Departament
d'Enginyeria Telemàtica

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Ph. D. Dissertation

Contributions to the Performance Evaluation and Improvement of the IPv6 Routing Protocol for Low-Power and Lossy Networks

Ph. D. candidate: Hamidreza Kermajani
Ph. D. advisor: Carles Gómez Montenegro

Acknowledgments

This dissertation would not be possible without the inspiration and support from many people. First and foremost, I would like to express my deepest appreciation to my advisor, Dr. Carles Gomez, for his efficient supervision, guidance, constant help and support throughout my PhD. He was always ready and available to guide me with his deep knowledge and broad experience. He showed me the path of a better life. It is my fortune and honor to have an advisor like him.

I would also like to thank all of my friends I met here in Barcelona, Arash Yazdani, Paola Garfias, Alberto José González Cela, André Rios, Hamze Khalili and Arman Kioumarsi, for their friendship and support during my PhD.

A special thanks to my family, for their unconditional love and support along the path of my academic pursuits. Words cannot express how grateful I am to my mother-in law, father-in-law, my mother, brothers and sisters for all of the sacrifices that you've made on my behalf.

At the end, I would like to express appreciation to my beloved wife Mahdieh and my dear son, Karen, who spent sleepless nights with and were always my support in the moments when there was no one to answer my queries and made life much easier and enjoyable than it would have otherwise been. I could have never achieved my goal without their encouragement, understanding, supports, and true love.

Contents

Index of figures

Index of tables

1. Introduction

This chapter has the following organization. The motivation for this Ph.D. is presented in section 1.1. Section 1.2 enumerates the results and contributions of the research work presented in this document. Finally, section 1.3 describes the structure of the document.

1.1. Motivation

Recent progress on wireless communications on the one hand, and technological improvements that allow to produce small, inexpensive and low-power wireless communication devices on the other, have caused Wireless Sensor Networks (WSNs) become increasingly important.

WSNs comprise sensor and actuator nodes that enable intelligent monitoring and control applications in a wide spectrum of environments including smart cities, home automation, remote health and precision agriculture to mention a few [1].

One of the main characteristics of WSNs is the fact that they are significantly constrained with respect to processing capabilities, memory and energy source. Another characteristic is that, being wireless, the links that interconnect these devices are lossy and usually support only low data rates. In certain IETF circles, networks of these characteristics are called Low Power and Lossy Networks (LLNs) [2].

In the last decade, many LLN technologies have emerged [1]. Whereas most protocol architectures were born without native IP support, there exists a tendency in the market towards IP convergence, since IP-based LLNs offer an open and standardized way of connecting LLNs to the Internet, thus enabling the Internet of Things (IoT). The IETF anticipated this need by creating the IPv6 over Low power WPAN (6LoWPAN) Working Group (WG) [3], which provides an adaptation layer to enable and optimize the transmission of IPv6 packets on top of IEEE 802.15.4 networks. However, 6LoWPAN needed a companion mechanism to enable the end-to-end delivery of IP packets: a routing protocol, since most LLN configurations are multihop. The IETF then created the Routing Over Low power and Lossy networks (ROLL) WG [4] in order to analyze the suitability of and/or create IP-based routing functionality for LLNs. The main outcome of the ROLL WG has been a new routing protocol called IPv6 Routing Protocol for LLNs (RPL) [5].

RPL was specifically designed to meet the requirements of WSNs [6-9] and is a central component of the IETF protocol suite for the IoT. Since RPL has already been deployed in millions of nodes [10], it is fundamental to characterize the properties of RPL, evaluate the influence of its main parameters and options on network performance, and analyze performance improvement possibilities. In this thesis, we present several original contributions in this field, which are enumerated in the next subsection.

1.2. Results and contributions

The major contributions of this PhD dissertation are as follows:

- Evaluation of the influence of the main RPL parameters on the network convergence process over IEEE 802.15.4 multihop networks of in terms of network characteristics such as size and density; proposal and evaluation of a mechanism that leverages an option available in RPL for accelerating network convergence.
- Development of an analytical tool to estimate the number of control messages transmitted in a static network which uses the Trickle algorithm (a transmission scheduling algorithm used in RPL, see section 3.2) under steady state conditions.
- Development of an analytical model for estimating the network convergence time of RPL in a static chain topology network of IEEE 802.15.4 nodes, in the presence of bit errors.
- Theoretical evaluation of the route change latency incurred by RPL when 6LoWPAN Neighbor Discovery (ND) is used. Study of the impact of the relevant 6LoWPAN ND and RPL parameters on path availability and the trade-off between path availability and message overhead.
- Development of a RPL simulator for OMNET++_4.3.1 using MiXiM-2.3.

As can be seen, the contributions focus on performance of RPL in both transient state, i.e. during the network formation process, and steady state.

1.3. Structure of this thesis

This document is organized in eight chapters. The novel contributions of the author are mainly presented in chapter 4, chapter 5, chapter 6 and chapter 7. The context for

the contributions is provided in the first three chapters of this document. The organization of this document is shown next.

Chapter 1 is the current introduction.

Chapter 2 provides an overview on WSNs. First, the chapter introduces the main concepts and constraints of WSNs, which should be taken into account while designing a routing protocol for WSNs. Next, we review the main features of IEEE 802.15.4, a popular radio interface for WSNs.

Chapter 3 first reviews the motivation and design requirements for RPL. Next, the chapter studies the main functionality of RPL and its transmission scheduling algorithm, Trickle. Subsequently, the routing metrics and constraints used in RPL for path selection are described. The last section of this chapter is devoted to RPL ancestors and other routing protocols used in different LLN technologies.

Chapter 4 presents a study on the impact of two important RPL parameters on the network convergence process in IEEE 802.15.4 multihop networks. We also propose and evaluate a mechanism for accelerating the DODAG convergence process, by leveraging an option available in RPL. The study has been carried out by extensive simulations for a wide range of conditions, considering different values for network parameters such as network size and density.

Chapter 5 presents an analytical model of the network convergence time in RPL under realistic assumptions, such as the presence of bit errors and the use of IEEE 802.15.4 at the physical and link layers. The model is developed for a static chain multihop topology, which also provides a lower bound on the network convergence time in a generic topology.

Chapter 6 presents an analytical model for the number of messages transmitted in a network that uses the Trickle algorithm. The model assumes a static network which is in steady state. However, for more dynamic WSNs, the model provides a lower bound on the number of message transmissions. The model is highly accurate for both synchronous and asynchronous networks.

Chapter 7 presents a theoretical evaluation of the route change latency incurred by RPL when 6LoWPAN ND is used . The chapter also studies the impact of the relevant 6LoWPAN ND and RPL parameters on path availability as well as the trade-off between path availability and message overhead.

Chapter 8 summarizes the main concluding remarks from this Ph.D. and points out future work directions.

2. Wireless Sensor Networks

In this chapter, we introduce the main concepts of Wireless Sensor Networks (WSNs), and give an overview on the IEEE 802.15.4 standard, which is the most prevalent radio interface for WSNs. The chapter is organized into three sections. Section 2.1 defines WSNs, their constraints, and the main WSN application areas. Section 2.2 introduces different topologies used in WSNs. Section 2.3 describes the main characteristics of the IEEE 802.15.4 standard and highlights the relevant features that will be mentioned in the subsequent chapters.

2.1. Main characteristics of WSNs

A WSN is a network that may be composed of a large set (hundreds or thousands) of distributed autonomous small and inexpensive low-power and low-cost sensors, actuators or smart objects which have significant limitations in terms of storage resources, processing power and energy.

The basic task of a WSN is to monitor an environment in order to sense some specific events or collect data, and send the information gathered to requested destinations. Due to the low cost and the ad-hoc nature of this kind of networks, WSNs can be used for different purposes including [١, 11-13]: environmental monitoring, healthcare, structural monitoring, logistics, seismic detection, military surveillance, inventory tracking, home and building automation, industrial monitoring and control, smart cities, etc.

Most wireless sensors have the ability of sensing, computation and communication among each other or with an infrastructure element like a sink node or gateway. The main components of a sensor node are the sensing unit, the processing unit, the memory unit, the communication unit, the power unit and, if necessary, Analogue to Digital Converters (ADCs), see Figure 2.1.

2.2. Topologies

As the network topology has a significant impact on the performance of a routing protocol, in this subsetion we review the main topologies used in WSNs.

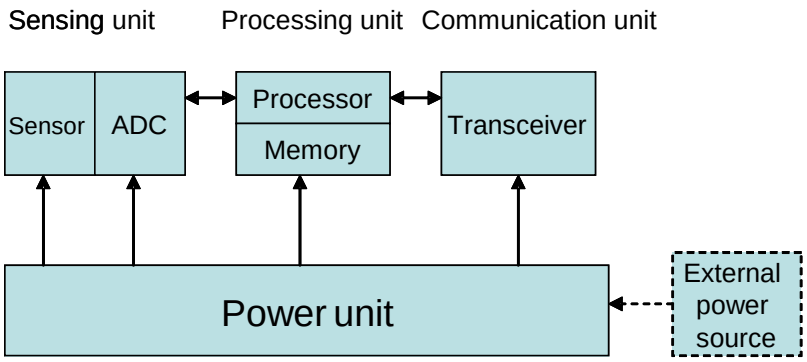


Figure 2.1. The main components of a sensor node [1]

There are different topologies that can be found in WSNs. These topologies can be classified in three main categories: star topology, tree topology and mesh topology. However, a WSN may use a combination of these topologies.

2.2.1. Star topology

In the star topology one of the sensor nodes plays the role of a sink node, and all other nodes are directly connected to the sink node, see Figure 2.2. Nodes can only communicate among each other through the sink node.

The main advantage of this toplogy is its simplicity. However, since all sensor nodes must be in the transmission range of a sink node, this toplogy cannot be used in a large scale network or when the geographical distance between a sender and its destination is large (e.g. greater than the transmission range).

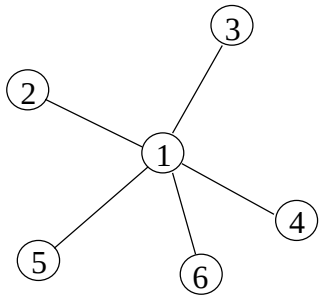


Figure 2.2. Star topology

2.2.2. Tree topology

The tree topology is a combination of connected multiple star topologies, see Figure 2.3.a). One node which is in the highest level of the tree, is named root of the tree and can play the role of a sink node in the network. The tree topology facilitates

routing data collected from sensors towards a root node. On the other hand, in case of root failure, the whole network will be crippled.

2.2.3. Mesh topology

In the mesh topology, all possible links that exist in the network may be used for communication, see Figure 2.3.b). When a destination is not in the transmission range of the source node, multihop transmission is utilized. An advantage of this topology is that if a path between two nodes fails, the nodes may be able to communicate with each other through an alternative route, if such route exists.

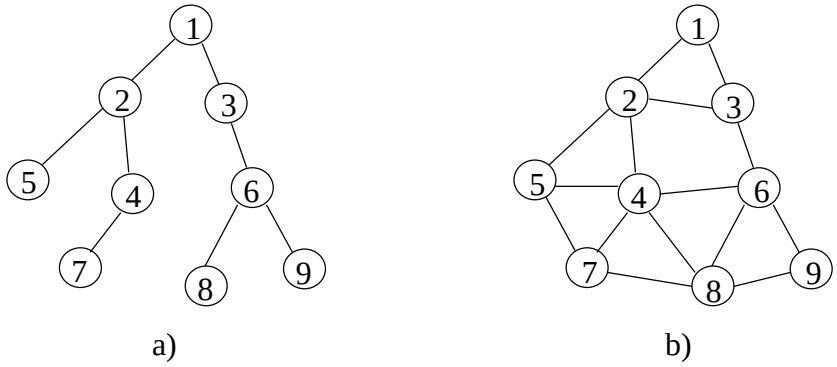


Figure 2.3. a) Tree topology; b) Mesh topology

2.3. IEEE 802.15.4 overview

IEEE 802.15.4 is the de-facto family of standard radio interfaces for low-cost and low-power, low-rate and short-range wireless communications. IEEE 802.15.4 specifies Physical (PHY) and Medium Access Control (MAC) layer [14].

2.3.1. Physical layers of IEEE 802.15.4-2003

This section reviews the physical layer (PHY) of the IEEE 802.15.4-2003 which is the most implemented variant of the IEEE 802.15.4 family. This PHY is responsible for the following four tasks:

1. Activation and deactivation of the radio transceiver.
2. Data transmission and reception.
3. Measurement of Energy detection (ED) and Link Quality Indication (LQI). The energy detection estimates the power of a signal received and it is used to accept or ignore a packet received. This parameter may also be used by an

upper layer in order to avoid the use of busy channels. The LQI indicates the strength/quality of a received packet and may be used by network or application layers.

4. Performing the Clear Channel Assessment (CCA) for Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) algorithm which is a function used by the MAC sub-layer.

The original specification, IEEE 802.15.4-2003, specifies the Radio Frequency (RF) link parameters, including modulation type, coding, spreading, symbol/bit rate, and channelization. This specification defines 27 channels spread across three different frequency bands: 1 channel in the 868 MHz band which is available in Europe, 10 channels in the 915 MHz band, available in the Americas region, and 16 channels in the 2.4 GHz band that is available worldwide.

The 2.4 GHz band is the most commonly used one among the bands available in IEEE 802.15.4. Channels in this band provide a raw data rate of 250 kbit/s.

