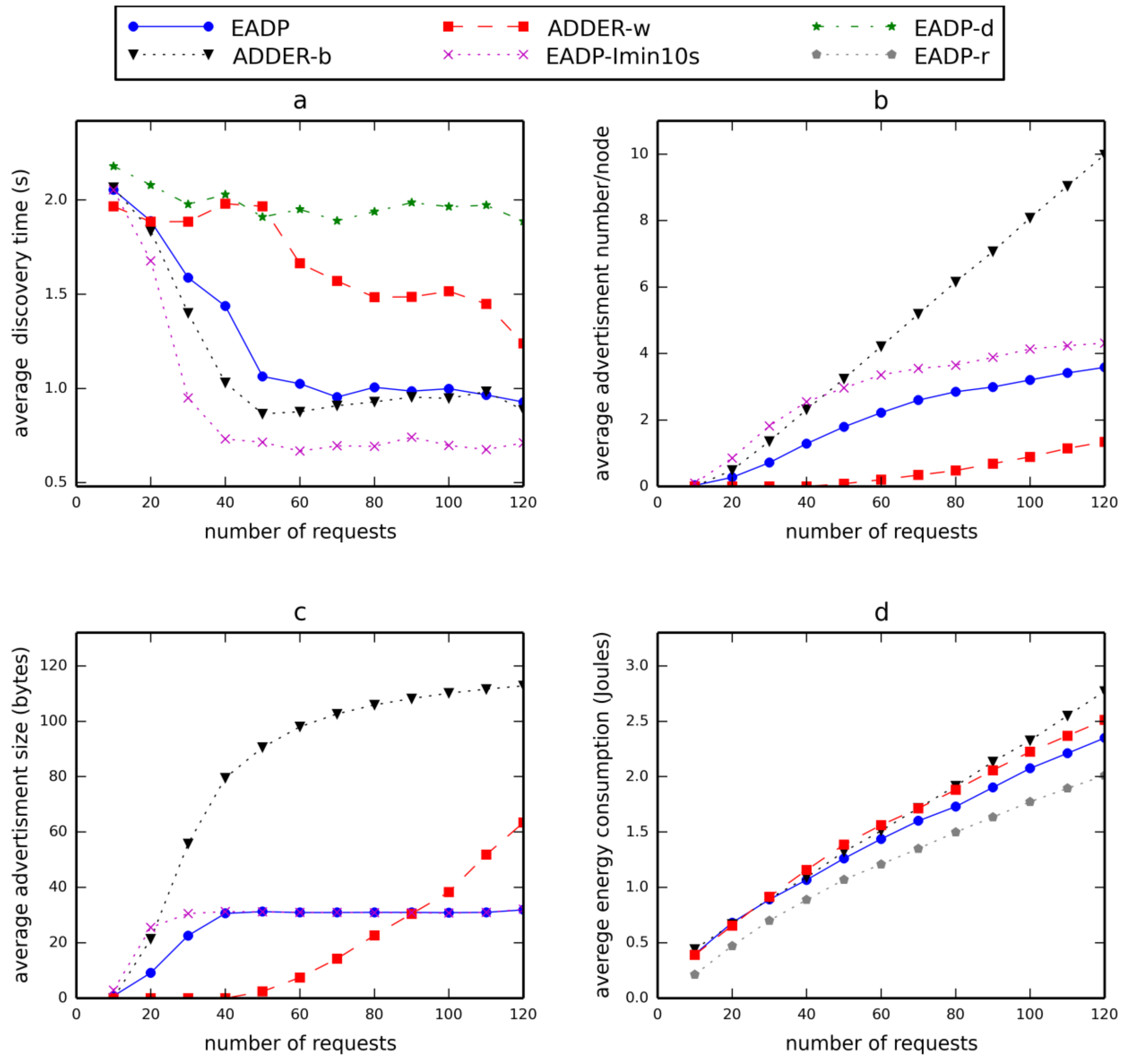


Figure 4-11 EADP time/cost performance

As can be seen from Figure 4-11 (a), queries would initially traverse the network to find requested services, thus resulting in slow average service discovery time. This stays constant for EADP-d over the course of time (increasing number of requests) as a consequence of disabling the push mode. However, it gets faster in the other evaluated protocols. This is realised thanks to enabling the push mode, which allowed information about services to be propagated and stored throughout intermediate nodes allowing them to answer subsequent requests. Nevertheless, by using the largest window, ADDER-w presented slower response times. Compared with ADDER-b, EADP registered comparable results especially after network convergence (more than 60 requests). Before that, ADDER-b achieved better discovery times. This is mainly caused by the listen-only period of the $Imin$ interval. Hence, if faster discovery times are further required, a smaller value of $Imin$ can be used. Thus, EADP with $Imin = 10s$ plot presented the best discovery times. However, such performance came at an additional cost compared to EADP, but it is still far less than that of ADDER-b. This is achieved thanks to exploiting the network-context to adjust the transmission window allowing the necessary information to be propagated as soon as it appears.

In ADDER-b, achieving good discovery times came at a high amount of generated overhead which scales linearly with the number of known services, as shown in Figure 4-11 (b). Whilst EADP generated more messages than ADDER-w, it sent far fewer messages than ADDER-b and provided comparable discovery times. When using an $Imin = 10s$, EADP generated more advertisements than one using an $Imin$ interval equal to that of ADDER-b. Although, such a cost is still far less than that of ADDER-b, it would be preferable to save it if alternative methods to achieve faster discovery time can be envisaged.

With regards to the impact of the number of known services on the size of an advertisement, Figure 4-11 (c) shows that EADP presented a lower advertisement size which converged and stayed constant, at less than 40 bytes. This is even better than ADDER-w, which sent fewer advertisements (Figure 4-11 (b)). Thus, in ADDER-w, and since each advertisement blindly contains all known entries, its size kept increasing with time and exceeded that of EADP. This makes it less reactive to an increasing number of known services although it does not achieve good discovery times. However, ADDER-b achieved good discovery times, but with the biggest average advertisement size, making it the worst in scalability terms as a result of sending big advertisements most often. This strengthens the analysis presented in section 4.7.

Finally, to evaluate the impact of the reverse-path mechanism on EADP's energy consumption, the reverse-path mechanism was employed to deliver unicast responses (EADP-r). As can be seen from Figure 4-11 (d) the protocols using RPL (EADP, ADDER-b, ADDER-w) registered comparable energy consumption values at the beginning (less than 30 requests). This might be explained by the fact that at the network bootstrapping, the routing traffic was the main source of energy consumption in the network. Such traffic continued to influence the energy consumption at the start of discovery operations making the evaluated protocols present approximately similar energy consumption values up to about 30 requests. However, after that and hence when discovery messages were the dominating traffic circulating in the network, EADP presented the lowest energy consumption compared to ADDER-b and ADDER-w. This was done thanks to mechanisms which minimised both the number and the size of generated push traffic, which saved the energy that would be consumed by transmitting such packets. On the other hand, by comparing EADP using the reverse-path (EADP-r) with its version using RPL, the reverse-path mechanism showed by far better energy savings. This is achieved by avoiding routing traffic.

4.8.3.2 Evaluation of the n-inconsistency approach

To evaluate the *n-inconsistency* approach, the number of advertised services was varied between 5 and 50. The average hit times and the average number of generated advertisements was measured, at the end of the simulation, for values of *n* equal to 1, 2, 4 and 6. Results are depicted in Figure 4-12 (a) and (b).

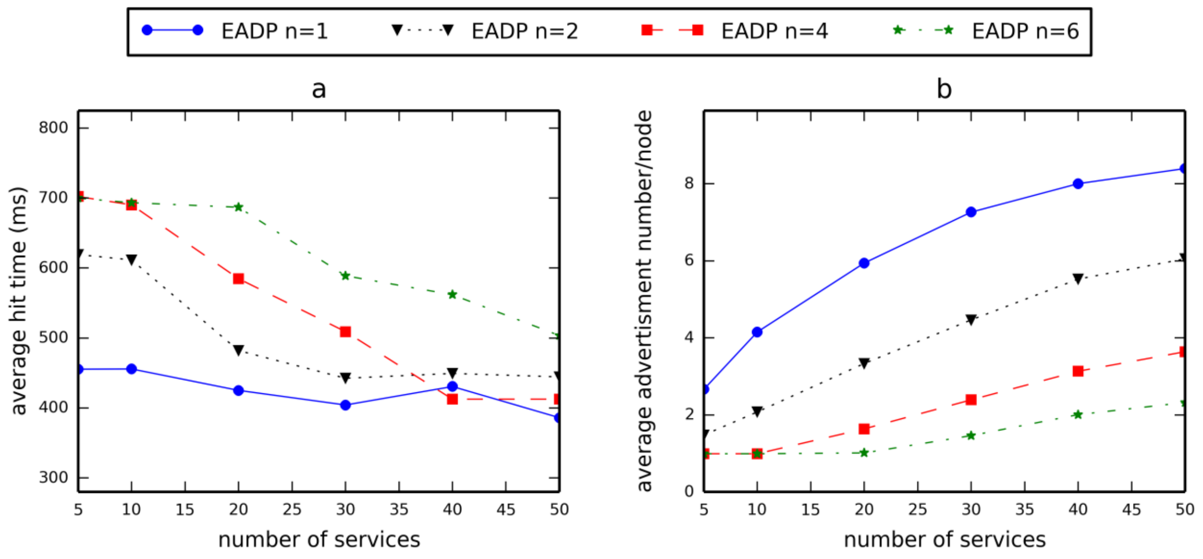


Figure 4-12 The n-inconsistency approach performance

As can be seen from these graphs, the *first-inconsistency* approach realised better hit times thanks to transmitting inconsistent service information as soon as it is available. This came at the cost of more generated traffic (Figure 4-12 (b)). For a small number of services (less than 20), the *first-inconsistency* approach realised the best hit times as it advertises as soon as an inconsistent service appears. The other approaches wait to gather *n inconsistent* services which take time especially when there are less and dispersed services. For more than 20 services, as the environment became service-rich, the *2-inconsistency* approach realised similar hit times to the *first-inconsistency* approach (Figure 4-12 (a)) with a good gain in the number of advertisements (Figure 4-12 (b)). This is done thanks to aggregating services in one message. A similar behaviour can be seen with the *4-inconsistency* approach when the environment was more service-rich (> 40 services).

4.8.3.3 Evaluation of the state maintenance mechanism

This section focuses on the evaluation of the state SM's delete-messages forwarding algorithm. The client sent 150 requests with different frequencies (1 request each 2, 6, 10 and 14 seconds). A delete-message for an already advertised service was triggered and the false negative percentage as the ratio of the number of hits, counted after triggering the delete-message, to the number of sent requests was measured (Figure 4-13 (a)); and the cost as the average number of generated packets per node was noted (Figure 4-13 (b)). This was done, following the specifications introduced in section 4.3 for the initial static network (using $k = 2$) and for the reference scenario, with $k = 1$ and $k = 2$.

As can be seen from Figure 4-13 (a), the false discovery percentage increased with increasing request frequencies. The algorithm registered better results with a redundancy constant $k = 2$ which allows nodes that did not receive the delete-message in the first transmission to get it in the second one. Thus, while in the static environment the mechanism registered its best performance with less than 10% false discovery in the highest request frequency tested (1 request each 2 seconds), in the mobile scenario, the mechanism registered a value of about 15% negative false discovery with the highest request frequency with $k = 2$ (the lowest was about 7%). This performance was realised with an average cost of about half a packet per node (Figure 4-13 (b)). This is half the cost of flooding, which theoretically can ensure zero false negatives. However, practically, it suffers from the broadcast storm problem [57]. In addition, its implementation adds to the complexity of the protocol.

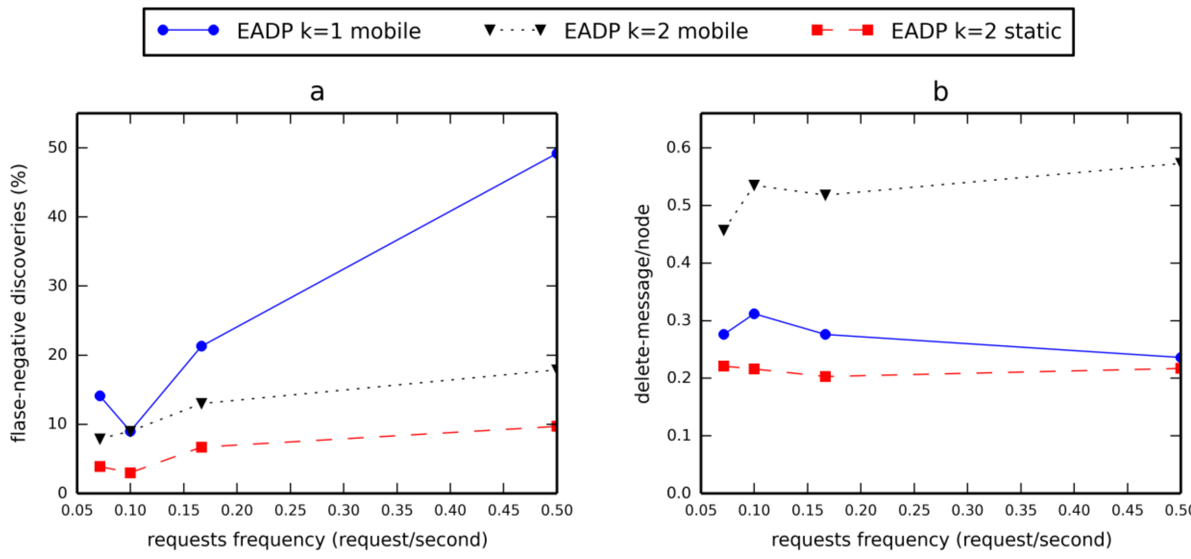


Figure 4-13 The explicit SM mechanism performance

4.8.3.4 The impact of network density on EADP

This experiment shows the impact of the network density on EADP performance when compared to ADDER-b and ADDER-w. As the density can be defined by: $density = N \times (\pi \times communication\_range^2/area)$ [148] and instead of varying the number of nodes, the side length of the square area was varied in steps of 50 metres from 250 to 450 metres. Obtained results are depicted in the first column of graphs in Figure 4-14.

As can be seen from Figure 4-14 (a), EADP showed comparable hit times to ADDER-b with about half the number of advertisements generated by ADDER-b (Figure 4-14 (b)) each containing less than 20% of data (Figure 4-14 (c)) at a density of about 3 neighbours per node. This is equivalent to the cost (number by size of advertisements) realised by ADDER-w which showed the worst hit times. With increasing densities, ADDER showed a high rise in the size of exchanged advertisements (Figure 4-14 (c)). For instance, in a network of 12.5 neighbours per node, ADDER-w and ADDER-b advertisement sizes were, respectively, more than 6 and 7 times those of EADP which kept the size of advertisements constant at less than 40 bytes. This suggests that EADP resists increasing network density while ADDER might not. This conclusion is also confirmed by EADP with $Imin = 10s$ plots. Thus, these plots show that using an $Imin$ of 10 seconds, allows EADP to present the best performance in hit times. While the size of an advertisement is density independent for both values of $Imin$ (thanks to the new

Trickle variant), EADP with $Imin = 10s$ generated more advertisements in order to achieve the best hit times. Again, this adds on the requirements for investigating better methods for achieving such hit times while saving the additional cost.

4.8.3.5 The impact of nodes' speeds on EADP

This experiment studies the impact of nodes' speeds on the performance of EADP, ADDER-b and ADDER-w. The maximum speed was varied by steps of 2 m/s from 0 to 8 m/s. Results are depicted in the second column of graphs in Figure 4-14.

With increasing speeds, the cost generated by the evaluated protocols increased. However, while the number of ADDER advertisements (Figure 4-14 (e)) remained approximately constant as nodes only send once per period, it increased slightly in EADP, but only measured about half that generated by ADDER-b at a speed of 8 m/s. Nevertheless, the size of an advertisement highly increased in ADDER. Thus with increasing speeds, nodes quickly carry services from one place to another which make them available from many nodes. Since ADDER includes all stored entries in its advertisements, their sizes keep increasing to contain all available services towards the end of the simulation (Figure 4-14 (f)). EADP on the other hand, kept the size of its advertisements constant with increasing speeds while showing comparable hit times to ADDER-b (Figure 4-14 (d)). For instance, at a speed of 6 m/s, EADP sent about half the number of ADDER-b advertisements, each containing about 6 times less data. This cost is equivalent to that realised by ADDER-w which registered the worst hit times. Note that with an $Imin = 10s$, EADP can bypass the drawbacks of the listen-only period and ensure better hit times than ADDER-b with about 7 times less cost in a speed of 8 m/s. However, such a cost is more than that of EADP with the same interval of ADDER-b, and requires enhancement. Finally, EADP's achievements suggest that it robustly resists increasing speeds.

Table 4-6 Average discovery success rate

Protocol	EADP-d	EADP	ADDER-w	ADDER-b
Discovery rate	74.16%	87.5%	78.33%	85%

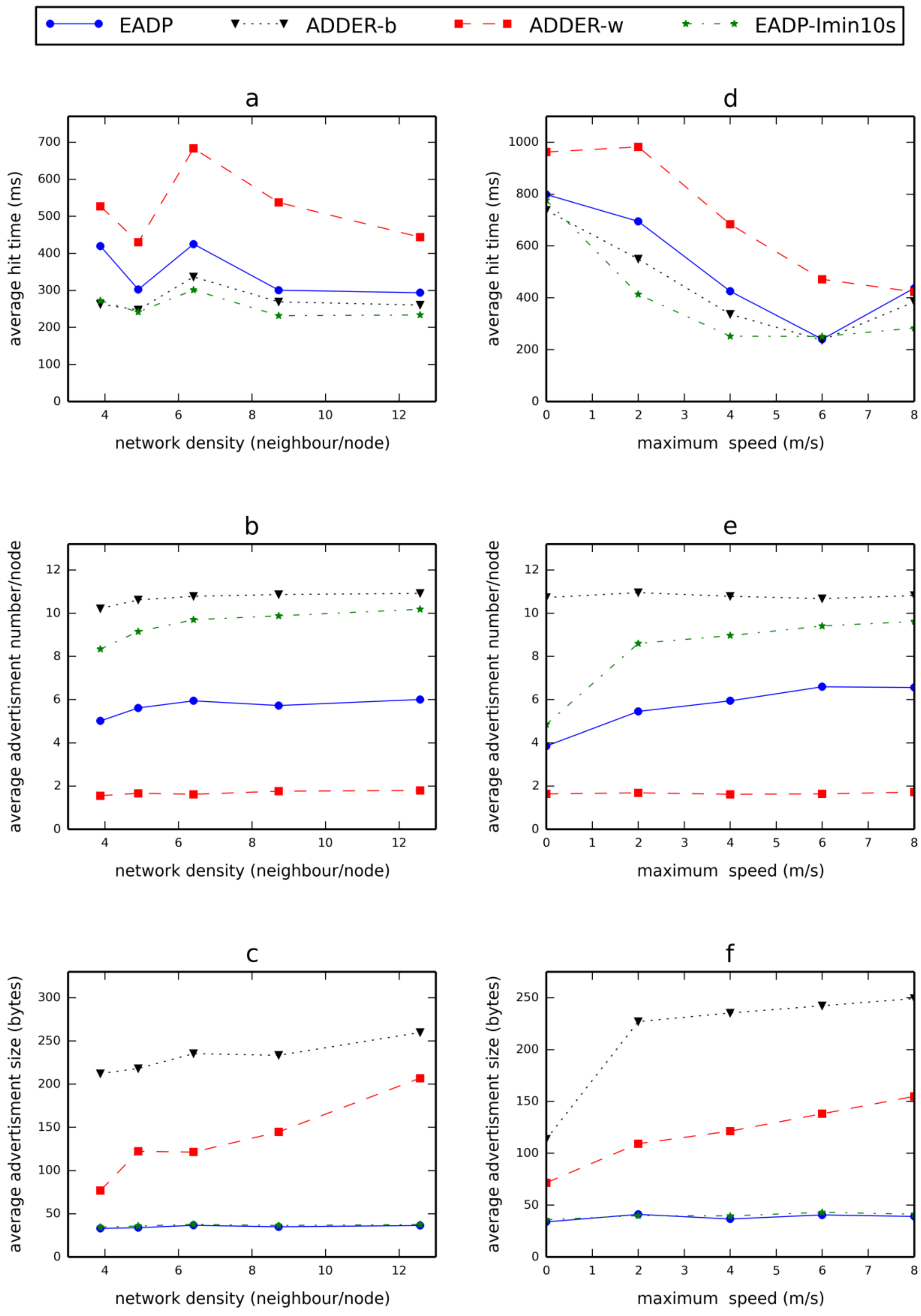


Figure 4-14 Impact of nodes' speeds and density on EADP performance

4.8.3.6 The practical discovery rate

While theoretically, the discovery rate of flooding pull protocols is expected to be 100%, practically this is not always the case. This is shown in Table 4-6, presenting the average practical discovery rates of EADP, ADDER-b, ADDER-w and EADP-d. This can be explained by losing request/reply messages caused by collisions as a result of high traffic circulating over long distances. Thus, enabling the push mode allows service information to be proactively propagated and stored in intermediate nodes which increase the practical discovery rate. From Table 4-6, EADP-d registered the worst rate because of disabling the push mode making requests and replies travel long distances thereby increasing the loss probability. ADDER-w showed the second worst discovery rate as it slowly propagated service information. EADP measured the best rate compared to ADDER-b as, in addition to propagating services fast, it minimises the number and size of the push mode messages, which reduces network load, thereby allowing more requests and replies to be delivered.

4.9 Summary

This chapter proposed, designed and evaluated EADP, an extensible adaptable discovery protocol targeting LLNs. EADP registered good performance, especially in realising a trade-off between optimal service acquisition times and minimal network overhead while ensuring a high discoverability of available services. However, many enhancements can be added to EADP, especially in controlling the pull mode based on blind flooding and enhancing the advertisement time of EADP without incurring the extra cost observed in the above results. These points will be the subject of the following chapter. Finally, it is worth noting that while EADP targets 6LoWPANs, the contributions discussed in this chapter are generic and can be used in other environments such as traditional WSN or MANET.

Chapter 5

TrickleSD: Optimised Scalable Trickle-based Service Discovery for LLNs

The previous chapter proposed EADP: an adaptable, extensible discovery protocol designed for pervasive CNNs. EADP focused on optimising the push mode while it relies on flooding for its pull mode. This chapter investigates lightweight mechanisms to substitute EADP's pull mode algorithm. From those, a generic Trickle version stands as a promising solution. However, Trickle's latency issue, observed in the previous chapter, might prevent its adoption for such a purpose. To address such an issue, this chapter presents, analyses and scrutinises a generic version of Trickle. Next, it addresses the author's criticisms of Trickle allowing the introduction of two main optimisations namely Optimised Trickle (Opt-Trickle) and Augmented Trickle. Subsequently, a time-efficient Trickle-based pull algorithm is proposed. Using such an algorithm to substitute EADP's pull algorithm, along with enhancements to other EADP's components, gives birth to the Trickle-based service discovery for LLNs (TrickleSD). The chapter ends with evaluations and discussions of its main contributions, namely, Opt-Trickle and TrickleSD.

5.1 Flooding substitution techniques

Flooding is widely deployed in wireless ad hoc and sensor networks. It is used to establish routes in unicast routing protocols (e.g., AODV [145], DSR [144], LOADng [149]), to deliver data in query-based protocols [92], and to realise SD [21], [127]. Such a usage comes from the simplicity, stateless nature, and ease of implementation of flooding. While this allows it to be implemented even in very constrained nodes, it makes flooding ineffective, in dense networks, in terms of energy consumption, bandwidth utilisation and reliability, as it may generate an abundant number of redundant transmissions and can lead to the broadcast storm problem [150].

To address the broadcast storm problem, a plethora of lightweight solutions has been proposed. Those can mainly be categorized into (adaptive) probabilistic-cancel approaches and delay-and-cancel techniques. The latter includes location-based [150], [151], counter-based [150], [152], [153], distance-based schemes [150], [153], and their adaptive versions. For instance, Trickle uses a counter-based approach where if nodes hear a message k times it suppresses its transmission. The suppression scheme deployed in SLIM [113] uses RSSI as an indicator of the distance to allow the farthest node from the originator to transmit first, other nodes hearing such a message delete their transmissions. This scheme seems very attractive to substitute EADP's pull mode algorithm. In this context, the author has studied such a scheme and has proposed an enhancement to its time/cost performance in [154]. However, with the introduction of a fourth parameter to Trickle, which allows its usage as a flooding substitute, the work in [154] has been abandoned in favour of Trickle. This move is also backed by the need to optimize Trickle's performance for the push mode identified in the previous chapter. Nevertheless, the experience and techniques proposed in [154] add to the contribution of enhancing Trickle (section 5.5.3).

5.2 Trickle as a flooding substitute

This section introduces Trickle with the fourth parameter proposed in MPL and discusses the latency issue introduced by the listen-only period.

5.2.1 Trickle with the fourth parameter

MPL [80] being currently standardized by the IETF introduced a fourth parameter – $TimerExpirations$ to the three parameters defined by Trickle (section 2.5.2).

- $TimerExpirations$: Specifies the number of Trickle timer expirations, since the last timer reset, which allows terminating Trickle's execution. Put another way, this is a response to the infinite duration of the Trickle's maintenance mode; not required by some applications such as flooding substitution.

This configuration parameter implies an additional variable: the expiration counter e.

- $\boldsymbol{e}$**:** A counter tracking the number of Trickle timer expirations that occurred since the last timer reset.

Integrating this modification to the 6-step algorithm presented in section 2.5.2, $\boldsymbol{Step\ 1}$, $\mathbf{5}$ and $\mathbf{6}$ can be modified as follows (modified parts are underlined):

- $\boldsymbol{Step\ 1}$: When Trickle starts execution, it picks I uniformly at random from $[Imin;\ Imin \times 2^{Imax}]$, sets e to zero and begins the first interval.
- $\boldsymbol{Step\ 2}$: At the start of an interval, Trickle resets c to 0 and picks t uniformly at random from $[I/2; I)$.
- $\boldsymbol{Step\ 3}$: Whenever a node hears a *consistent* transmission, Trickle increments c.
- $\boldsymbol{Step\ 4}$: At time t, Trickle transmits if and only if c is less than k ($c < k$). Otherwise, the transmission is suppressed.
- $\boldsymbol{Step\ 5}$: At the expiration of an interval, Trickle increments the expiration counter e. If e is equal to $TimerExpirations$, Trickle stops execution. Otherwise, Trickle doubles the current interval size I up to the time specified by $Imax$. Trickle then starts a new interval as in $\boldsymbol{Step\ 2}$.
- $\boldsymbol{Step\ 6}$: If an inconsistent transmission is received while I is greater than $Imin$, the receiver resets the Trickle timer. To do so, Trickle sets I to $Imin$, e to zero and starts a new interval as in $\boldsymbol{Step\ 2}$. Otherwise, Trickle does nothing. Note that the timer can also be reset by application-defined events external to Trickle.

5.2.2 The listen-only period

A noticeable point in the Trickle rules (particularly $\boldsymbol{Step\ 2}$), which is depicted in Figure 5-1 (c), is the so-called listen-only period spreading over the first half of each interval, hence dividing it into two main parts: a listen-only part and a transmission period. Indeed, this listen-only period is introduced in response to a challenging problem to Trickle called the short-listen problem [23].

The short-listen problem occurs because of non-synchronised intervals between neighbours. It has a drastic impact on Trickle's suppression mechanism, thereby on Trickle's scalability. Figure 5-1 (a) and (b) illustrate the effects of the short-listen problem in a single-hop network comprising three nodes. Figure 5-1 (a) shows the expected efficiency of the suppression mechanism if node intervals are synchronised. Thus, even when considering a worst-case of the random transmission time selection process making N1 transmit at the beginning of every interval, nodes N2 and N3 are able to catch this transmission before sending their own data and hence can suppress their transmissions. This allows Trickle to scale with k transmission per interval.

However, if node intervals are not synchronised (Figure 5-1 (b)), no node can hear N1's transmission before its own, hence nodes N2 and N3 keep competing to transmit. Suppose that N2 listens only for a short time before transmitting, then N3 will not be able to hear such a transmission before its own and hence decides to send, which makes the suppression mechanism useless in this particular case. As this problem is caused by nodes choosing to listen for short periods, it is dubbed short-listen problem in [23].

In the general case, [23] shows that the short-listen problem causes the number of transmissions per interval to scale as $O(\sqrt{N})$ (N being the number of nodes in a single-hop lossless network), instead of k in synchronised lossless networks or the aimed at $O(log(N))$ in lossy networks. Getting to synchronize node intervals and maintain synchronisation between them is a resource consuming task. Furthermore, even if node clocks can be synchronised, there is no guarantee that Trickle intervals can be too. Indeed, besides losses, Trickle itself can cause non-synchronised intervals.

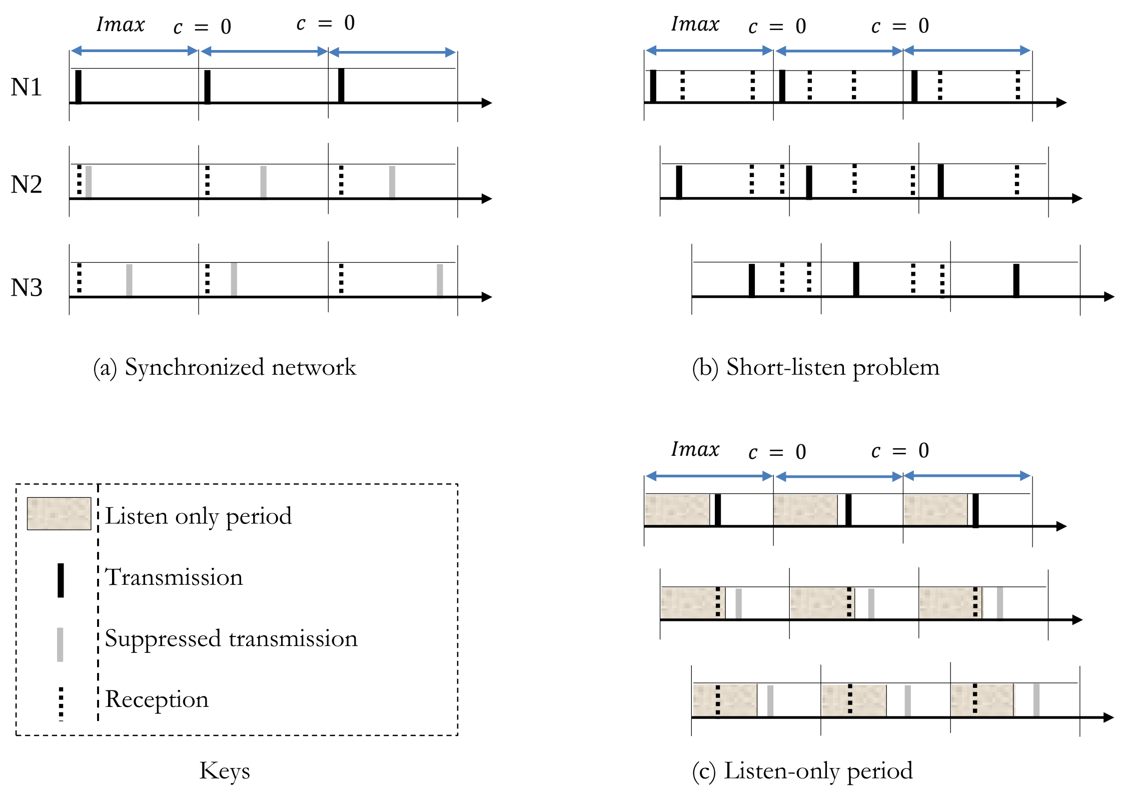


Figure 5-1 Short-listen problem and listen-only period with k = 1.

A simple, stateless and yet powerful solution is to impose a listen-only period at the start of each interval. In this period, a node only listens for incoming messages. Such a period has shown to bound the number of messages per interval by a constant inversely proportional to the size of the listen-only period [23]. However, a bigger listen-only period might have a dramatic impact on the propagation time as it delays a transmission by at least the length of the listen-only period at each hop. Opting for a fair time/cost trade-off, Trickle defaults to a listen-only period of a half-interval ($\boldsymbol{Step\ 2}$), which asymptotically bounds the number of transmissions per interval by $2 \times k$ in lossless networks [23]. As can be seen from Figure 5-1 (c), the default listen-only period allows nodes $\mathrm{N2}$ and $\mathrm{N3}$ to suppress their transmissions. This brings Trickle's scalability to the desired $O(log(N))$ in the general case of lossy networks.

5.2.3 Criticisms of the listen-only period

The subsection discusses the main issues caused by the listen-only period.

5.2.3.1 Increased inconsistency propagation time

As shown above, the listen-only period allows Trickle to scale logarithmically with network density at the expense of increased delays. Such delays have the most impact when resolving inconsistencies, as they postpone every transmission by at least half of an ($Imin$) interval. This makes the introduced delay heavily dependent on the value of $Imin$, further aggravating the latency in networks adopting relatively large $Imin$ values, such it is the case of EADP's push mode. Additionally, this $Imin$-dependent delay gets accumulated at every hop in multi-hop networks, which results in a considerable latency for a packet travelling long distances (in terms of hops).

5.2.3.2 Unbalanced transmission loads

Added to the aforementioned main issue, the listen-only period might introduce unbalanced transmission loads in the network, making some nodes transmit more than others. This issue is illustrated in Figure 5-2 which depicts Trickle intervals of three neighbours receiving an update from different senders. As can be seen from this figure, node $\mathrm{N1}$ has the biggest chance to transmit in the $Imin$-interval compared to nodes $\mathrm{N2}$ and $\mathrm{N3}$. Indeed, $\mathrm{N3}$ has zero-chance to transmit as its listen-only period is totally overlapped with the transmission period of $\mathrm{N2}$. Hence $\mathrm{N3}$'s transmission is explicitly prevented by the listen-only period, not by the suppression mechanism (more on this in the following section). From the next interval (I_1 in Figure 5-2), the chances of N2 and N3 to transmit slowly increase and keep increasing with increased interval sizes. For instance, in the fourth interval, the three nodes have similar transmission chances. This suggests that choosing bigger values of $Imax$ can help distributing Trickle's transmission load equitably.