Direct Sequence Spread Spectrum (DSSS) is used for spreading the spectrum of the signal in this standard. The modulations in IEEE 802.15.4-2003 are based on the Phase-Shift-Key (PSK) modulation. However, the 868/915 MHz bands use Binary PSK (BPSK) modulation and the 2.4 GHz band uses Offset-Quadrature PSK (O-QPSK). Table 2.1 shows the main physical layer features of IEEE 802.15.4-2003.

2.3.1.1.1. Physical Protocol Data Unit format

The main service of the physical layer is to enable the transmission and reception of IEEE 802.15.4 PHY protocol data unit (PPDU), the data unit that carries the upper layer data unit (i.e. MAC frame), across the physical radio cannel.

There are three components in the PPDU: the synchronization header (SHR), PHY header (PHR), and PHY payload. The structure of the PPDU is shown in Figure 2.4.

Table 2.1. Physical layer features of IEEE 802.15.4-2003 [1]

Frequency band	Number of channels	Spreading technique	Modulation	Symbol rate per channel (kbaud)	Bit rate per channel (kbit/s)
868 MHz	1	Binary DSSS	BPSK	20	20
915 MHz	10	Binary DSSS	BPSK	40	40
2.4 GHz	16	16-array DSSS	O-QPSK	62.5	250

SHR allows a receiving node to synchronize and lock into the bit stream. It is composed of a preamble field, which is used by the transceiver to obtain chip and symbol synchronization, and the Start-of-Frame Delimiter (SFD) that indicates the end of the SHR and the start of the data packet. The PHR contains the length of the PHY payload and finally, the PHY payload carries the MAC sublayer frame. The PHY payload is also called PHY Service Data Unit (PSDU).

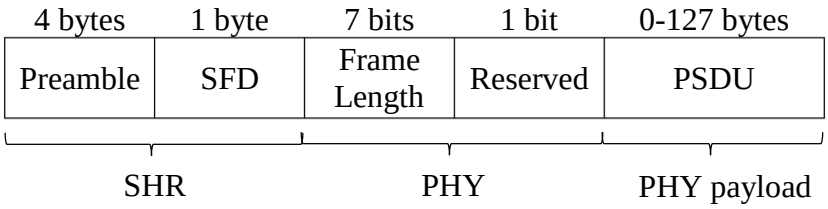


Figure 2.4. The IEEE 802.15.4 PPDU format

2.3.2. MAC layer of IEEE 802.15.4

The Medium Access Control (MAC) layer manages access to a physical radio channel shared between network nodes. At the MAC layer, IEEE 802.15.4 networks may be configured in two modes[1]: i) on the basis of a superframe structure delimited by beacons, or ii) in a beaconless mode. The latter is generally preferred due to its simplicity and also it does not require any specific network organization. For these reasons in this thesis we have assumed the beaconless mode of IEEE 802.15.4 MAC. Next we explain the operation of this mode.

2.3.2.1. Beaconless mode of IEEE 802.15.4 MAC

In the beaconless mode, nodes use unslotted CSMA/CA as the medium access mechanism. In this mode, the process for transmitting a data frame requires the sender to wait for an initial, uniformly random backoff time ranging from 0 to 2^{BE}-1, where *BE* is the current backoff exponent. Initially, *BE* is set to *macMinBE* and its maximum value is *macMaxBE*. The default values of *macMinBE* and *macMaxBE* are 3 and 5, respectively. In addition to the backoff exponent, *BE*, IEEE 802.15.4 uses another variable called *NB* which is used to count the number of times the CSMA/CA algorithm has been used while attempting the current transmission and it is initialized to 0 at the beginning. After waiting for the backoff time, the sender carries out a Clear Channel

[1] Note that we do not consider the recent MAC layer functionality mechanisms that have been added in IEEE 802.15.4e.

Assessment (CCA) to check the state of the medium. If the medium is idle during CCA time, the sender performs the transmission. Otherwise, the sender increases *BE* and *NB* by one unit (if *BE* < *macMinBE*) and does the same procedure up to *macMaxCSMABackoffs*+1 (while *NB* <= *macMaxCSMABackoffs*). After *macMaxCSMABackoffs*+1 unsuccessful attempts to access the channel, the procedure

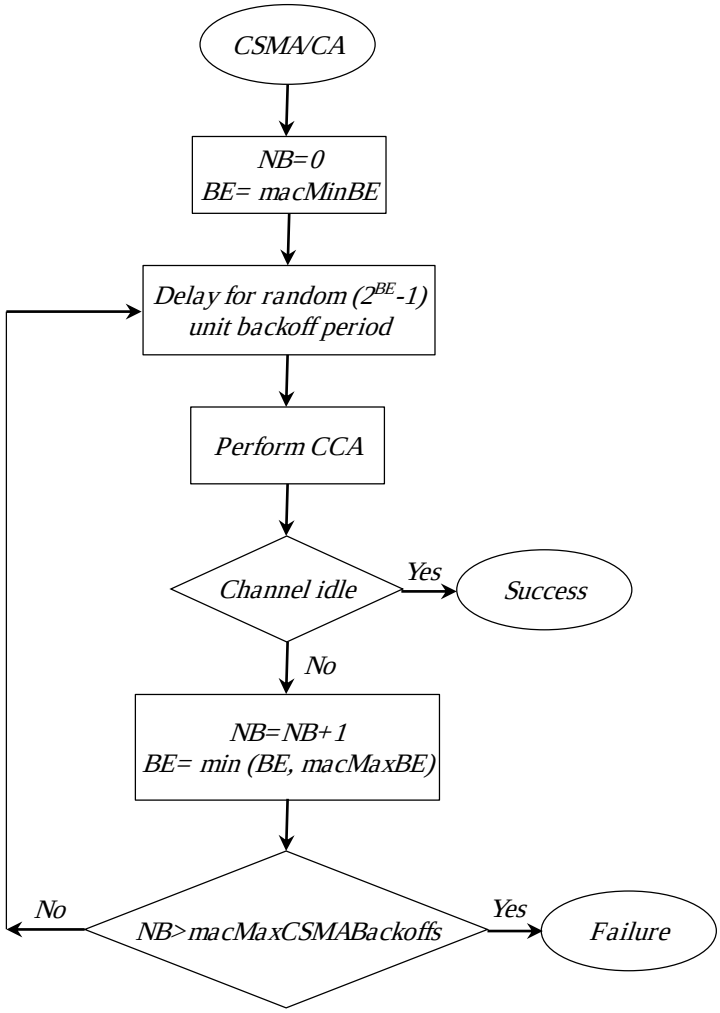


Figure 2.5. CSMA/CA beaconless mode algorithm

fails and the data frame is discarded. Figure 2.5. illustrates the steps of the CSMA/CA algorithm in the beaconless mode.

2.3.2.2. MAC packet data unit format

All MAC data frames are encapsulated in the PSDU. The format of a MAC data frame is shown in Figure 2.6.

Each MAC data frame is composed of the three following components:

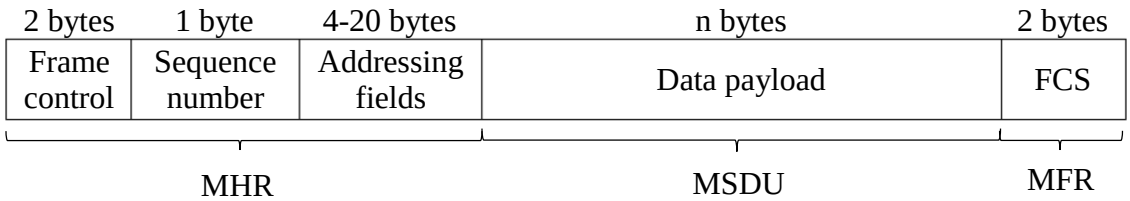


Figure 2.6. Format of the IEEE 802.15.4 data frame

1. MAC header (MHR): this part contains three different fields. The first is the Frame Control Field (FCF), which has a size of 2 bytes and includes information about the frame type, addressing fields and other control flags. The second field is the sequence number, which has a size of one byte. Finally, the addressing fields may include a two-byte destination PAN identifier, a 16- or 64-bit destination address, a two-byte source PAN identifier and a 16- or 64-bit source address
2. MAC payload (MSDU): its content depends on the frame type and it has a variable size.
3. MAC footer (MFR): this field contains a Frame Check Sequence (FCS) and has a size of 2 bytes.

3. IP-based routing protocol for LLNs

In this chapter we focus on RPL and its related mechanisms. First of all, we provide a RPL overview, including the motivation for its design and its basic functionality. Next we present a detailed description of a key RPL component, which is the Trickle algorithm. We also explain the objective function concept and neighbor discovery, and finally we review routing functionality related with RPL.

3.1. RPL overview

In this subsection, we first review the motivation for the design of RPL in section 3.1.1. The basic operation of RPL is provided in section 3.1.2. Section 3.1.3 explains routing methods in RPL. Point-to-Point communication mechanism in RPL is discussed in section 3.1.4 and section 3.1.5 presents neighbor unreachability detection in RPL. Finally, section 3.1.6 reviews different repair mechanisms in RPL.

3.1.1. Motivation for the design of RPL

Five years ago, the IETF ROLL Working Group (WG) wrote four documents [6-9] to describe ROLL application requirements, in order to assist in the selection or the development of a new routing protocol for LLNs. The application environments considered in these documents were home automation, building automation, industrial networks and urban networks, respectively.

Then, the IETF ROLL WG carried out a protocol survey [15] and specified five main criteria for routing over LLNs, extracted from the requirements that the above mentioned documents from different applications domains share. In this survey, it is considered that the ideal routing protocol to operate over LLNs should be able to satisfy all of the following criteria: "routing state", "loss response", "control cost", "link cost", and "node cost".

The first criterion, "routing state", indicates that the amount of memory used in each router in the routing protocol must not scale with number of nodes in the network. Instead, it should scale with the number of one-hop neighbors of a router.

The second criterion, "loss response", indicates that when a link failure occurs in LLNs, the routing protocol should try to repair the link failure by a local mechanism instead of triggering a global network re-optimization.

The “control cost” criterion proposes that, in order to maintain a network topology, the control traffic rate of a routing protocol should be bounded by the data traffic rate plus a small constant.

“Link cost” and “node cost” criteria are related to the metrics used by the routing protocol. The routing protocol must be able to satisfy node/link metrics and constraints in finding paths. Examples of these criteria include minimizing latency, or maximizing link reliability, power processing and available amount of memory or battery life of a node.

The IETF ROLL WG, after reviewing all the existing IETF routing protocols, see[2] Table 1, concluded that no existing routing protocol (in its current form) could fulfill the above mentioned criteria. Instead of adapting any of the existing routing protocols, the ROLL WG decided to develop a new routing protocol for IP-based LLNs. As a result, the IPv6 Routing Protocol for Low power and Lossy networks (RPL) [5] has been designed taking into account the above mentioned requirements.

Table 3.1. Evaluation of existing IETF routing protocols

Protocol	Routing state	Loss response	Control cost	Link cost	Node cost
OSPF/IS-IS [16-18]	fail	fail	fail	pass	fail
OLSRv2 [19]	fail	?	?	pass	pass
TBRPF [20]	fail	pass	fail	pass	?
RIP [21]	pass	fail	pass	?	fail
AODV [22]	pass	fail	pass	fail	fail
DYMO [23]	pass	?	pass	?	?
DSR [24]	fail	pass	pass	fail	fail

3.1.2. RPL basic operation

RPL is an IPv6-based proactive distance vector Routing Protocol for Low power and Lossy networks which has recently been developed by the IETF ROLL working group [5].

RPL builds Destination Oriented Directed Acyclic Graphs (DODAGs), based on routing metrics and constraints. A DODAG is a directed graph whereby all edges are oriented in such a way that no cycles exist. The edges are contained in paths oriented towards and terminating at one node that is called the root. A set of one or more

[2] In Table 1, the meaning of the word "pass" is that a given protocol has satisfactory performance according to the criterion. The value of "fail" means that the protocol cannot satisfy the criterion. Finally, the value of "?" means a protocol would require a supplementary document for better assessing whether the protocol passes the criteria or not.

DODAGs that try to provide a specific objective is called RPL instance (see Figure 3.1). In order to satisfy different constraints and criteria, a network may use multiple RPL instances concurrently, each of which may serve different and potentially antagonistic constraints or performance criteria. According to RPL rules, each node can only belong to one DODAG in the RPL Instance, but it can belong to multiple RPL instances concurrently.

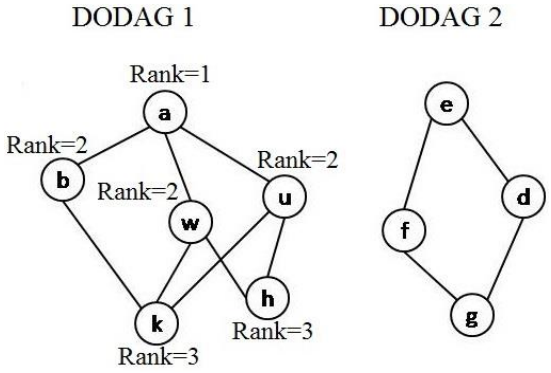


Figure 3.1. A network with a RPL instance and two DODAGs. Nodes a and e are the roots of DODAG 1 and DODAG 2. respectively.

Each RPL Instance is defined by a unique identifier within a network called *RPLInstanceID*. In order to distinguish DODAGs in a RPL Instance, RPL uses a unique identifier for each DODAG called *DODAGID*. RPL uses a sequential counter, incremented by the DODAG root, to identify the version of each DODAG called *DODAGVersionNumber*.

There are two types of DODAG in RPL: grounded and floating DODAG. A DODAG that, in addition to providing inner connectivity within itself, can provide connectivity to nodes in other networks, is called grounded DODAG. In contrast, a floating DODAG only offers connectivity between nodes within the DODAG and it cannot provide routes to nodes in other DODAGs.

For the construction and maintenance of the DODAG, RPL nodes transmit DODAG Information Object (DIO) messages via link-local multicast pseudo-periodically [25], further detail is given in section 3.2. A DIO message contains information that allows a node to discover a RPL instance, learn its configuration parameters, learn the Objective Function (OF) used and maintain the upward routing topology. In addition, the DIO message can include options. One of these options is the DODAG configuration option, which contains the necessary information to set the parameters in each new node: default lifetime, parameters for scheduling the

transmission of DIO messages, metric container, etc [5]. Default lifetime specifies the time of validity for all routes in a DODAG. RPL does not currently specify any default value for this parameter.

Each node in a DODAG selects a DODAG parent set, which is composed of the nodes that provide connectivity to the rest of the nodes in the DODAG. For example, in Figure 3.1, the parent set of node *k* is {*b*, *w*, *u*}. A node uses the *rank* property in order to select another node as a DODAG parent. The rank property is the individual position of a node relative to other nodes with respect to the DODAG root. In other words, the rank of a node abstracts the topological distance between a node and the DODAG root. Each node of a DODAG indicates its rank in the DIO messages it transmits. A node calculates its rank property by applying the Objective Function (OF) in use in the DODAG, and using the rank of its neighbors as an input. The OF defines how nodes use one or more metrics and constraints in order to determine their own rank. Some of the metrics and constraints that can be used in the OF are the following ones: Expected Transmission count (ETX), Latency, HoP-Count (HP), Link Quality Level (LQL), and Remaining energy.

When a node does not understand or support the RPL Instance's OF or advertised metric/constraint, it may join a DODAG as a leaf node. A leaf node is a node that does not extend the DODAG connectivity; i.e. it should not send DIO messages.

Another control message used in RPL is the DODAG Information Solicitation (DIS) message. When a new node wants to join a DODAG, it either can wait to receive DIO messages from nearby nodes or it can send a DIS message in order to request the immediate transmission of DIO messages from neighboring nodes. A DIS message can be sent either in multicast or unicast. When a node wants to receive information, such as the DODAG configuration option, from one of its parents, it unicasts a DIS to that parent. If the node does not have any parent and it is the first time that it wants to join a DODAG, it multicasts a DIS message.

When a node sends a DIS message and does not receive any DIO message in response, after some time the node may decide to be the root of its own floating DODAG and start to multicast DIO messages.

3.1.3. Upward and downward routing in RPL

The core RPL functionality defines two types of routes depending on the direction in which data are transmitted in a DODAG: upward and downward routes. An upward

(downward) route provides a path towards (from) the DODAG root from (to) non-root nodes. After joining a DODAG, each node learns its upward routes by choosing its parent set. Upward routes are used for data transmission from non-root nodes to the root. However, to establish downward routes RPL uses Destination Advertisement Object (DAO) messages which are issued by non-root nodes in order to propagate destination information upwards. RPL defines two modes of operation for downward routing in a DODAG, depending on the storage capabilities of a node: storing and non-storing mode. In the storing mode, all non-root nodes store downward routing tables for their sub-DODAG, which contain information obtained from the DAO messages. In non-storing mode, nodes do not store routing tables for their sub-DODAG. Instead, nodes downward packets by using source routes populated by the DODAG root.

3.1.4. P2P mechanism

3.1.4.1. P2P communication in the base RPL specification

In RPL, one-hop Point-to-Point (P2P) communication between two nodes is allowed, for both modes of operation, storing and non-storing modes. A node can directly send data packets to a one hop neighbor.

RPL uses the Multicast Destination Advertisement Object (Multicast DAO) message to provide P2P routes for one-hop communication between DODAG nodes. A node uses a multicast DAO message to advertise information about itself. This message can be sent to the link-local scope all-RPL-nodes multicast address.

However, if the destination is not within the source's range, this mechanism does not work. In such case, data have to be transmitted upwards and downwards following the DODAG structure, which may not be optimal.

3.1.4.2. Optimized P2P communication

Since the default P2P mechanism used in RPL may not be always optimal (i.e. there may exist shorter paths than the ones selected by RPL), a reactive mechanism for finding P2P routes has been designed [26].

Using this mechanism, a source node will be able to request a route to any other DODAG node. To this end, the source node forms a temporary DODAG that is rooted at itself by disseminating DIO messages which include a P2P Route Discovery Option (P2P-RDO). Such DIO messages are called route discovery messages. Route discovery message transmission is based on the Trickle algorithm. However, Trickle parameters

used to send route discovery messages can be different from the ones used to form and maintain the main DODAG.

All route discovery messages are sent via link-local multicast and all links between nodes in the temporary DODAG should have bidirectional reachability.

A source node announces Trickle parameter values, number of desired routes, route type (source route/hop-by-hop route), route constraints that the discovered route must satisfy and the lifetime of the temporary DODAG by including them in the P2P-RDO that will be carried by DIO messages. Using the route discovery message a source node can request one hop-by-hop route or up to four source routes per target. A source node can ask a target to inform about discovered routes by setting a flag of the route discovery message. In this case, upon receiving a route discovery message, the target sends back a P2P Discovery Reply Object (P2P-DRO) message to the source node in response.

3.1.5. Neighbor unreachability detection in RPL

RPL does not provide any mechanism for neighbor unreachability detection. However, the RPL specification suggests two mechanisms for that purpose: i) use of the Neighbor Unreachability Detection mechanism defined in IPv6 Neighbor Discovery specification [27]; ii) layer two notifications when layer two is acknowledged. Using a proactive approach such as Keepalive messages is not desirable, because it is expensive in terms of bandwidth and energy. Further details about neighbor unreachability detection mechanism in IPv6 are given in section 3.4.

3.1.6. DODAG repair in RPL

Repair is an important feature of a routing protocol. When a link or node failure happens in a network, this mechanism tries to heal the network topology with a minimum cost. The RPL core offers two repair mechanisms called local repair and global repair, respectively.

3.1.6.1. Local repair

When a node's parent set becomes empty and the node does not have any parent in the upward direction, or when a packet is sent out in the upward direction but the receiving router has greater rank than the sender of the packet, an inconsistency occurs. In these cases, local repair can be triggered to solve the problem.

In the first case, when the parent set of a node becomes empty, the node has to advertise a rank of *INFINITE_RANK* in order to inform its sub-DODAG that it no longer has any routes to the root. In addition it may become the root of a floating DODAG and reset its Trickle timer (see 3.2). There is a rule in RPL by which having a parent with a Rank of *INFINITE_RANK* in the parent set of a node is forbidden. In this way the nodes in the sub-DODAG must select another member of its parent set as a preferred parent, if any, to keep their connectivity. Otherwise, these nodes should do the same action.

RPL includes a reactive mechanism to carry out loop avoidance and detection. To this end, each data packet sent by RPL nodes contains a RPL Packet Information field [28] that allows RPL to detect loops while forwarding a data packet. RPL Packet Information defines control flags that are checked upon reception of a data packet. These flags are defined as follows:

- Down flag 'O' (1 bit flag): this flag is used to identify a packet direction. When the direction is downward this flag is set to 1, otherwise it is set to 0.
- Rank-Error flag 'R' (1 bit flag): 'R' flag is relative to 'O' flag. When a packet direction is wrong, based on the 'O' flag, the 'R' flag is set to 1, otherwise it is set to 1.
- Forwarding-Error 'F' (1 bit flag): this flag is used when the 'O' flag is set, i.e. in the downward direction. When the 'F' flag is set it indicates that the sender node does not have any route towards destination.

When a node receives a data packet with the 'O' flag set and the sender's rank is higher than its own rank, an inconsistency has occurred. Also if a data packet is received from a sender with lower rank while the 'O' flag is zero, an inconsistency happens.

When an inconsistency has occurred while forwarding a data packet, RPL nodes ignore it for the first time. This can be done by checking the 'F' flag within the packet received. If the 'F' is zero this means that it is the first time that an inconsistency has happened for this packet, in this case the receiving node should set the 'F' flag to 1. However, if the 'F' is 1, the packet will be discarded and the receiving node should reset its Trickle timer. This local mechanism, resetting the Trickle timer, will cause the receiving node to inform the sender node to adjust its rank.

Another reason that can cause an inconsistency is the reception of a multicast DIS message from a node that is interested in selecting the receiving node as its parent. The

DIS message can contain information, carried within the Solicited Information option, that specifies which receiving nodes must reset their Trickle timer. In addition, when a node receives a DIS message without a Solicited Information option it should reset its Trickle timer.

3.1.6.2. Global repair

DODAG roots can trigger a mechanism called global repair by incrementing the *DODAGVersionNumber* that is included in each DIO sent. Global repair causes a DODAG to form again from the beginning, which means that all nodes must select their parent set and subsequently compute their rank in the new DODAG version. Whenever a node hears a new *DODAGVersionNumber*, propagated in a received DIO message, it may immediately join the new DODAG by selecting the sender of a DIO message received as a parent. The node can also defer joining the new DODAG in order to receive a DIO from a DODAG that is preferred. However, in both cases, upon joining a new DODAG version, a node must not select as a parent another node that advertises an old *DODAGVersionNumber*.

Selecting the parent set in the new DODAG version is completely independent of the previous version. However, it is possible that a node chooses the same position, rank and parent set, in the new version. The RPL specification does not state when a DODAG root should increment its *DODAGVersionNumber* to form a new version of DODAG and leaves this decision up to implementation criteria.

3.2. The Trickle algorithm

The previous subsection introduced that DIO messages are disseminated pseudo-periodically, by using an algorithm called Trickle. Trickle is a transmission scheduling algorithm for local primitive communication between nodes in a network which is based on a consistency model [25]. When a network is in consistent state, nodes slow their communications rate exponentially, such that they transmit Trickle messages at most a few packets per hour. In contrast, when a node detects an inconsistency, it communicates Trickle messages quickly to resolve the inconsistency. At first, the Trickle algorithm was proposed for code propagation and maintenance in Wireless Sensor Networks (WSNs) [29] but it has been shown that it can be used for different purposes, such as control traffic timing, multicast propagation and route discovery [25].

The Trickle algorithm has gained relevance in recent years, since it has been standardized by the IETF as the mechanism that regulates the transmission of DIO messages, which are used to create the network graph in RPL [5].

The Trickle algorithm divides time into non-identical intervals in such a way that the size of the smallest interval is I_{min} and the size of the largest one is I_{max}. In each interval each node tries to send its Trickle message based on the Trickle rules. The Trickle algorithm works based on some parameters, variables and rules. We next review this algorithm.

There are three parameters to configure the Trickle algorithm:

- The minimum interval size, I_{min}.
- The maximum interval size, I_{max}.
- And a redundancy constant, *k*.

In addition, the Trickle algorithm has three variables which track the current status of the algorithm which are as follows:

- *I*: the current interval size.
- *t*: a time within the current interval.
- *c*: number of heard messages within the current interval.

The main operation of the Trickle algorithm is based on the following steps:

1. Initially, Trickle sets *I* to a value in the range [I_{min}, I_{max}].
2. At the beginning of each interval, Trickle sets *I* to a value in the range [I_{min}, I_{max}] and resets *c* to 0 and sets *t* to a random point from the range [*I*/2, *I*).
3. Whenever Trickle hears a transmission that is consistent it increments its own counter *c*.
4. At time *t*, Trickle transmits if, and only if, its counter *c* is lower than the redundancy constant, *k*; otherwise the transmission is suppressed.
5. Whenever the interval *I* expires, Trickle doubles the interval length (if the new interval length is greater than I_{max}, the new interval length is set to I_{max}). Then, the algorithm goes back to execute step 2.
6. If Trickle hears a transmission that is inconsistent, it must reset *I*, set this variable to I_{min}, start a new interval and execute step 2.

In RPL, the three main parameters of the Trickle algorithm, I_{min}, I_{max} and *k*, are set by the DODAG root. The values for these parameters are encapsulated in the DIO messages sent by the root. Hence, these parameter values are learnt by the root

neighbors upon DIO reception, and are included in the DIO messages sent by these neighbors. The same process happens for the rest of nodes that join the DODAG, and thus the parameter values originally announced by the root are propagated to the whole DODAG. In order to keep a low network convergence time, I_{min} is commonly used as the value for the first interval [30].

Figure 3.3 depicts an example of the Trickle transmission in the four-node network shown in Figure 3.2, where the value of redundancy constant, *k*, is 1.

In the first interval, node 2 sends a Trickle message before the other nodes because it has the lowest random value of *t* of all four nodes. Nodes 1 and 3 hear this transmission and increase their counter, *c*. Since node 4 is out of the coverage range of node 2, it does not hear this transmission, thus it does not increment its counter. The second minimum transmission time is the one selected by node 1, but this node suppresses its Trickle message because its counter's value, *c*, is 1 (to be able to send the Trickle message, the value of *c* has to be smaller than 1, i.e. $c < k$). Subsequently, node 4 transmits a Trickle message because it has not heard any transmission. The last node in the first interval is node 3, which suppresses its transmission, due to its counter's value.