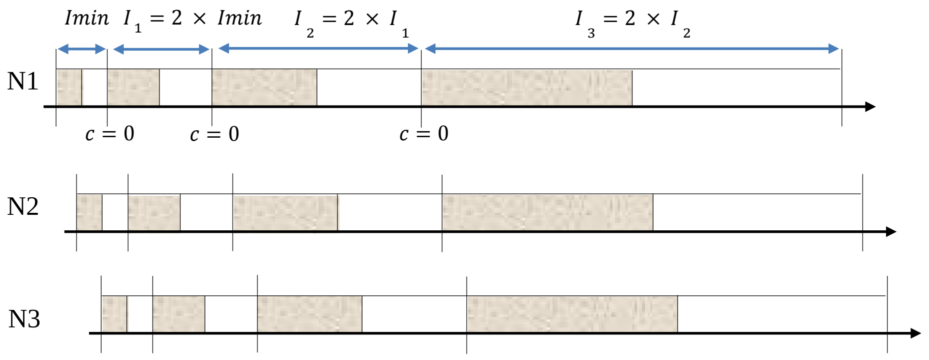


Figure 5-2 Trickle's load balancing issue

5.2.3.3 Explicit prevention of transmissions

This issue is illustrated via an example of a multi-hop lossless network depicted in Figure 5-3 (a). When a seed node S0 transmits an update, all its neighbours (S1 and N2 being two of them) shrink their intervals to $Imin$. At the end of its listen-only period, S1 transmits the received update. S1's neighbours which receive the update for the first time (N3 being one of them) will shrink their intervals to $Imin$. Before transmitting, N3 has to wait for at least the length of the listen-only period, which entirely overlaps with N2's transmission period, hence forcing N3 to suppress its transmission (Figure 5-3 (b)). This stops the update from reaching N4 and N5 in this interval, delaying them by at least another $Imin$ (which is the length of the next interval's listen-only period). This is with fewer chances for N3 to transmit compared to N2 and S1.

This problem is more visible in sparse networks or networks containing irregularities. It might also become worse if an application deploys smaller $Imax$ values; such is the case of MPL's proactive mode. Finally, it should be noted that opting for a $k > 1$ can minimise the likelihood of this problem, however, it adds to Trickle's cost.

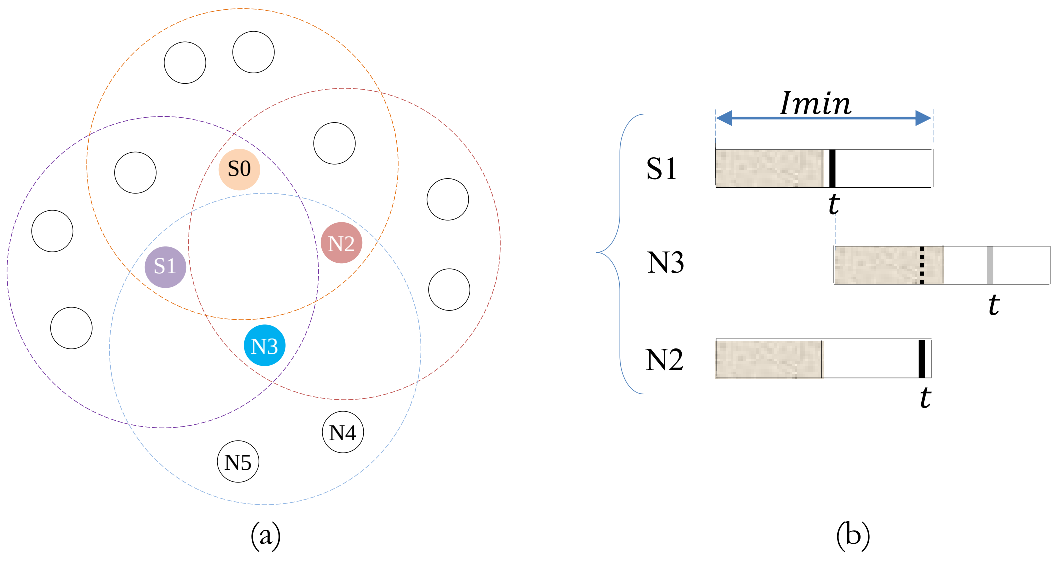


Figure 5-3 The listen-only period preventing nodes from transmitting (k = 1)

Other issues such as shortening the size of the contention interval can also arise. However, such issues are not discussed here since their effects are unpredictable as will be seen in section 5.5.2.

5.3 The Opt-Trickle algorithm

To address the above criticisms, this section introduces a simple, yet effective optimisation giving birth to the Optimised Trickle algorithm (Opt-Trickle).

5.3.1 The proposed optimisation

The proposed optimisation is based on a fundamental observation from $\boldsymbol{Step\ 6}$ of the Trickle algorithm. $\boldsymbol{Step\ 6}$ triggers the nodes receiving an inconsistency to immediately (assuming that receptions occur simultaneously) start new intervals of size $Imin$ (if $I > Imin$). This can present an implicit synchronisation of $Imin$-sized intervals between these nodes, which comes at no cost and exactly when needed. Such a synchronisation can allow these nodes to choose I from $[0; Imin)$ without experiencing a short-listen problem with each other. Based on this observation, the author proposes to modify $\boldsymbol{Step\ 2}$ of the Trickle algorithm as follows:

- ***Step* 2**: At the start of an interval, Trickle resets c to 0 and picks t uniformly at random from:
  - $[0;\ Imin)$, if the interval began as a result of ***Step* 6** (because of an *inconsistency* or in response to external *events*).
  - $[I/2; I)$, otherwise (the interval began as a result of ***Step* 1** or ***Step* 5**).

Note that neighbours can experience non-synchronised $Imin$-sized intervals as a result of losses and/or the multi-hop nature. Fortunately, an implicit synchronisation in the transmission periods of these intervals remains valid, as will be detailed in section 5.4. However, there is no guarantee of implicit synchronisation in the following intervals, and hence the listen-only period is deployed.

5.3.2 Expected latency achievements

As Trickle resolves inconsistencies in $Imin$-sized intervals, Opt-Trickle is expected to drastically decrease the propagation time of Trickle at virtually no extra cost. At first glance, it can be thought of the propagation time being halved. However, many parameters (e.g., $Imin$ value, network density, hop count) can influence the propagation time, allowing it to be much faster.

To quantify this latency gain, suppose that D neighbours have received an update and shrunk their intervals to $Imin$, thanks to the uniform choice of t, the expected time between successive transmissions is $Imin/(D+1)$. This gives an expected latency before a first node transmits the update, from the time of receiving it, of:

$$E_t(Opt - Trickle) = \frac{Imin}{D+1} \tag{5-1}$$

However, for Trickle and because it deploys the listen only period, this latency is:

$$E_t(Trickle) = \frac{Imin}{2} + \frac{Imin/2}{D+1} \tag{5-2}$$

If such an update is to be propagated over a L_p hops path, then Trickle's latency is:

$$E_t(Trickle) = L_p\left(\frac{Imin}{2} + \frac{Imin/2}{D+1}\right) \tag{5-3}$$

While Opt-Trickle can simply propagate such a packet in:

$$E_t(Opt - Trickle) = L_p\left(\frac{Imin}{D+1}\right) \qquad 5\text{-}4$$

Thus, Trickle's latency has a linear relationship with number of hops and $Imin$, while Opt-Trickle avoids this linearity and hence can achieve very fast propagations. Having briefly presented and shown the principal benefit of Opt-Trickle, the following section discusses its impact on Trickle's cost and scalability.

5.4 Scalability of Opt-Trickle

This section discusses Opt-Trickle's scalability. It starts by assuming a simple single-hop lossless network. Next, the assumption is relaxed by looking at multi-hop lossless networks and then by introducing losses in single- and multi-hop networks. New nodes joining the network are implicitly included in this analysis. Without loss of generality, $k = 1$ is assumed in what follows.

5.4.1 Lossless, single-hop networks

When a node $\mathrm{N1}$ propagates an update in a single-hop lossless network, all other nodes will receive it and, by Trickle's $\boldsymbol{Step\ 6}$, immediately start new $Imin$-sized intervals. This can constitute an implicit-synchronisation between these nodes. Hence, the short-listen problem is not experienced if they choose t from $[0;\ Imin)$, as shown in Figure 5-4. Note, however, that whichever receiver retransmits the update (e.g., $\mathrm{N2}$, $\mathrm{N3}$ or $\mathrm{N4}$ in Figure 5-4), it might experience short-listen with the second interval of the originator ($\mathrm{N1}$). The impact of this is discussed in section 5.4.5.

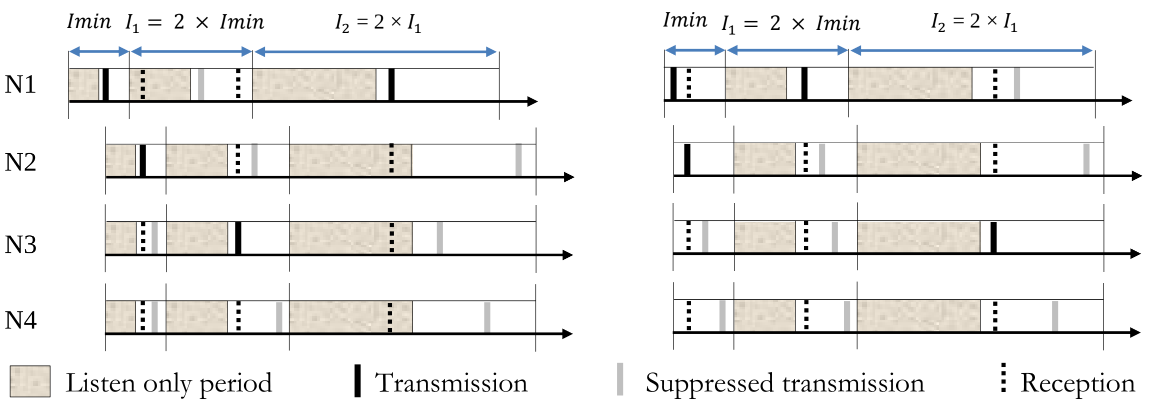
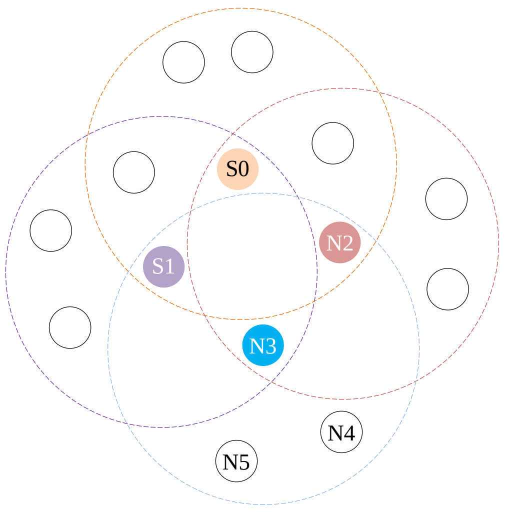


Figure 5-4 Trickle (left) and Opt-Trickle (right)

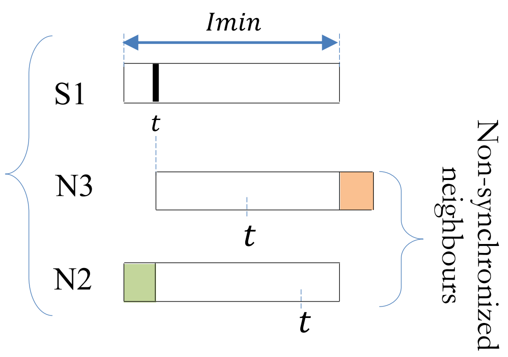
This idealistic case shows the basic intuition behind the proposed optimisation. It also clearly demonstrates that the listen-only period of the $Imin$ interval grows the interval skew between neighbours, for instance, between the originator (N1) and the other nodes depicted in Figure 5-4.

5.4.2 Lossless, multi-hop networks

The case of lossless, multi-hop networks can be explained with an example depicted in Figure 5-5 (a) (similar to the one presented in Figure 5-3 for Trickle). Suppose that a seed node S0 has D direct neighbours. As the network is lossless, all the D nodes (S1 and N2 being two of them) shrink their intervals to $Imin$ when receiving $S0's$ update. Opt-Trickle allows such nodes to choose t from $[0;\ Imin)$ as they are implicitly synchronised. Suppose now that S1 is the first to retransmit the update. S1's neighbours that receive the update for the first time (N3 being one of them) will shrink their intervals to $Imin$ and thereby can choose t from $[0;\ Imin)$ without experiencing a short-listen problem between each other. Other S1's neighbours hearing the retransmission simply suppress their transmissions.



(a)



(b)

Figure 5-5 Non-synchronised Imin intervals

The aforementioned process may result in non-synchronised $Imin$-sized intervals between nodes N2 and N3, which are neighbours competing to propagate the update, as shown in Figure 5-5 (b). This phenomenon challenges the implicit synchronisation observed in the previous idealistic case. Nevertheless, because N2 is still competing to transmit, meaning it did not send in the past part of its interval (the green rectangle in Figure 5-5 (b)), the beginning of the transmission periods of both N2 and N3 are implicitly synchronised. Therefore, either N2 or N3 transmits first; the other transmission will be suppressed and hence no short-listen problem would be experienced by either. Finally, it should be noted that as the network is lossless, N3 also cannot transmit in the orange part of its $Imin$ interval when $k = 1$ (Figure 5-5 (b)). This is because N2 would have sent before the end of its interval, and would have suppressed N3's transmission.

The implicitly imposed non-transmitting green and orange parts in the $Imin$ intervals of N2 and N3 allow for perfectly synchronised equal transmission periods that give the same transmission probability to N2 and N3. Thus, even if the suppression mechanism prevents N3 from transmitting in this interval, N3 will get approximately the same chance as N2 and S1 to transmit in the following interval. Note that if a $k > 1$ is used, the orange part of the interval is not guaranteed. However, this does not harm Opt-Trickle, as the beginnings of the transmission periods remain synchronised, which prevents the short-listen problem. In addition, all the neighbours are given the same chance to transmit as early as they can.

Other cases of non-synchronised $Imin$ intervals between neighbours have been experimentally observed, some of which are depicted in Figure 5-6. Nodes are deployed at every point in the grids shown in Figure 5-6 where corner nodes have 3 neighbours, border – non-corner – nodes have 5 neighbours and the others have 8 neighbours per node. Border node S0 initiates an update to be propagated in the network. In the three cases, whichever node of the two designated by a circle transmit, makes the $Imin$ interval of the node designated by a star non-synchronised with the other node designated by a circle. The difference between these cases is in the amount of skew in the $Imin$ intervals.

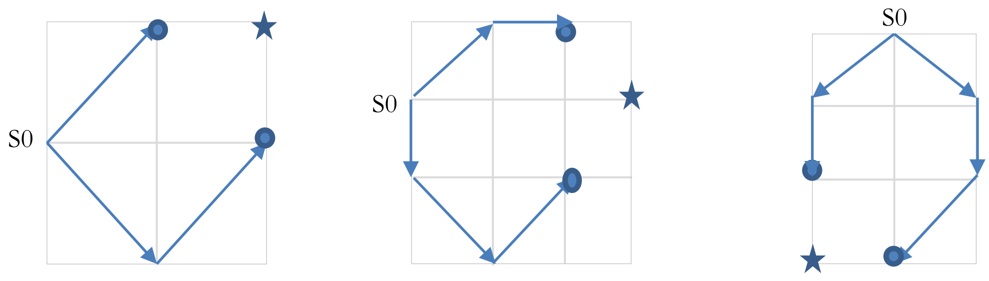


Figure 5-6 Observed non-synchronised Imin intervals.

Note that the observed non-synchronised *Imin* intervals in multi-hop physically lossless networks (recall that losses can also emerge from network dynamics) occurred between two (groups of) nodes. Nevertheless, for illustrative purposes, this case is generalised in Figure 5-7 in order to show that even if more than two neighbours have non-synchronised *Imin* intervals, the short-listen problem is not observed. Thus, Figure 5-7 depicts a case of 4 non-synchronised neighbours N1, N2, N3 and N4, which are competing to transmit an update using both Opt-Trickle (Figure 5-7 (a)) and Trickle (Figure 5-7 (b)). Note that it might be rare that the two algorithms can arrive at a similar situation.

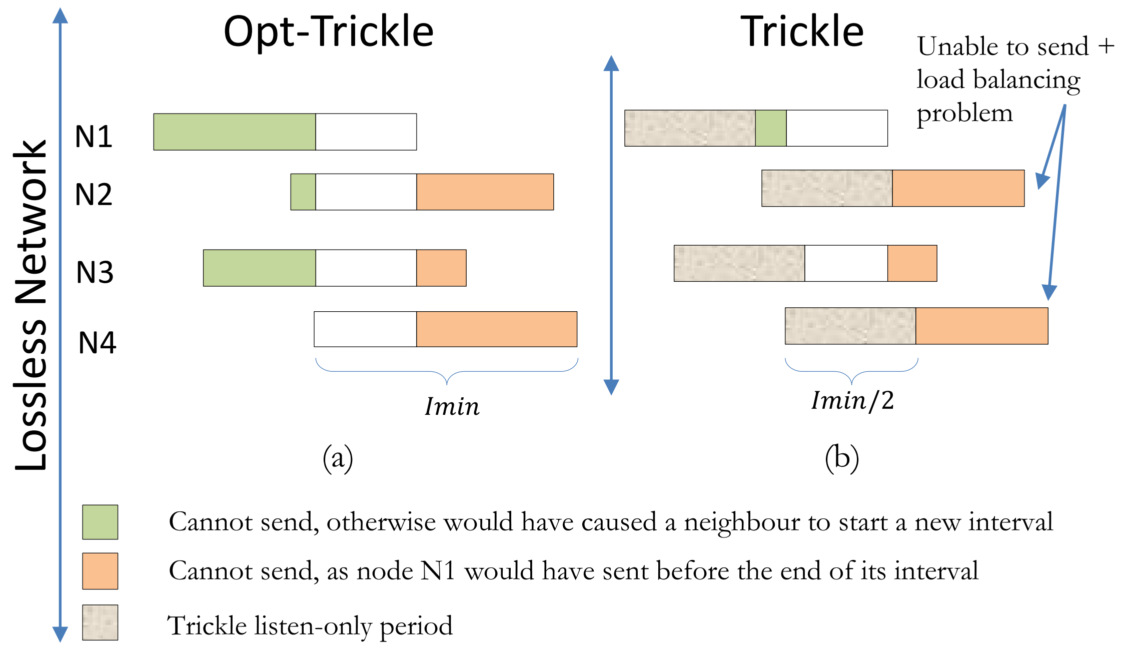

Opt-Trickle
Trickle
Unable to send + load balancing problem
Lossless Network
N1
N2
N3
N4
$Imin$
$Imin/2$
(a)
(b)
Cannot send, otherwise would have caused a neighbour to start a new interval
Cannot send, as node N1 would have sent before the end of its interval
Trickle listen-only period

Figure 5-7 Non-synchronised Imin intervals in lossless networks

Similarly to the discussion carried out regarding Figure 5-5, no node can transmit in the green part of the interval; since such transmissions would have caused other nodes to start new intervals (which is not the case in this example). Also, it is impossible for nodes N1, N2 and N3 to transmit in the orange part of their intervals when using a redundancy constant $k = 1$, as N1 would have transmitted by the end of its interval and thereby would have suppressed such transmissions. This leaves only the white parts of the intervals for potential transmissions. As can be seen from Figure 5-7 (a), the white parts of the intervals are fully synchronised between all the neighbours, giving them an equal chance to transmit. On the other hand, and despite the fact that this example might have favoured Trickle, Figure 5-7 (b) shows that two of the four neighbours are prevented from transmitting by the listen-only period. The two remaining nodes do not have the same chance to transmit.

5.4.3 Lossy, single-hop networks

The authors of [23] have shown that losses can cause Trickle to scale logarithmically with network density. This is also the scalability aimed at by Opt-Trickle.

In a lossy single-hop network, when a seed node $S0$ propagates an update, some of its neighbours will hear it, and others will miss it. The group of nodes hearing it ($S1$ being one of them) will immediately shrink their intervals and hence can choose t from $[0;\ Imin)$ without experiencing a short-listen problem. When a first node $S1$ from the group transmits, again some nodes will hear its transmission, and others will miss it. Let us consider what happen:

1. Nodes hearing $S1$'s transmissions can be divided into two categories:
    a. Nodes that have already heard $S0$'s transmission, which simply suppress their retransmissions.
    b. Nodes that have not heard $S0$'s transmission immediately shrink their intervals and can choose t from $[0, Imin)$ without experiencing a short-listen problem with each other.
2. Nodes not hearing $S1$'s transmission can also be divided into two categories:
    a. Nodes that have not also heard $S0$'s transmission; do nothing.
    b. Nodes that have heard $S0$'s transmission, started $Imin$-sized intervals in the past, and are still competing to transmit the update.

Clearly nodes in categories 1.b. and 2.b. might be competing to transmit the update and are not synchronised. Thankfully, the short-listen problem is not experienced between these categories. To simplify, take a node from category 1.b. and another from category 2.b., whichever transmits first, annuls the other's transmission unless the transmission is lost (nothing to do about it).

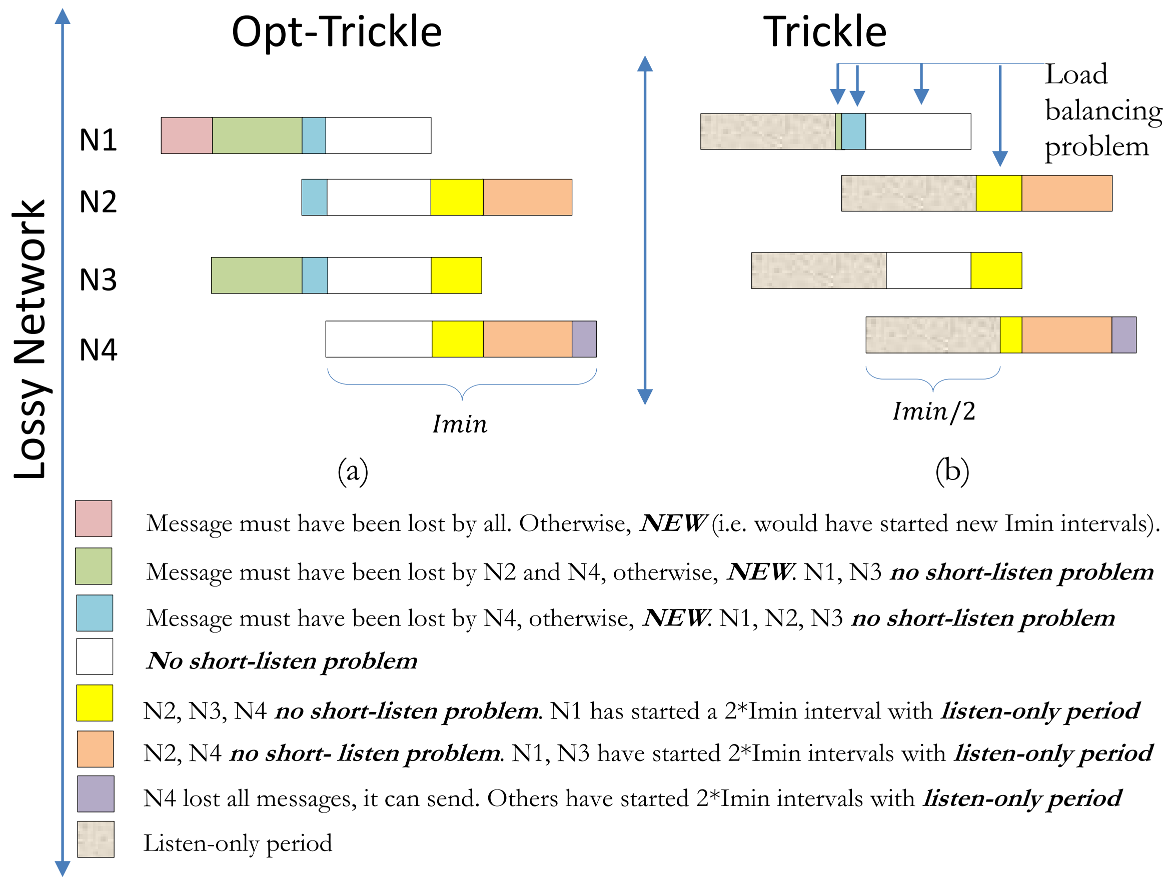
To generalize this case, let us suppose a single-hop network in which M non-synchronised (group of) nodes are competing to transmit a previously received update.

The remaining nodes are denoted by R. The following points examine what happens, in the M and R sets, when a first node $\mathtt{N1}$ from M transmits.

1. Suppose that H nodes from M will hear $\mathtt{N1}$'s transmission, hence they suppress their transmissions.
2. The remaining $M - H - 1$ nodes from M miss it because of losses; hence they continue competing to propagate the update.
3. Now consider that L nodes from R have heard the update for the first time. They start new $Imin$-sized intervals and they will be competing with the $M - H - 1$ nodes to propagate it.
4. The remaining $R - L$ nodes, which either did not hear $\mathtt{N1}$'s transmission because of losses or they are already aware of the update, keep quiet.

The only possibility for short-listen to occur is in step 3 of the above process. This case is examined in the example depicted in Figure 5-8, where $M = 4$, $L = 0$ and the remaining $R - L$ nodes are already aware of the update and hence are not shown.

Figure 5-8 discusses all possible transmission combinations of the four nodes ($\mathtt{N1}$, $\mathtt{N2}$, $\mathtt{N3}$ and $\mathtt{N4}$) in both Opt-Trickle and Trickle. Similarly to the analysis conducted in lossless multi-hop networks, and with the aid of coloured keys, Figure 5-8 (a) shows that short-listen is not experienced between these nodes when using Opt-Trickle. Moreover, the transmission periods of the nodes are fully synchronised, giving them an equal chance to transmit. On the other hand, Trickle might prevent some nodes from transmitting and may create unbalanced loads (Figure 5-8 (b)).



🟥 Message must have been lost by all. Otherwise, ***NEW*** (i.e. would have started new Imin intervals).

🟩 Message must have been lost by N2 and N4, otherwise, ***NEW***. N1, N3 ***no short-listen problem***

🟦 Message must have been lost by N4, otherwise, ***NEW***. N1, N2, N3 ***no short-listen problem***

⬜ ***No short-listen problem***

🟨 N2, N3, N4 ***no short-listen problem***. N1 has started a 2*Imin interval with ***listen-only period***

🟧 N2, N4 ***no short- listen problem***. N1, N3 have started 2*Imin intervals with ***listen-only period***

🟪 N4 lost all messages, it can send. Others have started 2*Imin intervals with ***listen-only period***

▨ Listen-only period

Figure 5-8 Non-synchronisation in lossy networks

5.4.4 Lossy, multi-hop networks

This section discusses the generic case of multi-hop lossy networks. Various cases of non-synchronised *Imin* intervals between neighbours can occur as a result of overlapping regions (Figure 5-7) or losses (Figure 5-8) or a combination of both. This scenario can be generalised to the cases discussed in sections 5.4.2 and 5.4.3. For instance, if one assumes that losses do not occur during *Imin* intervals, then such a situation is encapsulated in the generalised scenario depicted in Figure 5-7. Otherwise, neighbours' interactions can be captured by the generic case of losses illustrated in Figure 5-8. Fortunately, in both cases, Opt-Trickle does not only avoid the short-listen problem but also ensures synchronised transmission periods of *Imin* intervals.

5.4.5 The big picture

Having shown that Opt-Trickle does not suffer from the short-listen problem in $Imin$-sized intervals, this section puts these intervals in the larger context of Trickle's behaviour and determines whether Opt-Trickle preserves Trickle's scalability.

Let us start with the example of a perfect lossless single-hop network, depicted in Figure 5-4 (section 5.4.1). This example shows that Trickle deliberately prevents the originator node N1 from transmitting in the second interval (I_1 interval in Figure 5-4), as the transmission of N2, N3 or N4 in the $Imin$ interval forcibly coincides with the listen-only period of the second interval of N1. However, Opt-Trickle does not guarantee such a characteristic. Nevertheless, while Opt-Trickle allows all the nodes to transmit in the second interval, instead of only N2, N3 or N4 in the case of Trickle, the number of transmissions in the second interval is k for both algorithms.

Let us now take the generic case of lossy networks illustrated through the example depicted in Figure 5-9. In this case, because of losses, only N2 and N3 hear N1's update. N4 will receive the update from the second transmission (i.e. N2's transmission). The dashed red lines in Figure 5-9 show the transmit-listen interplay between $Imin$ interval transmissions and following intervals' listen-only periods. Although the $Imin$-sized intervals' transmissions does not experience short-listen with each other for both Trickle, Opt-Trickle, Trickle makes sure that such transmissions coincide with other intervals' listen-only periods (e.g., N1 and N2 in Figure 5-9), hence it might help to delete their transmissions. Opt-Trickle, however; does not provide such a guarantee. This can make Opt-Trickle transmit more messages in the second interval compared to Trickle.

As the second interval deploys the listen-only period, and as this additional cost is caused by losses, the number of transmissions in this interval still scales logarithmically with network density. Additionally, Trickle's transmit-listen interplay between second interval transmissions and the third interval's listen-only periods decreases. Thereby the additional cost which might be generated by Opt-Trickle in the third interval is much lower than that in the second interval. This continues, so that from the third interval, the two protocols might generate the same cost. It should be noted that while this section discussed the case of losses, the small additional cost is mainly caused by non-synchronised $Imin$ intervals which can also emerge in lossless multi-hop networks (section 5.4.2). In addition, it might be expected that the extra cost can slightly increase with increasing k. However, since this cost is density independent, it does not influence the logarithmic scalability. Therefore, Opt-Trickle preserves Trickle's scalability.

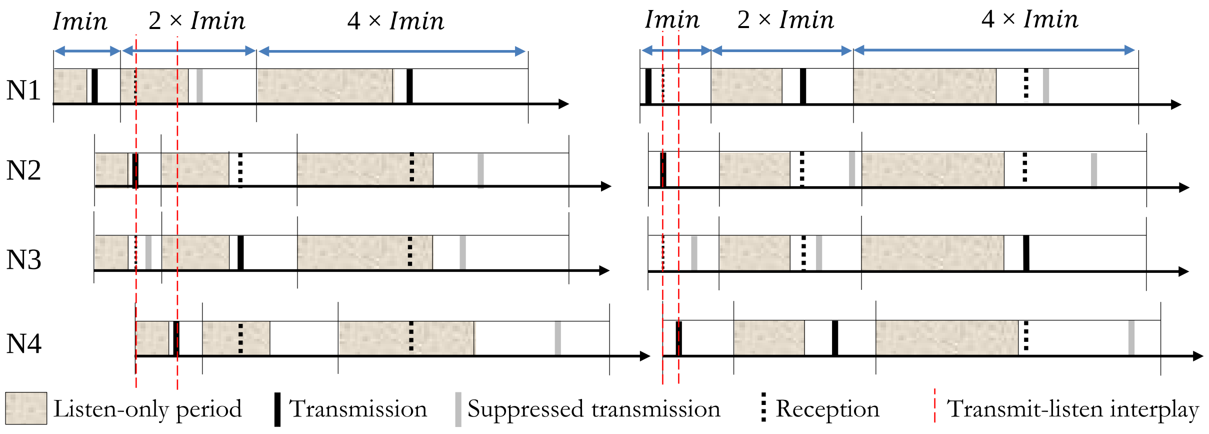


Figure 5-9 Transmit-listen interplay: Trickle (left) and Opt-Trickle (right).

5.5 Other benefits and implications of Opt-Trickle

In previous sections, it was shown that choosing t from $[0; Imin)$ can allow Opt-Trickle to propagate dramatically faster without resulting in a short-listen problem between competing neighbours. It was also demonstrated that although a small additional cost can occur in Opt-Trickle; this cost does not influence Trickle's scalability. In this section, some other benefits that can result from Opt-Trickle are outlined. They mainly address the remaining criticisms discussed in section 5.2.3.

5.5.1 Load balancing

Trickle inherits a balanced load distribution arising from the uniform random choice of transmission time. However, this balanced load can be challenged by the listen-only period as explained in section 5.2.3. As shown in that section, unbalanced load distribution has more chances to occur in small intervals (especially $Imin$-sized intervals), where it has the most impact. Additionally, it was shown in section 5.2.3 that the listen-only period of $Imin$ intervals may explicitly stop some transmissions, thus preventing parts of the network from being quickly updated. Throughout the above analysis (section 5.4), Opt-Trickle gave all competing nodes similar chances to transmit an update, which allows it to solve this serious issue. In what follows, the focus is on how Opt-Trickle helps to bring a balanced load distribution. To this end, the generic case of lossy networks depicted in Figure 5-9 is discussed below.

As can be seen from Figure 5-9, Trickle imposes, on every node, a wait of at least the size of the listen-only period before propagating an update. This skews the intervals of the receivers by at least $Imin/2$ from the originator. A receiver from those (e.g., node $\mathrm{N2}$ in Figure 5-9) has to wait for at least another $Imin/2$ before transmitting. As a result, a receiver of such an update (for instance, node $\mathrm{N4}$ in Figure 5-9) is again shifted by at least $Imin/2$ from $\mathrm{N2}$ and by $Imin$ from the seed. This process gets aggravated by heavy losses, which adds to interval skews. This in turn might give some nodes more chances to transmit in the following intervals as discussed in section 5.2.3. Opt-Trickle, however, does not impose any restriction on nodes competing to transmit an update. Hence, in addition to giving competing nodes the same chances to transmit, it allows for smaller interval skews as shown in Figure 5-9.

To see the impact of the above in practice, an experiment was conducted in TOSSIM. A Trickle application (similar to Setup 1 described later on in section 5.7.1) is deployed in a sparse grid topology of 15×15 nodes. The topology and link configurations are those of *15-15-sparse-mica2-grid.txt*[9] example available in TOSSIM. An artificial noise of -115 dBm was used to feed TOSSIM's noise model [155]. The standard deviation of the number of transmissions per node is measured as a metric of the load distribution: the smaller the standard deviation the better balanced the transmission loads. Obtained results are presented in Table 5-1. As can be seen from this table, both Trickle and Opt-Trickle try to provide a balanced load distribution between nodes. Opt-Trickle provides better load balancing than Trickle for both small and big values of $Imin$ with the biggest gap observed with small $Imin$ values. To further get a feel of the dispersion of transmissions between nodes, Figure 5-10 presents the transmission topography of Trickle and Opt-Trickle for both values of $Imin$ presented in Table 5-1.