In the second interval, first node 3 transmits a Trickle message, thus all other nodes increase their counters since all of them hear this transmission. In the rest of this interval no other node transmits because the value of *c* is 1 for all nodes, and this value is not smaller than the redundancy constant value.

3.3. Objective functions

The Objective Function (OF) defines how RPL nodes select and optimize routes within a RPL Instance. The OF used in a DODAG is indicated within DIO messages using a Metric Container suboption. Whereas several OFs can be defined and supported by a RPL implementation, the RPL specification only mandates the support of the Objective Function Zero (OF0), see 3.3.1.

The set of node and link metrics and constraints that are desirable to consider in LLNs and can be used in RPL are listed in [31]. A routing metric can be used as a constraint as well. In general, this specification divides routing metrics in two categories: aggregated and recorded. An aggregated metric will be adjusted on each receiving node in a DODAG, e.g. hop count, etc. If a recorded metric is used, each node adds a sub-object reflecting the local computation of the metric, e.g. LQI, ETX, etc. The

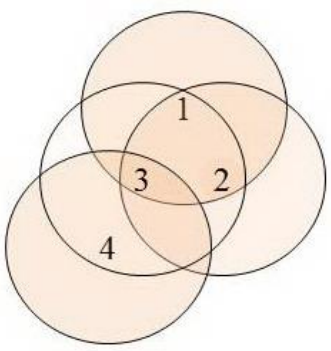


Figure 3.2. A network with four nodes where the circles show the corresponding coverage range of each node.

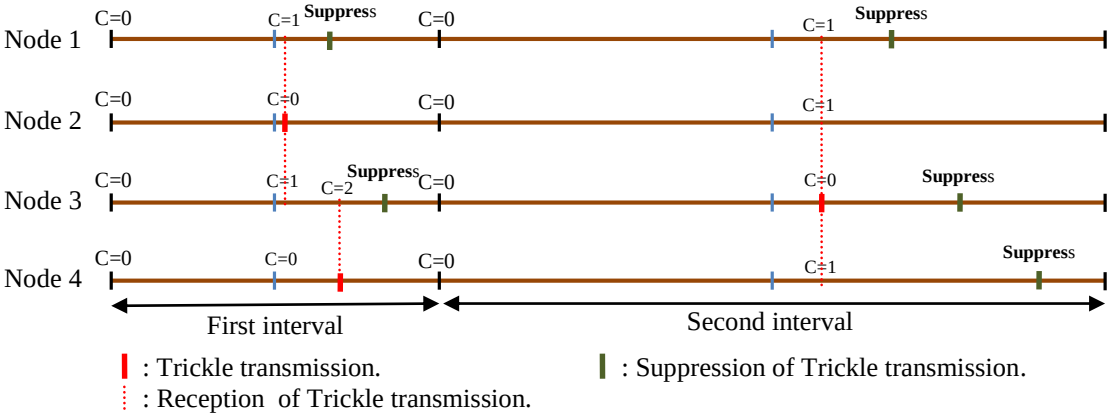


Figure 3.3. Trickle transmission in a synchronous network with the four nodes shown in Figure 3.2.

aggregated metrics are classified in four different types: additive, multiplicative, minimum and maximum. The type of each metric is indicated within a Metric Container carried within the DIO message.

Using multiple metrics and constraints in RPL is allowed. To this end, a DIO message can carry Multiple Metric Containers. It is very important that the mechanism used for the calculation of node/link metrics and constraints has to be the same for all nodes and links.

In general, metrics and constraints used in wireless networks may try to satisfy at least one of the following goals:

- Minimize delay
- Maximize probability of data delivery
- Maximize path throughput
- Minimize energy consumption
- Equally distribute traffic load

Link metrics/constraints that are listed in [31] for possible use in RPL are as follows: Throughput, Latency, LQL, ETX, and Link Color where Energy and Hop-Count are specified as node metrics/constrains.

In the following we present the OFs proposed as of the writing for RPL: OF0 and MRHOF.

3.3.1. Objective Function Zero

In order to find optimized routes in a RPL instance, using the same OF by DODAG nodes is very important, otherwise sub-optimal paths may form. It can also happen when different implementations want to cooperate together in the absence of an identical OF: correct interoperation between these implementations cannot be guaranteed. In this sense, Objective Function Zero (OF0) [32] is specified as the default OF for RPL in order to provide correct interoperation between different implementations of RPL.

RPL nodes use the OF when they want to join a DODAG. This can be done by selecting a neighboring node as a preferred parent that presents a lower rank along with connectivity to a grounded root. To this end, a node scans all possible neighbors and assigns a value by using OF0 to each corresponding link, which is called *step_of_rank*. The value assigned to each link is based on link properties that can be static or dynamic. The exact method of *step_of_rank* computation is left to the implementation in OF0. In the case of using static link properties the result leads to a rank that is analogous to hop-count. However, OF0 recommends use of dynamic metrics such as ETX instead of the statics ones. The implementation should set the step_of_rank between the fixed constants *MINIMUM_STEP_OF_RANK* and *MAXIMUM_STEP_OF_RANK*. Since the range of *step_of_rank* may not always be enough to strongly distinguish links of different types, e.g. powered over battery-operated or wired over wireless, OF0 allows the implementation to multiply *step_of_rank* by a factor that is called rank_factor. The implementation should set the rank_factor between the fixed constants *MINIMUM_RANK_FACTOR* and *MAXIMUM_RANK_FACTOR*. Sometimes after multiplication of *step_of_rank* by *rank_factor* an implementation needs to stretch the result by adding a configurable parameter that is called *stretch_of_rank*, in order to enable a node for selecting at least one feasible parent. This stretching can range from 0 to the fixed constant *MAXIMUM_RANK_STRETCH*. Although stretching the

step_of_rank mechanism is allowed by OF0, it is not recommended, due to the possibility of loop creation.

After the computation of the *step_of_rank*, *rank_factor* and *stretch_of_rank*, OF0 computes a *rank_increase* value by applying the following equation:

$$rank_{increase} = \left(rank_{factor} * step_{of_{rank}} + stretch_{of_{rank}}\right) * MinHopRankIncrease \quad (3.1)$$

where *MinHopRankIncrease* is the minimum increase in rank between a node and any of its DODAG parents and it is defined in the RPL specification by the default value of *DEFAULT_MIN_HOP_RANK_INCREASE*.

Finally a node's rank can be computed as follows:

$$Node's\ rank = Parent's\ rank + rank\_increase \quad (3.2)$$

3.3.2. Minimum Rank with Hysteresis Objective Function

Minimum Rank with Hysteresis Objective Function (MRHOF) [33] is another OF proposed for RPL that can only be used with an additive metric that must be minimized on the paths selected for routing. In other words MRHOF finds a path with the minimum cost based on the metric advertised in metric container. To do so, when a new path is found, MRHOF switches to the new path if and only if the new path cost is smaller than the cost of current path plus a given threshold, denoted by *PARENT_SWITCH_THRESHOLD* in MRHOF specification. The cost of a path is the summation of the selected metric of the links or nodes along the path.

When a node wants to join a DODAG, it scans all of its feasible neighbors in order to calculate the corresponding metric to each of these neighbors/links, depending on the metric advertised in the metric container which is link or node metric. In parallel, the node has to compute the cost of each path by adding the calculated metric to the corresponding neighbor's rank, which is advertised by that neighbor through its sent DIO messages. After these computations, the node selects a neighbor that offers the minimum path cost as its preferred parent and advertises this cost as *cur_min_path_cost* in each DIO transmission. DODAG roots must set its *cur_min_path_cost* to *MIN_PATH_COST* which is the default value for root nodes. In the case of using a link metric when the metric of the link to a neighbor is not available, the path cost for the path through that neighbor should be set to *MAX_PATH_COST*. Also, while using a node metric that is not available the path cost through all the neighbors should be set to

MAX_PATH_COST. When a node does not have metrics for calculating the path cost of its neighbors it should join one of its feasible neighbors as a leaf node.

MRHOF uses ETX as default metric in the absence of metric container. In this case, a node computes the ETX metric for each of its reachable neighbors and selects the minimum one as preferred parent. In the case of using ETX as the metric, a node sends the path cost chosen as its rank.

Some of the node/link metrics that can be used by MRHOF are the following ones: Node Energy, Hop-Count, Latency, Link Quality Level and ETX.

3.4. IPv6 and 6LoWPAN Neighbor Discovery

In this section we review the shortcomings of IPv6 Neighbor Discovery to operate over LLNs and the reasons that caused the IETF 6LoWPAN WG to design an optimized version of the IPv6 ND for LLNs, denoted 6LoWPAN ND. Next we focus on the neighbor unreachability detection mechanism in 6LoWPAN ND and we will study how this mechanism can be used in RPL.

3.4.1. Overview of IPv6 Neighbor Discovery and 6LoWPAN Neighbor Discovery

The IPv6 Neighbor Discovery protocol (ND) [27] has defined a set of important mechanisms for Address Resolution, Duplicate Address Detection, Redirect and Router Discovery along with Prefix and Parameter Discovery for IPv6 networks. However, IPv6 ND is not suitable for LLNs. One reason is the aggressive use of multicast signaling, which leads to link layer broadcast in IEEE 802.15.4 networks and consumes excessive energy and bandwidth. Another reason is that IPv6 ND does not support sleeping nodes [34].

In order to solve the problems of IPv6 ND on top of LLNs, the 6LoWPAN WG designed an optimized version of the IPv6 ND for LLNs (we refer to this ND version as ‘6LoWPAN ND’) [34].

3.4.2. 6LoWPAN ND Neighbor Unreachability Detection

In 6LoWPAN ND, each node registers its IPv6 address along with its link layer address in a neighbor cache entry of its default routers. This method is different from the one used in IPv6 ND [27]. In 6LoWPAN ND the address registration can be done by

sending a Neighbor Solicitation (NS) message, which in addition to the IPv6 and link layer addresses, includes the node registration lifetime [34]. The receiving router expects that the sending node will be reachable during the registration lifetime specified. After successful registration, the node also expects that the router will be available for the same lifetime. Whenever a node wants to check whether its default routers are still reachable or not, it performs Neighbor Unreachability Detection (NUD). NUD is a mechanism that detects the failure of a neighbor or the failure of the forward path to the neighbor [27].

A node is responsible for maintaining its neighbor cache entries in its routers by performing re-registrations, even when the node does not have data packets to send. Sending data to the router does not serve as a re-registration. In fact, other nodes may want to send data to this node. For this reason, the node repeats sending NS messages to the router periodically before the registration lifetime expiration. In order to save power, sending the NS message for re-registration and NUD can be combined together. In the storing mode of RPL, nodes can use DAO messages instead of NS messages. NUD can only be performed in storing mode, because in non-storing mode, nodes cannot store the addresses of their children in a routing table. Sending the NS message will be repeated up to *MAX_UNICAST_SOLICIT* times using a minimum timeout of *RETRANS_TIMER* until the node receives a Neighbor Advertisement (NA) message from the router in response, or a DAO-ACK if the DAO message has been used in the storing mode of RPL.

The reachability of the router can be acknowledged by using different mechanisms: i) layer two notifications (e.g. by using link layer acknowledgments) or ii) upper layer mechanisms, such as hints from transport layer protocols. However, layer two-based mechanisms may not always be available. Hence, RPL relies by default on 6LoWPAN ND for neighbor reachability maintenance.

3.5. RPL ancestors

RPL has inherited a lot of its mechanisms from other protocols. In other words, these protocols can be considered to be RPL ancestors. In the following we will give a short overview on these protocols.

In [35] the authors posed two principles for a routing protocol over wireless sensor networks. These principles are used in design of RPL. The first principle is data path validation, “traffic quickly discovers and fixes routing inconsistencies”, and the second

one is adaptive beaconing, "extending the Trickle algorithm to routing control traffic reduces route repair latency and sends fewer beacons". These principles were presented in the Collection Tree Protocol (CTP) [36] for the first time.

CTP is a tree-based protocol in which nodes construct and maintain one or more minimum-cost trees by sending routing messages (called as beacons). Beacon transmission is based on the Trickle algorithm. Each beacon sent includes an estimate of the sender's route cost to a tree root, collection point. CTP uses ETX as its routing metric to estimate of this cost. The definition of node's cost is the cost of its next hop plus the cost of its link to the next hop, parent, and the route's cost is the sum of the costs of its links. Nodes that operate as collection points advertise a cost of zero. When a node wants to join a tree, based on receiving beacons, it must select the one that has the minimum ETX as its parent, next hop. Whenever a node wants to send data packets to the collection point it sends the data to its parent and the next hop does the same until the packets reach the collection point. This approach allows to carry out the transmission of multipoint-to-point traffic. However, CTP does provide any mechanism for point-to-point traffic and also it is not an IP-based protocol. Another difference between CTP and RPL is that CTP is link layer-dependent. CTP expects that the data link layer provides the following mechanisms:

- An efficient local broadcast address.
- Synchronous acknowledgments for unicast packets.
- A protocol dispatch field to support multiple higher-level protocols.
- Single-hop source and destination fields.

Another protocol from which RPL has inherited ideas was proposed in an IPv6-based network architecture for WSNs presented in [37]. The operation of this protocol is similar to that of CTP, e.g. control message transmission is based on the Trickle algorithm and parent selection is based on ETX and Hop count. However, this protocol is an IP-based protocol and also the roots store routing tables for all destinations that have sent a packet to the root while joining the network. Each datagram sent includes a Record Route Option that records the nodes visited on the way towards the root. After receiving a datagram, the root reverses the Record Route Option and creates an entry for destination received in its routing table, i.e. like in the non-storing mode of RPL. Another proposed mechanism deals with inconsistencies. A node that detects an

inconsistency must reset its Trickle timer in order to trigger a local repair to resolve the problem. RPL uses the same mechanism.

3.6. Routing in other LLN technologies

In this section we overview other routing technologies for LLNs. However none of these technologies are IP-based.

3.6.1. Routing in ZigBee

ZigBee is a wireless networking technology developed by the ZigBee Alliance for low-data rate and short-range applications [38]. The ZigBee protocol stack is composed of four main layers: the physical (PHY) layer, the medium access control (MAC) layer, the network (NWK) layer, and the application (APL) layer. In addition, ZigBee provides security functionality across layers (see Figure 4.a). The two lower layers of the ZigBee protocol stack are defined by the IEEE 802.15.4 standard, while the rest of the stack is defined by the ZigBee specification.

ZigBee defines three device roles:

- The ZigBee coordinator, which corresponds to an IEEE 802.15.4 PAN coordinator
- The ZigBee router
- The ZigBee end device

The latter is normally a simple device with very low capabilities. The ZigBee NWK layer specifically supports addressing and routing for the tree and mesh topologies. The tree topology, which is adequate for data collection, is rooted at the ZigBee coordinator. This scheme includes a mechanism for address assignment, which also facilitates multihop data delivery.

In a mesh topology, routes are created on demand and are maintained using a set of mechanisms based on the ad hoc on-demand distance vector (AODV) routing protocol [22] . This solution is used for arbitrary point-to-point traffic. The ZigBee PRO solution also offers many-to-one routing for communication between several devices and a central controller or sink node. This node may reply back to the devices using source routing. Only ZigBee coordinators and routers participate in routing operations.

3.6.2. Routing in Z-Wave

Z-Wave is a wireless protocol architecture developed by ZenSys (now a division of Sigma Designs) and promoted by the Z-Wave Alliance for automation in residential and light commercial environments [39]. The main purpose of Z-Wave is to allow reliable transmission of short messages from a control unit to one or more nodes in the network [39]. Z-Wave is organized according to an architecture composed of five main layers: the PHY, MAC, transfer, routing, and application layers (see Figure 4.b).

Z-Wave defines two types of devices: controllers and slaves. Controllers poll or send commands to the slaves, which reply to the controllers or execute the commands.

The Z-Wave routing layer performs routing based on a source routing approach. When a controller transmits a packet, it includes the path to be followed in the packet. A packet can be transmitted over up to four hops, which is sufficient in a residential scenario and hard-limits the source routing packet overhead. A controller maintains a table that represents the full topology of the network. A portable controller (e.g., a remote control) tries first to reach the destination via direct transmission. If that option fails, the controller estimates its location and calculates the best route to the destination. Slaves may act as routers. Routing slaves store static routes (typically toward controllers) and are allowed to send messages to other nodes without being requested to do so.

Slaves are suitable for monitoring sensors, in which the delay contributed by polling is acceptable, as well as for actuators that perform actions in response to activation commands. Routing slaves are used for time-critical and non-solicited transmission applications such as alarm activation.

3.6.3. Routing in INSTEON

INSTEON [40] is a solution developed for home automation by SmartLabs and promoted by the INSTEON Alliance. One of the distinctive features of INSTEON is the fact that it defines a mesh topology composed of RF and power line links. Devices can be RF-only or power-line-only, or can support both types of communication.

INSTEON devices are peers, which means that any of them can play the role of sender, receiver, or relayer. Communication between devices that are not within the same range is achieved by means of a multihop approach that differs in many aspects from traditional techniques. All devices retransmit the messages they receive, unless

they are the destination of the messages. The maximum number of hops for each message is limited to four (as in Z-Wave). The multihop transmission is performed using a time slot synchronization scheme, by which transmissions are permitted in certain time slots, and devices within the same range do not transmit different messages at the same time. These time slots are defined by a number of power line zero crossings. RF devices not attached to the power line can transmit asynchronously, but the related messages will be retransmitted synchronously by RF devices attached to the power line. In contrast to classical collision avoidance mechanisms, devices within the same range are allowed to transmit the same message simultaneously. This approach, which is called simulcast, relies on the very low probability of multiple simultaneous signals being cancelled at the receiver.

3.6.4. Routing in Wavenis

Wavenis is a wireless protocol stack developed by Coronis Systems for control and monitoring applications in several environments, including home and building automation. Wavenis is currently being promoted and managed by the Wavenis Open Standard Alliance (Wavenis-OSA). It defines the functionality of physical, link, and network layers [41]. Wavenis services can be accessed from upper layers through an application programming interface (API) (see Figure 4.c).

Wavenis defines only one type of device. The Wavenis network layer specifies a four-level virtual hierarchical tree. The root of the tree may play the role of a data collection sink or a gateway, for instance. A device that joins a Wavenis network intends to find an adequate parent. For this purpose, the new device broadcasts a request for a device of a certain level and a sufficient quality of service (QoS) value. The QoS value is obtained by taking into consideration parameters such as received signal strength indicator (RSSI) measurements, battery energy, and the number of devices that are already attached to this device.

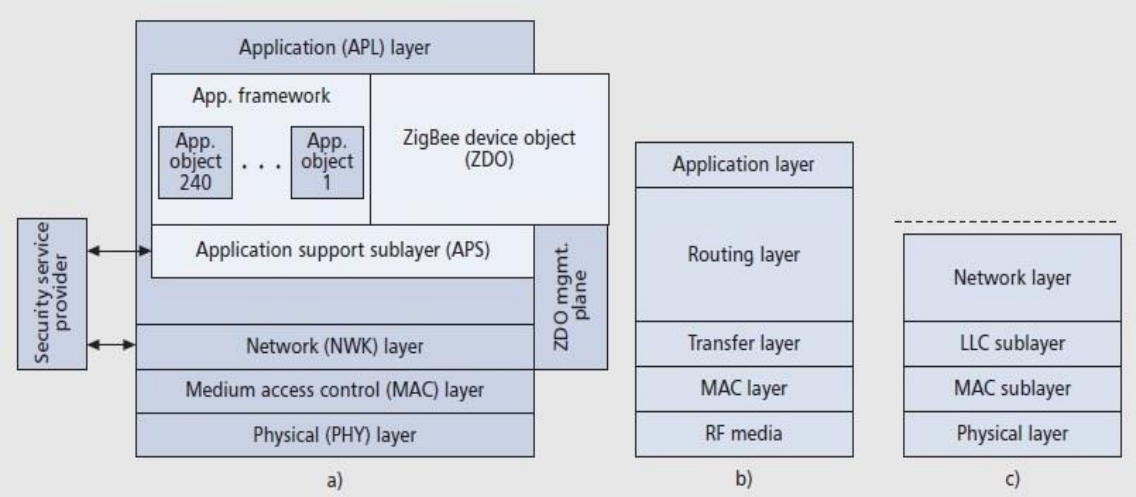


Figure 3.4. Protocol architectures: a) ZigBee; b) Z-Wave; c) Wavenis [12].

4. Network convergence process in RPL over IEEE 802.15.4 multihop networks: improvement and trade-offs

This chapter first studies the influence of two important RPL parameters: the redundancy constant, *k*, and the minimum interval, I_{min}, on the network formation process in different network sizes with different densities in RPL networks. We also propose and evaluate a mechanism that leverages an option available in RPL for accelerating the network convergence process.

This chapter is organized in six parts as follows. Section 4.1 introduces and provides the motivation for this chapter. Section 4.2 describes the simulation environment and methodology used in the chapter. Section 4.3 studies the impact of the main RPL parameters on performance of the network convergence process. Section 4.4 evaluates a mechanism proposed by the author for accelerating network convergence, leveraging an option available in RPL. Section 4.5 reviews related work with this chapter, finally, Section 4.6 presents the main conclusions of this chapter.

4.1. Introduction

Despite its novelty, RPL has already been a subject of study [42-53]. Most of the literature focuses mainly on evaluating RPL behavior in steady state [48-53]. However, performance of RPL during network convergence may be critical, since it may significantly affect user experience (e.g. when a user expects fast network creation to fulfill a certain action) and it is fundamental to global network recovery due to topology changes. Nevertheless, RPL network convergence has received limited attention. The studies that consider RPL performance in transient state do not provide a deep analysis, since they do not focus on the joint influence of RPL parameters and mechanisms, and network characteristics (such as network size and density), on network convergence performance [42-49, 53].

To this end, as we mentioned before, in this chapter we investigate by simulation the influence of the main RPL parameters, the redundancy constant (i.e. the main parameter of the Trickle suppression mechanism) and the value of I_{min} (the smallest size of the Trickle intervals), on the network convergence process, and we also propose and

evaluate a mechanism that leverages an option available in RPL, the DIS messages, for spurring the network convergence process. The performance parameters we consider are network convergence time, network join time and message overhead. In order to assist the derivation of conclusions, we also evaluate the number of collisions during network convergence. With the aim of obtaining comprehensive results, we carry out extensive simulations for a wide range of conditions in terms of network size and density. We consider multihop networks whereby the nodes use IEEE 802.15.4, i.e., the most prevalent radio interface for WSNs. In order to achieve realistic results, we have tuned the simulation environment in order to accurately model the link behavior observed in real experiments.

Results show that RPL network convergence performance depends dramatically on the use and adequate configuration of key parameters and mechanisms. The findings and contributions of this chapter provide a guideline for configuring and selecting adequately crucial RPL parameters and mechanisms for achieving high network convergence performance, on the basis of network characteristics such as size and density, as well as a characterization of the related performance trade-offs.

4.2. Simulation environment and methodology

This section presents the simulation environment and methodology used to evaluate the network convergence process of RPL in IEEE 802.15.4 multihop networks.

In order to carry out our simulations, we used OMNeT++ [54], a well-known C++-based discrete event simulator, jointly with MiXiM, an OMNeT++ framework created for various types of wireless networks [55]. We implemented RPL for this simulation environment. As a side contribution, we have made the simulation code publicly available [56]. The simulated network nodes were static and located following a uniformly random spatial distribution in two-dimensional square areas. We configured the nodes to use the 2.4 GHz band IEEE 802.15.4 physical layer, and the beaconless mode functionality. We set the MAC queue length of the nodes to 1 in order to replicate the value of this parameter in CC2420, a widely used IEEE 802.15.4 radio chip [57]. We used the log-normal shadowing propagation model available in MiXiM. In order to achieve accurate and realistic link behavior in our simulation framework, we carried out experiments and tuned the parameter settings of the simulated propagation model based on the experimental results. The experiments comprised communication in a link

between two TelosB nodes [58] running TinyOS [59]. Table 4.1 shows the main physical and link layer parameter settings used in the simulations.

Table 4.1. Main physical and link layer simulation parameters.

Parameter	Value
Communication range (m)	9.96
Carrier frequency (GHz)	2.4
Carrier sense sensitivity (dBm)	-95
Transmit power (dBm)	-25
MAC queue length	1

In order to evaluate the influence of the network size in our study, we considered 3 different scenarios, denoted small, medium and large network scenarios. The area of the large network scenario is 100 x 100 m^2, which is 5 times greater than the medium network area. There is the same relationship between the medium and small network scenario areas. We also considered the network density in our study, by evaluating node degree values of 5, 10 and 15. Note that this range of network densities covers from relatively sparse networks to highly connected and dense networks. As a result, we simulated a variety of network sizes and densities, ranging from 8-node to 483-node networks, which cover a wide range of use cases. The number of nodes and the average rank for each network size and density scenario are shown in Table 4.2.

For each individual combination of network size scenario, network density and

Table 4.2. Number of nodes and average rank for each network size scenario and density.

Network size	Node degree	Number of nodes	Average rank
Small	5	8	3.09
	10	14	3.30
	15	21	3.30
Medium	5	34	6.52
	10	66	6.34
	15	99	5.78
Large	5	162	16.76
	10	322	12.43
	15	483	10.74

protocol configuration, we simulated 1500 different randomly generated topologies, and evaluated 20 instances of the network convergence process for each topology, yielding a total of 30000 network convergence instances. We discarded the results of DODAGs not formed in less than 10000 seconds. The minimum and maximum percentage of times that DODAGs formed in less than 10000 seconds are 50.14% and 99.97%, which correspond to the large network scenario when the node degree is 5, and the small network scenario when the node degree is 15, respectively.

4.2. Evaluation: influence of the main RPL parameters on network convergence performance

In this section we study the influence of the two most crucial RPL parameters on the performance of the network convergence process, in the network scenarios presented in the previous section. The two considered RPL parameters are the redundancy constant, *k*, and the minimum interval, I_{min}. This section comprises two subsections, which focus respectively on the impact of each one of the aforementioned RPL parameters on network convergence performance.