[9] https://github.com/tinyos/tinyos-main/tree/master/tos/lib/tossim/topologies

Table 5-1 Load balancing metric

$Imax = 8 \times Imin$ $k = 1$	Load Balancing Metric		
	Opt-Trickle	Trickle	%
$Imin = 24ms$	9.34	12.36	24.43
$Imin = 2000ms$	2.13	2.30	7.39

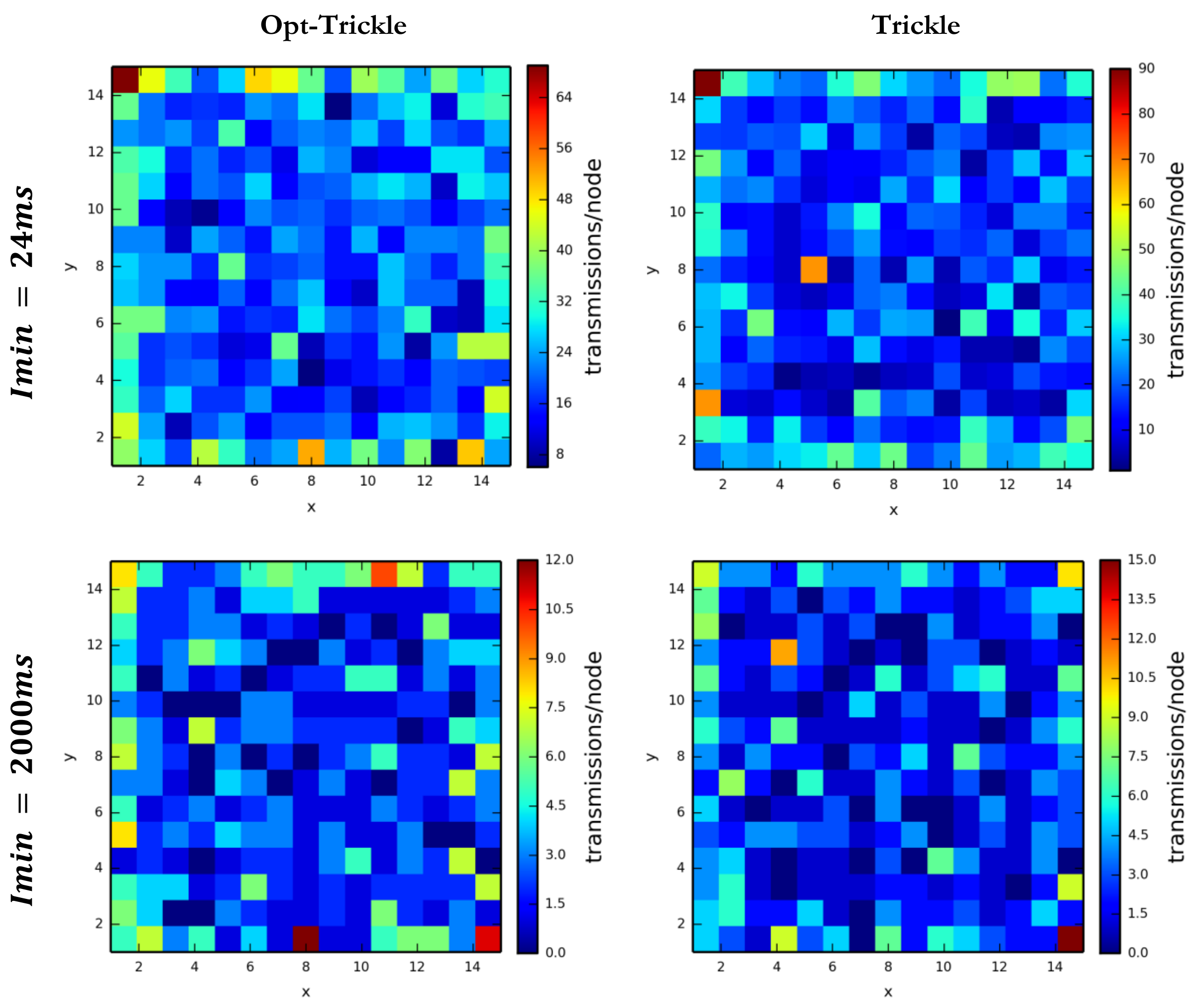


Figure 5-10 Transmission topography

As can be seen from Figure 5-10, both Trickle and Opt-Trickle generally show balanced transmission loads in the centre of the network, which is presented by a similar number of transmissions per node. However, border nodes tend to send more messages in both algorithms. This border effect can be explained by the fact that border nodes have fewer neighbours, and hence they receive fewer messages and make fewer suppressions. This is confirmed from the reception topography presented in Figure 5-11.

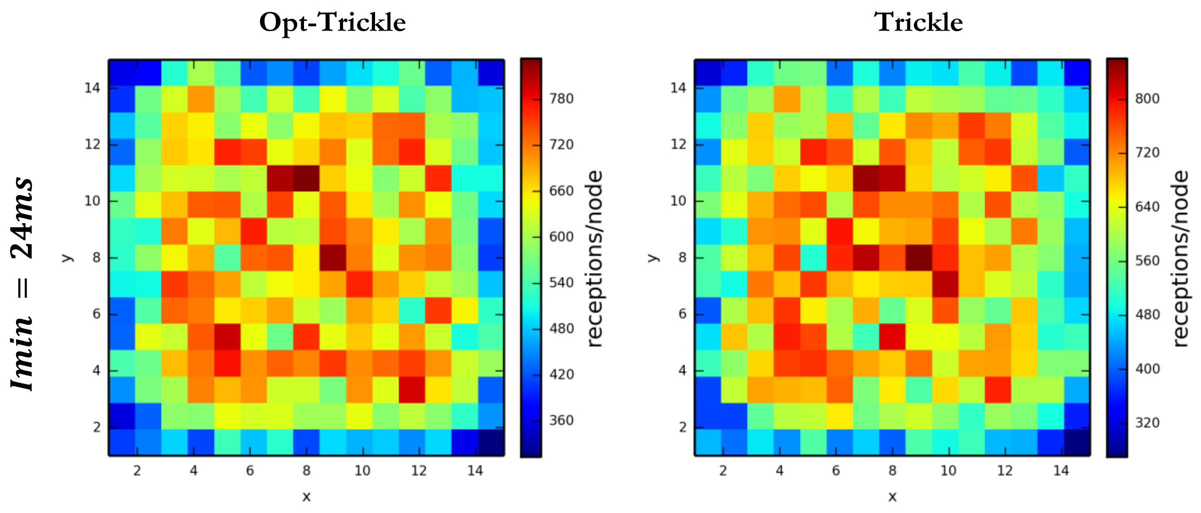


Figure 5-11 Reception topography

5.5.2 Propagation patterns

Having analysed the scalability and benefits of Opt-Trickle, it is interesting also to analyse its other behaviours, especially the propagation patterns and hop count.

Figure 5-12 depicts the behaviours of Trickle and Opt-Trickle in the $Imin$ interval for the sake of discussing their propagation patterns. By imposing a half-interval listen-only period at the start of an $Imin$ interval, Trickle inherits a wavelike propagation pattern. Such a pattern prevents the next update wave to start before the end of the current one. Thus, when a node $S0$ transmits an update (first wave), its neighbours shrink their intervals to $Imin$ and wait for $Imin/2$ before contending to transmit. If one of these neighbours, for instance, $S1$, transmits the update (second wave), its non-updated neighbours shrink their intervals to $Imin$ and wait for half the interval before deciding to transmit. Waiting for this $Imin/2$ time gives $S0$'s neighbours all the necessary time to transmit before the start of the next wave.

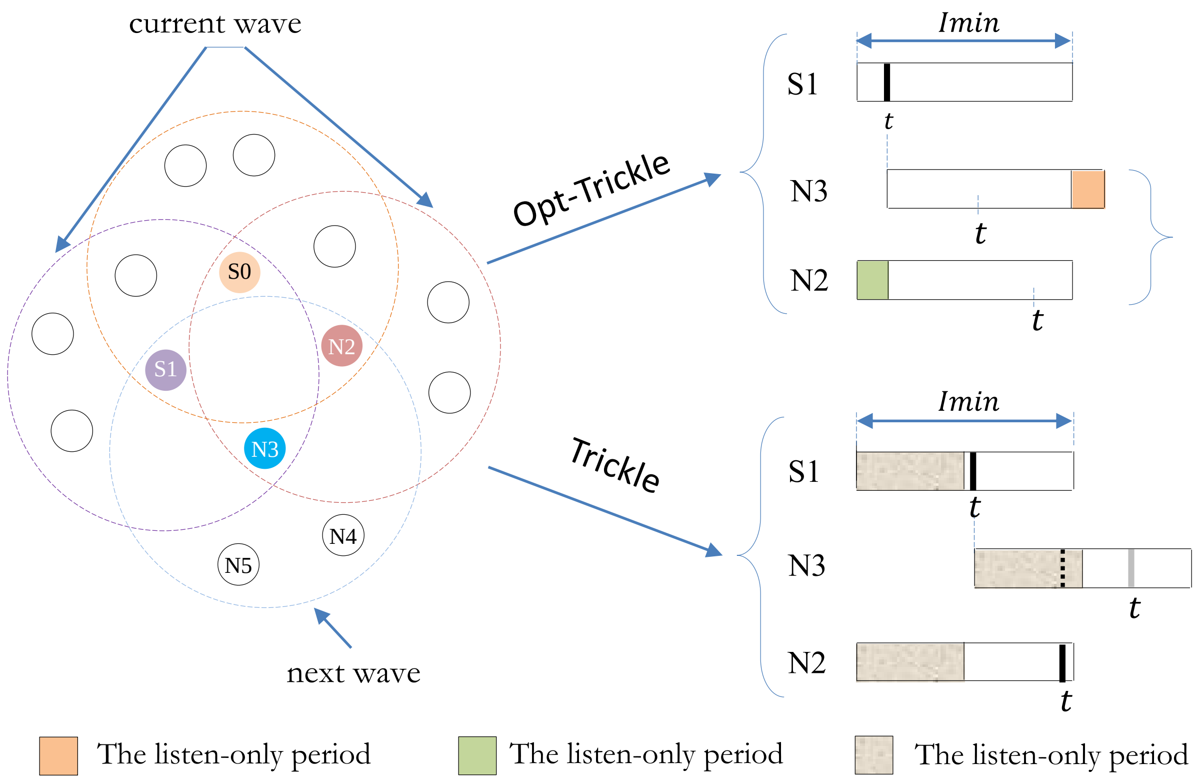


Figure 5-12 Wavelike propagation

This wavelike propagation depicted in Figure 5-13 is obtained from running Trickle and Opt-Trickle on a 400-node network deployed in a 20x20 grid in Contiki. The seed originating the update is located at the upper left corner. Configuration and link details of such a deployment are described in section 5.7.1 below. Besides the wavelike propagation pattern, the half *Imin* interval listen-only period could ensure important features. Hence, it implicitly imposes that the next wave cannot start before the end of the current one which minimises the number of contenders in an *Imin* interval to only the nodes of the current wave. This in turns reduces contentions and chances of collisions and hidden terminals and adds to the efficiency of the suppression mechanism especially when opting for very small *Imin* values. Note that while this wavelike propagation might be only observable in prefect lossless networks, the causes behind it stay the same and this discussion also applies to lossy networks.

By taking out the half-interval listen-only period from the *Imin* interval, Opt-Trickle does not suffer from the short-listen problem, but it might not provide the wavelike propagation ensured by Trickle thanks to the same half-interval listen-only period. For instance, the next wave in Figure 5-12 (e.g., N3's transmission) can start before the end of the current wave (e.g., before N2's transmission). Hence, Opt-Trickle allows for a mix between previous, current and following waves which results in random propagation patterns as shown in Figure 5-13. Thus, as can be seen from the heat maps of Figure 5-13, Trickle can ensure the wavelike propagation independently from the value of $Imin$ while Opt-Trickle might have different propagation patterns depending on the value of $Imin$. For instance, it might achieve a wavelike propagation for small values of $Imin$ under certain conditions, while realising generally free random propagation patterns. This random behaviour frees Trickle from the wavelike propagation pattern and hence allows Opt-Trickle to achieve very fast propagations. Opt-Trickle's propagation pattern may also minimise the effect of the dynamic behaviour discussed in [76].

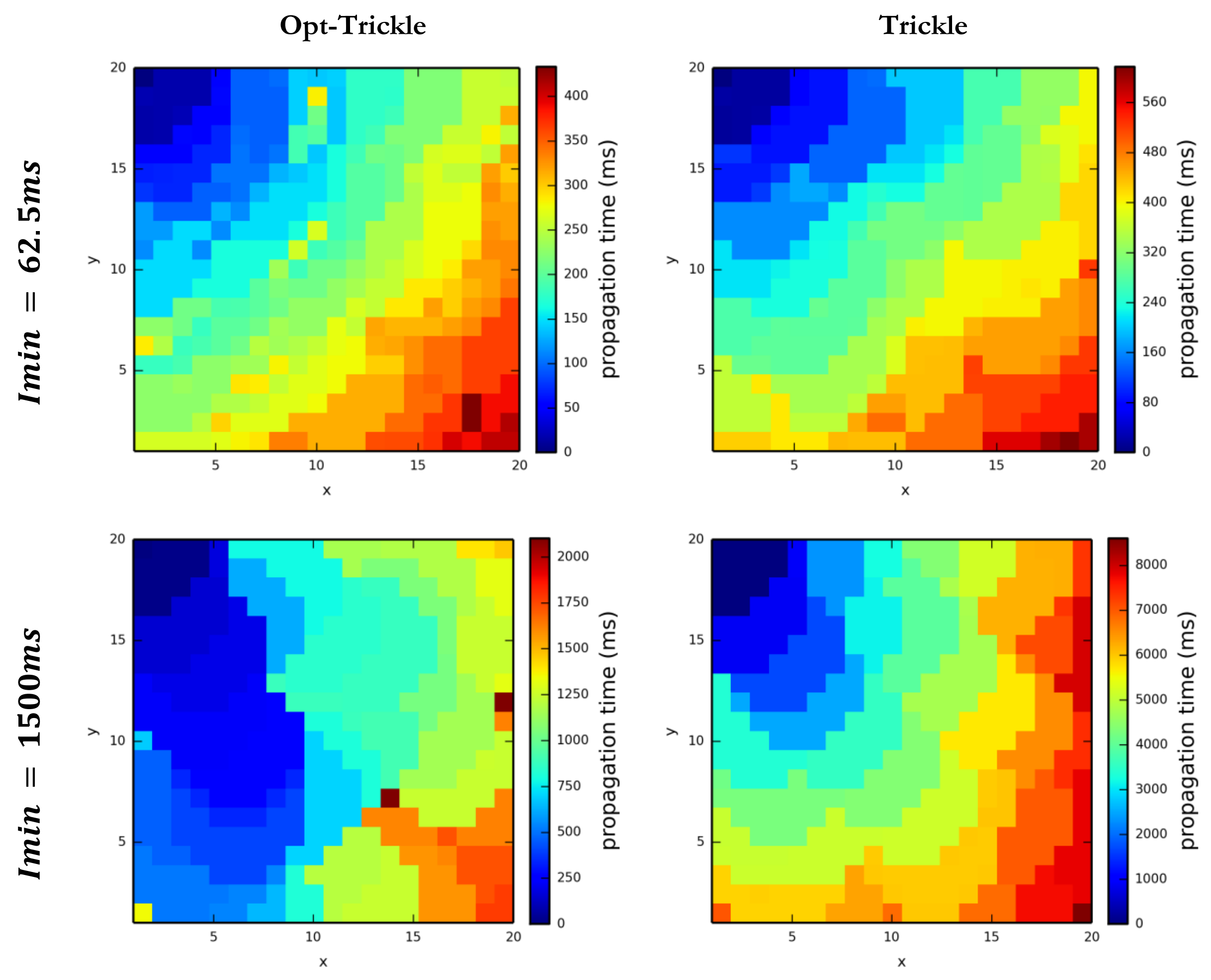


Figure 5-13 Propagation patterns of Trickle and Opt-Trickle

The different propagation patterns of Trickle and Opt-Trickle can have other implications on the hop count, the number of inconsistent transmissions and other Trickle behaviours. Hence, by allowing a free random propagation, Opt-Trickle might help the suppression mechanism and can generate fewer messages in the $Imin$ interval as will be observed in section 5.7.2. It was also observed that Opt-Trickle takes longer paths than Trickle to achieve consistency. However, it is complex to exactly predict the impact and relationships between the observed propagation patterns and Trickle behaviours (latency, number of transmissions, hop count, etc.) without a detailed thorough analysis. Such an analysis is left for future work.

5.5.3 Augmented Trickle

Having seen the benefits of Opt-Trickle and its propagation patterns, this section proposes to augment the Trickle's uniform selection of transmission time in the $Imin$ interval using context information in order to expand its applicability further. As demonstrated in section 5.2.3.3, Trickle might explicitly prevent some nodes from propagating new information. Therefore, Augmented Trickle works by default over Opt-Trickle. In this section, Opt-Trickle executions are categorised into three modes:

- **Proactive propagation of updates:** A node receiving an update shrinks its interval to $Imin$ in order to propagate it.
- **Reactive response to out-dated information:** A node receiving out-dated data, shrink it is interval to $Imin$ in order to bring the sender up-to-date.
- **Active maintenance of network consistency:** Nodes keep gossiping about the information in order to detect anomalies.

Augmented Trickle only biases Opt-Trickle's transmission time selection in the proactive and reactive modes (i.e. in the $Imin$ interval) as shown in Figure 5-14. The unbiased random time selection is preserved in the active mode in order to achieve balanced loads. The proposed augmentation is given in equation 5-5.

$$t = U(0, (1 - \alpha) \times Imin) \qquad 5\text{-}5$$

The bias parameter α depends on the available local information and the execution mode. Thus, α can be essentially distinguished into two parameters: α_p for the proactive mode and α_r for the reactive mode. The definitions of such parameters depend on the available local information. For instance, if the RSSI and LQI are used as metric, a distance-based heuristic would be to give the farthest nodes a priority to transmit the update (proactive mode) first in order to increase the chances of reaching newer nodes [113], [150], [153] and hence minimise the number of transmissions and decrease propagation latency. On the other hand, in the reactive mode closer nodes to the out-dated sender would be preferred to send first as to suppress the maximum number of redundant transmissions and minimise hidden terminals. Moreover, this helps balancing loads with the proactive mode.

Other heuristics can be envisaged depending on the locally available information such as node's power. For instance, a mains-powered node would be preferred to transmit than a battery powered one. The neighbour cache of the 6LoWPAN-ND (section 2.4.3) can also be exploited for a similar purpose. In addition, exploiting the neighbour cache to get estimates of locale densities can even allow for an adaptive value of k per node. These augmentations along with the fourth parameter allow Trickle to encompass a wide range of algorithms including flooding and all the flooding substitution techniques described in section 5.1. Finally, it should be noted that in the particular case of $\alpha_p = \alpha_r = 0$ for all nodes, Augmented Trickle falls back to Opt-Trickle.

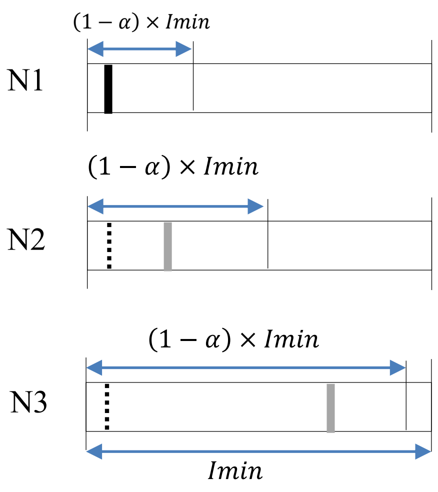


Figure 5-14 Augmented Imin interval of Trickle

5.6 TrickleSD: Trickle-based service discovery for LLNs

Having optimised Trickle, this section incorporates such optimisations to build reliable, time-efficient and cost-effective service discovery, and proposes TrickleSD. TrickleSD combines the mechanisms developed above with those incorporated in EADP. Indeed, TrickleSD mainly replaces EADP's UA with a new algorithm, described below, and responds to the shortcomings of EADP's Trickle-based SA and SM algorithms discussed in the previous chapter. Note that by building on Trickle's reliability feature, TrickleSD addresses the unreliability of the underlying UDP protocol in a lightweight effective way.

5.6.1 The user agent algorithm

For the pull mode and since Trickle is used to substitute EADP's flooding, it is mandatory to deploy the fourth parameter of Trickle in order to stop its execution after a sufficient number of intervals (required for reliability). Indeed, without such a parameter, Trickle would have been impractical for use in forwarding client requests. Thanks to the $query\_seq$ field included in each request message (section 4.2); Trickle operations in the pull mode are possible without modification to the generic request message format. To this end, the TrickleSD UA defines the additional configuration parameters depicted in Table 5-2. Note that since Trickle is now deployed in both pull and push modes of TrickleSD, hereafter, the Trickle parameters of the push mode (section 4.4) are referred to as PUSH_IMIN, PUSH_IMAX and PUSH_K.

Table 5-2 TrickleSD UA's configuration parameters

Configuration parameter	Meaning
PULL_IMIN	Pull mode Trickle's minimum interval size.
PULL_IMAX	Pull mode Trickle's maximum interval size.
PULL_K	The redundancy constant deployed in the pull mode to stop redundant request forwarding.
PULL_EXPIRATIONS	The number of timer expirations that allows killing the timer managing a particular request.

The use of Trickle in the UA follows a parallel approach where each request is managed by a separate Trickle timer. Thus, when the UA initiates a new service request, a Trickle timer is created to manage its forwarding. A node receiving such a request for the first time inserts it in its request cache and then asks the matchmaker to match it with the node's local directory entries. If a service matches, a reply is generated by the RA. Otherwise, the UA investigates the distance travelled by the entry and compares it with REQUEST_DISK. Depending on the results, it decides whether to abort or forward the request. If a forward decision is made, the UA creates a new Trickle timer to manage its forwarding. The Trickle timer follows Opt-Trickle operations. A Trickle timer is maintained for a number of PULL_EXPIRATIONS, used for reliability reasons, after which it gets destroyed to save memory resources. This UA algorithm is summarised in Figure 5-15.

User Agent Pseudo Code

```
evaluate_ incoming_request (request)
IF new_request THEN
    Insert request in the request cache
    Call the matchmaker component
    IF request_match THEN
        call reply agent
    ELSE
        IF distance < REQUEST_DISK THEN
            Create new Trickle timer
            Trickle_inconsistency (request)
        ELSE
            Abort the request
        ENDIF
    ENDIF
ELSE
    Trickle_consistency (request)
ENDIF
```

Figure 5-15 The pull mode algorithm

5.6.2 TrickleSD

TrickleSD mainly substitutes the EADP's UA algorithm (section 4.3) with the one proposed in the previous section. Since TrickleSD's UA, SA, and SM algorithms deploy Trickle as an underlying mechanism, any combination of Trickle, Opt-Trickle and/or Augmented Trickle can be used with TrickleSD. It is outside the scope of this chapter to evaluate them all. For the sake of answering the questions addressed in this thesis in a generic way, this chapter opts for using Opt-Trickle for the UA, SA and SM. It should be recalled that changing the Trickle version underlying TrickleSD does not require any changes to its other components, and it is always compatible with the TrickleSD protocol presented in this chapter. Finally, it should be remembered that as in EADP, the push mode can be disabled. Thanks to the reliable, time-efficient, Trickle-based pull mode algorithm, TrickleSD can be used to fulfil reliably the requirements of broadcast services (section 3.2.2).

Finally, it should be noted that the SA (section 4.4) and the SM (section 4.5.1) can be made generic by using the fourth Trickle parameter. Indeed, deploying Opt-Trickle with the fourth parameter allows the use of a separate Trickle timer to manage delete-message forwarding.

5.6.3 Managing the request cache

In order for the TrickleSD's UA to operate, a request cache table is maintained by every node. The current implementation opts for a simple structure containing all request entries. However, more efficient structures such as the ones used by MPL are envisaged in order to save memory. Independently of the adopted structure, TrickleSD provides cache management techniques, when low on memory, aiming to get better performance even in congested networks. Thus, once a Trickle timer managing a request reaches its PULL_EXPIRATIONS, the timer is destructed to save memory but the necessary information about the request are kept for a little longer in order to avoid loops. When receiving a new request, a node low in memory, purges entries with destructed timers starting with the ones having oldest sequence numbers. Likewise, the garbage collector can periodically delete such entries.

Having introduced and discussed Opt-Trickle and TrickleSD, the remainder of this chapter is devoted to evaluating such contributions. It starts by evaluating Opt-Trickle and then moves on to TrickleSD.

5.7 Evaluation of Opt-Trickle

Since Opt-Trickle changes the well-known Trickle assumptions, it makes sense to provide a separate, thorough and generic evaluation of its behaviour to demonstrate that it does not break Trickle even in severe network cases.

5.7.1 Evaluation methodology

Realistic simulations and public testbed experiments were conducted in order to evaluate the performance of Opt-Trickle. To put results into context, Opt-Trickle is compared with Trickle and Short-Trickle; a version of Trickle without the listen-only period. An abstract Trickle application is developed in Contiki where a seed node periodically injects new packets (identified by new sequence numbers) in the network. In such an abstract application:

- Receiving a packet with the same sequence number implies a consistency.
- Receiving a new packet (greater sequence number than receiver's version) implies an inconsistency for which the receiver updates its data and contend to propagate the update (proactive mode).
- Receiving an old packet (smaller sequence number than receiver's version) implies also an inconsistency. In this case, the receiver shrinks its interval to $Imin$ and contends to transmit its data in order to bring neighbours up to date (reactive mode).

At first, only one update is generated (Setup 1) in order to get a clear understanding of Opt-Trickle. Then periodic updates were injected (Setup 2) for the sake of deliberately creating heavy inconsistent traffic to show the impact of the transmit-listen interplay (section 5.4.5). This application is developed over UDP using uIPv6, at the network layer, and the IEEE 802.15.4 CSMA/CA algorithm, at the MAC layer. At the RDC layer, a non-duty-cycled network using the NullRDC protocol (NullRDC just keeps the radio always on) was in operation. This allows focusing on Opt-Trickle's performance rather than RDC effects, which are discussed in the following chapter.

In all experiments, the focus is on the following three main performance metrics: the number of transmissions (per interval), the consistency time (time from issuing an update until all the nodes get updated) and the number of inconsistent transmissions. The varied parameters are the minimum interval size, network density, redundancy constant, physical success rate, transmission power, and network topology. The configurations used in the experimentations are depicted in Table 5-3.

Table 5-3 Main evaluation parameters of Opt-Trickle

Parameter	Value
Duration of one simulation / #iterations / #nodes	600s / 25 / 400
Medium / Transmission range / Throughput	UDGM / 50m (single-hop: 500m) / 250kbps
Network area (x, y)	300m x 300m
Message payload	20 Bytes
MAC retransmissions	0 (Trickle takes care of retransmissions)
MAC initial backoff	0 (Trickle takes care of randomisation)
RDC / MAC / Adaptation	NullRDC / CSMA / 6LoWPAN

5.7.2 Results and discussions

The main simulation results discussed in this section are from evaluating setup 1 in a dense reference scenario containing 400 nodes deployed in a 20x20 grid, which gives a network density of around 36 neighbour/node. The observed network diameter was about 13 hops. The discussion starts by analysing results from multi-hop networks, and then moves to analyse the results obtained from the Indriya testbed [15]. Finally, the small additional cost is quantified. Unless otherwise stated, the default value of $Imin$ is one second and that of k is one. Each simulation runs for 10 virtual minutes and is repeated 25 times. The following graphs report the mean value of the 25 runs.

5.7.2.1 Multi-hop networks

This section discusses Opt-Trickle performance, in multi-hop networks, when varying network density, k, $Imin$ and loss rate. To vary the loss rate, the reception probability of a packet, which is proportional to the square of the distance between a sender-receiver pair, was varied in Cooja. Obtained results when varying the network density are depicted in Figure 5-17, and the remaining results are depicted in Figure 5-18.

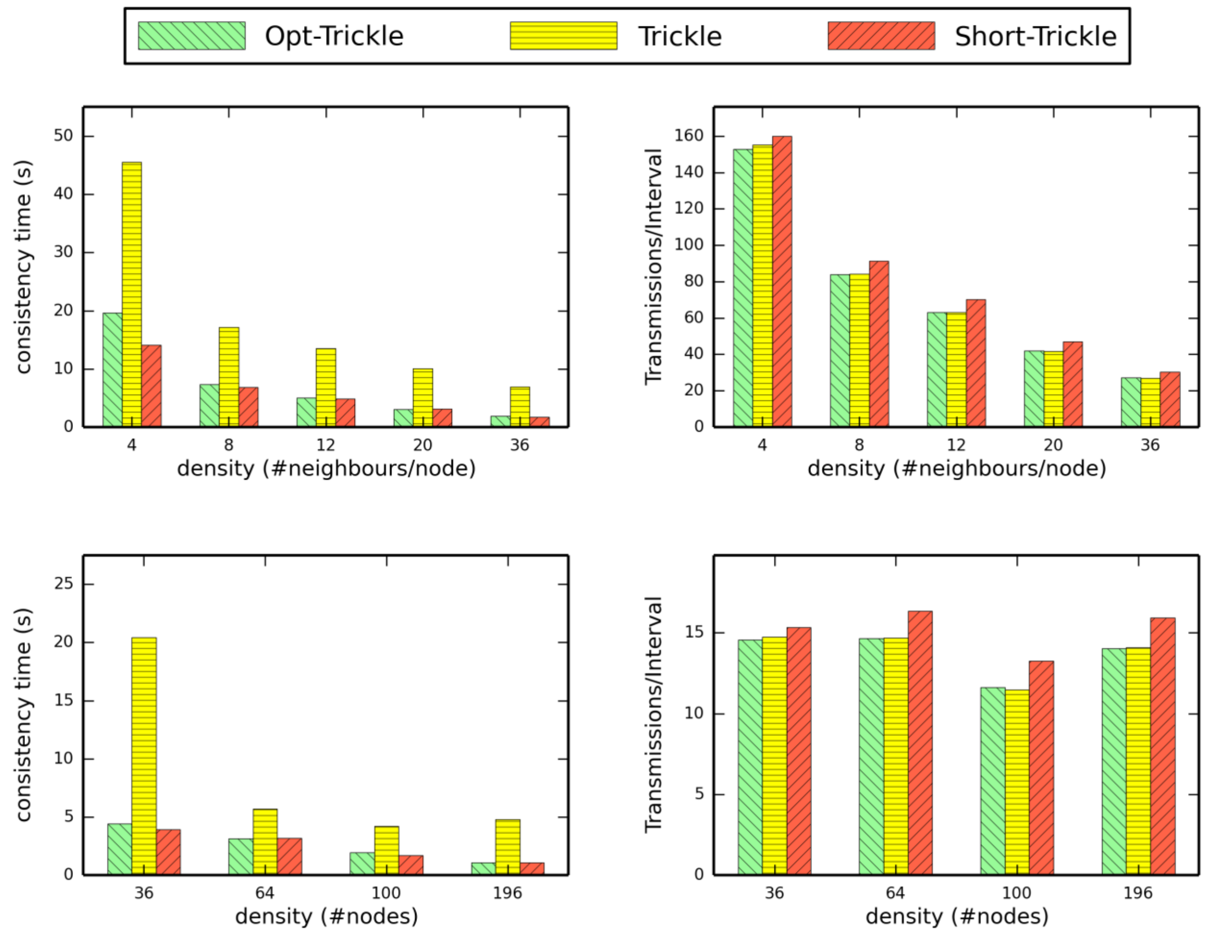


Figure 5-16 Impact of density on Opt-Trickle in multi-hop networks

Figure 5-16(a) shows the consistency time and transmissions/interval registered by Opt-Trickle, Trickle and Short-Trickle under different network densities. As can be seen from this figure, Opt-Trickle propagated more than two times faster than Trickle when varying network density. The biggest difference was observed in a sparse network of 4 neighbours/node density. Such a performance is achieved at approximately the same cost as Trickle, and is even lower in sparse networks. This can be explained by the fact that Trickle prevents some nodes from transmitting in the *Imin* interval, which might require more transmissions in order to achieve consistency in sparse networks. Compared to Short-Trickle, Opt-Trickle achieved similar consistency times, with the gap decreasing with increased network density. This is so, since in sparse networks the first propagation wave might get stopped by the suppression mechanism, before it starts again in the second interval, where Opt-Trickle deploys the listen-only period. Finally, Short-trickle generated the biggest cost since it suffers from the short-listen problem. These results are confirmed when varying the number of nodes as shown in the second row of graphs.

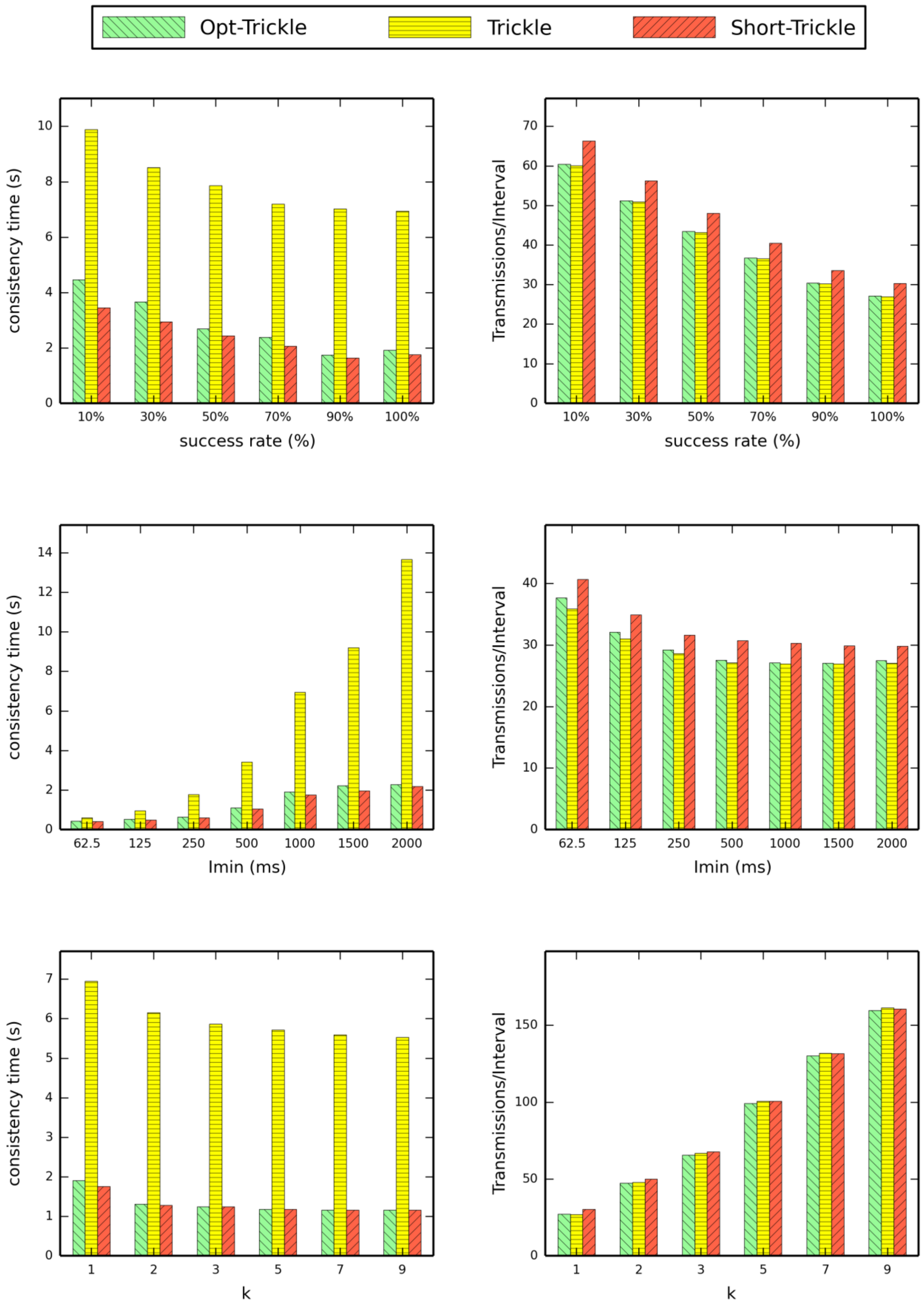


Figure 5-17 Opt-Trickle performance in multi-hop networks

The impact of losses is illustrated in the first row of graphs in Figure 5-17. As can be seen from this figure, Opt-Trickle approached the propagation time of Short-Trickle even in a worst case of 90% configured loss rate. The gap between the two decreased with increasing success rates. This is explained by the fact that in lossy networks there are more chances of losing an update for the first time which postpones its delivery to following intervals. In such intervals, Opt-Trickle deploys the listen-only period, while Short-Trickle does not, allowing it to propagate faster. In all cases, the consistency time of Short-Trickle and Opt-Trickle was about four times lower than that of Trickle. While Short-Trickle achieved such a performance by generating more packets, Opt-Trickle generated approximately the same cost as Trickle. On closer examination, Opt-Trickle generated slightly more packets than Trickle, which are more visible in lossy networks. This is explained by the transmit-listen interplay benefiting Trickle, which is quantified in details in section 5.7.2.3.

The second row of graphs in Figure 5-17 depicts the performance of the evaluated protocols when varying $Imin$ in a physically lossless network. As expected, Opt-Trickle achieved network consistency as quickly as Short-Trickle. Interestingly, the propagation time of Opt-Trickle does not heavily depend on $Imin$ such is the case for Trickle, thereby allowing Opt-Trickle to propagate new updates seven times faster in an $Imin$ of two seconds. This gap is expected to increase with increased $Imin$ values. While Short-Trickle achieved such a propagation speed generating more messages, the cost of Opt-Trickle is similar to that of Trickle. It should be noted, however, that a small difference in the cost of the two protocols can be observed, and it is more visible for smaller $Imin$ values. This can be due to two main reasons; losses and unbalanced load distribution, both benefiting Trickle in dense networks. Thus, even though the network is physically lossless, losses can always occur because of collisions and hidden terminals, which are more likely to occur in small contending periods, i.e. smaller $Imin$ intervals. On the other hand, since Trickle explicitly prevents some nodes from transmitting, it minimises the number of contenders in an $Imin$ interval, which in turn minimises losses.

Finally, the third row of graphs in Figure 5-17 presents the performance of the evaluated algorithms when varying k. As can be seen from this figure, increasing k decreased slightly the consistency time while increased considerably the cost of the three protocols.

Concerning individual protocol performance, these graphs show that Opt-Trickle propagated about 3.5 times faster than Trickle while also generating fewer messages when increasing k. This can be explained by the fact that in physically lossless dense networks, Opt-Trickle suffers less from the transmit-listen interplay, while at the same time benefiting from its implicit synchronisation in an $Imin$ interval.

5.7.2.2 Empirical study

Opt-Trickle was also evaluated in the public large-scale Indriya testbed (section 2.6.2.3) [156]. At the time of experimentation almost all of middle floor nodes were off, leaving just 65 motes and a good opportunity to test in an irregular, faulty real-world scenario. Using Setup 2, the seed (node 21 in the third floor, Figure 2-24) injected a new packet every 60 seconds. This is so to create a network dominated by inconsistent traffic, in order to show the impact of the observed small additional cost. Each experiment was run for 30 minutes and was repeated three times. The default value of $Imin$ was half a second and that of k was one. As in the simulations, the graphs report the mean. Figure 5-18 presents the consistency time and the cost expressed by the number of transmissions/interval registered by Opt-Trickle, Trickle and Short-Trickle when varying $Imin$, k and the transmission power.

The first row of graphs in Figure 5-18 depicts Opt-Trickle's performance when varying $Imin$. As can be seen from these graphs, Opt-Trickle achieved network consistency faster than Trickle, while approximately generating a similar number of transmissions. The small additional cost is due to the transmit-listen interplay discussed earlier. Note that even in this very irregular faulty network Opt-Trickle's propagation speed is less affected by the value of $Imin$. Concerning the number of generated inconsistent transmissions and similarly to previous results, Opt-Trickle generated fewer packets in the $Imin$ interval when the interval size was greater than 125ms. In an $Imin$ = 62.5ms, Trickle transmitted fewer packets, since it explicitly prevented some nodes form contending and hence minimised collisions and hidden terminals.

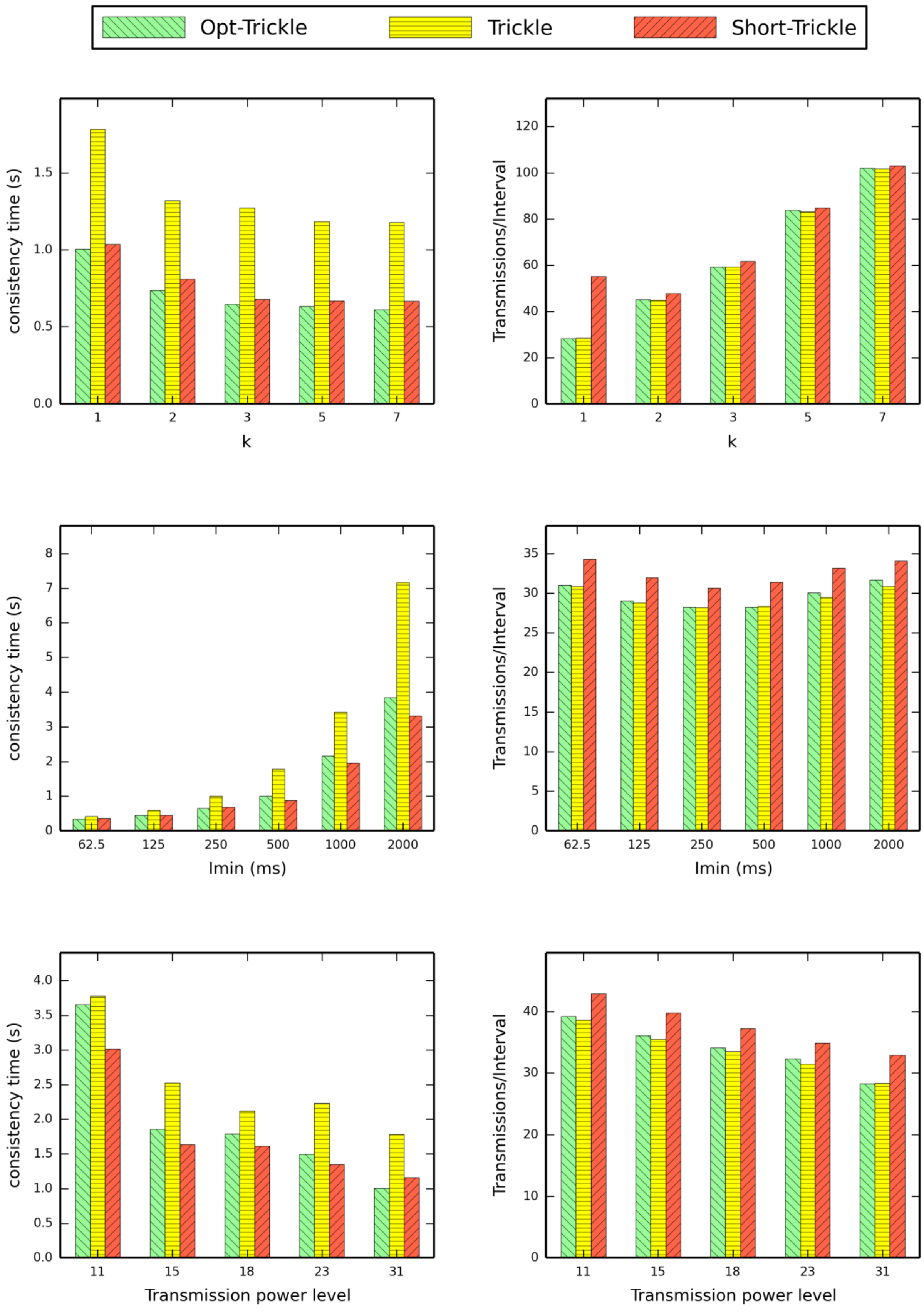


Figure 5-18 Opt-Trickle performance in the Indriya testbed

When varying k, as can be observed from the second row of graphs in Figure 5-18, Opt-Trickle provided the best of both Trickle and Short-Trickle, even in this faulty irregular network experiencing heavy inconsistent traffic. Thus, it propagated new updates as quickly as Short-Trickle, which is about two times faster than Trickle, at a similar transmission cost.

Finally, the third row of graphs depicted in Figure 5-18 shows the performance of the evaluated protocols when varying transmission power. Varying transmission power plays a double role; it changes both the density and the success rate of the network. As can be seen from these graphs, Opt-Trickle propagated about two times faster than Trickle, even in a quite lossy, less connected network (power level 15). As expected and for the reasons discussed earlier, Opt-Trickle's consistency time approached that of Short-Trickle, with a cost similar to that of Trickle.

5.7.2.3 Quantifying the additional cost

This subsection discusses the small extra cost observed in some of the above graphs. To this end, Figure 5-19, Figure 5-20 and Figure 5-21 depicts the number of transmissions generated by Trickle and Opt-Trickle in the second, third and remaining intervals in multi-hop, single-hop and the Indriya testbed respectively. The single-hop network is derived from the 400-node reference scenario by increasing the transmission range such that each node can reach directly every other node in the network. Overall, these figures show that the small additional cost does not violate the logarithmic scalability of Opt-Trickle and it largely disappears after a few intervals following $Imin$.

Figure 5-19 and Figure 5-20 show that Opt-Trickle's extra cost is density independent in both single-hop and multi-hop networks, and that it largely disappears as soon as the third interval. The main cause behind it is the transmit-listen interplay. As shown in section 5.4.5, such a cost is mainly caused by losses and the multi-hop nature of the network, and it might increase with increasing k. Indriya's results also confirm that this cost disappears as early as the third interval (Figure 5-21). In the second and third intervals, and although Trickle sends fewer packets because of the transmit-listen interplay, the difference between the costs of the two protocols is very small. This might be due to the moderate density of the network.

Concerning the number of generated messages in the *Imin* interval, these results show that Opt-Trickle does not incur more traffic in this interval. Conversely, it generally generated fewer inconsistent packets than Trickle. This can be due to the free propagation pattern (section 5.5.2) of Opt-Trickle which might have allowed it to suppress more inconsistent packets. This was also observed when varying k. However, when opting for small *Imin* values in dense networks, it was observed that Opt-Trickle generated more inconsistent messages than Trickle which contributed to the overall cost of Opt-Trickle depicted in the second row of graphs in Figure 5-17.

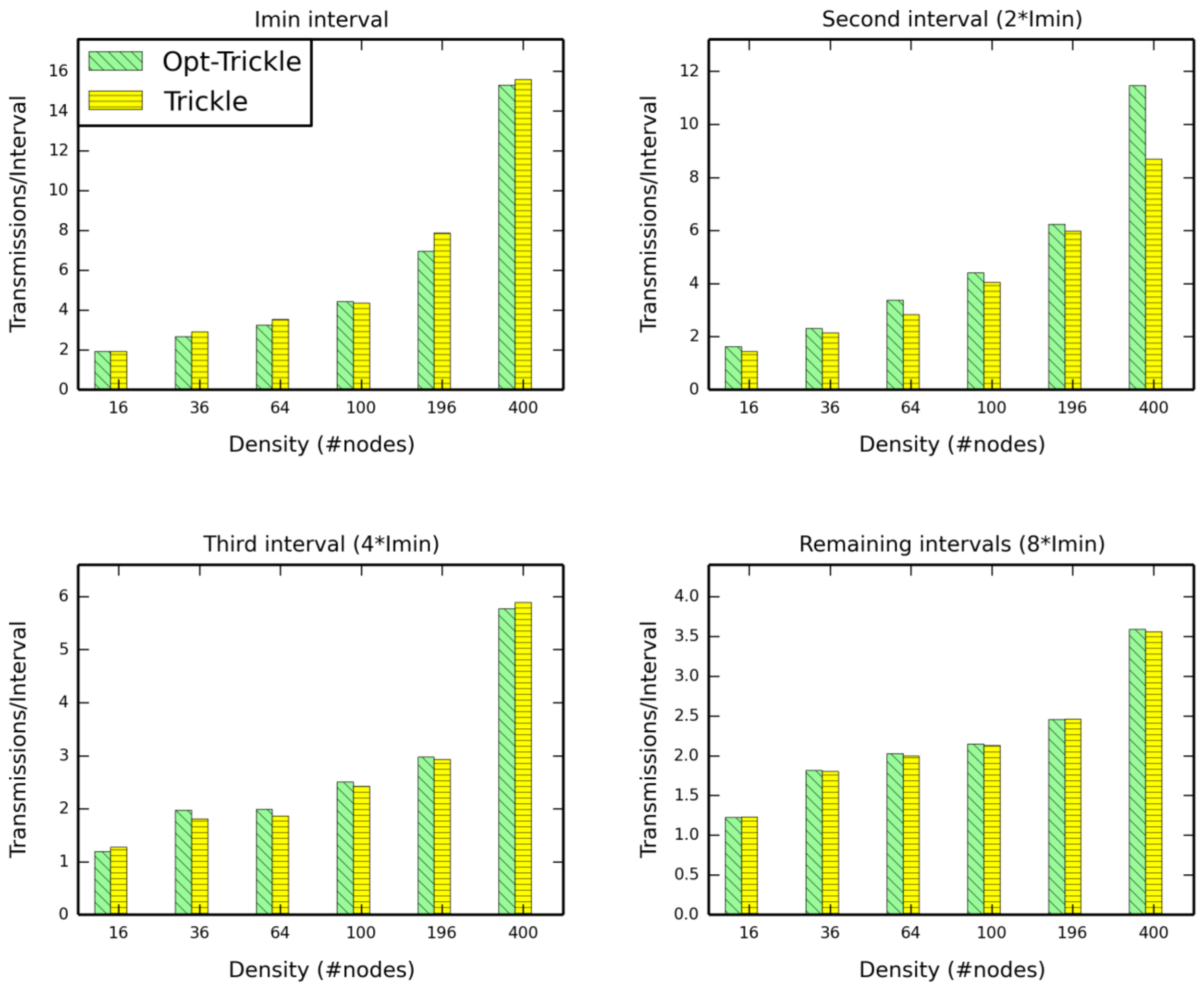


Figure 5-19 Quantifying the additional cost in single-hop networks

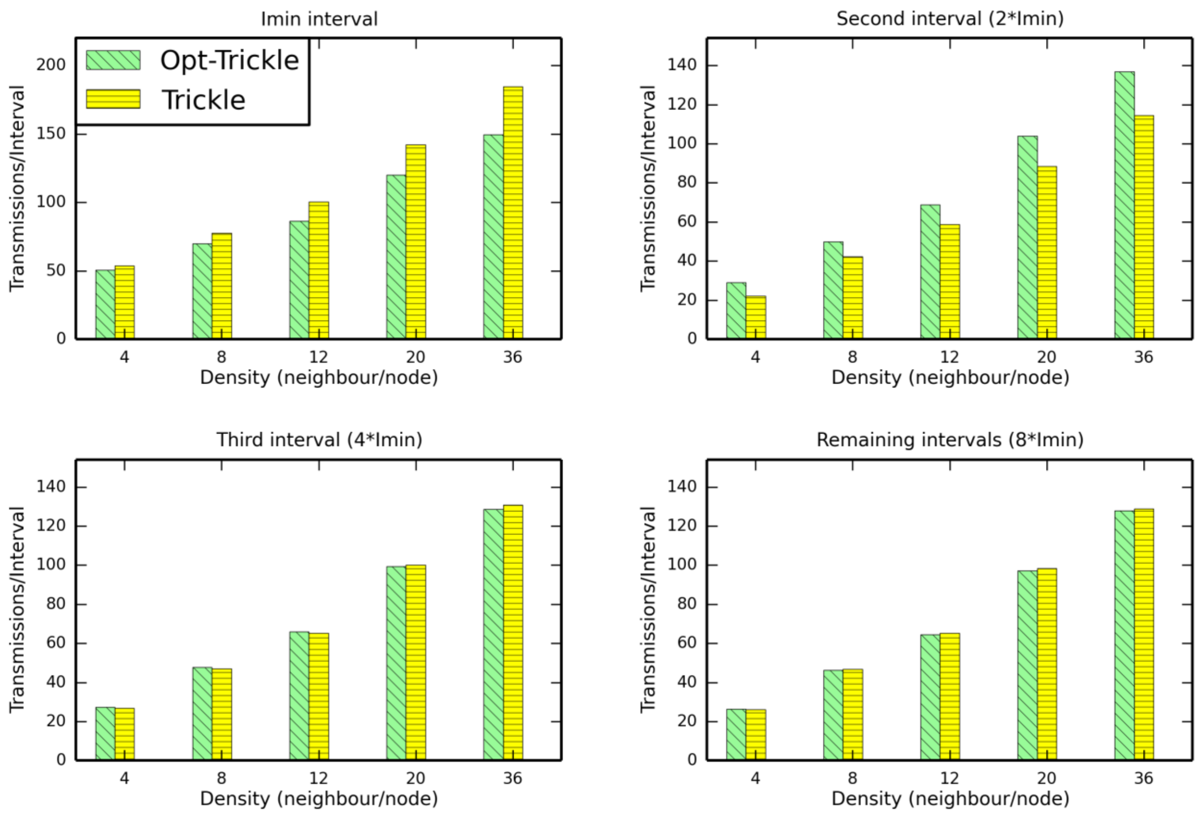


Figure 5-20 Quantifying the additional cost in multi-hop networks

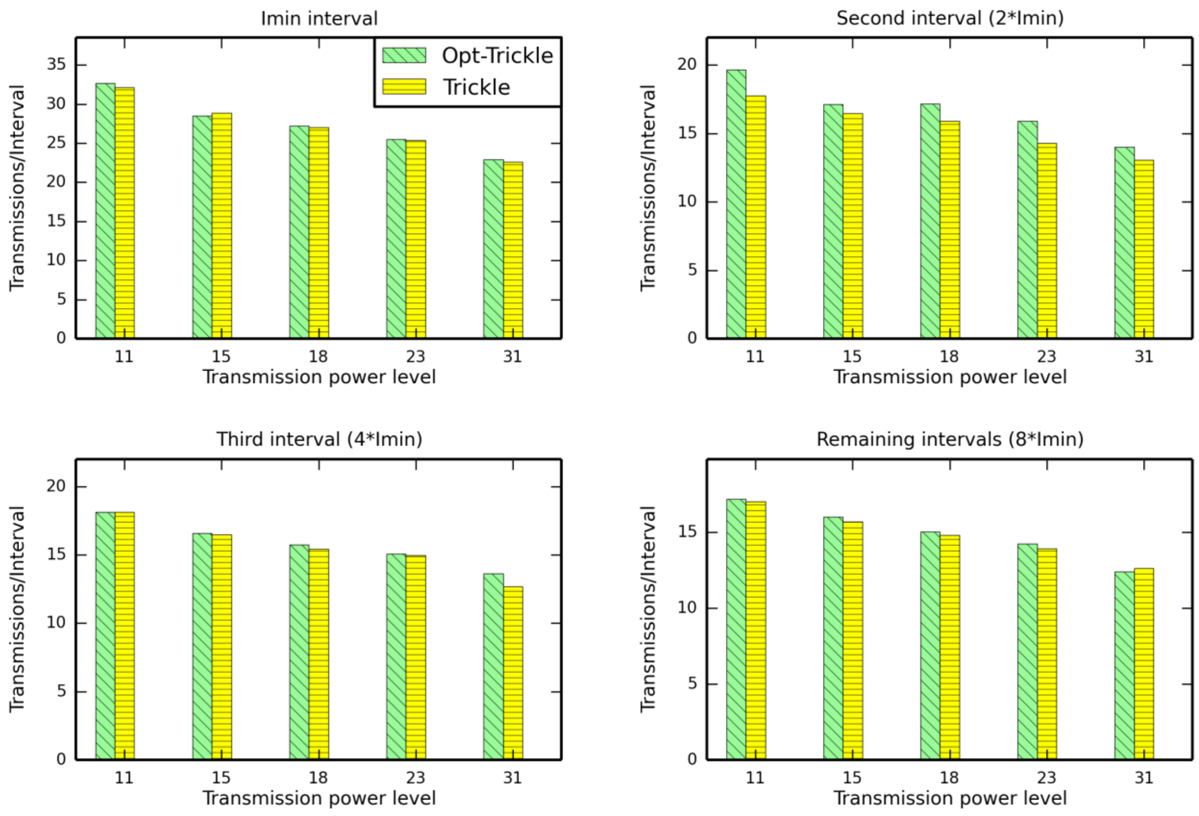


Figure 5-21 Quantifying the additional cost in the Indriya testbed

5.8 Evaluation of TrickleSD

This section evaluates the performance of TrickleSD. To put TrickleSD results into context, it was compared with the EADP protocol, which achieved far better performance than ADDER [127] used as a benchmark for EADP evaluations in the previous chapter (section 4.8). To see the impact of the push mode on both TrickleSD and EADP, it was disabled in the TrickleSD-d and EADP-d versions, respectively. The evaluated protocols' variants are summarised in Table 5-4.

Table 5-4 Evaluated protocols' variants (scenario #2)

Protocol variant	**Description**
EADP	The EADP protocol in default settings as described in Table 4-4.
EADP-d	The EADP protocol when disabling the push mode as in Table 4-4.
TrickleSD	The TrickleSD protocol having an adaptable period between $[Imin, Imax]$, enabling both push and pull modes and using RPL as the underlying routing protocol.
TrickleSD-d	TrickleSD as in the above configuration having the push mode disabled.

5.8.1 Evaluation methodology

In the previous chapter, EADP was evaluated in a large scale network scenario of 100 nodes in both static and mobile environments to see its performance in emergency response scenarios and similar applications. In this evaluation, a second scenario targeting home automation systems and similar IoT applications is considered to assess the performance of EADP and TrickleSD in such fast growing applications of LLNs. To this end, a reference network of 31 nodes (one border router and 30 motes) was randomly deployed in a square area of 300m×300m as shown in Figure 5-22. The configuration and link parameters of such network are directly extracted from the *'rpl-udp.csc'*[10] example available in Contiki. A client generated periodic requests every 5 seconds looking for a service provided in the network. Each node provides one service which is proactively

[10] https://github.com/contiki-os/contiki/blob/master/examples/ipv6/rpl-udp/rpl-udp.csc

advertised over the network for the ADVERTISEMENT_DISK. One service provider, which was at least 5 hops away from the client, had matching responses.

At the RDC layer and because of issues observed with the ContikiMAC protocol, which will be addressed in the following chapter, this section uses another RDC called X-MAC [53] with a channel check rate of 8 Hz, which gives a worst link latency of 125ms. A small random delay called jitter [157] is used by the EADP's pull mode flooding in order to avoid collisions that might arise from simultaneous retransmissions. To make fair comparisons, the maximum jitter value of the EADP's flooding algorithm is made equal to the PULL_IMIN value of TrickleSD, which is recommended to be at least 2-3 times the worst link latency [24]. Similarly to Chapter 4's evaluations, each experiment was repeated 10 times modifying for each the seed of the random number generator.

In addition to the performance metrics described in section 4.8.2, this section defines and measures the following new metrics:

- **Normalised pull traffic per node**: This metric measures the amount of traffic generated, on average, by an intermediate node in order to forward a single service request. In the case of a network-wide flooding, this parameter would be equal to 1 since, theoretically, every node would forward a request once. This metric measures the cost-effectiveness of the pull mode algorithm and contributes hugely to the scalability and cost-effectiveness of an SDP.
- **Average hit success rate:** This metric measures the capacity of an SDP to find available requested services. It is measured as the ratio between the number of requests to the number of unique hits at the provider. This distinction between hit and discovery success rates is made since achieving a good hit success rate is the responsibility of a discovery protocol while ensuring high delivery rate of service replies back to the client is the responsibility of the underlying routing protocol. Therefore, the hit success rate reflects the ability of a protocol to reliably find available services and hence contributes largely to the reliability of an SDP.

Furthermore, the network energy consumption is proxied by the percentage of time the radio was on, commonly known as the radio duty cycle.

- **Network radio duty cycle:** the percentage of time a node's radio transceiver was on averaged over all the nodes. Reporting power consumption as radio duty cycles has two main advantages. Firstly, it preserves the accuracy of the results since the transceiver's energy consumption has a linear relationship with its on-time [90], [147], and, secondly, it allows comparison of results across hardware platforms which may have different power consumption factors per component [147]. To measure the radio duty cycles, the PowerTrace tool distributed with Contiki was used [158]. Note that the radio duty cycle is the metric used to indicate energy consumption in the remaining of this document. The main parameters used in this evaluation are summarised in Table 5-5.

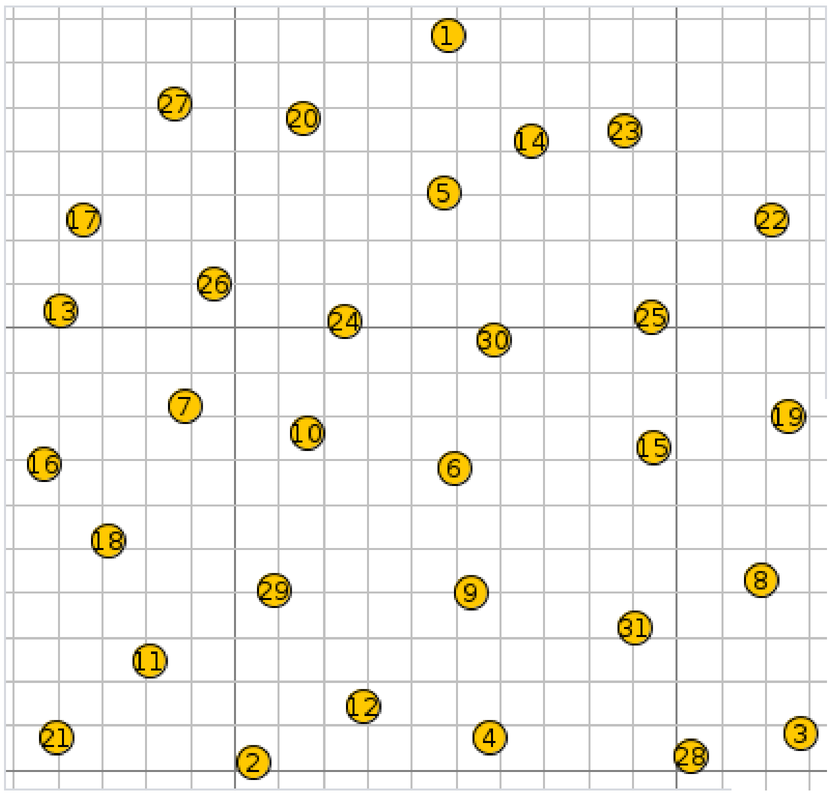


Figure 5-22 Reference scenario for evaluating TrickleSD (scenario #2)

Table 5-5 Experimental parameters (scenario #2)

Configuration parameter	**Value**
Duration of one simulation/ #iterations / #nodes	600s / 10 / 31
Medium / range / Throughput	UDGM / 50m / 250kbps
Network area (x, y)	300m x 300m
PULL_IMIN = max jitter / PULL_IMAX	500ms / 2^{10} × PULL_IMIN
PULL_EXPIRATIONS	1
PULL_K / PUSH_K	1 / 1
PUSH_IMIN/ PUSH_IMAX	40s / 160s
REQUEST_RETRANSMISSION_COUNTER	0
REQUEST_DISK / ADVERTISEMENT_DISK	6 / 4
Underlying routing protocol	RPL
RDC / MAC / Adaptation	X-MAC / CSMA-CA / 6LoWPAN

5.8.2 Results and discussions

Figure 5-23 depicts the discovery time, number and size of advertisements, pull mode generated traffic, the hit and discovery rates and the network radio duty cycle of both TrickleSD, TrickleSD-d, EADP and EADP-d, when varying the execution time (proxied by the number of requests). The results are the mean of 10 runs.

As can be seen from Figure 5-23 (a), the discovery time of TrickleSD and EADP services decreased over time as a result of the push mode. However, TrickleSD achieved the best discovery time thanks to exploiting Opt-Trickle's latency improvements to achieve faster advertisements. Thus, TrickleSD responses started coming faster as soon as network deployment. This is achieved with approximately the same advertisement cost of EADP as can be seen from Figure 5-23 (c). Thus, while TrickleSD's push mode sent slightly more advertisements than that of EADP, the average size of such advertisements was smaller than those generated by EADP (Figure 5-23 (d)). This is achieved thanks to Opt-Trickle characteristics discussed in section 5.7.2. The discovery time of TrickleSD-d and EADP-d remained generally constant over the course of time as a consequence of disabling the push mode. However, it could be observed that TrickleSD-d times were slightly smaller than those of EADP-d. This may be due to the fact that TrickleSD-d generates less traffic (Figure 5-23 (b)) allowing less congestion for the replies.

Concerning the traffic generated in the pull mode (Figure 5-23 (b)), TrickleSD in both its versions generated considerably less traffic than EADP. This is mainly realised by deploying Opt-Trickle as a substitute of flooding. Thus, even in this sparse network, TrickleSD still allows to cut the number of unproductive pull traffic by about half compared to that of EADP. Specifically, each node in TrickleSD-d generated about 40% less pull overhead than its EADP-d counterpart. In TrickleSD, a similar observation can be drawn. Interestingly, the normalised generated pull cost per request decreases with increasing times thanks to the push mode which allows closer nodes to answer the requests and hence stop their propagation.

The achievements of the push mode also allowed both TrickleSD and EADP to realise high hit and discovery rates as can be seen from Figure 5-23 (e) and (f). Thus, as depicted in Figure 5-23 (e), the hit success rate of TrickleSD approached 100% as soon as starting network operations, thanks to its fast advertisement propagations governed by Opt-Trickle. Slightly after this, EADP achieved the same hit patterns. Those hit achievements are also accompanied with higher discovery rates since the routing protocol does not have to route the responses over long distances. Hence, TrickleSD achieved good discovery rates right from the start while EADP achieved only about 80% discovery rates until up 20 requests while it achieved a 100% discovery rate after that, thanks to the services being advertised.

When disabling the push mode, discovery and hit rates dropped dramatically (Figure 5-23 (f)). Thus, while hit rates achieved above 94% for both TrickleSD-d and EADP-d, discovery rates dropped to below 80% for both. This latter can be explained by the fact that the routing protocol has to deliver responses over long distances which increases the loss rate. For the former, it might be surprising to see flooding slightly outperformed Opt-Trickle in terms of hit time with around 98% and 94% respectively. This is because the configuration of TrickleSD's pull mode in this evaluation (PULL_EXPIRATIONS = 1 and PULL_K = 1) was focusing on minimising the cost which rendered it too impassionate in suppressing request transmissions. Nevertheless, in dense networks or when opting for bigger values of PULL_K and/or PULL_EXPIRATIONS, TrickleSD-d can achieve better and reliable hits as has been discussed in Opt-Trickle evaluations.