4.2.1. Influence of the redundancy constant on the network convergence process

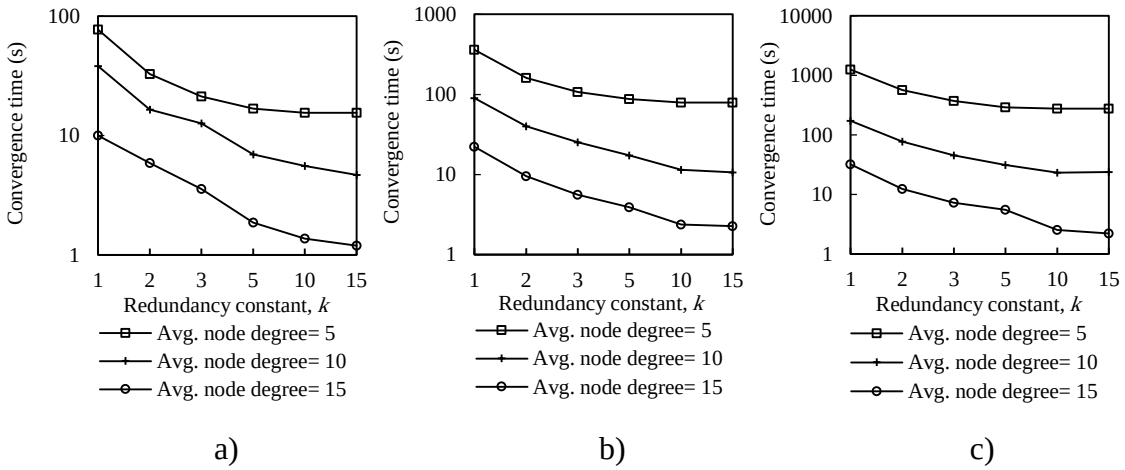
One of the RPL parameters that affects network convergence performance to a greatest extent is the redundancy constant, *k*. As presented in section 3.2, this parameter limits the number of transmitted messages per Trickle interval in a given coverage area. In order to analyze the influence of the redundancy constant on the network convergence process, we simulated the network size and density scenarios shown in Table 4.2 for a range of redundancy constant values between 1 and 15, which includes the default value for this parameter stated by the RPL specification (i.e. $k = 10$) [5]. For the results presented in this subsection, the rest of RPL parameters were set to the default values stated in the RPL specification. We evaluated network convergence time, DODAG join time of a node, number of DIO messages transmitted, and number of collisions.

Figure 4.1 shows average network convergence time in small, medium and large network scenarios, for different node density and *k* values. To obtain the convergence

time of a network, we calculated the time from the instant in which the DODAG root starts its first Trickle interval until the instant in which all nodes have joined the DODAG. We consider that a non-root node joins a DODAG when it receives a DIO message for the first time.

As it can be seen in Figure 4.1, using low values for the redundancy constant yields high network convergence time in all network size and density scenarios considered. This behavior is mainly due to the message suppression mechanism in the Trickle algorithm. When the root node transmits its first DIO message, the neighbors of the root that receive this DIO message schedule themselves to send their own first DIO message. However, when *k* is set to a low value, only a few neighbors of the root will be allowed to transmit their DIO message. If the rest of root neighbors (if any) hear a number of DIO messages greater than or equal to *k*, they will suppress their DIO message transmission in the current interval. As a result, nodes that are only neighbors of the latter root neighbors have to wait for subsequent intervals to have the opportunity to receive their first DIO message. The same phenomenon happens as DIO messages propagate through the network, which finally leads to high network convergence time.

As the redundancy constant increases up to medium values, the network convergence time decreases. This occurs because, with greater redundancy constant values, the probability that nodes suppress their DIO messages is lower. Therefore, the probability that a node receives its first DIO message earlier is greater. However, network convergence times do not vary significantly as the value of *k* increases beyond



a) b) c)

Figure 4.1. Average network convergence time in three network size scenarios, for different densities, as a function of the redundancy constant, k: a) Small network scenario, b) medium network scenario, and c) large network scenario, respectively.

the values around the node degree. The reason for this behavior is that for such values of *k*, the number of DIO message suppressions is low, since many nodes have a number of neighbors lower than *k*, and thus varying *k* within the mentioned range does not significantly affect the network convergence time.

As is also shown in Figure 4.1, network convergence time decreases with the node degree since greater network density provides better network connectivity and a greater amount of opportunities for a node to receive DIO messages from its neighbors.

By analyzing Figure 4.1 results in deeper detail, we found that the relative impact of the redundancy constant on network convergence time does not vary significantly with the network size when networks are sparse. However, as the network density increases, the improvement that can be achieved by increasing the redundancy constant grows with the network size. For example, when the node degree is 15, in the small size scenario, a network convergence time decrease factor of 8.3 can be achieved by using $k = 15$ in comparison with the result obtained for $k = 1$. However, for the same node density, the network convergence time decrease factor that can be achieved in the large network scenario is 14.5.

Figures 4.2, 4.3 and 4.4 show the Cumulative Distribution Function (CDF) of the network convergence time for all the network size and density scenarios considered. The results illustrated by these figures evidence the dramatic impact of the redundancy constant on network convergence time. Increasing the redundancy constant value causes the percentage of DODAGs formed in a given time to increase as well. For example, in the medium network scenario, when the node degree is 5 (Figure 4.3.a)), and *k* is equal to 1, more than 80% of the DODAGs are formed in less than 120 s, while the same ratio of DODAGs are formed in less than 18 s for $k \geq 2$. As the network density grows, network convergence times decrease as well. Still in the medium network scenario, when the node degree is 10 (Figure 4.3.b)), and *k* is set to 1, 80% of the DODAGs are formed in less than 2 s, whereas the same ratio of DODAGs are formed in less than 0.6 s for $k \geq 2$. In the same scenario, for a node degree of 15 (Figure 4.3.c)), 90% of the DODAGs are formed for any value of *k* in less than 0.6 s.

On the other hand, the reader may observe the wave shape that can be appreciated in Figures 4.2, 4.3 and 4.4, especially when the node degree is low. This behavior is due to the interval duplication of the Trickle algorithm, and reflects the distribution of the

number of intervals needed to completely form a DODAG, as well as the size of the interval in which a DODAG is formed.

Another useful performance result is the time it takes for a node to join a DODAG. Figure 4.5 shows the CDF of the DODAG join time for a node in the considered network scenarios. When the average node degree is 5 and *k* is equal to 1, around 80% of the nodes have joined the DODAG in 0.1 s, 1.1 s and 15 s in the small, medium and large network size scenarios, respectively. These times are significantly smaller than the average network convergence times shown in Figure 4.1. Therefore, a relatively small fraction of the nodes require a large amount of time to join a DODAG. Whether this is a critical aspect for the user depends on the application requirements the network is deployed for.

As shown in Figures 4.2-4.5, the difference between the curves for $k = 1$ and for $k = 2$ is greater than the difference between the curve for $k = 2$ and any curve for $k > 2$. This result illustrates that the network convergence time and join time improvement that can be achieved by increasing the redundancy constant decreases as the redundancy constant grows.

In order to provide insight on the influence of the redundancy constant on network convergence performance, we also evaluated the average number of DIO message transmissions and collisions that occur from the first DIO message transmission by the root until the instant in which all nodes join the DODAG, for all the network sizes and densities considered. The corresponding results are shown in Figure 4.6. The average number of DIO message transmissions increases as the redundancy constant grows up to the node degree. In fact, increasing the redundancy constant within this range of values leads to a lower number of DIO message suppressions. When the redundancy constant is greater than the node degree, varying this RPL parameter does not have a significant effect on the number of DIO messages transmitted, since the number of message suppressions is very low. The number of collisions varies with *k* similarly to the number of DIO messages transmitted, since the number of DIO message collisions grows with the number of DIO messages that are actually transmitted. Note that despite the number of DIO message collisions increase with the redundancy constant, the network convergence time decreases with the redundancy constant. This happens because, as the redundancy constant grows, even though a subset of the additional DIO messages transmitted are lost due to collisions, a greater number of DIO messages are received

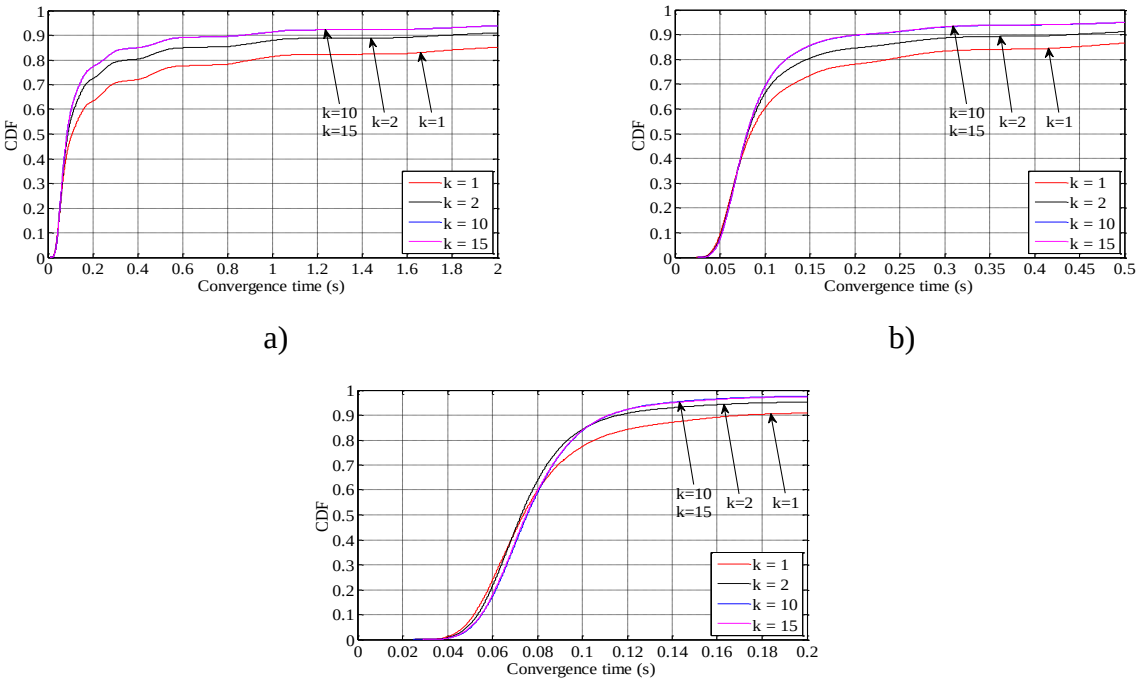


Figure 4.2. CDF of network convergence time in the small network size scenario. The average node degrees are 5, 10 and 15 in figures a), b) and c), respectively.

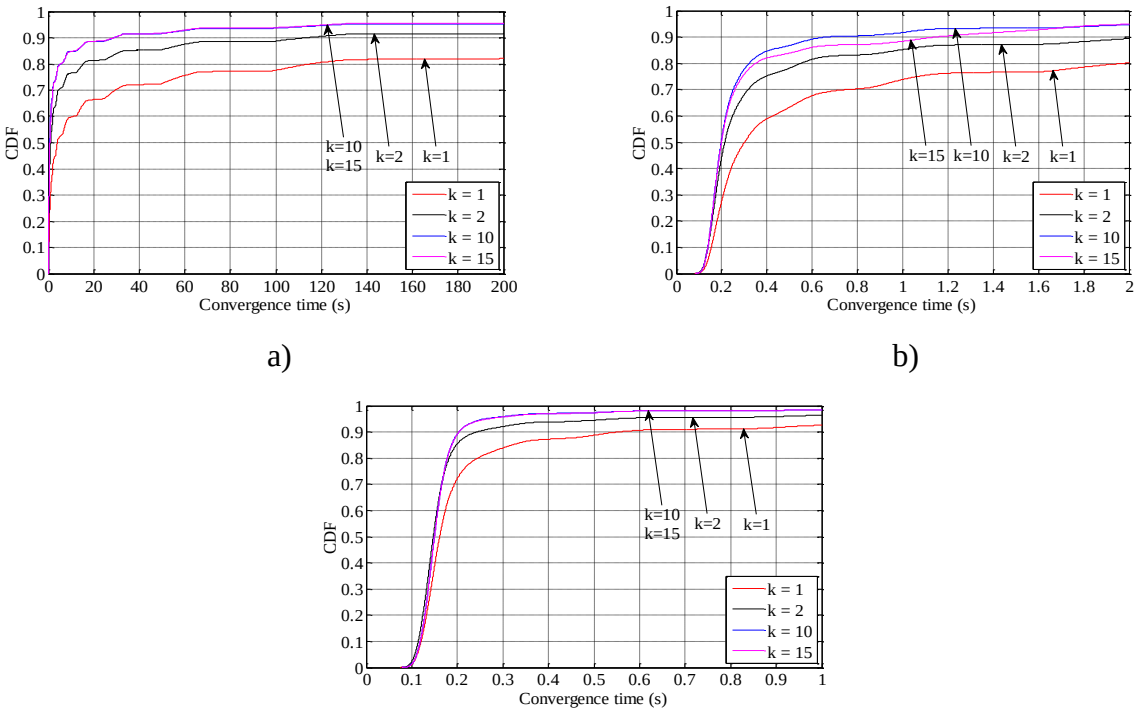


Figure 4.3. CDF of network convergence time in the medium network size scenario. The average node degrees are 5, 10 and 15 in figures a), b) and c), respectively.

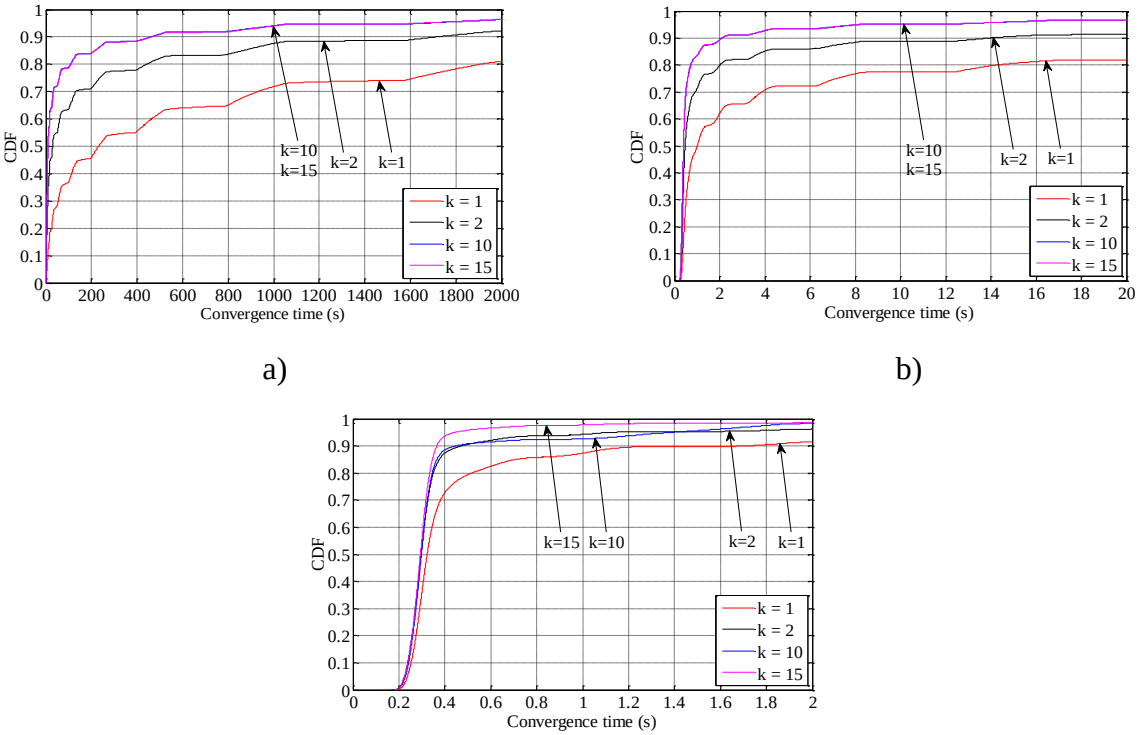


Figure 4.4. CDF of network convergence time in the large network size scenario. The average node degrees are 5, 10 and 15 in figures a), b) and c), respectively.

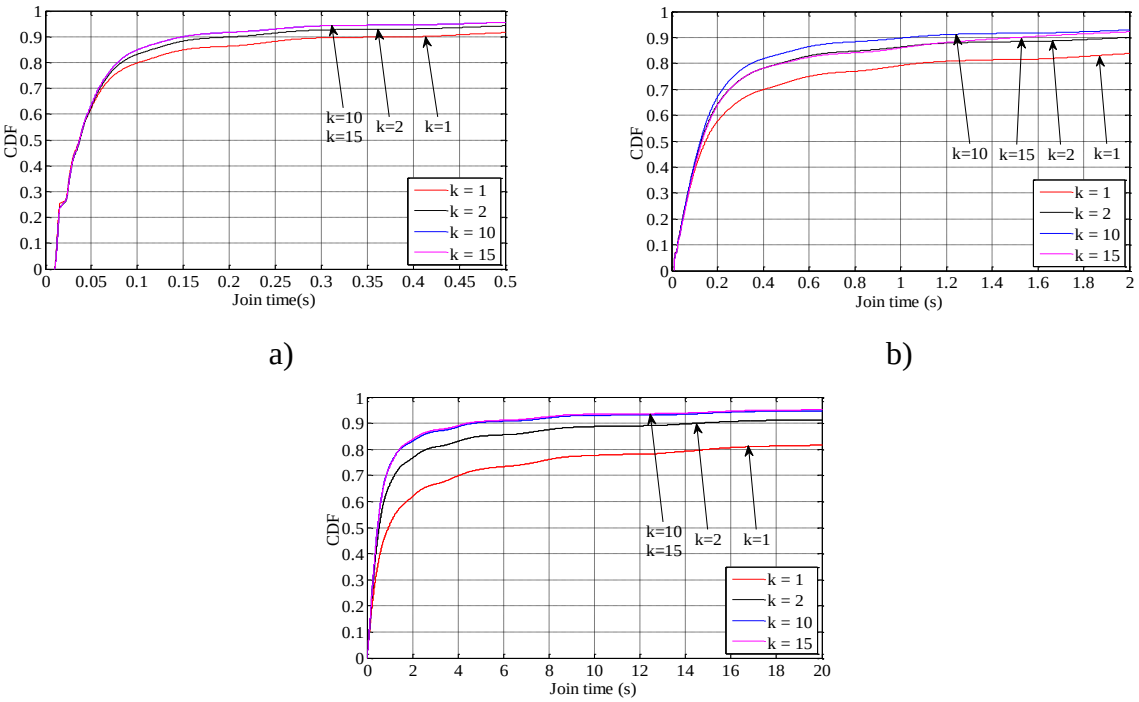


Figure 4.5. CDF of nodes join time for an average node degree of 5: a) small, b) medium, and c) large network size scenarios.

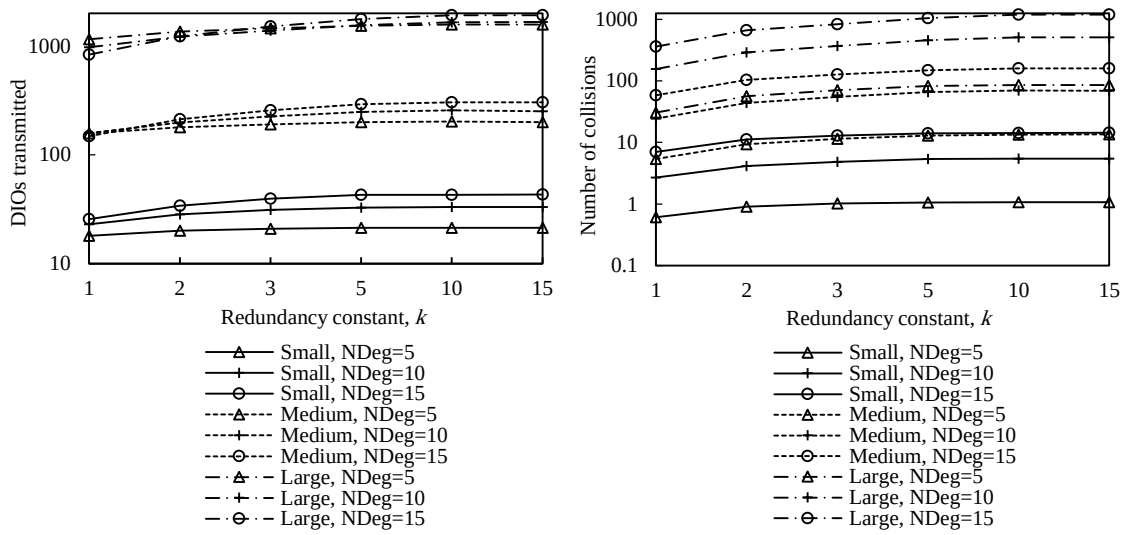


Figure 4.6. Average number of DIO messages transmitted (left) and collisions (right) in three network size scenarios, for different densities, as a function of the redundancy constant, k. NDeg denotes the average node degree.

correctly by non-DODAG nodes. Finally, we found that the relative influence of *k* on the number of DIO messages transmitted, and to a lower extent, on the number of collisions, increases with both network size and network density. As network size grows, the influence of *k* increases due to the multiplicative effect of a greater hop count between the root and any other node.

From the results shown in Figures 4.1-4.6, it can be concluded that there is a tradeoff between network convergence time (and node join time) and the number of DIO messages sent, which depends on the redundancy constant. In order to achieve a low network convergence time, the redundancy constant should be set to a value equal or close to the average node degree, since greater redundancy constant values might cause an unnecessary increase in the number of DIO messages transmitted and collisions in very dense areas of a network, which however would not yield a network convergence time decrease. The influence of the redundancy constant grows with both the network density and size.

4.2.2. Influence of the minimum interval on the network convergence process

In this subsection we study the effect of the minimum interval, I_{min}, on the network convergence process. We evaluate the impact of I_{min} on the network convergence time, number of DIO message transmissions and number of collisions during the network

convergence process. We also analyze the relationship among these performance parameters. The RPL specification states that the default value for I_{min} is 8 ms [5]. However, in order to investigate the influence of I_{min} on network convergence performance, we evaluate different I_{min} values in the same scenarios and for the same redundancy constant values considered in the previous subsection.

Figure 4.7 shows the average network convergence time when I_{min} is set to 4 ms, 8 ms and 16 ms in the scenarios and for the settings presented in Section 4.2. As it can be seen in Figure 4.7, decreasing the minimum interval yields a reduction of network convergence time regardless of the network size, the network density or the redundancy constant, for the range of I_{min} values considered. The reason is that when I_{min} decreases, the length of Trickle intervals decreases as well, therefore nodes send their DIO messages earlier and more frequently, and as result a DODAG forms faster. Another important consideration that can be inferred from Figure 4.7 is that halving the I_{min} value decreases the network convergence time by a factor smaller than two. As we already mentioned, when I_{min} decreases, the number of DIO message transmissions during network convergence increases; as a consequence, the number of collisions increases as well (see Figures 4.8-4.10). The number of collisions increase does not

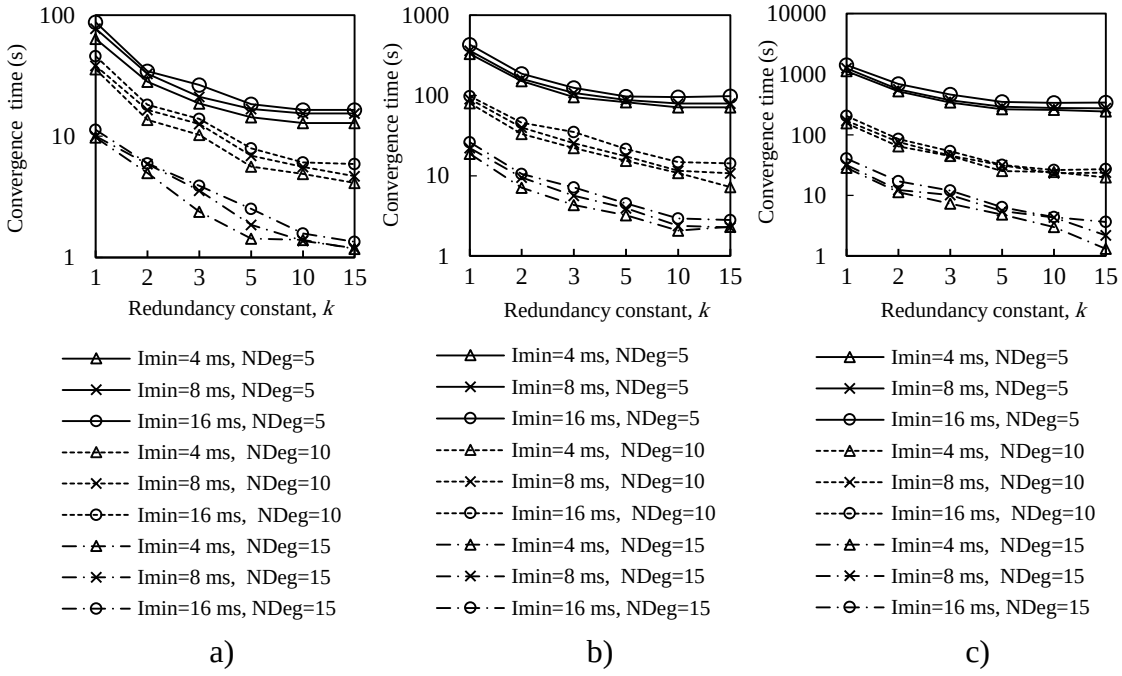
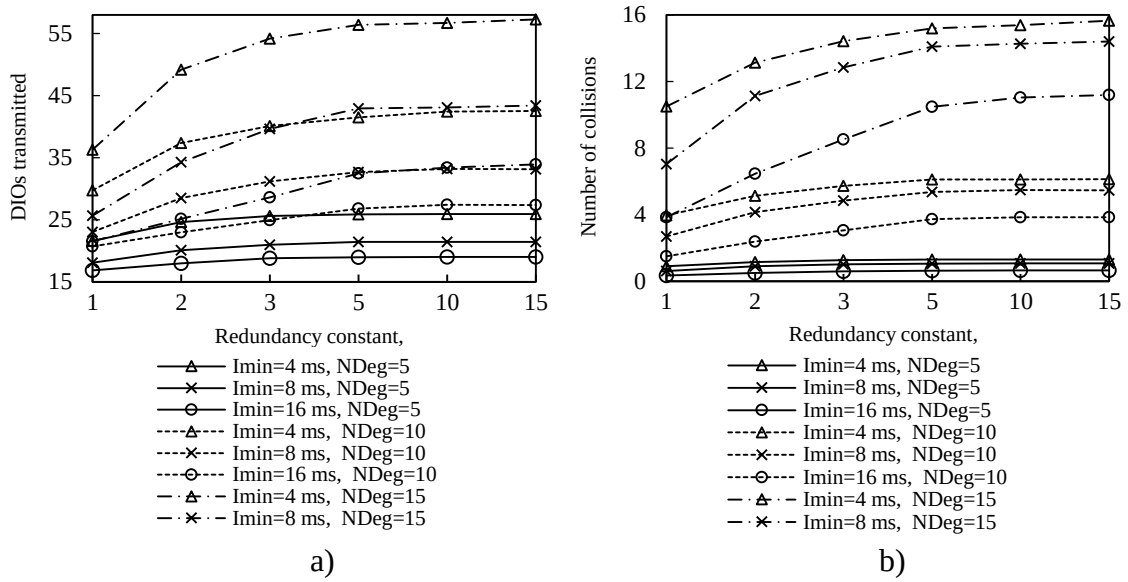
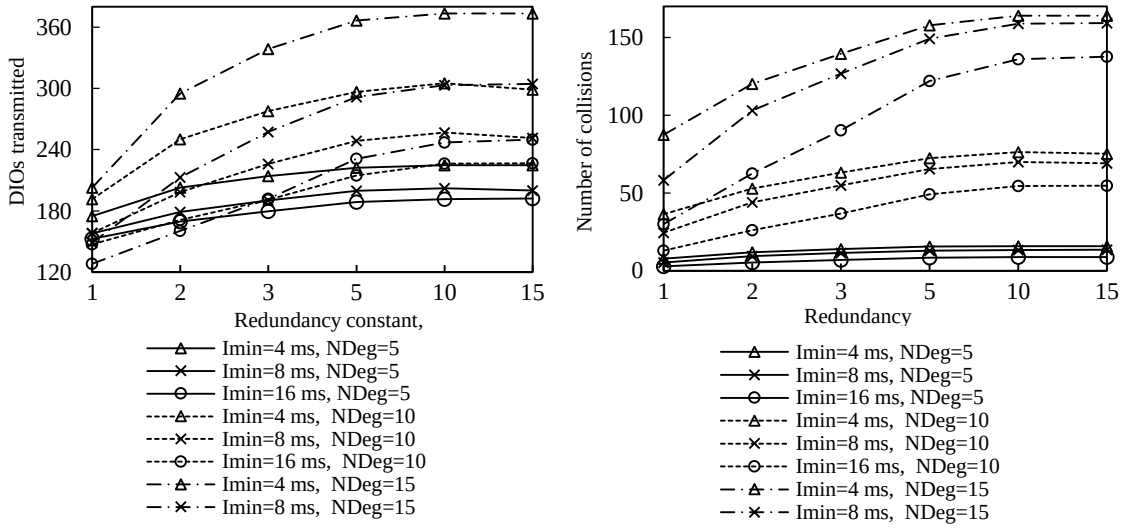