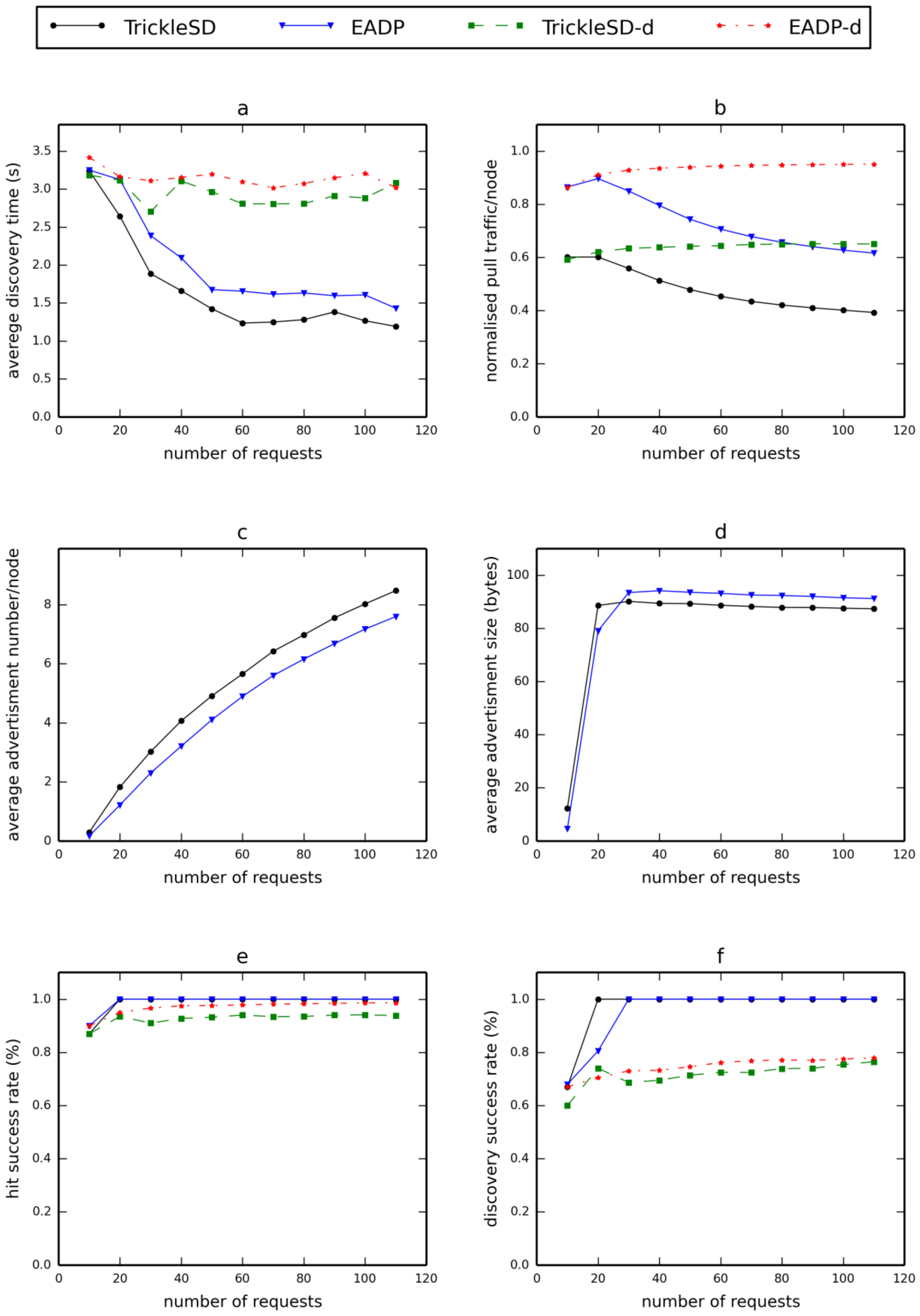


Figure 5-23 TrickleSD's time/cost performance

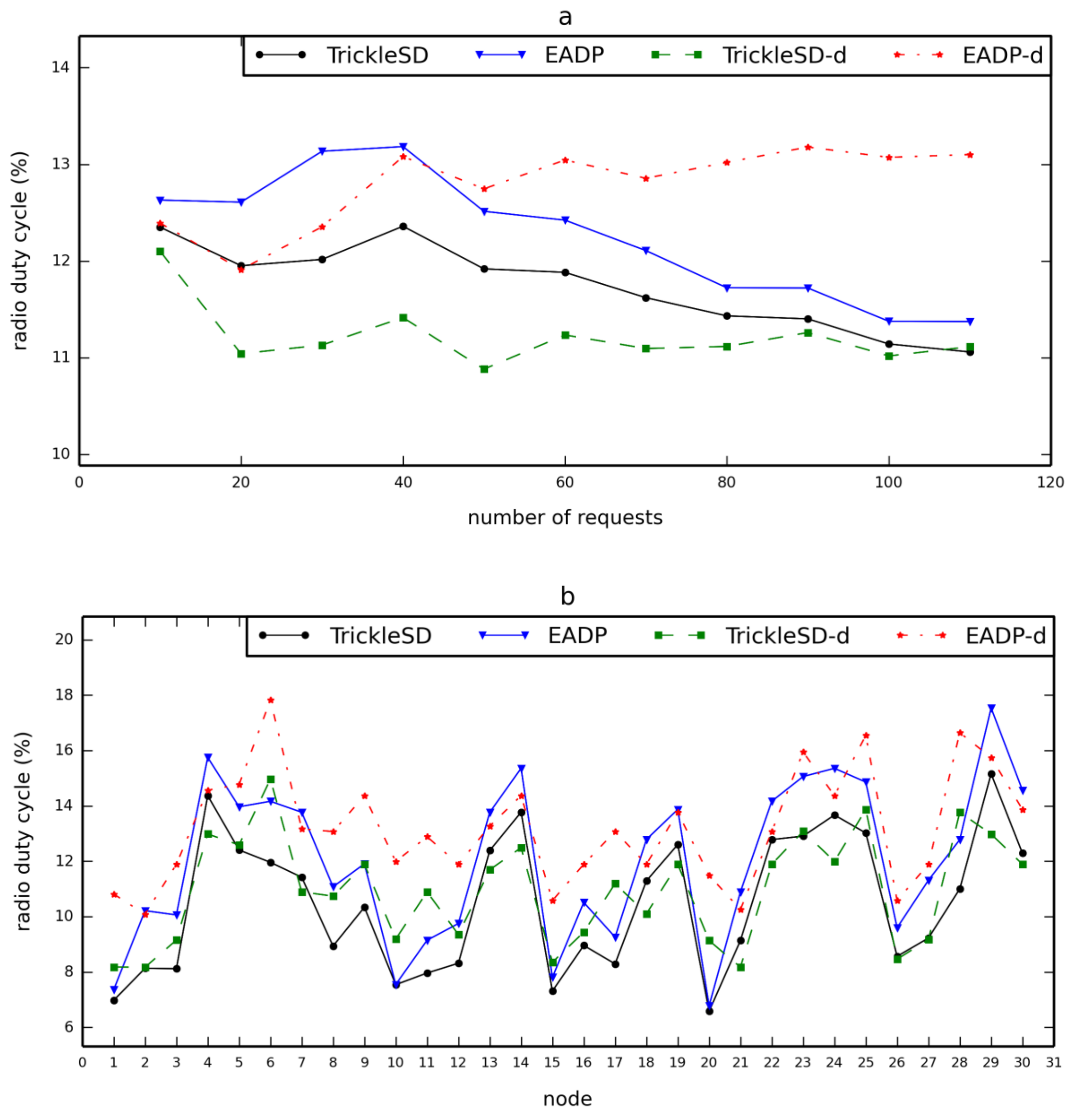


Figure 5-24 TrickleSD's energy distribution

Figure 5-24 presents the radio duty cycle of the evaluated protocols. Overall, Figure 5-24 (a) shows that the radio duty cycles of EADP and TrickleSD decreased over time, thanks to pull mode traffic reduction (Figure 5-23 (b)), while those of TrickleSD-d and EADP-d remained constant since such protocols kept generating the same amount of traffic. When it comes to comparing specific protocol performance, it is clear from this figure that TrickleSD outperformed EADP in both versions. Thus, TrickleSD-d consumed the smallest energy as a result of disabling the push mode and enabling the Trickle-based pull mode.

Finally, Figure 5-24 (b) shows the distribution of the energy consumption between nodes. The distribution depicts generally balanced loads for all protocols taking into account border effects and irregularities in the network.

5.9 Discussions

Having presented and evaluated Opt-Trickle and TrickleSD and because of the expected impact of Opt-Trickle, this section is set apart to situate it among the existing work trying to alter Trickle's behaviour. Such works can be divided into two categories: those attempting to tweak Trickle's behaviour in specific use-cases, such as in RPL and MPL, and those studying Trickle's behaviour in the generic case, similarly to Opt-Trickle.

RPL (section 2.5.1) relies on Trickle for controlling the frequency of DIO (DODAG Information Object) messages, which constitute the building block of the DODAG. Therefore, Trickle plays a significant role in the convergence time and stability of RPL networks. This motivated researchers to study Trickle in order to predict the performance of RPL networks. For instance, [159] tries to make Trickle fair to all RPL nodes. Thus, it proposes to bias the uniform choice of transmission times by giving the nodes that sent fewer packets in the past more chances to transmit in the future. The authors of [160] study the effect of non-synchronised Trickle intervals on RPL's generated control traffic and propose a readjustment of Trickle intervals in order to gradually re-establish synchronisation. In the context of RPL, Opt-Trickle can help achieving better convergence times.

MPL describes a way of using Trickle to realise reliable multicast routing in LLNs. To this end, MPL introduced the fourth Trickle parameter, which is also employed by TrickleSD. MPL uses Trickle to manage multiple data items in both MPL's proactive and reactive modes. Thus, it deploys, for the proactive mode, parallel Trickle approaches similar to the ones applied in [78], [142] and uses serial approaches [143] for the reactive mode. Opt-Trickle can provide better time efficiency to MPL.

Other works focusing on analytically modelling Trickle's behaviour in generic cases are reported in [161]–[163]. These works try to provide mathematical tools that can analytically predict the message count and the propagation time of Trickle. For instance, a detailed analytical study of Trickle's behaviour is reported in [163] where the message count and propagation time are analytically modelled as a function of a generalised listen-only period. In a recent work [164] published in parallel to our Opt-Trickle work [165], the authors moved the parametric listen-only period to the $Imin$ interval. Although, the theoretical modelling presented in [164] only treats lossless line-topology networks, the results confirm the experimental findings discussed in [165]. Generally speaking, while analytical models help to understand the dynamics of Trickle, they assume simplistic, lossless and regular network deployments. In addition, such models neither consider realistic radio propagation patterns nor model contentions and collisions, which is an oversimplification of LLN dynamics.

5.10 Summary

This chapter focused on two main parts namely the optimisation of the Trickle algorithm (Opt-trickle) and the proposition of the TrickleSD protocol. Opt-Trickle results showed noticeable performance enhancements regarding the time efficiency of Trickle while preserving its scalability. Building upon these achievements, TrickleSD showed important discovery performance improvements over EADP especially regarding scalability, reliability and time efficiency. However, it is worth noting that EADP might still be preferred in small very constrained-node networks as it uses the simplest stateless flooding algorithm for its pull mode.

The contributions to enhancing Trickle allow the expansion of its usage even further. Thus, Trickle was introduced to manage code propagation in non-IP based CNNs, the work done in [45] brings it into the IP world, and the contributions of this chapter generalised Trickle's usage to encompass a variety of algorithms including flooding and most of the lightweight flooding substitution algorithms in both IP and non-IP networks. One of the new usages of the optimised Trickle algorithms is presented in the flexible TrickleSD protocol constituting the second main contribution of this chapter. TrickleSD showed good time/cost performance even under less efficient RDC protocols. The following chapter investigates new methods to enhance broadcast under RDC for the sake of providing a better time/cost performance for both EADP and TrickleSD.

Chapter 6

Link-layer Consideration: Improving Broadcast Communication under RDC

EADP and TrickleSD take advantage of the broadcast nature of the wireless channel to achieve efficient cooperative discovery tasks. Indeed, without broadcast, zero-configuration discovery operations would have been impossible. However, broadcast communication is fundamentally more costly than unicast in radio duty-cycled networks. This chapter starts with an overview of broadcast importance in CNNs before presenting a critical analysis of broadcast handling in RDCs. Subsequently, two main generic contributions to enhance broadcast performance in duty-cycled networks are proposed. This is followed by a comprehensive analysis of latencies and power consumption of unicast and broadcast communication patterns along with evaluations and discussions of the proposed contributions. Finally, the benefits of such contributions when deployed with EADP and TrickleSD are demonstrated.

6.1 Multicast and broadcast in CNNs

Multicast –the process of delivering a message to multiple destinations– has many interesting applications in CNNs. Examples include network configuration and administration; firmware installation and updates; resource, route and neighbourhood discovery. In 6LoWPANs, *"IPv6 level multicast packets MUST be carried as link-layer broadcast frames in IEEE 802.15.4 networks"* [6]. By this requirement, multicast packets are sent as link-layer broadcasts in 6LoWPAN networks.

Foreseeing the importance of multicast for the IoT, the IETF is standardizing MPL. Multicast is also specified as a communication pattern in CoAP [82]. Such a group communication pattern is ratified in RFC 7390 [166]. Moreover, multicast forms the basis for zero-configuration networking via the mDNS/DNS-SD suite. While mDNS/DNS-

SD are not explicitly designed for LLNs, there are ongoing and very active efforts to adapt them to LLNs [99], [134], [135], [167]. One of such efforts is introduced in the following chapter. Furthermore, multicast has an abundance of potential uses in various IoT applications such as building control where it is frequently employed in actuation tasks. Indeed, all Trickle-based applications rely on broadcast to achieve simple, reliable and efficient data dissemination. Moreover, broadcast can realise efficient anycast –the process of delivering a message to at least one destination– and can be used in opportunistic routing approaches [168].

In non-duty-cycled 6LoWPANs, because of the wireless medium nature, sending a packet to a particular receiver (unicast) or to all surrounding nodes (broadcast) consumes the same energy, rendering broadcast very efficient since a single transmission reaches multiple destinations [45]. In duty-cycled 6LoWPANs; however, this might not be the case. Depending on the adopted strategy of managing nodes' sleep/wakeup periods, multicast may consume more energy than unicast. Thus, in current asynchronous RDC strategies, broadcast transmissions are fundamentally more costly. The reasons behind this are discussed in the following section.

6.2 Broadcast handling under RDCs

By exploiting acknowledgements, RDC protocols discussed in section 2.3.4 are generally optimised for unicast, not for broadcast. For instance, receiver-based protocols hardly support broadcast. Indeed, RIT does not handle broadcast communications at all [41]. Sender-initiated RDCs support broadcast using mainly two approaches. The first approach, which uses data-strobes is deployed in ContikiMAC (the default RDC protocol in Contiki), and BoX-MAC-2 (the default RDC protocol in TinyOS). A broadcast communication with 1 sender and 4 receivers in ContikiMAC is presented in Figure 6-1. The second approach, used in CSL, adheres to a strobed preamble similar to that of X-MAC shown in Figure 6-2. It avoids the wasted energy represented by grey lines in Figure 6-2 using the *rendezvous* time embedded in every chirp (section 2.3.4.3). By decoupling receiver-energy from the length of the CCI, the additional broadcast energy wastage mainly affects the senders. Thus, in both cases, either the chirp or the data is repeatedly sent for the whole CCI. In addition to this extra cost, the lack of acknowledgments in broadcasts and the associated lack of reliability make the Unicast Burst Forwarding (UBF) mechanism discussed in section 2.3.4.4 very inefficient with broadcast. Since burst forwarding is very important for both EADP and TrickleSD, section 6.3 introduces a Multicast Burst Forwarding (MBF) mechanism but first, the following subsections discusses advantages and issues of using data-strobes for broadcast.

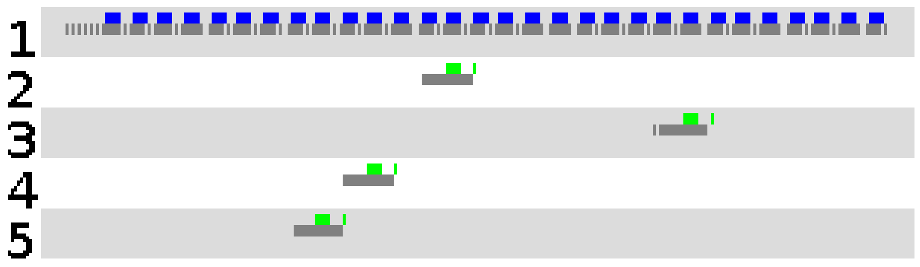


Figure 6-1 Broadcast Communication in ContikiMAC; CCI = 125ms

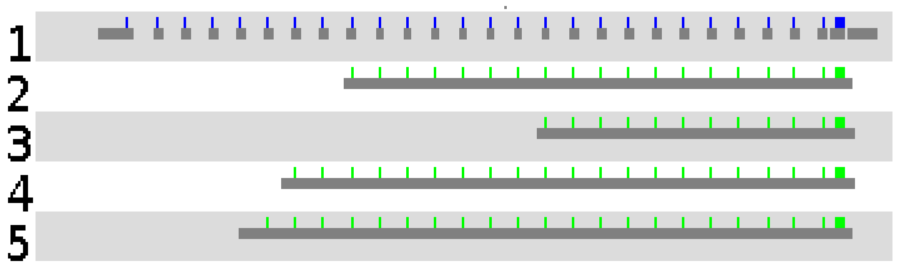


Figure 6-2 Broadcast in the Contiki implementation of X-MAC; CCI = 125ms.

Finally, it is worth noting that Figure 6-1, Figure 6-2 and similar figures in this chapter are generated using the Cooja Timeline tool [169], which visualises precise behaviours of network protocols in real-time. The tool delivers the radio state using colour codes: transmission (blue), reception (green), radio on (grey), radio off (white) and interference (red) as can be seen from Figure 6-1 and Figure 6-2.

6.2.1 Advantages of broadcast handling via data-strobes

The main advantages of broadcast handling using data-strobes are set out in the following subsections.

6.2.1.1 Efficient layer-2 anycast

One of the main advantages of broadcast handling using data strobes is efficient anycast communication. Indeed, by adopting such a mechanism (Figure 6-1), anycast can be achieved with similar performance to unicast. For instance, a node waking up first and responding to particular protocol criteria can acknowledge the transmission. This allows the sender to stop its transmission early which consequently conserves energy and channel utilisation. Stopping the transmission early also enables other receivers to save energy. Indeed, a new network metric based on this concept, called Expected Duty Cycles (EDC), was introduced in [168]. Such a metric is applied to RPL in [170], where it demonstrated noticeable performance improvements. In this work, layer-2 anycast can provide another reply-storms-avoidance-mechanism for the reply agent (section 4.6.1).

6.2.1.2 Gained processing time

Another advantage can arise from the fact of allowing the receivers to receive a frame as soon as they wake up and thus giving them sufficient time to perform required processing in the period during which the sender is still transmitting.

6.2.1.3 Robust to rapid interferences

Data-strobes might achieve better reliability for broadcast since the data itself is repeatedly transmitted. Thus, if a receiver wakes-up and receives a corrupted signal, it might stay awake and receive the following data. Indeed, this mechanism provides a means of remedying rapid interferences such as those caused by microwave ovens operating in the 2.4 GHz band [56].

6.2.2 Issues of broadcast handling via data-strobes

This section discusses the main issues arising from using data-strobes for broadcast handling. Such issues are basically due to an inherent collision problem characterizing broadcast handling using data-strobes, which can be seen in Figure 6-1.

6.2.2.1 Multi-hop forwarding

If a packet has to be delivered through multi-hop, a systematic collision problem depicted in Figure 6-1 might cause it to backoff at every hop thus incurring significant delays in the delivery time of such a packet. Indeed, if any node 1, 2, 3 or 4 in Figure 6-1 attempts to immediately retransmit a received packet, it fails since node 1 is still transmitting. The node then backs-off to try retransmitting again. In fact, this is the issue which prompted the introduction of a small delay when evaluating EADP (section 4.8.1). It is also the same issue that led to using X-MAC when evaluating TrickleSD (section 5.8.1).

6.2.2.2 Extended backoff periods

It is clear from the above that RDC introduces delays in the communication. Such delays might affect the CSMA/CA medium access strategy deployed by IEEE 802.15.4, especially the duration of the backoff period. Thus, to avoid failing the second attempt at retransmission, the CSMA/CA implementation in Contiki assumes a worst case link latency and waits for a randomized extended backoff period [55] of at least the size of CCI before attempting to retransmit again. Hard-coupling the backoff period to the CCI is mainly imposed by the systematic collision problem depicted in Figure 6-1. Finally, it should be noted that the radio is disabled during this period, and, therefore, there is no additional energy consumption.

6.2.2.3 Delay-and-cancel based forwarding

Delay-and-cancel mechanisms rely on deferring the retransmission of received packets in order to achieve better performance. Thus, all the delay-and-cancel mechanisms discussed in section 5.1, including Opt-Trickle, work on the assumption that a packet is received simultaneously by all receivers. This assumption is broken by data-strobes broadcast handling. While such an issue impacts both random deferring mechanisms such as Opt-Trickle (section 5.3), and deterministic deferring mechanisms such as Augmented Trickle (section 5.5.3), it is more of a threat to deterministic ones.

6.2.2.4 Cooperative feedback-based reliability

Broadcast is known to lack reliability since broadcast transmissions are not acknowledged [4]. In order to achieve cost-effective reliability, some protocols rely on the concept of negative acknowledgement (NACK). Unlike ACK, NACK is only triggered if a receiver detects that some data is missing. To avoid NACK implosion, receivers detecting inconsistencies generally defer their NACK transmissions and employ suppression techniques similar to the above case (section 6.2.2.3). Consequently, a node trying to transmit a NACK might fail because of the inherent collision of the data-strobes broadcast. In this research, such a problem might affect the operability of the RA mechanism discussed in section 4.6, which proposes a cooperative feedback-based mechanism in order to avoid potential service cache reply storms.

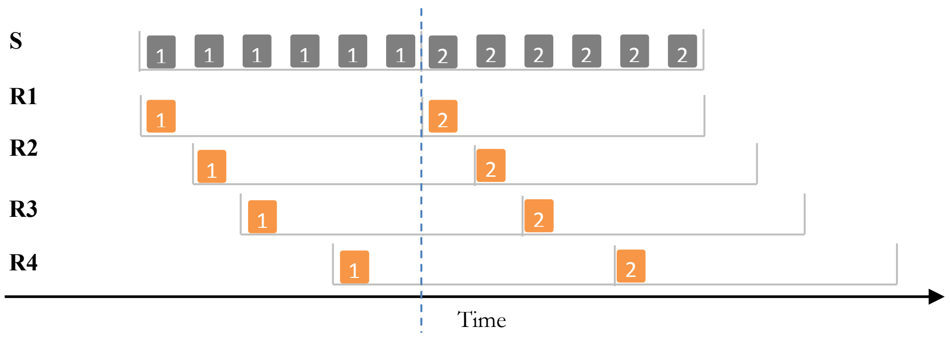
In addition to the need for an MBF mechanism outlined in section 2.3.4.4, these discussions point to the requirement for a mechanism that addresses the fundamental limitation of broadcast handling using data-strobes. Such a mechanism will be the subject of section 6.4. MBF is introduced in the following section.

6.3 Multicast burst forwarding

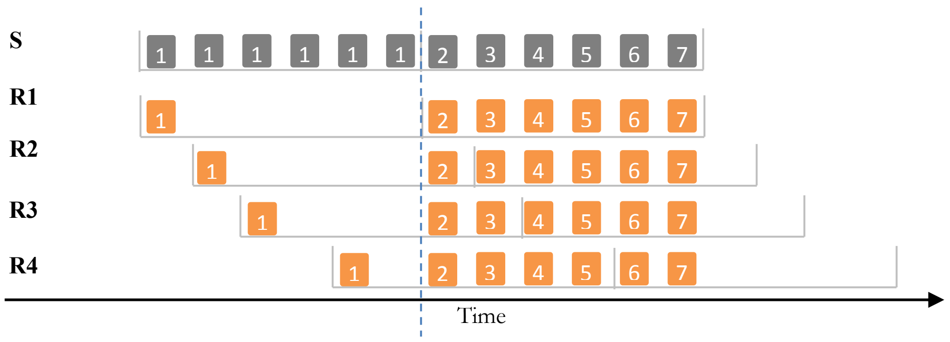
This section introduces multicast burst forwarding. It starts by presenting the underlying mechanism, its features, reliability and the impact of data strobes.

6.3.1 The MBF mechanism

Since broadcast transmissions are not acknowledged, MBF works on the assumption that if a node is awake it will receive a transmitted frame (or chirp). Thus, an MBF sender repeatedly transmits the first frame with the pending bit set to 1 for the whole transmission period. This is in order to awake all receivers and inform them to stay awake. The sender then transmits subsequent frames only a few times, ideally once. This avoids unnecessary repetitions required to awake receivers and thus saves a considerable amount of senders' energy, minimises communication latency, and increases throughput.



(a) ContikiMAC broadcast



(b) The MBF mechanism combined with ContikiMAC

Figure 6-3 Multicast burst forwarding combined with ContikiMAC.

Figure 6-3 presents the MBF mechanism when integrated with ContikiMAC. Figure 6-3 (a) shows how a burst of multicast packets is transmitted in ContikiMAC while Figure 6-3 (b) depicts MBF's behaviour to bursts. To awake all receivers, the first frame has to be repeatedly transmitted over the whole check period as in ContikiMAC. Because receivers are now awake, subsequent frames will be sent just a few times; ideally once (Figure 6-3 (b)). This allows more frames to be transmitted in a particular period of time, thereby increasing throughput and decreasing latency.

6.3.2 MBF features and practical considerations

With MBF, data transmissions are considerably shortened to just one or a few frames which save considerable energy. MBF is also important as frames do not have to undergo the CSMA/CA process for accessing the channel, which further reduces latency. This is particularly important since extended backoffs are generally deployed with RDCs. On the other hand, if any frame in the burst undergoes collisions, the burst transmission is interrupted. Therefore, the CSMA back-off is called thus allowing competing senders to access the channel. In unicast burst forwarding, not receiving acknowledgments can also interrupt the transmission and thereby trigger CSMA back-offs. Unfortunately, MBF lacks such a feature; resulting in a sender possibly monopolising the channel for longer periods. To overcome this issue, MBF uses a timer to decide on a burst's maximum duration. On the receiver side, a guard timer is employed to stop burst reception mode when waiting longer for lost frames.

It is important to note that the issues with data-strobes, discussed in section 6.2.2, imply an additional energy cost for every first frame in an MBF burst. Thus, when receiving the first frame, node R1 in Figure 6-3 (b) has to stay awake for approximately the whole channel check interval just waiting for the second frame to arrive. A similar observation can be made with the other three nodes (R2, R3 and R4 in Figure 6-3 (b)) with smaller wasted on-times. Indeed, this is a serious problem that might even prevent the adoption of MBF because of the energy wastage that grows linearly with the number of receivers. Section 6.4 introduces a technique to solve this issue along with the ones discussed in section 6.2.2.

6.3.3 The case of fragmentation

Particular care has to be taken with multi-hop forwarding of fragmented packets which is known to be problematic in LLNs since losing a fragment causes a packet to be discarded [56]. MBF, as described above, provides an effective way of treating fragmentation in a hop-by-hop fashion. Indeed, MBF's fragmentation/reassembly at each hop can save network resources and still benefit from MBF's throughput and latency improvements. However, one might consider a *forward-then-construct* strategy which might benefit better from MBF in terms of end-to-end throughput and latency. In such an approach, a node first tries to forward the fragments it receives without waiting for complete reassembly. When all the fragments are received, the node constructs the packet and delivers it to the upper layer. However, *forward-then-construct* may waste network resources by transmitting fragments which will be (later on) discarded. In this chapter, the default strategy is the hop-by-hop fragment forwarding.

6.3.4 Reliability

Benefiting from its energy savings, MBF can offer best-effort reliability by repeating a frame transmission more than once. Rather than for simplicity reasons, MBF does not improve on multicast reliability though it provides attractive features to build effective reliability mechanisms. Building such reliable mechanisms is outside the scope of this chapter, and it is left for future investigations. Finally, it is worth noting that MBF is expected to work also for best-effort unicast burst transmissions (i.e. if a sender does not require acknowledgements). This cannot be achieved without excessive resource consumption, increased latencies and decreased throughput when using unicast burst forwarding.

6.4 Addressing the issues of broadcast via data-strobes

This section presents a simple, yet effective method for addressing the issues of broadcast handling in data-strobes. Note that an intuitive solution would be to impose on every receiver to wait for the size of CCI before any attempt of transmission. This, however, would introduce more delays in the communication and will only solve the issue of multi-hop forwarding discussed above (section 6.2.2.1).

6.4.1 The proposed solution

Inspired by the *rendezvous* time used in CSL, this section proposes incorporating in every data strobe the time remaining until the end of the current transmission. This information, termed *synchronisation* time, is then extracted from every received data strobe thereby enabling the receivers to synchronise their subsequent operations based on the sender's activity (e.g., wait until the end of the current transmission before any attempt to transmit). In the case of MBF, *synchronisation* time allows the nodes receiving a first frame in the burst to sleep until the expected transmission of the second frame.

Compared to the *rendezvous* time, the *synchronisation* time may not require precise timing. Thus, late-skews resulting from small additional drifts do not greatly affect the performance of *synchronisation* time. Also, accounting for processing times allows the *synchronisation* time to be robust to early-skews. This tolerance enables flexible implementations of *synchronisation* time. However, precise implementations are required if it has to be used for strict implicit synchronisation between receivers.

In addition, and unlike the *rendezvous* time, *synchronisation* time can be offered to upper layers willing to benefit from it. For instance, a protocol can exploit the *synchronisation* time to perform some secondary operations while waiting to transmit, (e.g., read/write to flash). Indeed, *synchronisation* can even be used as a metric. For example, a node that can do its processing in the order of magnitudes of the *synchronisation* time can be considered as a better candidate forwarder than a node which takes longer times. However, while the *rendezvous* time is embedded in the *chirp*, the current implementation of the *synchronisation* time reserves 2 bytes from each data frame. This might not be an issue for non-full frames, but it could present a trade-off for full frames. Note that the current ContikiMAC implementation already takes 2 bytes from each data frame in order to ensure the minimum frame size required by the 2-CCA wakeup mechanism. Optimizing these bytes and using them for the *synchronisation* time might be an option. Finally, it is worth recalling that the *synchronisation* time is only required for broadcast frames. In the case of MBF, it is only necessary for the first frame of a burst.

6.4.2 Implications on CSMA/CA

The *synchronisation* time allows decoupling the CSMA/CA backoff period from the RDC's channel check interval. This decoupling opens avenues for new research looking at better CSMA/CA and RDC interactions by finding a better extended backoff period trade-off. In addition, a sender finding the channel busy can snoop on-going transmissions and get the *synchronisation* time to be used as a hint for calculating the next backoff period.

Having discussed and presented solutions to address broadcast shortcoming under RDCs, the remainder of this chapter is devoted to evaluating the performance of unicast/broadcast communications and the proposed techniques. First, a comparison of unicast and broadcast primitives is provided in order to get a feel for the additional cost required for broadcasts. Next, evaluations of MBF and *synchronisation* time are presented. Finally, the advantages of these optimisations, when used with EADP and TrickleSD, are discussed.

6.5 Broadcast and unicast performance under RDCs

This section quantifies multicast/unicast latencies and energy consumptions in duty-cycled single-hop 6LoWPAN networks. It evaluates the performance of multicast/unicast under the following conditions: (i) different radio duty cycling mechanisms; (ii) varied sleep periods which has a direct impact on the latency, throughput and power consumption; and (iii) when varying the packet frequency rate. The performance metrics of interest to this study are the radio duty cycle as a proxy of consumed energy (section 5.8.1) and the transmission latency.

6.5.1 Evaluated RDC protocols

To achieve the aims of this experiment, three RDC protocols were considered namely: ContikiMAC and X-MAC as representatives of sender-initiated approaches, and LPP as an example of a receiver-initiated approach. Their implementations, which slightly differ from the original specifications are studied and summarised as follows:

- X-MAC implementation in Contiki provides additional optimisations by deploying the ContikiMAC *phase-lock* mechanism (section 2.3.4.3) for unicast. It also uses the ContikiMAC broadcast mechanism by default.
- LPP implementation also includes the *phase-lock* mechanism. To respond to probes, LPP senders use hardware acknowledgments. The default Contiki LPP configurations were used. LPP broadcasts are also managed by the ContikiMAC mechanism
- ContikiMAC implementation combines the original protocol with UBF [56].

6.5.2 Evaluation methodology

Experiments were conducted using both simulations and a local testbed. The former provided a controlled experimental environment and the latter an authentic framework for validation of the simulation results.

In the Cooja simulator [89], a star-topology network comprising 5 emulated Tmote Sky motes was set (Figure 6-4). The base station (node 1 in Figure 6-4) was periodically sending data to node 3 in the case of unicast and to all the other four nodes in the event of broadcast. As can be seen from Figure 6-4, all the nodes are within the transmission range of node 1 and the links are perfect (100% reception ratio). This is so to mitigate all other effects rather than RDC in order to get a comprehensive understanding of RDC impact on multicast/unicast performance.

For the testbed, a similar configuration was set up using AS-XM1000[11] motes (a variant of the Tmote Sky mote) in an environment with working Wi-Fi deployments (Figure 6-4). In both simulation and testbed experiments, each test was run for 10 minutes. Channel 26 (Figure 2-6) was used to minimise interference with coexisting 2.4 GHz technologies such as Wi-Fi. The default CCI was 125ms with 65-byte message payload. A

[11] http://www.advanticsys.com/shop/asxm1000-p-24.html

transmission rate of 1 packet every 15 seconds was used. This rate might not show the full benefits of the *phase-lock* mechanism used in unicast. The full benefit of such mechanism is shown when varying the transmission rate. Unless differently stated, these are the default values used in the evaluations below. Each testbed experiment was repeated three times.

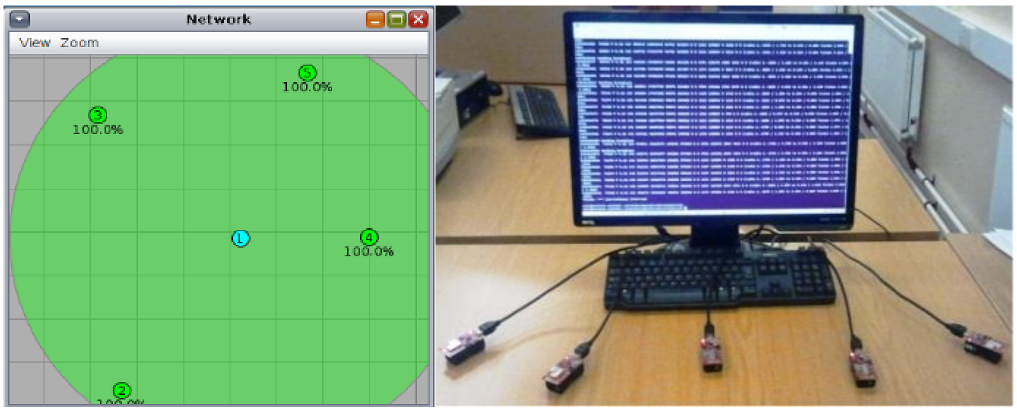


Figure 6-4 Simulation and testbed setups

Table 6-1 Simulation parameters

Parameter	**Value**
Simulation time / #nodes	300, 600s / 5, 10
Medium / range / bandwidth	UDGM / 50m / 250kbps
Traffic type / rate	CBR / variable rates
MAC / Adaptation	CSMA-CA / 6LoWPAN
RDC layer	LPP, X-MAC, ContikiMAC

6.5.3 Results and discussions

This section reports obtained unicast/multicast performance under different: RDC protocols, channel check rates, transmission rates and payload sizes.

6.5.3.1 Multicast and unicast performance under RDCs

In this experiment, only the underlying RDC protocol was changed. Figure 6-5 presents obtained results.

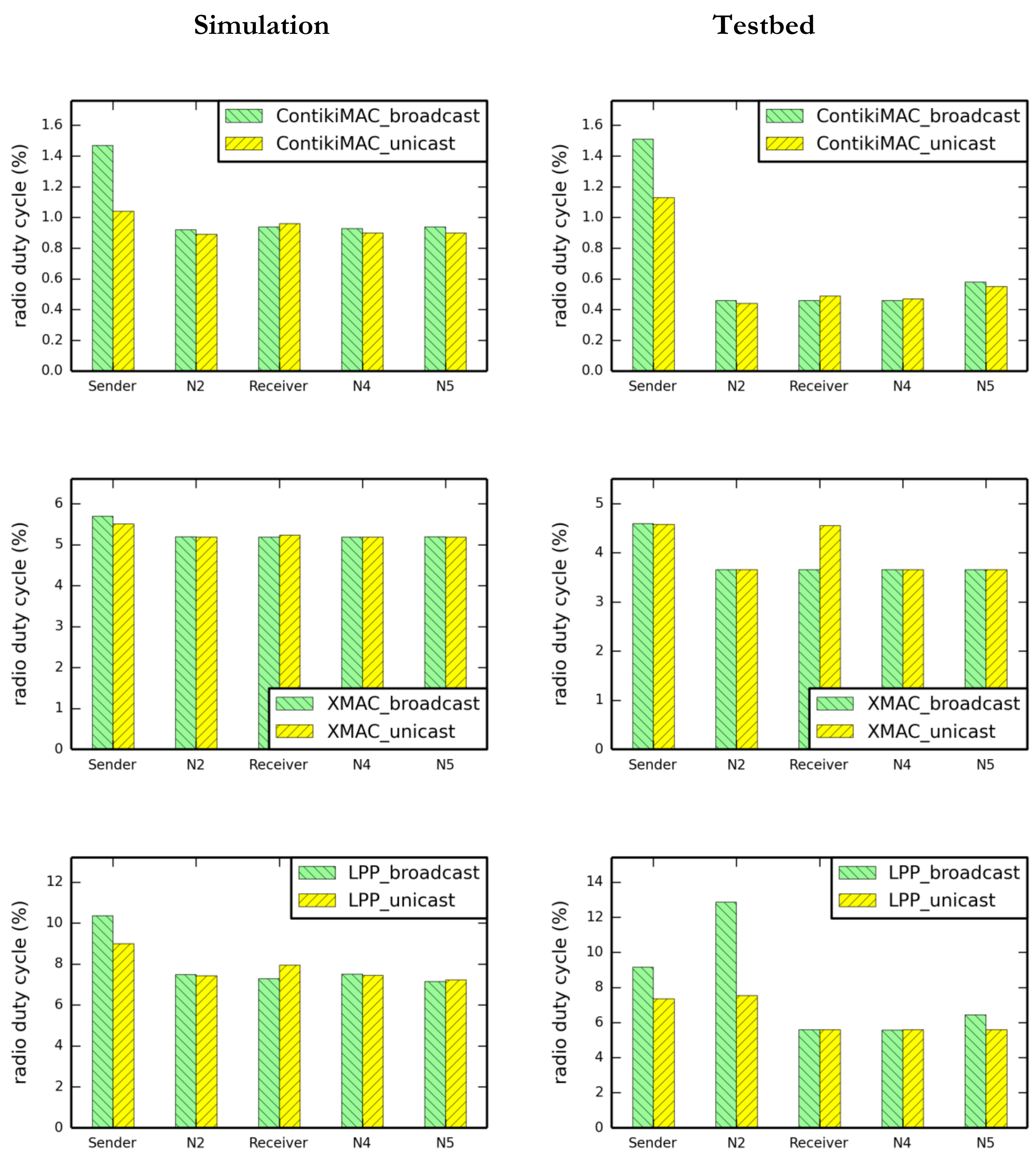


Figure 6-5 Unicast/Multicast radio duty cycles under different RDCs

As can be seen from this figure, ContikiMAC registered the best radio duty cycle which is approximately less than a third of that of X-MAC. LPP registered the worst radio duty cycle with about 10% on time. When it comes to the comparison between unicast and broadcast, it is clear from this figure that sending broadcast packets consumed more energy than sending unicast ones. Thus, in ContikiMAC, sending in broadcast kept the radio on for 1.47% of the time while unicast transmissions kept it on for 1.04% of the time. This represents about 30% additional on-time. It can be explained by the mechanisms of optimising unicast discussed in section 2.3.4.3. However, receiving unicast packets consumed more energy. For the other nodes, although they were not involved in the unicast communication, they consumed approximately the same energy as in broadcast. The small additional energy consumed by broadcast was due to receiving the actual data before returning to sleep. The same pattern can be seen in X-MAC and LPP but with less difference between broadcast and unicast 3% and 23% respectively.

The testbed results show a similar trend to that observed in simulation as can be seen from Figure 6-5. Thus, ContikiMAC registered the best radio duty cycle followed by X-MAC and then LPP. Regarding the unicast/multicast performance, the same conclusion as in simulations can be drawn. For instance, the unicast receiver in X-MAC consumed about 20% more energy than broadcast reception. Finally, it should be noted that LPP duty cycles were unstable. This is because LPP implementation in Contiki uses less accurate timers.

As ContikiMAC implements the same mechanism for both unicast and broadcast, has the most stable implementation, and because it presented the best radio duty cycle and showed the largest difference ratio between broadcast and unicast, the following experiments focus on studying unicast/broadcast performance in ContikiMAC.

6.5.3.2 Varying transmission frequency and CCI

In these experiments, unicast/multicast power consumptions when varying transmission latency and CCI were compared. The payload was kept fixed at 65 bytes. The RDC registered in both Broadcast (B) and Unicast (U) by: (i) the sender BS-RDC and US-RDC; (ii) the transmission activity at the sender BTx-RDC and UTx-RDC; and (iii) the average RDC registered by other nodes (receivers) BR-RDC and UR-RDC were measured and reported in Figure 6-6.

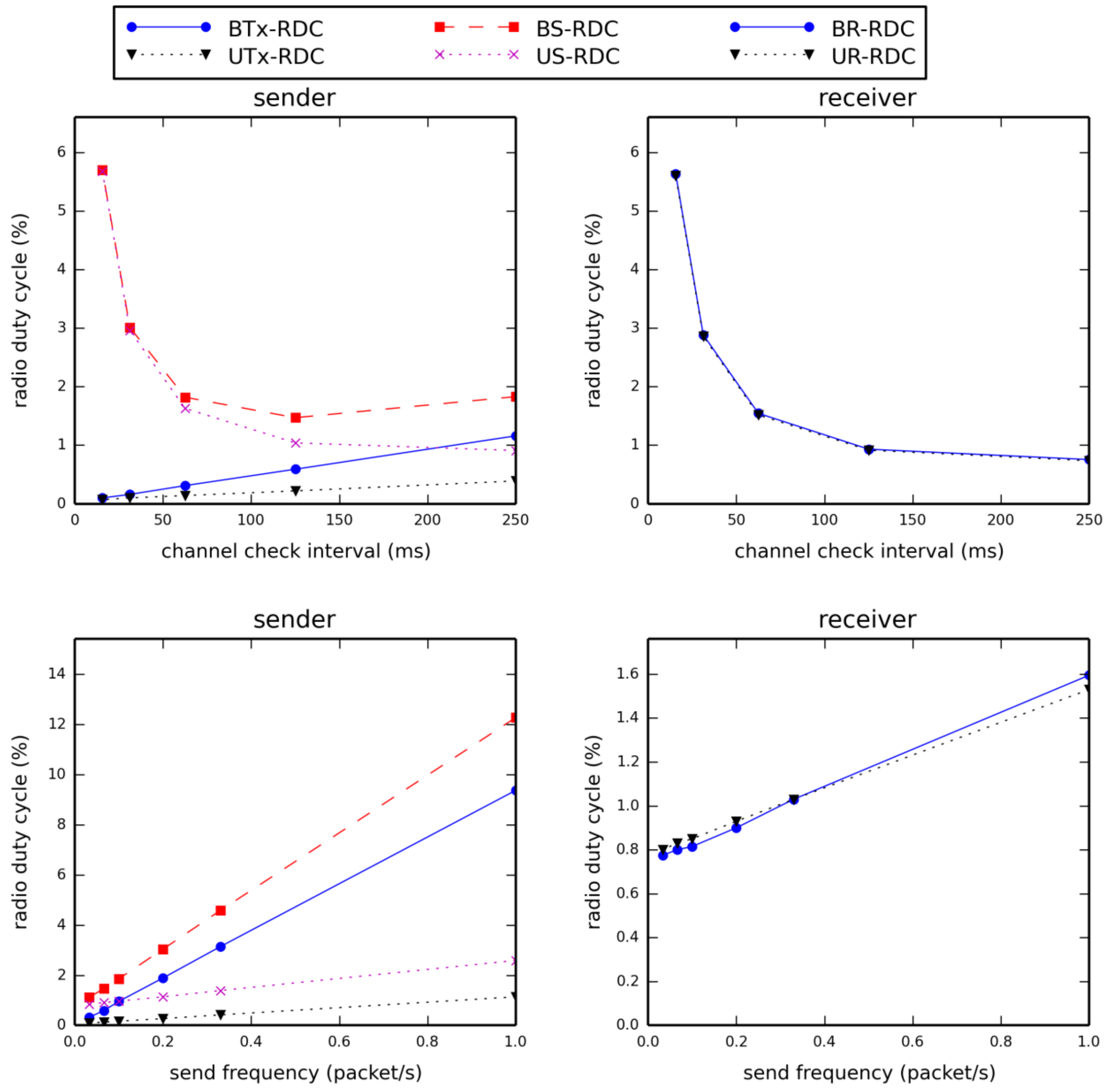


Figure 6-6 Channel check rate impact on broadcast/unicast duty cycles

The first row of graphs in Figure 6-6 show that unicast and broadcast consumed approximately the same energy up to a CCI of 62.5ms. This is explained by the fact of decreasing sleep-periods, which minimised the benefits of the *phase-lock* mechanism used in unicast transmissions. Indeed, the ContikiMAC implementation disables this mechanism if the CCI is below 16ms. However, for a CCI bigger than 62.5ms, unicast transmissions showed noticeable energy savings which increased with increasing check intervals. For instance, at a CCI of 250ms, unicast consumed about half the energy consumed by broadcast. This is realised thanks to the *phase-lock* mechanism. This trend is also confirmed by the transmission activity at the sender (Tx-RDC) graphs.

The second row of graphs shows that the gap in the energy consumption, of the sender, between broadcast and unicast increases with increasing transmission frequency. This is due to the fact that in such cases unicast benefits better from the *phase-lock* mechanism which allows it to save noticeable energy. At the receiver, however, unicast and broadcast showed comparable energy consumptions when varying both CCI and the send frequency. This is explained by the fact that the receivers' energy is decoupled from the length of the wakeup signal.

6.5.3.3 Multicast and unicast single-hop latencies

Table 6-2 presents the one-hop latency, measured at the application layer, when sending a multicast/unicast non-fragmented message (65 bytes payload) and a fragmented message (120 bytes payload) under the three representative radio duty cycling protocols: ContikiMAC, X-MAC and LPP, and for a non-duty-cycled network using the NullRDC protocol. In each experiment, about 40 messages were sent and the average time taken to receive a message is reported in Table 6-2.

As can be seen from this table, RDC protocols introduce latency at each hop. Thus, the best RDC protocol delivered a non-fragmented packet about 4 times late when compared to a non-duty-cycled network. The worst RDC delivered a packet about 6 times late. Generally speaking, broadcast non-fragmented packets were delivered earlier than unicast ones. This continued to be the case for all protocols except ContikiMAC for fragmented packets. Thus, under ContikiMAC, a fragmented broadcast packet was delivered about 35% of the time later than a unicast one. This confirms the gains brought by UBF.

Table 6-2 Unicast/broadcast transmission latencies (ms)

Packet / **Protocol**	**non-fragmented**		**fragmented**	
	unicast	**broadcast**	**unicast**	**broadcast**
X-MAC	155	112	409	365
LPP	91	160	360	269
ContikiMAC	113	112	125	168
NullRDC	24	23	35	34

6.6 Evaluation of MBF and synchronisation time

6.6.1 Performance evaluation of MBF

In this section, the same evaluation methodology and experimental design described in section 6.5.2 were used.

6.6.1.1 Senders' energy consumption

This experiment evaluates the energy consumption of MBF. To this end, the size of transmitted packets is varied in order to trigger fragmentation and hence create bursty traffic. Bursts are of great interest for TrickleSD as the push mode can generally generate packets that get fragmented. Also, fragmentation is of great interest in 6LoWPANs where an IPv6 packet might be fragmented into at least 18 fragments in *route-over* configurations and, at most, 32 fragments in a worst case *mesh-under* configuration [171]. Since MBF mainly affects senders' energy, the duty cycle of the sender along with the sender's transmission activity were measured for both MBF and ContikiMAC. Obtained results are depicted in Figure 6-7.

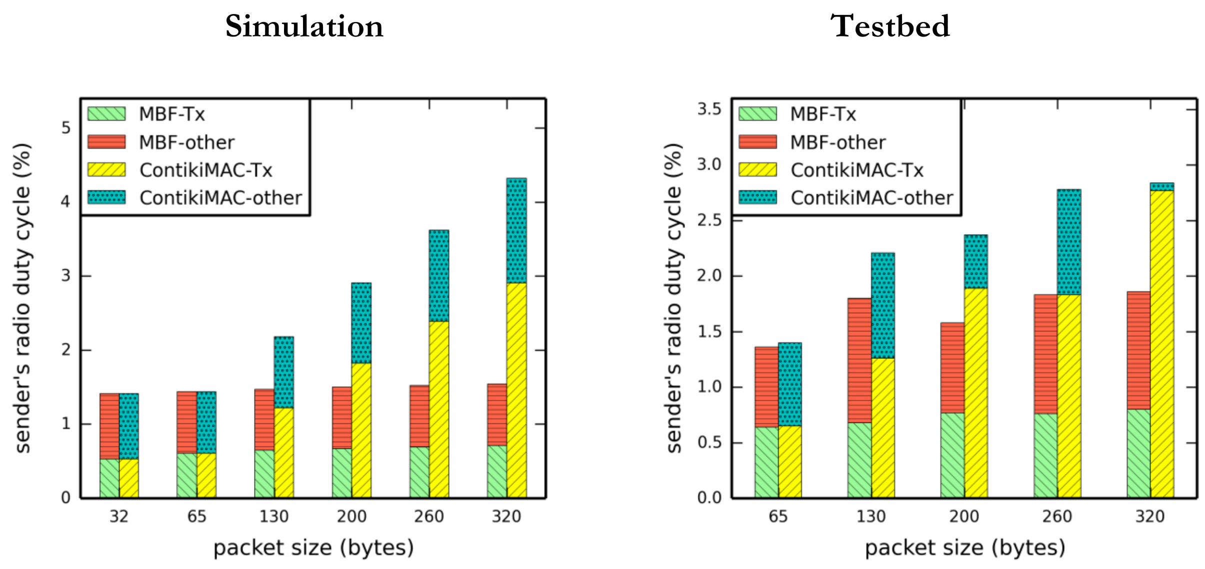


Figure 6-7 Energy consumption of an MBF and a ContikiMAC sender

As can be seen from the above figure, both MBF and ContikiMAC registered similar performance for non-fragmented packets where no burst is generated. However, for an application payload greater than 130 bytes, MBF registered important energy savings. Thus, simulation results show that for a packet of 260 bytes payload, the average radio activity at the sender registered around 1.5% using MBF while it registered around 3.6% on-time when using ContikiMAC. This is confirmed by the testbed results. Such a performance is achieved as a result of avoiding unnecessary repetitions used by ContikiMAC in order to wake up the receivers. This evidence is clearly reported by the plots of the transmission activity at the sender.

6.6.1.2 Throughput

Having shown the energy benefits brought by MBF to multicast burst transmissions, this section evaluates MBF's maximum communication throughput when compared with ContikiMAC and with a non-duty-cycled network running NullRDC. To do so, an experiment, where the sender sent 65-payload messages as fast as it can, was designed. For each run, the number of transmitted packets, as well as the reception ratio, the transmission latency and the throughput, expressed as packets per second (pps) were recorded. Each experiment ran for 5 minutes. Obtained results are depicted in Table 6-3.

Table 6-3 MBF throughput

Protocol / Metrics	ContikiMAC	MBF	NullRDC
Sender RDC (%)	79.81	31.45	100
Receiver RDC (%)	5.7	95.29	100
Sent Packets	2069	14576	21716
transmission time (ms)	91	31	24
Throughput (pps)	6.88	48.59	72.38

As can be seen from Table 6-3, MBF increased the available communication throughput by about 8 times when compared with the maximum throughput available using ContikiMAC. This was accompanied by a remarkable decrease in the transmission latency from more than 90ms per packet in ContikiMAC to around 30ms per packet with MBF. Throughput performance is mainly registered because of not sending repeated frames thus freeing the channel for transmitting useful ones. The latency performance is primarily due to the fact of keeping receivers active waiting for the data. Compared with a non-duty-cycled network, MBF registered comparable maximum available throughput and latency while allowing for energy savings, especially at the sender. Finally, it is worth noting that [158] has presented throughput figures for both NullRDC and ContikiMAC which are in concordance with our findings.

6.6.1.3 Single-hop latency

In this experiment, a burst of multicast traffic is generated by triggering fragmentation. Results of averaging the latency over 300 messages are depicted in Figure 6-8.

As can be seen from Figure 6-8, a sender employing MBF transmitted multicast bursts at a rate two to three times faster than ContikiMAC for 130-byte and 320-bytes sized messages respectively. For instance, a 320-byte packet was transmitted in less than 200ms using MBF while ContikiMAC spent around 600ms to transmit it. This experiment shows the importance and benefits brought by MBF to fragmented packets. Thus, by transmitting fragments faster, MBF allows for larger packets to be transmitted over LLNs and be reconstructed within the reassembly time.

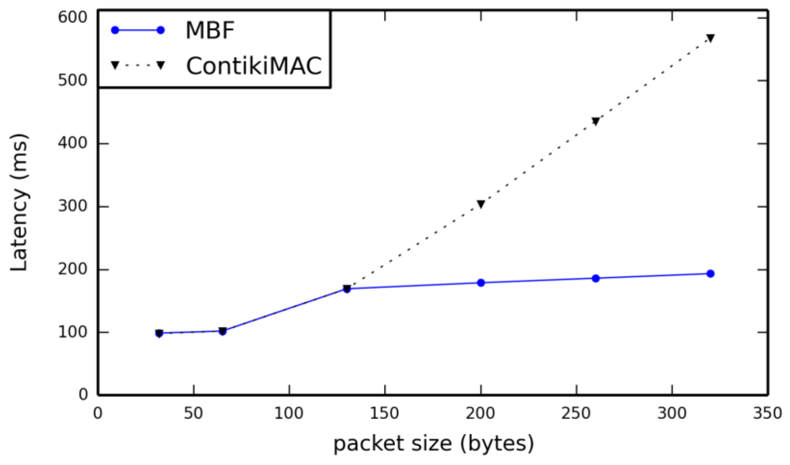


Figure 6-8 MBF's single-hop latency

6.6.1.4 Impact on aggregation

Aggregation allows LLN protocols to minimise the number of transmissions by aggregating smaller packets into a larger one. TrickleSD exploits this concept in its push mode. For instance, the bursts arising in the TrickleSD push mode can be caused by aggregation. An experiment was designed to show the impact of aggregation on energy consumption. In this experiment, the sender either transmits 200 packets of 32 bytes payload, half this number (100 packets) with double payload (64 bytes) and so on until sending 25 packets of 260 bytes payload. Obtained results are depicted in Figure 6-9.

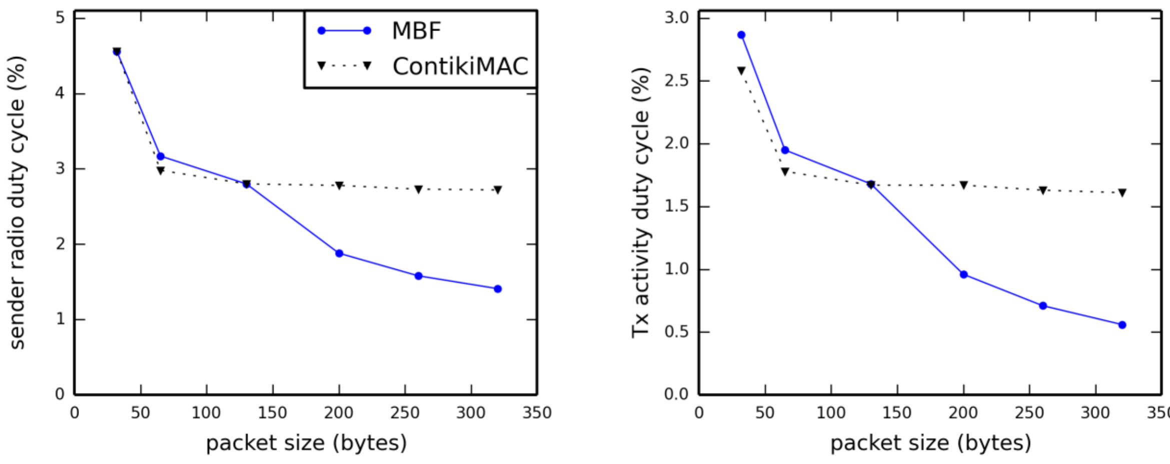


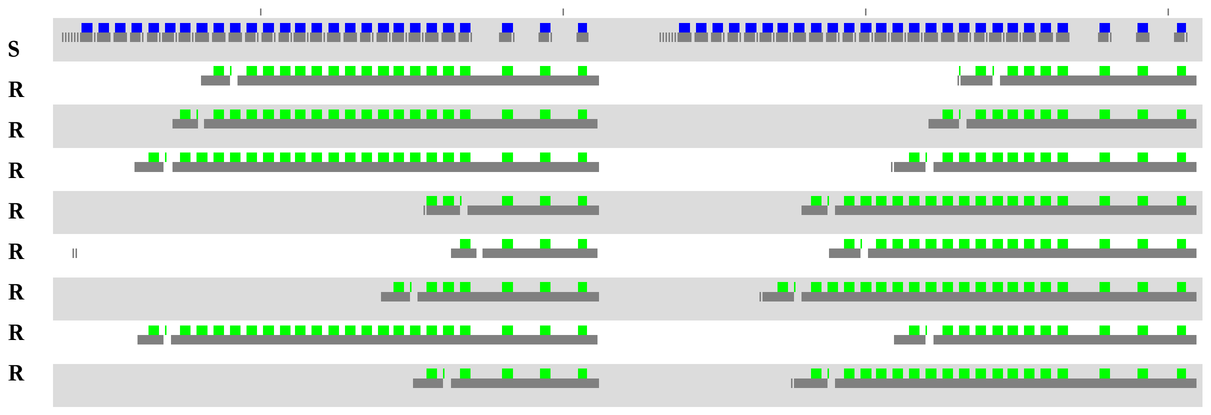
Figure 6-9 Aggregation performance

For both MBF and ContikiMAC, aggregation showed a noticeable reduction in energy consumption while not leading to fragmentation (less than 100-byte payload). However, aggregated packets leading to fragmentation have shown no benefits when using ContikiMAC, making aggregation not worthwhile. Conversely, with MBF, a noticeable gain in energy consumption can be obtained when aggregating data. Thus, sending 25 packets of 260-bytes payload consumed around 25% less energy than that of sending the same amount of information in packets of 32-bytes payload.

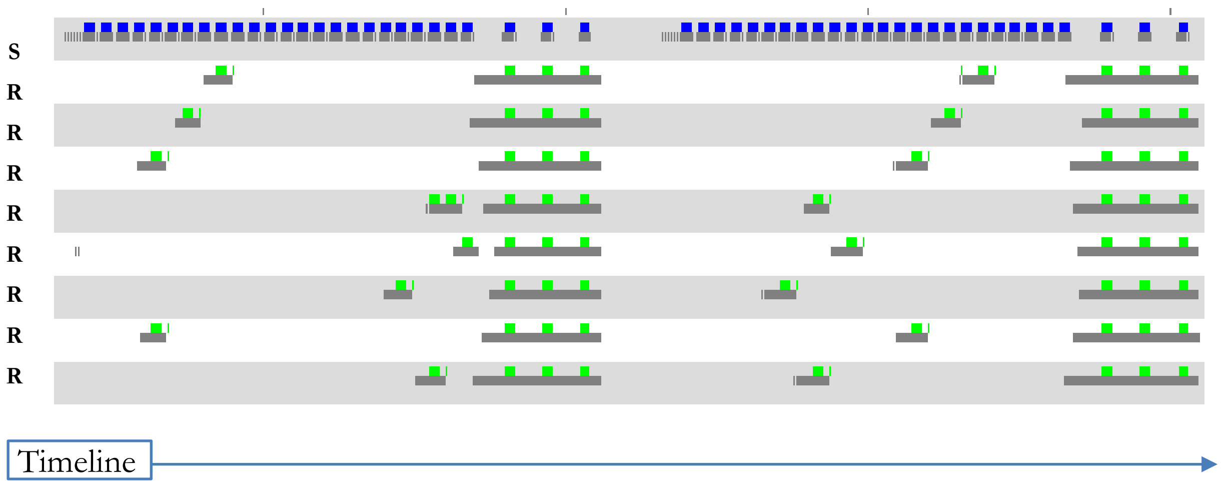
6.6.1.5 Receivers' energy consumption

The above subsections focused on the benefits of MBF to senders' energy, throughput, latency and benefits for aggregation. However, the impact on receivers' energy was not shown. This is because receivers' energy is already decoupled from the length of the CCI by ContikiMAC. Hence, MBF does not enhance on receivers' energy. On the contrary, when deployed alone, MBF causes the receivers to waste more energy as they have to stay awake every time a first frame in a burst is received. Nonetheless, the energy consumption of MBF receivers is still far better than that of ContikiMAC-UBF; currently implemented as default in Contiki, as demonstrated in [172]. To see the implication of MBF on receivers' energy, an experiment was carried out in Cooja with one sender and 8 receivers. The sender periodically broadcasted messages of 300-byte payloads.

Figure 6-10 shows the radio activity of the sender and receivers reported by the Cooja Timeline tool over a period of 375ms. Figure 6-10 (a) visualises the benefits of MBF (in the same period of time, ContikiMAC only transmitted half a message while MBF transmitted two messages). However, the cost implied on the receivers was dramatic. To address this issue, the *synchronisation* time mechanism must be deployed with MBF in order to get all the benefits. Thus, when coupled with the *synchronisation* time, MBF can achieve independent energy consumption for the receivers as can be seen from Figure 6-10 (b). Indeed, the *synchronisation* time allows a receiver to sleep until just before the start of the transmission of the second frame which makes the cost of reception totally independent from the length of the CCI.



(a) MBF



(b) MBF + *synchronisation* time

Figure 6-10 MBF and synchronisation time performance

6.6.2 Synchronisation time impact on EADP

The previous subsection showed the importance of the *synchronisation* time to save receivers' energy for MBF when used with ContikiMAC. Indeed, the *synchronisation* time can solve a broader range of issues including those discussed in section 6.2.2. This section demonstrates the performance of the *synchronisation* time for the case of multi-hop forwarding of multicast traffic in order to show its importance for the hit time and throughput for protocols like EADP. To this end, 10 emulated Tmote Sky motes [90] were deployed in a 9-hop line topology network (Figure 6-11) in Cooja [89]. This third scenario is used in order to show the impact of *synchronisation* time on EADP performance in other very prominent IoT applications of LLNs including street lighting and vehicular network applications.

All nodes run EADP with the push mode disabled (EADP-d) which results in a limited flooding algorithm. A node on the left side of the network periodically broadcasts a service request to be flooded into the network. From the SD performance metrics defined in section 4.8.2, the average hit time is of importance to this evaluation. It is reported when varying the distance (in terms of the number of hops) between the client and the provider and when varying the CCI. The average hit time can also provide a proxy for the end-to-end throughput. Therefore, a discovery protocol ensuring smaller hit times can achieve higher throughput. To put the results of EADP-d with *synchronisation* time (EADP-d with ContikiMAC-Sync) in context, it was compared to that of EADP-d with ContikiMAC. Obtained results are depicted in Figure 6-12, Figure 6-13 and Figure 6-14.

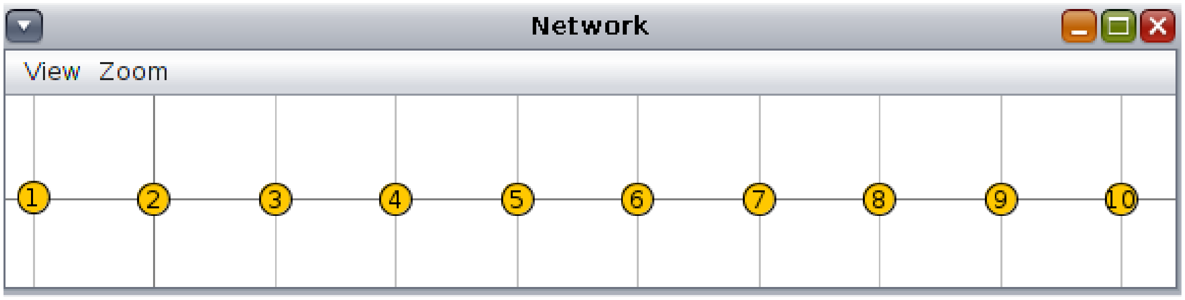


Figure 6-11 Line topology network (scenario #3)

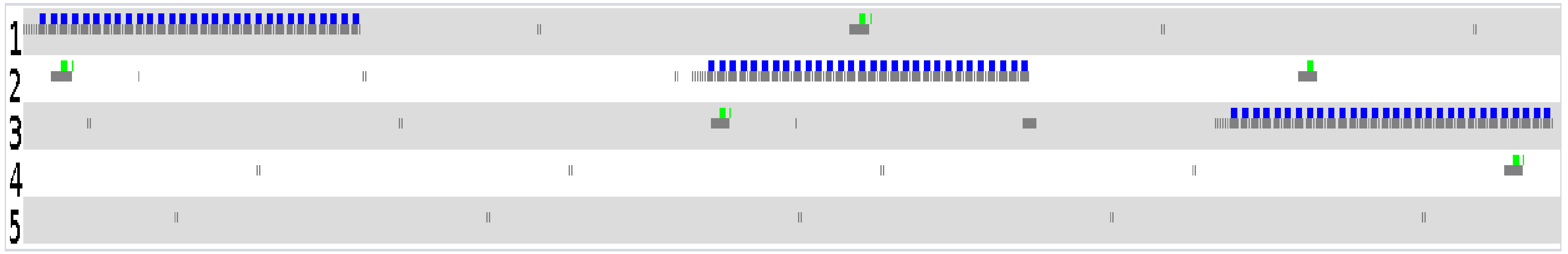


Figure 6-12 EADP-d with ContikiMAC, CCI = 125ms

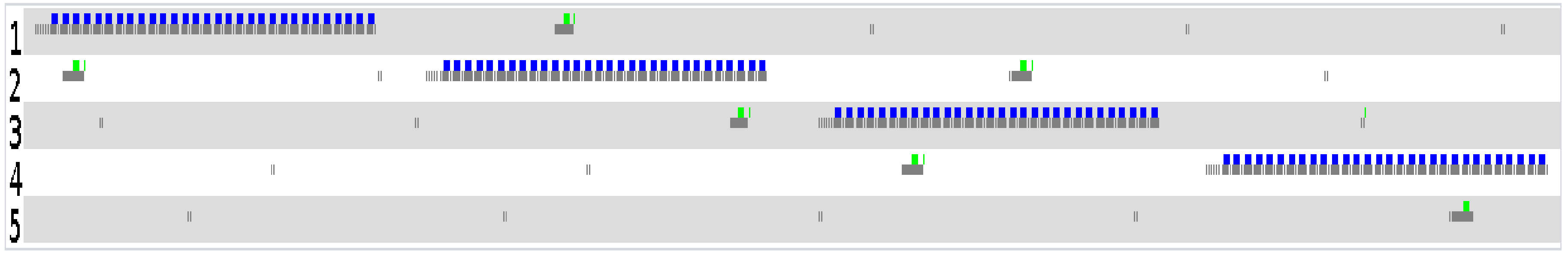


Figure 6-13 EADP-d with ContikiMAC + synchronisation time, CCI = 125ms

Figure 6-13 shows the benefits of *synchronisation* time in terms of end-to-end throughput and average hit time for EADP-d. When compared to that of EADP-d with ContikiMAC (Figure 6-12), EADP-d with ContikiMAC-Sync allowed a request message to reach four neighbours instead of three in the same amount of time using a CCI of 125ms. Figure 6-14 (a) quantifies the depicted performance under the same CCI and shows the average hit time of EADP-d as a function of the distance between the client and the service provider(s). As can be seen from this figure, EADP-d with ContikiMAC-Sync allowed a request to always hit the provider earlier than that of EADP-d with ContikiMAC with the maximum of 2/3 early hit time achieved at 9 hops.

To illustrate how this performance behaves when varying CCI, Figure 6-14 (b) keeps the distance fixed at 6 hops and depicts the average hit time when varying the CCI. As can be seen from Figure 6-14 (b), increasing CCI from 125ms to 500ms increased the gap in the average hit time between EADP-d with ContikiMAC and EADP-d with ContikiMAC-Sync. For instance, at a CCI of 500ms, the proposed optimisation allowed a request to hit the provider in less than half of the time registered by EADP-d with ContikiMAC.

Finally, while the results in Figure 6-14 explicitly address the average hit time, implicitly, they contain information about the EADP-d throughput. Thereby, achieving hits in less time implies that EADP-d with ContikiMAC-Sync can handle more requests in a particular period of time. For instance, achieving a hit time of less than half of that of EADP-d with ContikiMAC in Figure 6-14 (b), allows EADP-d with ContikiMAC-Sync to achieve twice the end-to-end throughput of EADP-d with ContikiMAC.

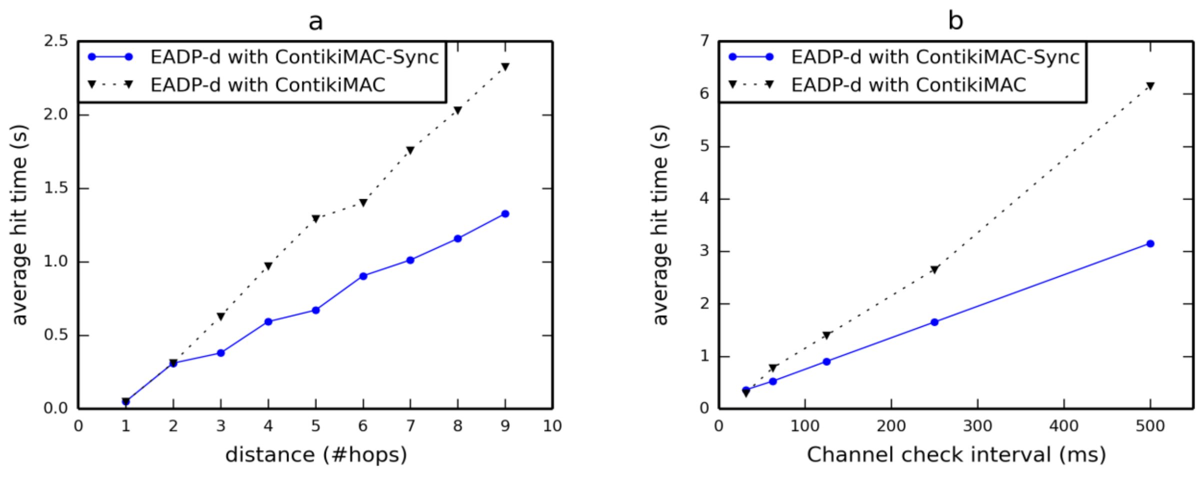


Figure 6-14 Average hit time of EADP-d with ContikiMAC-Sync

6.7 TrickleSD with MBF and synchronisation time

MBF and *synchronisation* time aim to provide generic mechanisms for supporting high throughput, low latency and low-power tasks required by multicast protocols in LLNs. Examples of protocols that can benefit from MBF and *synchronisation* time include MPL [80], EADP and TrickleSD. In the previous section, the impact of the *synchronisation* time on EADP's pull mode was shown. In this section, the performance of MBF and *synchronisation* time when used with TrickleSD's push mode is evaluated.

This experiment shows the impact of both MBF and *synchronisation* time (MBF-Sync) on TrickleSD's push mode. As mentioned earlier (section 6.6.1.4), push mode bursts are mainly caused by aggregating service descriptions leading to fragmentation. Thus, although the TrickleSD service agent ensures an upper bound on the number of service descriptions contained in an aggregated advertisement (section 4.7), a burst can always emerge, especially when verbose service descriptions are used.

The evaluation was carried out in the Indriya testbed. TrickleSD's pull mode was disabled, and a network-wide push mode was in place. PUSH_IMIN was set to 20 seconds, PUSH_IMAX to 160 seconds and PUSH_K to 1. Every node provides one service whose description get advertised and cached throughout the network. The parameter varied in this experiment is the size of a service description and the metric in focus is the average network duty cycle when using MBF-Sync and ContikiMAC. Each experiment ran for 30 minutes. Obtained results are depicted in Figure 6-15.

Figure 6-15 (a) shows the average network radio duty cycle consumed by TrickleSD for different service description sizes. The average radio duty cycle increased with increasing description sizes. However, while it showed a steep increase with ContikiMAC from around 2% radio duty cycle for descriptions of 20 bytes size to around 12% for descriptions of 80 bytes size, it only slightly increased when using MBF-Sync from 1.5% to around 2%. This could be explained by the fact that bigger service descriptions along with TrickleSD aggregation of multiple service entries in one advertisements allow MBF to save noticeable senders energy benefiting from shortening transmissions (section 6.6.1.1) and the gains brought by aggregation (section 6.6.1.4), while the *synchronisation* time allowed receivers to avoid wasting extra energy (section 6.6.1.5). ContikiMAC on the other hand lacking both mechanisms was unable to cope with frequent bursty traffic generated in this experiment without wasting a lot of energy. Finally, Figure 6-15 (b) shows how the energy consumption is distributed among the nodes involved in the experiment for a service description size of 80 bytes. Note that the node IDs in the x-axis of Figure 6-15 (b) are as numbered in the Indryia testbed [91] and the gaps mean that the corresponding node IDs were off when running this experiment.

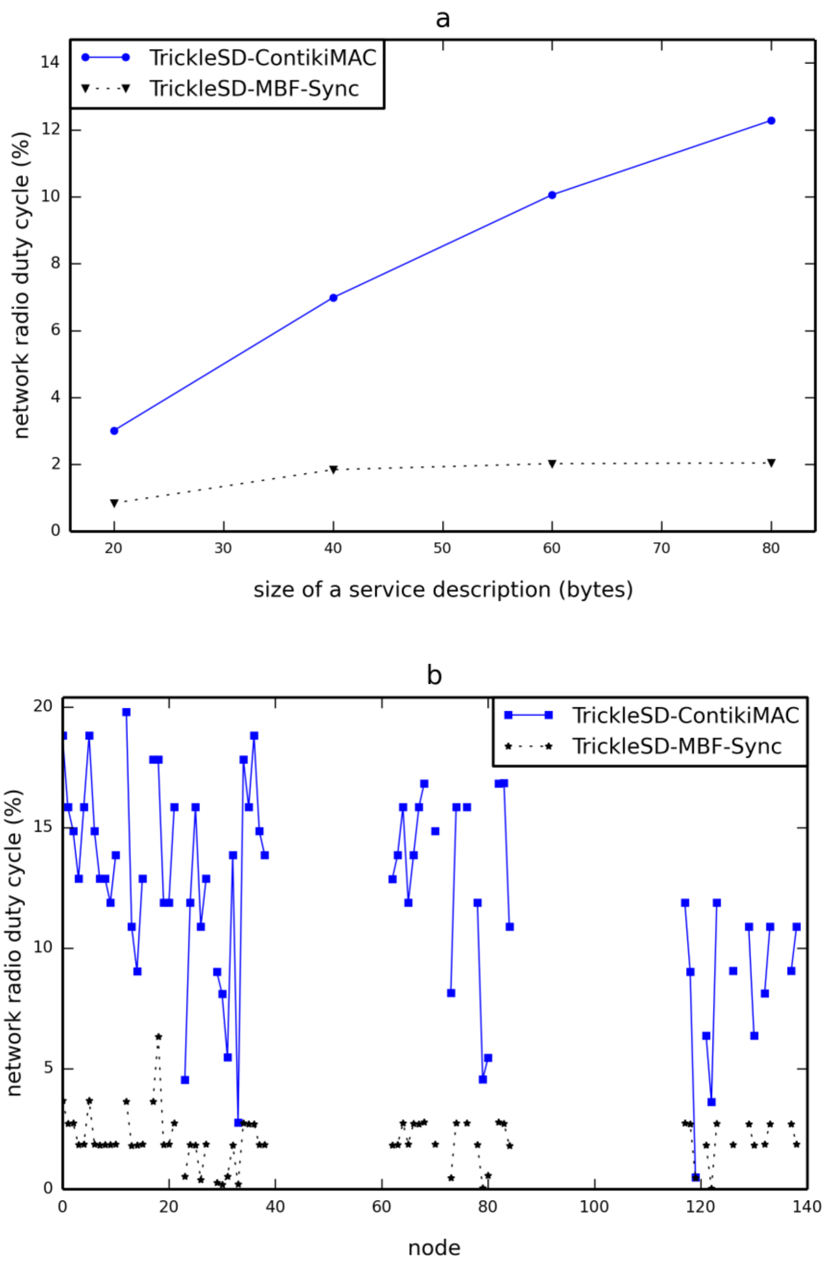


Figure 6-15 TrickleSD push mode performance in the Indriya testbed

6.8 Summary

This chapter investigated the effects of RDC on the performance of EADP and TrickleSD. It started by discussing broadcast handling under RDCs and extracting their advantages and drawbacks. Subsequently, two main contributions were introduced. The first proposed multicast burst forwarding: a mechanism that enables LLNs to respond to multicast burst requirements in terms of latency and throughput while leveraging RDCs to save energy. The second responded to a systematic problem with broadcast handling using data-strobes strategies.

These mechanisms were combined with ContikiMAC and showed important improvements. The performance of unicast/multicast communication patterns in duty-cycled 6LoWPAN networks was also analysed. Generally speaking, the introduced optimisations addressed the essential drawbacks of broadcast under RDCs; however, broadcast is still less efficient than unicast in terms of energy consumption and end-to-end delay. Finally, when used with EADP and TrickleSD, the proposed optimisations permitted the provision of attractive features for saving energy and time. The following chapter tackles the description and matchmaking component of EADP and TrickleSD and investigates ways to substitute some of their mechanisms using unicast.

Chapter 7

Standards-based Descriptions for TrickleSD Services

Previous chapters proposed EADP and TrickleSD –two SDPs optimised for CNN applications. So far, such protocols have focused on the dissemination part of SD and hence contributed various broadcast-based mechanisms for advertising and discovering available service information. This chapter presents the contributions on the service description and matchmaking part of the developed protocols. It opts for standards-based descriptions in order to foster seamless integration of CNNs with the Internet. To this end, integration of the previous solutions with two main service descriptions, namely DNS-SD and CoRE link format are investigated. Necessary changes and specific optimisations depending on the adopted service description along with the use of unicast instead of multicast, whenever possible, are also discussed. When directories are deployed, a possibility for using hybrid directory-based and directory-less approaches is also shown. Finally, a proof-of-concept implementation and evaluation of an EADP/DNS-SD integration is discussed.

7.1 Interoperable discovery operations

In the IoT, heterogeneous devices with different characteristics are expected to be interconnected. Figure 7-1 presents a general discovery task in the IoT. LLN nodes issue requests to locate suitable services which can be available locally or remotely over the Internet. On the other hand, local and remote non-LLN clients interested in LLN services find them by issuing requests to the LLN. Such an interoperable discovery is still a challenge today [173]. A first element to allow seamless operations in this vision resides on the adoption of well-established description technologies for CNNs [174]. This approach avoids maintenance and interoperability problems related to developing new descriptions. Furthermore, it leverages well-established, well-tested and well-understood technologies that respond to most of the requirements of SD. Examples of such technologies being considered in CNNs are DNS-SD and CoRE link format discussed in Chapter 3 (section 3.4).

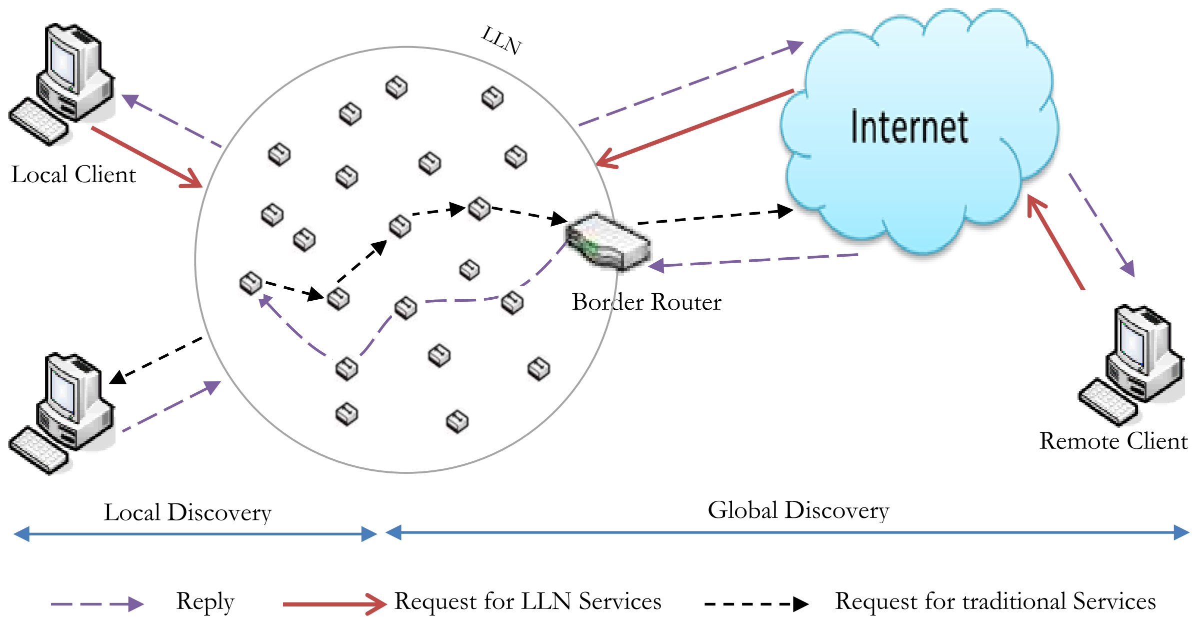


Figure 7-1 Interoperable service discovery in the IoT

So far, the matchmaking and description component of EADP and TrickleSD was deliberately left generic in order to allow their adoption by various formats and needs. This chapter proposes to complete the SD architecture by introducing coupling of EADP and TrickleSD with well-established descriptions that can be used in both local and global service discovery (Figure 7-1). Note that the study presented in this chapter has informed the design of the experiments of previous chapters, especially concerning the size of a service description entry. Finally, it is worth noting that while the following sections focus on TrickleSD integrations since it presents the optimised generic SD solution proposed in this research, they are equally applicable to EADP, which might be preferred in small very constrained networks. Indeed, the proof-of-concept evaluation discussed in section 7.5 is for integrations of DNS-SD with EADP.

7.2 DNS-SD for CNNs

As discussed in section 3.4.2.1, DNS-SD defines conventional usage of DNS messages and resource records to facilitate the discovery of services available in a network. It mainly specifies how a particular service instance can be described and accessed using the PTR, SRV, TXT and A/AAAA records. Table 7-1 presents an example of representative CNN service description using DNS-SD. The service considered in this table is a simple light service ($light1$), representing a type of CNN services available in street lighting, home automation and similar IoT applications.

Table 7-1 DNS-SD description of a light service

Record	Role	Usage in CNNs
PTR	assigns the instance $light1$ to the service $\_coap.\_udp$	$\_coap.\_udp\ IN\ PTR\ light1.\_coap.\_udp$
SRV	gives the target host and port of the service instance $light1$	$light1.\_coap.\_udp\ \ IN\ SRV\ 0\ 0\ 5683\ node1.local.$
TXT	key/value pairs convoying additional information. In this example, TXT contains the URI of the instance $light1$	$IN\ TXT\ path = /light/27$
AAAA	maps the hostname $node1$ providing the service instance $light1$ to an IPv6 address	$node1.local.IN\ AAAA\ fdfd::1234$

The following subsection discusses considerations of DNS-SD usage in constrained-node networks before introducing solutions for integrating DNS-SD with TrickleSD (the integrations are similarly applicable to EADP).

7.2.1 Considerations for DNS-SD usage in CNNs

Since clients search for DNS-SD services by requesting the PTR records of a $< service >.< domain >$ (section 3.4.2.1), the $\_subtype$ feature of the $service$ is very important when considering DNS-SD in CNNs. Indeed, the use of $\_subtype$ allows CNN clients to request for a narrower set of results. For instance, a selective query of subtype ($\_light.\_sub.\_coap.\_udp$) of the basic service type ($.\_coap.\_udp$) will only return

PTRs of light services. This feature is very important for a DNS-SD deployment in CNNs since many subtypes might exist in the network. Indeed, even with this feature, DNS-SD might still generate an abundant number of results in service-rich dense CNN deployments. Concerning the $domain$, as shown in section 3.4.2.1, it can be $.local$ when used in a local-scope CNN network. In multi-hop CNNs, the $domain\ .site$ may be used for a site-wide discovery as suggested in [138].

The multi-step discovery process of DNS-SD (section 3.4.2.1) might be very consuming for scares CNN resources and can prevent scalability if it is not controlled. To avoid such a process, RFC 6763 recommends using the *additional* section to include additional records believed to be subsequently requested by the client. For instance, the PTR response message can contain in its *additional* section SRV, TXT and AAAA records which are required to fully locate the service. However, including these records grows the size of the message and might exceed the MTU if DNS-SD is to be adopted for CNNs. Thus, some related work such as [134] recommend the disabling of this feature for CNNs, others such as [136] use it at the basis for optimisations. This feature is revised in the following section, and some recommendations are provided.

Finally, since mDNS, which allows DNS-SD usage for zero-configuration, plug-and-play operations is developed for traditional single-hop networks, a usage of DNS-SD with TrickleSD to provide zero-configuration operations for CNNs is introduced below.

7.2.2 TrickleSD with DNS-SD

The following subsections propose a usage of DNS-SD with TrickleSD. In this usage, the identification number in the DNS message header (Figure 3-9) is mapped directly to the $query\_seq$ number used by EADP and TrickleSD generic request message (section 4.2). This enables using the standard DNS message format as container of TrickleSD requests.

The proposed integrations of DNS-SD with TrickleSD are presented as a proof-of-concept for a first step into allowing a seamless integration of CNNs running TrickleSD/DNS-SD with traditional local networks running mDNS/DNS-SD and global Internet services using unicast DNS/DNS-SD. For the sake of ensuring backward compatibility, TrickleSD/DNS-SD adopts standardized DNS-SD messages for its requests and replies. However, to reduce traffic and adapt to CNN needs, TrickleSD/DNS-SD advertisements might be optimised. The optimised TrickleSD/DNS-SD advertisement uses the generic advertisement format described in section 4.2 which is not compatible with DNS messages. This, however, does not break compatibility with existing mDNS/DNS-SD implementation since if a node does not understand an advertisement it silently drops it.

Now, when used with DNS-SD services, TrickleSD's push mode can be tuned to only advertise the information necessary to facilitate DNS-SD interactions instead of exchanging verbose information that might not be used. Recommendations on how to use the push mode with DNS-SD are given in the following subsections. Note that many of the attempts into using DNS-SD for CNNs focus on single-hop networks [99], [134], [135] and result in designing lightweight mDNS implementations. TrickleSD can be used in such a case to replace components of mDNS. The following focuses on the generic case of multi-hop networks.

7.2.2.1 First solution

In this solution, the push mode can be used to only advertise PTR records with the AAAA records in the *additional* section. This way, nodes can find matching PTRs locally or from nearby nodes using the pull mode. Building on this idea, a node finding a PTR responding to its needs can retrieve the address of the provider from the *additional* section. When the address is retrieved, the client can achieve subsequent discovery stages via unicast which provides better reliability and time/cost performance under RDCs as shown in the previous chapter. Note that using unicast requires the availability of a routing protocol within the network.

7.2.2.2 Second solution

In this approach, a node takes full advantage of the push mode and the fact that much redundant information is present in the 4 DNS-SD records necessary for discovery (see Table 7-1). Such information is compacted, similarly to [136], in just one TrickleSD entry to be advertised. This case could be useful for the nodes to get the service information locally in a purely push mode, especially if the advertised service is of crucial importance. It could also be important when the number of services is smaller than the number of clients. The compact record format is included in the service field (*s*) of the generic format of an EADP and TrickleSD entry (Figure 4-3).

In both solutions and depending on the application, nodes might be allowed to respond on behalf of others or be prohibited from doing so. Prohibiting the nodes from responding on behalf of others has the drawback of not supporting the discovery of the services hosted by sleepy nodes. Note that if the push mode is totally disabled, using the pull mode to request for the PTR with the AAAA records and then relying on unicast instead of broadcast for the remaining discovery stages provide attractive features in terms of energy consumption, reliability and latency.

Finally, it should be noted that a new standardisation work has been recently started at the DNSSD working group which is expected to bring new optimisations to DNS-SD usage in CNNs [13]. The work in this chapter could provide input for the working group and future specifications of the working group might be used with TrickleSD.

7.3 TrickleSD with resource discovery

The CoRE working group defines specific discovery capabilities targeting CNNs. Those include a pull mode direct approach [124] and a centralised resource directory solution [112]. Both strategies use, by default, the CoRE link format (section 3.4.1) transported in CoAP messages (Figure 3-8) to enable discovery. However, many other formats can be used to describe resources and services deployed over CoAP. Examples include CBOR [175], JSON [176], and SenML [177]. This section shows how TrickleSD can be used with resource discovery. A method describing how TrickleSD can be used with the resource directory will be presented in section 7.4.

To realise resource discovery, CoAP relies on IP multicast. Thus, TrickleSD's pull mode can be used to substitute such a protocol in multi-hop networks, especially as MPL, the only currently considered multicast protocol for LLNs, has been criticised for its latency [178]. When used with CoAP messages, the field *query_seq* in the generic TrickleSD request message can be directly mapped to the message-ID field in a CoAP message (Figure 3-8), which allows TrickleSD's pull mode operations to work with resource discovery without the need to define new CoAP options. However, since CoAP request and response messages contain a Token value used to map a request-response pair (section 3.4.1), the $query\_seq$ field will be used only to detect potential request duplications.

The push mode of TrickleSD might be used to advertise the resource attributes necessary for efficient resource discovery. Such attributes might include the resource type (rt) along with the provider's IP address in order to achieve filtered unicast-based resource discovery. Finally, if nodes are authorized to respond on behalf of others, a node not having the advertised resource information in its local directory can find it from nearby nodes and hence minimise generated traffic and speed up the discovery process. This way, TrickleSD can be used to discover the services (endpoints) responding to particular need and the resource discovery mechanism integrated with CoAP will be used to get a fine-grained list of the resources hosted by a specific node via unicast.

7.4 Hybrid directory-based and directory-less discovery

While TrickleSD is developed as a zero-configuration solution that enables discovery in an ad hoc manner, its features allow it to be used as a hybrid protocol if local directories are deployed. For instance, both DNS-SD and CoRE envisage the usage of directories to manage large networks. For instance, a DNS-SD server might be deployed locally with a conventional DNS server to manage discovery in a configured network. Likewise, the CoRE working group is developing the resource directory (RD) solution for a similar purpose. In addition to facilitating local discovery, directory-based solutions, if available, can allow efficient global discovery throughout the Internet (Figure 7-1).

The default process to discover the presence of an RD in the network is to use the resource discovery mechanism described in section 3.4.1 [112]. Thus, nodes issue multicast requests to the "All CoAP Nodes" address [82] looking for the resource type $rt = rd$. Such a process has been shown to be inefficient since the overhead of discovering the RD is proportional to the number of nodes [99] and hence it grows linearly with network size (message complexity in $O(N)$, N being the number of non-RD nodes in the network). This costly multicast-based process gets aggravated under RDCs as discussed in the previous chapter. Furthermore, since the RD has to respond to each node's request separately, the number of responses also grows linearly with the number of nodes. TrickleSD's push mode provides an efficient solution to this problem. Thus, having the RD advertise its presence to the network using TrickleSD's push mode; the nodes will simply skip the discovery of the RD and start using unicast primitives to register with and/or look-up the RD. Indeed, using TrickleSD's push mode the overhead of discovering an RD is proportional to the number of RDs which is very small compared to the number of nodes (typically one RD in the network) and hence the message complexity is in $O(RD)$. In addition, this method saves all the traffic incurred by generating the responses since no requests/responses are exchanged.

Besides providing an efficient way to discover the RD, TrickleSD enables hybrid directory-based and directory-less discovery solutions. Thereby, a node might start by locating whether an RD is advertised, and, if so, it uses it for discovery. Otherwise, it issues a fully distributed lookup. Similarly, TrickleSD can be used for hybrid unicast/multicast DNS-SD, which is recently being considered in [179] for traditional IP networks.

7.5 Evaluation of EADP/DNS-SD integration

This proof-of-concept evaluation discusses results from an EADP/DNS-SD integration in Contiki OS. However, it is worth noting that by using specific TrickleSD configuration parameters, a TrickleSD execution could be mirrored by the results discussed below. Thus, a value of PULL_K = ∞ and PULL_EXPIRATIONS = 1 makes the TrickleSD execution fall to that of EADP.

7.5.1 Evaluation methodology

EADP/DNS-SD was implemented in Contiki and evaluated in Cooja using the third simulation scenario of a line topology network presented in the previous chapter (Figure 6-11). Two instances ($light1$ and $temp1$) of a service ($\_coap.\_udp$) were provided in the network. A client, placed at different distances from the provider, issues a DNS-SD lookup every 15 seconds by sending a PTR query for $\_coap.\_udp$. After getting the list of available instances, it chooses one and submits a query for its SRV record followed by another for TXT records and finally an AAAA query to resolve the provider's hostname to an IPv6 address. From the SD performance metrics defined in section 4.8.2, the following are of interest to this study:

- The average discovery time measured from sending the PTR query until receiving all the records necessary for discovery.
- The network duty cycle as indicator of the energy consumed by the evaluated protocols during simulation time.
- The average discovery success rate (it only considers the queries for which all the records necessary for discovery were received).

Since work on DNS-SD for CNNs is still in early stages, the existing related work only consider the case of single-hop networks resulting in optimised implementation of the mDNS/DNS-SD suite. Because this section evaluates a new usage of DNS-SD in multi-hop networks, the experiments compares two use-cases of the broadcast-based integration of EADP with DNS-SD. To this end, EADP/DNS-SD advertisements were governed by solution 2 proposed in section 7.2.2.2. Both EADP and EADP-d were evaluated with DNS-SD to see their benefits and drawbacks. The underlying RDC protocol used in this evaluation is ContikiMAC with a channel check rate of 16 Hz (giving a CCI of 62.5ms). The main configuration and simulation parameters used in this experiment are depicted in Table 7-2.

Table 7-2 EADP/DNS-SD simulation parameters (scenario #3)

Configuration parameter	**Value**
Duration of one simulation/ #iterations / #nodes	360s / 1 / 10
Medium / range / Throughput	UDGM / 50m / 250kbps
PUSH_K	1
PUSH_IMIN/PUSH_IMAX	10s / 80s
REQUEST_RETRANSMISSION_COUNTER	0
REQUEST_DISK / ADVERTISEMENT_DISK	6 / 5
Underlying routing protocol	RPL
RDC / MAC / Adaptation	ContikiMAC / CSMA-CA / 6LoWPAN

7.5.2 Results and discussions

The average discovery time, network radio duty cycle and the average success rates of EADP-d/DNS-SD and EADP/DNS-SD protocols when varying the distance between the client and the service provider are depicted in Figure 7-2.

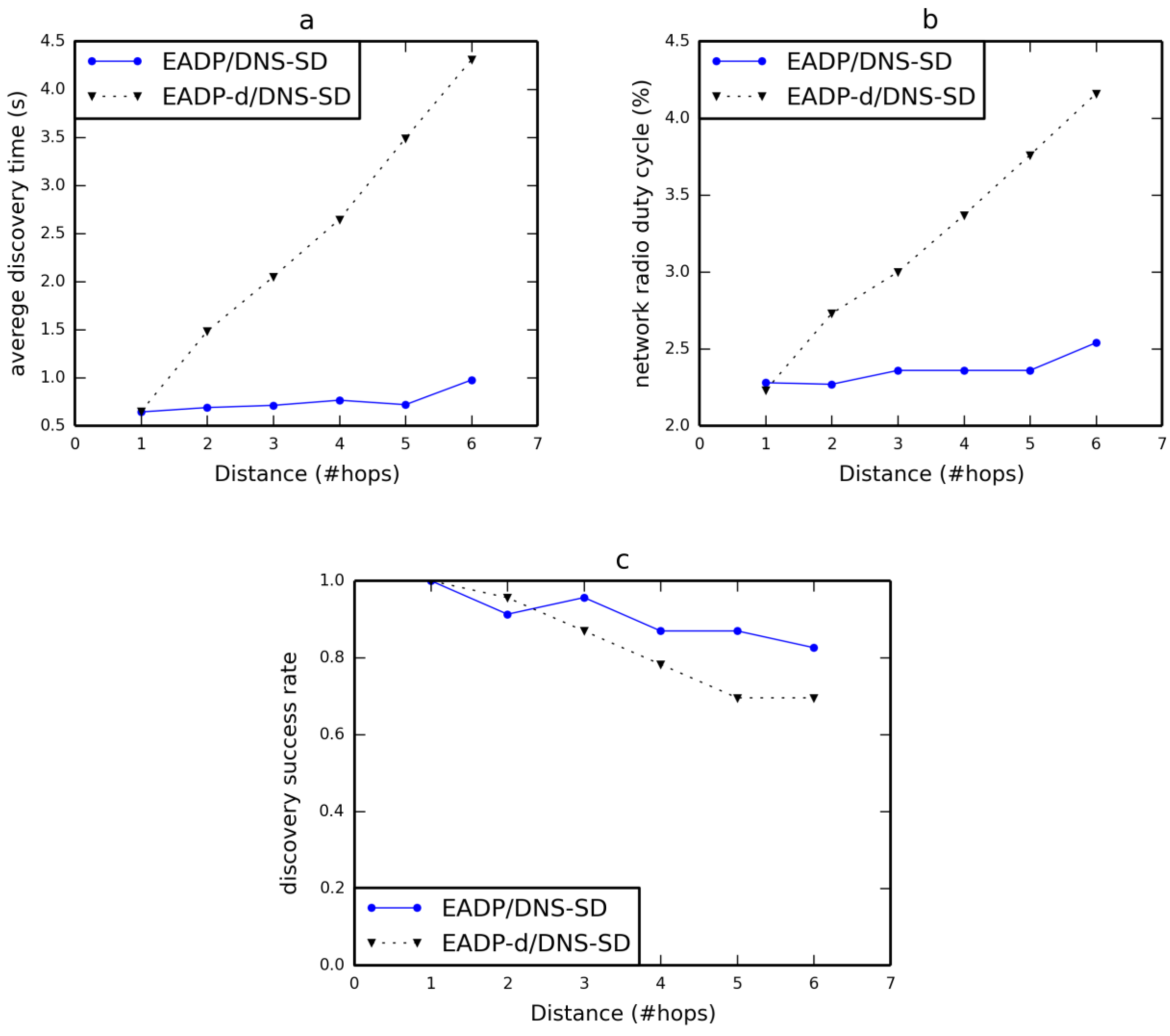


Figure 7-2 Simulation results of EADP/DNS-SD

As can be seen from Figure 7-2 (a), the average discovery time increases with distance, however while EADP-d/DNS-SD keeps it increasing as a result of the multi-step DNS-SD discovery process spanning long distances, EADP/DNS-SD keeps the discovery time relatively constant at a low value, which provides a stable user experience regardless of the service location. In addition, Figure 7-2 (b) shows a noticeable reduction in network energy consumption for EADP/DNS-SD. This is achieved thanks to advertising compact messages of available service instances, which minimised distances travelled by verbose requests and replies. This figure may change with frequent network churn, which triggers more advertisements; however, because of advertisements' compact format, it can be argued that a good gain in energy over time can be registered.

Finally, Figure 7-2 (c) shows that the discovery success rate decreases with increasing distance. When compared with EADP-d/DNS-SD, EADP/DNS-SD registered a better discovery success rate. This can be explained by the fact of reducing the distances travelled by lookups and responses, which minimises the probability of losing messages.

7.5.3 Code size discussion

The protocols and algorithms developed in this research are designed to fit in constrained devices, especially targeting to be implemented in Class 1 constrained devices (section 2.2.2). All the prototype implementations and evaluations throughout this research project were implemented in the Tmote Sky mote platform and its variants, a well-known representative hardware for Class 1 constrained devices. Hence, all the algorithms can fit the constrained devices of Class 1 and hence can work with liberty of space on Class 2 devices.

7.6 Summary

This chapter presented proof-of-concept integrations of EADP and TrickleSD with DNS-SD and CoRE link format; two well-known, standards-based service description formats. It also showed how to take advantage from such descriptions to substitute some of TrickleSD's broadcast primitives by unicast ones, which are less costly in duty-cycled CNNs. Furthermore, the potential of using TrickleSD to achieve efficient discovery of directories or hybrid directory-based and directory-less SD was shown. Proof-of-concept results of an EADP/DNS-SD integration are promising. However, since DNS-SD was not designed with CNN constraints in mind, more investigations are required to fully develop a deployable, interoperable solution based on DNS-SD. Indeed, beside the recently started investigations at the DNSSD working group, ideas for providing query filtering to DNS-SD are being explored in [180]. Finally, it is worth noting that EADP and TrickleSD are not tailored to any service description, and their component-based architecture allows them to be used with a multitude of service description formats that shape their operations.

Chapter 8

Conclusions and Future Work

This chapter summarises the contributions of this thesis. It briefly shows how the previous chapters address the problem of pervasive SD in LLNs. Limitations of the proposed approaches are discussed, and potential avenues for future research are highlighted.

8.1 Summary of contributions

This research proposed new service discovery approaches and protocols aimed at realising zero-configuration low-power operations in the IoT. The development and design of such solutions were guided by the challenges introduced in Chapter 1 and the requirements extracted in section 3.5.

Having identified the gap in service discovery literature for 6LoWPAN networks (Chapter 3), the author designed EADP; an Extensible, Adaptable hybrid push-pull Discovery Protocol for 6LoWPANs. EADP contributed a new Trickle variant along with many interesting mechanisms regarding time efficiency and response to network dynamics detailed and discussed in Chapter 4. Its performance was formally analysed and extensively evaluated in the same chapter. EADP showed important performance achievements concerning both discovery latency and push mode generated traffic. However, while EADP presented many attractive features, it is far from being optimal. Particularly, EADP's pull mode inefficiencies in terms of generated cost and its push mode time efficiencies required enhancement. Therefore, EADP was optimised in Chapter 5.

Building on the above, Chapter 5 proposed three main contributions. It proposed an optimisation for the well-known Trickle algorithm that addresses its main drawbacks concerning latency while preserving its scalability. Such an optimisation enables further expansion of its reach and allows it to be used for SD while responding to the requirements of time efficiency. Based on this optimisation, a second contribution, Augmented Trickle, proposed methods to bias Trickle's random transmission time selection process with metrics freely available either from the received packet such as RSSI and LQI, or by using neighbourhood information collected by the 6LoWPAN-ND. These optimisations formed the building blocks for proposing a Trickle-based pull mode algorithm to replace the EADP's flooding-based one. The chapter also incorporated optimised Trickle in both the push mode and state maintenance mechanisms of the EADP protocol which gave birth to the TrickleSD protocol. The optimisations along with TrickleSD were thoroughly analysed, evaluated and discussed in the same chapter. The extensive evaluation included both cycle-accurate simulations and public large-scale testbed experiments.

Being based on broadcast communication, the above protocols could suffer inefficiencies in duty-cycled networks. To respond to this, Chapter 6 was set apart to investigate the performance of broadcast under duty-cycling mechanisms used in LLNs to achieve better energy budget thereby extending the network lifetime. Thus, besides proposing a comprehensive analysis of energy consumption and latencies, two main contributions were proposed in Chapter 6 in order to enhance broadcast communication in radio duty-cycled networks. The first contribution addressed the problem of multicast burst forwarding while the latter responded to a systematic problem with a class of widely deployed RDC protocols. Both cycle-accurate and local testbed experiments were carried out to assess their performance. The contribution showed important time/cost enhancements compared with existing schemes. However, they are still far from achieving unicast performance. Potential ideas for exploiting unicast to the greatest extent possible should be considered.

While pursuing such ideas and attempting to complete the proposed solutions with interoperable descriptions, the previous chapter proposed integrations of TrickleSD with two well-known description formats that would foster seamless integration of CNNs in the Internet of things. It also showed how to take advantages of their features in order to substitute some TrickleSD broadcast primitives with unicast ones and thereby benefit from the unicast performance demonstrated in Chapter 6. The previous chapter also proposed methods for using TrickleSD in order to achieve hybrid directory-based and directory-less discovery tasks using RD or DNS-SD servers whenever available.

Finally, the proposed contributions were implemented and evaluated in major CNN operating systems namely TinyOS (particularly parts of Chapter 5) and Contiki OS as the main development platform. EADP, TrickleSD and the integration with DNS-SD were evaluated in three representative trending IoT application scenarios representing: (i) emergency response and similar application scenarios (Chapter 4); (ii) home automation systems and similar applications (Chapter 5); (iii) street lighting and vehicular network scenarios (Chapter 6 and 7); and (iv) a large-scale publicly available testbed (Chapter 6). The proposed generic Opt-Trickle algorithm deployed in TrickleSD was extensively evaluated in large scale simulations including 400 nodes in both single- and multi-hop network scenarios along with the large scale Indryia testbed in order to show that it does not violate Trickle assumptions in accordance with RFC 6206. The MBF mechanism was separately evaluated in single-hop networks in both simulation and local testbed experiments and was evaluated when integrated with TrickleSD and synchronisation time in the large scale Indryia testbed. Added to this last evaluation, the synchronisation time idea was also evaluated with EADP in Chapter 6. Finally, all the above contributions showed that they could achieve lightweight memory footprint that fits the constraints of today's Class 1 devices.

8.2 Broader impact

This thesis has contributed another small step towards zero-configuration, plug-and-play IoT applications via the EADP and TrickleSD protocols. This is expected to stimulate new research in the field especially since plug-and-play IoT functionalities are estimated to be widely deployed beyond 2025 [83]. Hence, the mechanisms and techniques developed in this research are expected to go beyond addressing the problem of pervasive service discovery in constrained environments. They have a broader impact and can be used to address a multitude of similar problems. For instance, the proposed optimisations concerning the Trickle algorithm are generic enough to be able to benefit all Trickle-based applications by modifying only a single line of their codes. In particular, the two Internet standards based on Trickle (RPL and MPL) can be greatly enhanced by deploying the Trickle optimisations developed in this research. New use-cases of Trickle are envisaged, for instance, with Opt-Trickle, it is now possible to replace flooding used in multitude routing protocols including the reactive routing protocol being considered for LLNs, LOADng [149]. Likewise, the contributions at the RDC layer could be applied to RDC protocols in a generic manner and might open doors for new solutions in the field of radio duty cycling.

8.3 Future directions

This research demonstrated the feasibility of pervasive service discovery in LLNs. Many enhancements and techniques are envisaged including:

- **Message compression:** Very recently the IETF has standardised the Generic Header (and header-like) Compression (GHC) format for 6LoWPAN networks [181]. Investigations for using GHC to compress TrickleSD messages, especially when coupled with verbose DNS-SD, are planned for future work.
- **Network traffic reduction:** Looking for ways to minimise unproductive traffic by avoiding replying to known answers is a potential method for reducing generated traffic. In addition, enhanced algorithms for avoiding broadcast reply storms should be considered. Methods to make the push mode context-aware would greatly reduce the amount of unproductive traffic. For instance, allowing only the most popular services to be advertised lead to a significant saving in unproductive overhead.
- **Radio duty cycling:** In order to ensure robustness, timeliness and better energy conservation, ways of looking at multichannel operations for broadcast communications are planned. Working on a reliable multicast burst forwarding mechanism to enhance MBF's reliability is also envisaged. Finally, techniques for combining data-strobes with chirp-based strobes in the same RDC protocol would greatly enhance existing RDC protocols.
- **In-depth investigations of service descriptions:** A proof-of-concept integration, implementation and evaluation of EADP and TrickleSD with DNS-SD were presented in Chapter 7. In addition, integrations with CoRE link format have been also discussed in the same chapter. However, to fully understand and

propose better integrations, future work in this direction should be carried out. For instance, looking for a better naming service that fits CNN constraints should provide an interesting direction to be investigated. Consideration of other description formats is also planned.

- **Formal analysis:** Although some analysis of the message and time complexity of EADP and Opt-Trickle were made, more analysis is required to fully formulate the time/cost behaviours of the proposed algorithms and analytically prove the performance of TrickleSD.
- **Securing TrickleSD:** TrickleSD is intended to operate on local LLN networks. To be successfully deployed, authentication, authorisation and security mechanisms should be integrated. The IETF has recently chartered two working groups [182], [183] addressing such issues in constrained environments. The outcomes of such works might be exploited to secure TrickleSD.
- **Semantic TrickleSD:** Exploiting the full potential of the service and node contexts in ubiquitous environments to personalise TrickleSD services is another attractive feature for future investigations.
- **Work on Trickle:** The Opt-Trickle algorithm showed noticeable latency improvements over Trickle. However, it might lack the wavelike propagation pattern of Trickle. Investigations into how to enhance Opt-Trickle along with investigations of new techniques for Augmented Trickle have been already started and preliminary results are promising.

8.4 Concluding remarks

Supporting zero-configuration IoT interactions is emerging as a promising paradigm for the future. Indeed, if IoT technologies are to be widely deployed, they should be made so easy for ordinary people that they can go buy a 'thing' and it will work 'out-of-the-box', without the need for the user to carry out any further configuration. The results of this research set foundations for such a direction and showed that zero-configuration plug-and-play operations in the IoT are feasible. However, fully achieving such a vision and bringing it to life requires cross-disciplinary efforts and a shared desire to realise a truly felt Internet of things.

REFERENCES

[1] M. Weiser and J. S. Brown, 'The coming age of calm technology', in *Beyond calculation*, Springer, 1997, pp. 75–85.

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

[3] J.-P. Vasseur and A. Dunkels, *Interconnecting smart objects with IP: the next Internet*. Burlington, MA: Morgan Kaufmann Publishers/Elsevier, 2010.

[4] *IEEE standard for local and metropolitan area networks. Part 15.4*, IEEE Std 802.15.4™-2011. New York: Institute of Electrical and Electronics Engineers, 2011.

[5] S. Deering and R. Hinden, 'Internet Protocol, Version 6 (IPv6) Specification', *RFC 2460, IETF*, Dec. 1998.

[6] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, 'Transmission of IPv6 Packets over IEEE 802.15.4 Networks', *RFC 4944, IETF*, Sep. 2007.

[7] B. Guo, D. Zhang, Z. Yu, Y. Liang, Z. Wang, and X. Zhou, 'From the internet of things to embedded intelligence', *World Wide Web*, vol. 16, no. 4, pp. 399–420, Sep. 2012.

[8] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*. Chichester, U.K.: J. Wiley, 2009.

[9] 'More than 50 billion connected devices', Ericsson, white paper 284 23-3149 Uen, Feb. 2011.

[10] T. Erl, *SOA: Principles of Service Design*, 1st edition., vol. 1. Upper Saddle River: Prentice Hall, 2008.

[11] J. Zhang, B. Iannucci, M. Hennessy, K. Gopal, S. Xiao, S. Kumar, D. Pfeffer, B. Aljedia, Y. Ren, M. Griss, S. Rosenberg, J. Cao, and A. Rowe, 'Sensor Data as a Service – A Federated Platform for Mobile Data-centric Service Development and Sharing', in *Proceedings of the 2013 IEEE International Conference on Services Computing*, Washington, DC, USA, 2013, pp. 446–453.

[12] 'Constrained RESTful Environments (core) - Charter'. [Online]. Available: https://datatracker.ietf.org/wg/core/charter/. [Accessed: 25-May-2015].

[13] 'Extensions for Scalable DNS Service Discovery (dnssd) - Charter'. [Online]. Available: https://datatracker.ietf.org/wg/dnssd/charter/. [Accessed: 25-May-2015].

[14] 'Bonjour Overview: Bonjour Concepts'. [Online]. Available: http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/NetServices/Articles/about.html. [Accessed: 25-May-2015].

[15] 'OASIS Web Services Dynamic Discovery (WS-Discovery) Version 1.1'. [Online]. Available: http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html#_Toc234231817. [Accessed: 25-May-2015].

[16] 'UPnP APIs (Windows)'. [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/aa382303(v=vs.85).aspx. [Accessed: 25-May-2015].

[17] Yaron Y. Goland, Ting Cai, Paul Leach, Ye Gu, and Shivaun Albright, 'Simple Service Discovery Protocol', *Internet Draft, IETF*, Oct. 1999.

[18] E. Guttman, C. Perkins, J. Veizades, and M. Day, 'Service Location Protocol, Version 2', *RFC 2608, IETF*, Jun. 1999.

[19] 'UDDI Version 3.0.2'. [Online]. Available: https://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm. [Accessed: 27-May-2015].

[20] D. Singh, U. S. Tiwary, H.-J. Lee, and W.-Y. Chung, 'Global Healthcare Monitoring System Using 6Lowpan Networks', in *Proceedings of the 11th International Conference on Advanced Communication Technology - Volume 1*, Piscataway, NJ, USA, 2009, pp. 113–117.

[21] A. Kovacevic, J. Ansari, and P. Mahonen, 'NanoSD: A Flexible Service Discovery Protocol for Dynamic and Heterogeneous Wireless Sensor Networks', in *2010 Sixth International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, 2010, pp. 14–19.

[22] M. Becker, 'Modelling and Optimisation for the Efficient Discovery of Services in Wireless Sensor Networks', PhD, TZI, University Bremen, 2014.

[23] P. Levis, N. Patel, D. Culler, and S. Shenker, 'Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks', in *In Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, 2004, pp. 15–28.

[24] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, 'The Trickle Algorithm', *RFC 6206, IETF*, 2011.

[25] Thanh Dang, Kaisen Lin, Chieh-Jan Mike Liang, and Omprakash Gnawali, 'The Net2 Protocol Benchmark', *TinyOS Enhancement Proposals (TEP)*, 2010.

[26] Vasseur, JP., 'Terms Used in Routing for Low-Power and Lossy Networks', *RFC 7102, IETF*, 2014.

[27] C. Bormann, M. Ersue, and A. Keranen, 'Terminology for Constrained-Node Networks', *RFC 7228, IETF*, May 2014.

[28] '21 Ideas for the 21st Century'. [Online]. Available: http://www.businessweek.com/1999/99_35/2121_content.htm. [Accessed: 25-May-2015].

[29] '10 Emerging Technologies That Will Change the World - MIT Technology Review'. [Online]. Available: http://www2.technologyreview.com/featured-

story/401775/10-emerging-technologies-that-will-change-the/3/. [Accessed: 25-May-2015].

[30] Zach Shelby and Martti Huttunen, '6LoWPAN: The Wireless Embedded Internet (Companion Exercise Slides)', 2010.

[31] 'DASH7 Alliance'. [Online]. Available: http://www.dash7-alliance.org/. [Accessed: 25-May-2015].

[32] 'Z-Wave : Home control'. [Online]. Available: http://www.z-wave.com/. [Accessed: 25-May-2015].

[33] 'Bluetooth Specification Adopted Documents'. [Online]. Available: https://www.bluetooth.org/en-us/specification/adopted-specifications. [Accessed: 25-May-2015].

[34] 'EnOcean Alliance'. [Online]. Available: http://www.enocean-alliance.org/en/home/. [Accessed: 25-May-2015].

[35] IEEE Communications Society, Power Line Communications Standards Committee, Institute of Electrical and Electronics Engineers, and IEEE-SA Standards Board, *IEEE standard for low-frequency (less than 500 kHz) narrowband power line communications for smart grid applications*. 2013.

[36] J. Nieminen, T. Savolainen, M. Isomaki, B. Patil, Z. Shelby, and C. Gomez, 'Transmission of IPv6 Packets over BLUETOOTH Low Energy', *Internet Draft, IETF*, May 2015.

[37] A. Brandt and J. Buron, 'Transmission of IPv6 Packets over ITU-T G.9959 Networks', *RFC 7428, IETF*, Feb. 2015.

[38] 'IEEE 802.15 WPAN™ Task Group 4 (TG4)'. [Online]. Available: http://ieee802.org/15/pub/TG4.html. [Accessed: 29-May-2015].

[39] L. Benini, E. Farella, and C. Guiducci, 'Wireless sensor networks: Enabling technology for ambient intelligence', *Microelectronics Journal*, vol. 37, no. 12, pp. 1639–1649, Dec. 2006.

[40] Patrick Kinney, 'IEEE 802.15.4 Tutorial', *ZigBee Alliance*, 19-Nov-2003. [Online]. Available: https://docs.zigbee.org/zigbee-docs/dcn/03-1323.pdf. [Accessed: 25-May-2015].

[41] *IEEE standard for local and metropolitan area networks. Part 15.4, Amendment 1: MAC sublayer*. New York: Institute of Electrical and Electronics Engineers, 2012.

[42] *IEEE standard for local and metropolitan area networks. Part 15.4, Amendment 3: Physical Layer (PHY) Specifications for Low-Data-Rate, Wireless, Smart Metering Utility Networks*. New York: Institute of Electrical and Electronics Engineers, 2012.

[43] 'Texas Instruments CC2420 Datasheet'. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2420.pdf. [Accessed: 25-May-2015].

[44] Y. Xu, J. Heidemann, and D. Estrin, 'Geography-informed Energy Conservation for Ad Hoc Routing', in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, New York, NY, USA, 2001, pp. 70–84.

[45] J. W.-Y. Hui, 'An Extended Internet Architecture for Low-power Wireless Networks: Design and Implementation', PhD, University of California, Berkeley, 2008.

[46] A. Dunkels, L. Mottola, N. Tsiftes, F. Österlind, J. Eriksson, and N. Finne, 'The announcement layer: Beacon coordination for the sensornet stack', in *Wireless sensor networks*, Springer, 2011, pp. 211–226.

[47] W. Ye, J. Heidemann, and D. Estrin, 'An energy-efficient MAC protocol for wireless sensor networks', in *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002, vol. 3, pp. 1567–1576.

[48] T. van Dam and K. Langendoen, 'An Adaptive Energy-efficient MAC Protocol for Wireless Sensor Networks', in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2003, pp. 171–180.

[49] R. Musaloiu-E., C.-J. M. Liang, and A. Terzis, 'Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks', 2008, pp. 421–432.

[50] Y. Sun, O. Gurewitz, and D. B. Johnson, 'RI-MAC: A Receiver-initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks', in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, New York, NY, USA, 2008, pp. 1–14.

[51] J. Polastre, J. Hill, and D. Culler, 'Versatile low power media access for wireless sensor networks', in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 95–107.

[52] S. Unterschütz, C. Renner, and V. Turau, 'Opportunistic, receiver-initiated data-collection protocol', in *Proceedings of the 9th European conference on Wireless Sensor Networks*, Berlin, Heidelberg, 2012, pp. 1–16.

[53] M. Buettner, G. V. Yee, E. Anderson, and R. Han, 'X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks', in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2006, pp. 307–320.

[54] A. El-Hoiydi and J.-D. Decotignie, 'WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks', in *Algorithmic Aspects of Wireless Sensor Networks*, S. E. Nikoletseas and J. D. P. Rolim, Eds. Springer Berlin Heidelberg, 2004, pp. 18–31.

[55] D. Moss and P. Levis, 'BoX-MACs: Exploiting physical and link layer boundaries in low-power networking', *Computer Systems Laboratory Stanford University*, pp. 116–119, 2008.

[56] S. Duquennoy, F. Österlind, and A. Dunkels, 'Lossy links, low power, high throughput', in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, 2011, pp. 12–25.

[57] 'ZigBee Alliance'. [Online]. Available: http://www.zigbee.org/. [Accessed: 25-May-2015].

[58] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, 'WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control', 2008, pp. 377–386.

[59] 'ISA100.11a Technology Standard'. [Online]. Available: http://www.nivis.com/technology/ISA100.11a.php. [Accessed: 27-May-2015].

[60] J. Hui and P. Thubert, 'Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks', *RFC 6282, IETF*, 2011.

[61] C. Gomez, E. Kim, D. Kaspar, and C. Bormann, 'Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing', *RFC 6606, IETF*, May 2012.

[62] T. Narten, E. Nordmark, W. A. Simpson, and H. Soliman, 'Neighbor Discovery for IP version 6 (IPv6)', *RFC 4861, IETF*, Sep. 2007.

[63] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann, 'Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)', *RFC 6775, IETF*, Nov. 2012.

[64] 'TinyOS Home Page'. [Online]. Available: http://www.tinyos.net/. [Accessed: 25-May-2015].

[65] 'Contiki: The Open Source Operating System for the Internet of Things'. [Online]. Available: http://www.contiki-os.org/index.html. [Accessed: 25-May-2015].

[66] Reza Khoshdelniat, '6LoWPAN Applications and Internet of Things', presented at the 30th APAN Meeting, Hanoi, Vietnam, 09-Aug-2010.

[67] G. Mulligan, C. Williams, and D. Huo, 'Mobility considerations for 6LoWPAN', *Internet Draft, IETF*, 2010.

[68] C. E. Perkins, D. B. Johnson, and J. Arkko, 'Mobility Support in IPv6', *RFC 6275, IETF*, Jul. 2011.

[69] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert, 'Network Mobility (NEMO) Basic Support Protocol', *RFC 3963, IETF*, 2005.

[70] 'IPv6 over Networks of Resource-constrained Nodes (6lo) - Documents'. [Online]. Available: https://datatracker.ietf.org/wg/6lo/documents/. [Accessed: 25-May-2015].

[71] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander, 'RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks', *RFC 6550, IETF*, Mar. 2012.

[72] M. Goyal, E. Baccelli, M. Philipp, A. Brandt, and J. Martocci, 'Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks', *RFC 6997, IETF*, Aug. 2013.

[73] J. Ko, A. Terzis, S. Dawson-Haggerty, D. E. Culler, J. W. Hui, and P. Levis, 'Connecting low-power and lossy networks to the internet', *Communications Magazine, IEEE*, vol. 49, no. 4, pp. 96–101, 2011.

[74] P. Levis, E. Brewer, D. Culler, D. Gay, S. Madden, N. Patel, J. Polastre, S. Shenker, R. Szewczyk, and A. Woo, 'The emergence of a networking primitive in wireless sensor networks', *Communications of the ACM*, vol. 51, no. 7, pp. 99–106, 2008.

[75] 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*, 2009, pp. 1–14.

[76] A. Chlipala, J. Hui, and G. Tolle, 'Deluge: data dissemination for network reprogramming at scale', *University of California, Berkeley, Tech. Rep*, 2004.

[77] K. Lin and P. Levis, 'Data discovery and dissemination with dip', in *Proceedings of the 7th international conference on Information processing in sensor networks*, 2008, pp. 433–444.

[78] G. Tolle and D. Culler, 'Design of an application-cooperative management system for wireless sensor networks', in *Proceeedings of the Second European Workshop on Wireless Sensor Networks.*, 2005, pp. 121–132.

[79] T. Dang, N. Bulusu, W.-C. Feng, and S. Park, 'Dhv: A code consistency maintenance protocol for multi-hop wireless sensor networks', in *Wireless Sensor Networks*, Springer, 2009, pp. 327–342.

[80] J. Hui and R. Kelsey, 'Multicast Protocol for Low power and Lossy Networks (MPL)', *Internet Draft, IETF*, 2014.

[81] R. T. Fielding, 'Architectural styles and the design of network-based software architectures', PhD, University of California, Irvine, 2000.

[82] Z. Shelby, K. Hartke, and C. Bormann, 'The Constrained Application Protocol (CoAP)', *RFC 7252, IETF*, 2014.

[83] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, 'Internet of Things (IoT): A vision, architectural elements, and future directions', *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[84] 'IPSO Alliance'. [Online]. Available: http://www.ipso-alliance.org/. [Accessed: 25-May-2015].

[85] 'The Thread Group'. [Online]. Available: http://threadgroup.org/Home.aspx. [Accessed: 25-May-2015].

[86] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, and E. Brewer, 'TinyOS: An operating system for sensor networks', in *Ambient intelligence*, Springer, 2005, pp. 115–148.

[87] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler, 'The nesC language: A holistic approach to networked embedded systems', in *Acm Sigplan Notices*, 2003, vol. 38, pp. 1–11.

[88] P. Levis, N. Lee, M. Welsh, and D. Culler, 'TOSSIM: Accurate and scalable simulation of entire TinyOS applications', in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 126–137.

[89] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, 'Cross-level sensor network simulation with cooja', in *31st IEEE Conference on Local Computer Networks*, 2006, pp. 641–648.

[90] J. Polastre, R. Szewczyk, and D. Culler, 'Telos: enabling ultra-low power wireless research', in *Fourth International Symposium on Information Processing in Sensor Networks, 2005. IPSN 2005*, 2005, pp. 364–369.

[91] 'INDRIYA: A Wireless Sensor Network Testbed'. [Online]. Available: http://indriya.comp.nus.edu.sg/motelab/html/motes-info.php. [Accessed: 07-May-2015].

[92] Abdelmounaam Rezgui, 'Service-Oriented Sensor-Actuator Networks', PhD, Blacksburg, Virginia, USA, 2007.

[93] J. Leguay, M. Lopez-Ramos, K. Jean-Marie, and V. Conan, 'An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks', in *33rd IEEE Conference on Local Computer Networks (LCN)*, 2008, pp. 740 –747.

[94] M. S. Thompson, 'Service discovery in pervasive computing environments', PhD, Blacksburg, Virginia, USA, 2006.

[95] E. Meshkova, J. Riihijärvi, M. Petrova, and P. Mähönen, 'A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks', *Computer Networks*, vol. 52, no. 11, pp. 2097–2128, 2008.

[96] S. H. Chauhdary, M. Cui, J. H. Kim, A. K. Bashir, and M.-S. Park, 'A Context-Aware Service Discovery Consideration in 6LoWPAN', in *Third International Conference on Convergence and Hybrid Information Technology (ICCIT)*, 2008, vol. 1, pp. 21 –26.

[97] F. M. Anwar, M. T. Raza, S.-W. Yoo, and K.-H. Kim, 'ENUM Based Service Discovery Architecture for 6LoWPAN', in *2010 IEEE Wireless Communications and Networking Conference (WCNC)*, 2010, pp. 1 –6.

[98] X. Wang and H. Huang, 'A Service Model for 6LoWPAN Wireless Sensor Networks', *International Journal of Distributed Sensor Networks*, vol. 2013, Jun. 2013.

[99] B. Villaverde, R. Alberola, A. Jara, S. Fedor, S. 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.

[100] J. Su and W. Guo, 'A survey of service discovery protocols for mobile ad hoc networks', in *International Conference on Communications, Circuits and Systems (ICCCAS)*, 2008, pp. 398 –404.

[101] A. N. Mian, R. Baldoni, and R. Beraldi, 'A Survey of Service Discovery Protocols in Multihop Mobile Ad Hoc Networks', *IEEE Pervasive Computing*, vol. 8, no. 1, pp. 66 –74, Mar. 2009.

[102] F. M. Anwar, S.-W. Yoo, and K.-H. Kim, 'Survey on service discovery for Wireless Sensor Networks', in *2010 Second International Conference on Ubiquitous and Future Networks (ICUFN)*, 2010, pp. 17 –21.

[103] Huaglory Tianfield, 'Context-Aware Service Discovery in Pervasive Environments: A Survey', *International Transactions on Systems Science and Applications*, vol. 7, no. 3/4, pp. 314–338, Dec. 2011.

[104] M. Heni and R. Bouallegue, 'Adaptive service discovery and proactive routing protocol for Mobile Ad hoc Network', in *Mediterranean Microwave Symposium (MMS), 2011 11th*, 2011, pp. 193 –196.

[105] M. Skjegstad, F. T. Johnsen, and J. Nordmoen, 'An emulated test framework for service discovery and manet research based on ns-3', in *5th International Conference on New Technologies, Mobility and Security (NTMS)*, 2012, pp. 1–5.

[106] C. N. Ververidis and G. C. Polyzos, 'A routing layer based approach for energy efficient service discovery in mobile ad hoc networks', *Wireless Communications and Mobile Computing*, vol. 9, no. 5, pp. 655–672, May 2009.

[107] F. Sailhan and V. Issarny, 'Scalable Service Discovery for MANET', presented at the International Conference on Pervasive Computing and Communications (PerCom), 2005, pp. 235–244.

[108] 'SOA Manifesto'. [Online]. Available: http://www.soa-manifesto.org/. [Accessed: 25-May-2015].

[109] Frank T. Johnsen and Trude Hafsøe, 'Service Advertisements in MANETs (SAM): A Decentralized Web Services Discovery Protocol', in *2010 IEEE GLOBECOM Workshops (GC Wkshps)*, 2010, pp. 1674–1678.

[110] K.-H. Kim, W. Baig, S. Yoo, S. Park, and H. Mukhtar, 'Simple Service Location Protocol (SSLP) for 6LoWPAN', *Internet Draft, IETF*, Oct. 2009.

[111] T. A. Butt, I. Phillips, L. Guan, and G. Oikonomou, 'TRENDY: an adaptive and context-aware service discovery protocol for 6LoWPANs', in *Proceedings of the Third International Workshop on the Web of Things*, Newcastle, United Kingdom, 2012, pp. 2:1–2:6.

[112] Z. Shelby and C. Bormann, 'CoRE Resource Directory', *Internet Draft, IETF*, Nov. 2014.

[113] G. Cugola and A. Margara, 'SLIM: Service Location and Invocation Middleware for Mobile Wireless Sensor and Actuator Networks', *International Journal of Systems and Service-Oriented Engineering (IJSSOE)*, vol. 1, no. 3, pp. 60–74, 2010.

[114] C. Jardak, E. Meshkova, J. Riihijarvi, K. Rerkrai, and P. Mahonen, 'Implementation and Performance Evaluation of nanoIP Protocols: Simplified Versions of TCP, UDP,HTTP and SLP for Wireless Sensor Networks', in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2008, pp. 2474 –2479.

[115] M. Shukla, R. Kishore, and R. Kumar, 'Integration of Cluster Based Routing and Mobile Service Discovery Protocol for Manets: A Novel Approach', in *Nirma University International Conference on Engineering (NUiCONE), Ahmedabad, India*, 2009.

[116] L. Mingxue, H. Jing, Z. Rongfu, W. Xianqing, and H. Xiaoxin, 'Tree-Based Service Discovery in Mobile Ad Hoc Networks', in *IEEE Asia-Pacific Services Computing Conference (APSCC)*, 2010, pp. 206–211.

[117] R. S. Marin-Perianu, J. Scholten, P. J. M. Havinga, and P. H. Hartel, 'Cluster-based service discovery for heterogeneous wireless sensor networks', *International Journal of Parallel, Emergent and Distributed Systems*, vol. 23, no. 4, pp. 325–346, Aug. 2008.

[118] P. Faltstrom and M. Mealling, 'The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)', *RFC 3761 , IETF*, Apr. 2004.

[119] J. Mäenpää, J. J. Bolonio, and S. Loreto, 'Using RELOAD and CoAP for wide area sensor and actuator networking', *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, pp. 1–22, 2012.

[120] C. Jennings, B. Lowekamp, E. Rescorla, S. Base, and H. Schulzrinne, 'REsource LOcation And Discovery (RELOAD) Base Protocol', *RFC 6940, IETF*, Jan. 2014.

[121] M. Liu, T. Leppanen, E. Harjula, Z. Ou, A. Ramalingam, M. Ylianttila, and T. Ojala, 'Distributed resource directory architecture in Machine-to-Machine communications', in *IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2013, pp. 319–324.

[122] F. Palmieri, 'Scalable service discovery in ubiquitous and pervasive computing architectures: A percolation-driven approach', *Future Generation Computer Systems*, vol. 29, no. 3, pp. 693–703, Mar. 2013.

[123] J. Mitsugi, S. Yonemura, H. Hada, and T. Inaba, 'Bridging UPnP and ZigBee with CoAP: protocol and its performance evaluation', in *Proceedings of the workshop on Internet of Things and Service Platforms*, 2011, p. 1.

[124] Z. Shelby, 'Constrained RESTful Environments (CoRE) Link Format', *RFC 6690, IETF*, 2012.

[125] M. Nidd, 'Service discovery in DEAPspace', *IEEE Personal Communications*, vol. 8, no. 4, pp. 39 –45, Aug. 2001.

[126] C. Campo, C. García-Rubio, A. M. López, and F. Almenárez, 'PDP: A lightweight discovery protocol for local-scope interactions in wireless ad hoc networks', *Computer Networks*, vol. 50, no. 17, pp. 3264–3283, Dec. 2006.

[127] G. Oikonomou, I. Phillips, L. Guan, and A. Grigg, 'ADDER: Probabilistic, Application Layer Service Discovery for MANETs and Hybrid Wired-Wireless Networks', in *Ninth Annual Communication Networks and Services Research Conference (CNSR)*, 2011, pp. 33 –40.

[128] S. Helal, N. Desai, V. Verma, and C. Lee, 'Konark - a service discovery and delivery protocol for ad-hoc networks', in *IEEE Wireless Communications and Networking (WCNC)*, 2003, vol. 3, pp. 2107 –2113 vol.3.

[129] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin, 'Toward Distributed service discovery in pervasive computing environments', *IEEE Transactions on Mobile Computing*, vol. 5, no. 2, pp. 97 – 112, Feb. 2006.

[130] X. Li, N. Santoro, and I. Stojmenovic, 'Localized Distance-Sensitive Service Discovery in Wireless Sensor and Actor Networks', *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1275–1288, 2009.

[131] Frank T. Johnsen, 'Pervasive Web Services Discovery and Invocation in Military Networks', PhD, Norwegian Defence Research Establishment (FFI), 2011.

[132] S. Cheshire and M. Krochmal, 'Multicast DNS', *RFC 6762, IETF*, 2013.

[133] S. Cheshire and M. Krochmal, 'DNS-based service discovery', *RFC 6763, IETF*, 2013.

[134] 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', in *Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, 2012, pp. 731–738.

[135] R. Klauck and M. Kirsche, 'Bonjour contiki: a case study of a DNS-based discovery service for the internet of things', in *Proceedings of the 11th international conference on Ad-hoc, Mobile, and Wireless Networks*, Berlin, Heidelberg, 2012, pp. 316–329.

[136] R. Klauck and M. Kirsche, 'Enhanced DNS message compression - Optimizing mDNS/DNS-SD for the use in 6LoWPANs', in *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2013, pp. 596–601.

[137] 'ZigBee IP Specification'. [Online]. Available: https://docs.zigbee.org/zigbee-docs/dcn/13/docs-13-0148-00-0mwg-zigbee-ip-features-benefits.pdf. [Accessed: 25-May-2015].

[138] K. Lynn and D. Sturek, 'Extended Multicast DNS', *Internet Draft, IETF*, 2012.

[139] Z. Shelby and M. Vial, 'CoRE Interfaces', *Internet Draft, IETF*, Nov. 2014.

[140] P. Mockapetris, 'Domain Names - Implementation and Specification', *RFC 1035, IETF*, Nov. 1987.

[141] F. Gramegna, S. Ieva, G. Loseto, and A. Pinto, 'Semantic-enhanced resource discovery for CoAP-based sensor networks', in *5th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*, 2013, pp. 233–238.

[142] J. W. Hui and D. Culler, 'The dynamic behavior of a data dissemination protocol for network programming at scale', in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 81–94.

[143] O. Gnawali, K.-Y. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler, 'The tenet architecture for tiered sensor networks', in *Proceedings of the 4th international conference on Embedded networked sensor systems*, 2006, pp. 153–166.

[144] D. Johnson, Y. Hu, and D. Maltz, 'The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4', *RFC 4728, IETF*, Feb. 2007.

[145] C. Perkins, E. Belding-Royer, and S. Das, 'Ad hoc On-Demand Distance Vector (AODV) Routing', *RFC 3561, IETF*, Jul. 2003.

[146] A. Dunkels, 'The ContikiMAC Radio Duty Cycling Protocol', SICS, Technical Report T2011:13, Dec. 2011.

[147] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, 'Software-based on-line energy estimation for sensor nodes', in *Proceedings of the 4th workshop on Embedded networked sensors*, 2007, pp. 28–32.

[148] Nirupama Bulusu, Deborah Estrin, Lewis Girod, and John Heidemann, 'Scalable Coordination for Wireless Sensor Networks: Self-Configuring Localization Systems', in *6th IEEE International Symposium on Communication Theory and Application (ISCTA)*, Ambleside, UK, 2001, pp. 1–6.

[149] T. Clausen, J. Yi, A. Verdiere, A. Niktash, Y. Igarashi, H. Satoh, U. Herberg, C. Lavenu, T. Lys, and J. Dean, 'The Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)', *Internet Draft, IETF*, Oct. 2014.

[150] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, 'The broadcast storm problem in a mobile ad hoc network', *Wireless networks*, vol. 8, no. 2–3, pp. 153–167, 2002.

[151] A. Durresi, V. K. Paruchuri, S. S. Iyengar, and R. Kannan, 'Optimized broadcast protocol for sensor networks', *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1013–1024, 2005.

[152] P. Kyasanur, R. R. Choudhury, and I. Gupta, 'Smart Gossip: An Adaptive Gossip-based Broadcasting Service for Sensor Networks', in *2006 IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, 2006, pp. 91–100.

[153] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih, 'Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network', *IEEE Transactions on Computers*, vol. 52, no. 5, pp. 545–557, May 2003.

[154] B. Djamaa, N. Aouf, M. Richardson, and B. Walters, 'Enhancing Delay-based Packet Forwarding Schemes in Wireless Sensor Networks', in *4th Annual International Conference on Energy Aware Computing Systems and Applications (ICEAC)*, 2013, pp. 12–17.

[155] H. Lee, A. Cerpa, and P. Levis, 'Improving wireless simulation through noise modeling', in *6th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2007, pp. 21–30.

[156] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda, 'Indriya: A low-cost, 3D wireless sensor network testbed', in *Testbeds and Research Infrastructure. Development of Networks and Communities*, Springer, 2012, pp. 302–316.

[157] T. Clausen, C. Dearlove, and B. Adamson, 'Jitter Considerations in Mobile Ad Hoc Networks (MANETs)', *RFC 5148, IETF*, Feb. 2008.

[158] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, 'Powertrace: Network-level power profiling for low-power wireless networks', Swedish Institute of Computer Science, Technical Report T2011:05, Mar. 2011.

[159] C. Vallati and E. Mingozzi, 'Trickle-F: Fair broadcast suppression to improve energy-efficient route formation with the RPL routing protocol', in *Sustainable Internet and ICT for Sustainability (SustainIT)*, 2013, pp. 1–9.

[160] J. Eriksson and O. Gnawali, 'Poster Abstract: Synchronizing Trickle Intervals', in *Proceedings of the 11th European conference on Wireless sensor networks (EWSN 2014)*, Oxford, 2014.

[161] 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, Dec. 2012.

[162] M. Becker, K. Kuladinithi, and C. Görg, 'Modelling and Simulating the Trickle Algorithm', in *Mobile Networks and Management*, Springer, 2012, pp. 135–144.

[163] T. M. M. Meyfroyt, 'Modeling and analyzing the Trickle algorithm', MsC, Eindhoven University of Technology, Eindhoven, The Netherlands, 2013.

[164] T. M. M. Meyfroyt, S. C. Borst, O. J. Boxma, and D. Denteneer, 'A data propagation model for wireless gossiping', *Performance Evaluation*, vol. 85–86, pp. 19–32, Mar. 2015.

[165] B. Djamaa and M. Richardson, 'Optimizing the Trickle Algorithm', *IEEE Communications Letters*, vol. 19, no. 5, pp. 819–822, May 2015.

[166] A. Rahman and E. Dijk, 'Group Communication for the Constrained Application Protocol (CoAP)', *RFC 7390, IETF*, Oct. 2014.

[167] B. Djamaa and M. Richardson, 'Towards Scalable DNS-Based Service Discovery for the Internet of Things', in *Lecture Notes in Computer Science. Ubiquitous Computing and Ambient Intelligence. Personalisation and User Adapted Services*, Springer, 2014, pp. 432–435.

[168] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, 'Low power, low delay: opportunistic routing meets duty cycling', in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, 2012, pp. 185–196.

[169] F. Österlind, J. Eriksson, and A. Dunkels, 'Cooja TimeLine: a power visualizer for sensor network simulation', in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, Zurich, Switzerland, 2010, pp. 385–386.

[170] S. Duquennoy, O. Landsiedel, and T. Voigt, 'Let the tree Bloom: scalable opportunistic routing with ORPL', in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, Rome, Italy, 2013.

[171] P. Thubert and J. Hui, 'LLN Fragment Forwarding and Recovery', *Internet Draft, IETF*, Nov. 2014.

[172] Badis Djamaa, Mark Richardson, Peter Barker, and Mohamed Aissani, 'Multicast Burst Forwarding in Constrained Networks', presented at the 81st IEEE Vehicular Technology Conference, Glasgow, to appear.

[173] 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, Oct. 2014.

[174] H. Tschofenig and J. Arkko, 'Report from the Smart Object Workshop', *RFC 6574, IETF*, 2012.

[175] C. Bormann and P. Hoffman, 'Concise Binary Object Representation (CBOR)', *RFC 7049, IETF*, 2013.

[176] T. Bray, 'The JavaScript Object Notation (JSON) Data Interchange Format', *RFC 7159, IETF*, Mar. 2014.

[177] C. Jennings, Z. Shelby, and J. Arkko, 'Media Types for Sensor Markup Language (SENML)', *Internet Draft, IETF*, Oct. 2012.

[178] G. Oikonomou, I. Phillips, and T. Tryfonas, 'IPv6 Multicast Forwarding in RPL-Based Wireless Sensor Networks', *Wireless Personal Communications*, vol. 73, no. 3, pp. 1089–1116, Jun. 2013.

[179] S. Cheshire, 'Hybrid Unicast/Multicast DNS-Based Service Discovery', *Internet Draft, IETF*, Nov. 2014.

[180] A. Aggarwal, 'Optimizing DNS-SD query using TXT records', *Internet Draft, IETF*, Jul. 2014.

[181] C. Bormann, '6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)', *RFC 7400, IETF*, Nov. 2014.

[182] ‘Authentication and Authorization for Constrained Environments (ace) - Charter’. [Online]. Available: https://datatracker.ietf.org/wg/ace/charter/. [Accessed: 25-May-2015].

[183] ‘DTLS In Constrained Environments (dice) - Charter’. [Online]. Available: https://datatracker.ietf.org/wg/dice/charter/. [Accessed: 25-May-2015].