Figure 4.7. Influence of Imin on the network convergence time in the small, medium and large network scenarios with different average node degrees as a function of the redundancy constant, k, when the I_{min} value is set to 4 ms, 8 ms and 16 ms. a) small, b) medium, and c) large network size scenarios.



a) b)

Figure 4.8. Average number of DIOs transmitted and collisions in the small network scenario for different average node degrees, as a function of the redundancy constant, k, based on the different values of I_{min}.

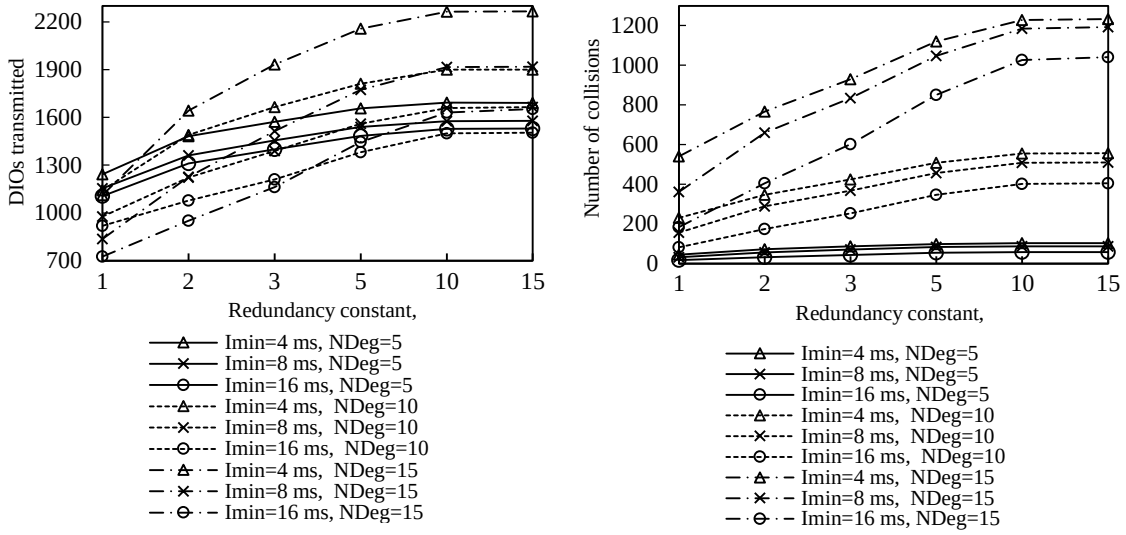


a) b)

Figure 4.9. Average number of DIOs transmitted and collisions in the medium network scenario for, different average node degrees, as a function of the redundancy constant, k, based on the different values of I_{min}.

avoid the network convergence time decrease with I_{min}, but it is the reason why this decrease is smaller than the I_{min} decrease.

Analyzing in deeper detail the number of DIO messages transmitted during network convergence (Figures 4.8.a), 4.9.a) and 4.10.a)), we observe that as both the redundancy constant and the network density grow, the influence of I_{min} becomes greater. This is due to a lower amount of DIO message suppressions in the network. As a consequence



a) b)

Figure 4.10. Average number of DIO messages transmitted and collisions in the large network scenario for different average node degrees, as a function of the redundancy constant, k, for different values of I_{min}.

of a greater number of DIO message transmissions, the number of collisions also grows with I_{min} (see Figures 4.8.b, 4.9.b) and 4.10.b)). The relative influence of I_{min} on the number of collisions does not vary with the redundancy constant, except for a high network density, whereby as the redundancy constant grows, the influence of I_{min} becomes smaller. This happens because for a large I_{min} value (e.g. I_{min} = 16 ms), the number of collisions is low for low redundancy constant values. However, as the redundancy constant grows, and when the network is dense, the probability of collision significantly increases due to the greater number of DIO message transmissions in the network. On the contrary, for low I_{min} values (e.g. I_{min} = 4 ms), the number of collisions is already high for low redundancy constant values, since such I_{min} values approach the DIO message transmission time in the 2.4 GHz IEEE 802.15.4 physical layer (i.e., 2.82 ms), and thus the number of collisions increase with the redundancy constant is smaller than that observed for greater I_{min} values.

From the study carried out in this subsection, we conclude that there exists a tradeoff between network convergence time and DIO message overhead (and collisions) that depends on I_{min}. Varying I_{min} in the range of values considered in this study has a greater quantitative effect on the number of DIO messages sent than on the network convergence time. Fine-tuning I_{min} has a greater impact on the number of DIO messages transmitted during network convergence as both the redundancy constant and network

density grow, whereas the relative impact of I_{min} on network convergence time does not vary significantly with the redundancy constant, network size or network density.

4.3. DIS-Trickle: a mechanism for accelerating network convergence

This section proposes and evaluates a mechanism for leveraging DIS messages in order to accelerate RPL network convergence. In some scenarios a node may have to wait for a long time to receive a DIO message (and thus, to join a DODAG). As we described in section 3.1.1, a node may discover nearby DODAG nodes in a short time by sending DIS messages, which will trigger a quick response (in the form of a DIO message) from neighbors that have already joined a DODAG. However, the RPL specification does not state any rule or scheduling algorithm for the transmission of DIS messages [5]. In this section we propose and evaluate DIS-Trickle, a mechanism for accelerating the network convergence process by using DIS messages. We first present DIS-Trickle and then evaluate its influence on network convergence performance.

4.3.1. DIS-Trickle design

DIS-Trickle comprises two components: an initial delay and a scheduling algorithm. This subsection describes in detail both components.

For nodes that may send DIS messages, it is beneficial to introduce an initial delay before the first DIS message transmission. This is motivated by the fact that DIO messages need time to propagate through the network, and thus nodes at a multihop distance from the root node must have the opportunity of receiving DIO messages before sending premature, and unnecessary, DIS messages. The value of the initial delay has to be set carefully on the basis of the network size and density. Such an initial delay is also used in the COOJA/ContikiRPL implementation, whereby it is set by default to 5 s [30]. However, 5 s may be a too large value for the characteristics of many networks, including the ones considered in this chapter. Based on simulation results that are discussed later in subsection 4.3.2, we set the initial delay to 200 ms, which yields good performance in different network size and density scenarios, in terms of network convergence time, number of DIS and DIO messages transmitted, and collisions.

After the initial delay, in network zones where nodes have not yet received any DIO message, these nodes may attempt to transmit DIS messages at the same time,

which may yield an *initial DIS storm* and a high number of collisions. On the other hand, it is not necessary that all non-DODAG neighbors of a DODAG node send a DIS message to trigger a DIO message transmission from the latter, since all these neighbors have the opportunity of receiving a DIO message sent by the DODAG node. In order to solve the *initial DIS storm* problem, as well as to limit unnecessary DIS message transmissions, we propose applying the Trickle algorithm to schedule the transmission of DIS messages. Since the main purpose of using DIS messages is the decrease of nodes' DODAG join time, we disable the Trickle interval size doubling mechanism in DIS-Trickle. On the other hand, in order to decrease the number of DIS message transmissions and the probability of message collisions while sending DIS messages, we set the redundancy constant of DIS-Trickle, k_{DIS}, to 1. Another important parameter of DIS-Trickle is the interval size, denoted I_{DIS}, which defines the time between consecutive DIS message transmissions. The value of this parameter should be specified on the basis of the DIS response time, defined as the time between the transmission of a DIS message and the corresponding DIO message reply from a DODAG member. On the other hand, the time between consecutive DIS messages should not be unnecessarily large.

In order to assist the determination of I_{DIS}, we next calculate the minimum and maximum DIS response time, assuming that nodes use a 2.4 GHz band IEEE 802.15.4 interface in the beaconless mode (see Figure 4.11). Let node A, which is interested in joining a DODAG, start the procedure for transmitting a DIS message at time t_1. Let node B be already a DODAG member. The physical transmission of the DIS message starts after the backoff period and the CCA. We should note that, by default, the minimum and maximum backoff times in IEEE 802.15.4 are 0 ms and 16.96 ms, respectively. On the other hand, the time needed for performing the CCA on Telos B

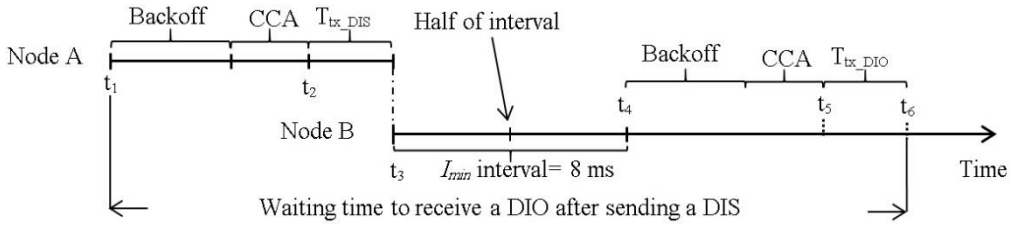


Figure 4.11. Calculation of the DIS response time. The figure illustrates the maximum delay between the transmission of a DIS message and the reception of a DIO message in response, where node A is interested in joining a DODAG and node B is already a DODAG member.

nodes running TinyOS is 3 ms. The physical layer of node A starts to send the DIS message at time t_2, and it takes 1.34 ms (denoted by T_{tx_DIS} in Figure 4.11) to transmit this message. On the other hand, node B, which has started to receive this DIS message at time t_2, finishes receiving the DIS message at time t_3, it immediately resets its Trickle timer to I_{min} (which is set to 8 ms by default) and schedules itself to transmit a DIO message in the second half of the Trickle interval (of size I_{min}). A DIO message transmission can take place at any point of the second half of the current interval, i.e. it can suffer a random delay between 4 ms and 8 ms (the latter is illustrated in Figure 4.11). Finally, the physical transmission of this DIO message (which takes 2.82 ms) starts, after the backoff period and the CCA, at time t_5, and node A finishes receiving the DIO message at time t_6. Considering the duration of all the periods included between t_1 and t_6, the DIS response time is a value between 14.16 ms and 52.08 ms. From the basis of this analysis, we tested different values for I_{DIS}, (see the results obtained and discussed later in subsection 4.3.2), and based on these tests we finally set I_{DIS} to 30 ms, since it yields good performance in terms of both network convergence time and number of DIO and DIS messages transmitted during network convergence. Table 4.3 shows the default values of the DIS-Trickle parameters used in our simulations.

Table 4.3. DIS-Trickle parameter configuration

Parameter	Value
Initial delay (ms)	200
Interval length, I_{DIS} (ms)	30
Number of doublings	0
DIS-Trickle redundancy constant, k_{DIS}	1

4.3.2. DIS-Trickle evaluation

In this subsection, we evaluate DIS-Trickle and discuss the results obtained. We assume the default value of I_{min} = 8 ms, analyze the influence of the redundancy constant on network convergence when DIS-Trickle is used, and consider the same range of network density and size scenarios evaluated in Section 4.2. We analyze the network convergence time, total number of RPL (i.e. DIO and DIS) messages

transmitted, and collisions during network convergence. We also study the impact of DIS-Trickle parameters on performance.

As can be seen in Figure 4.12, using DIS-Trickle decreases network convergence time by 2 to 3 orders of magnitude, regardless of the network density and size. Actively requesting DIO messages constitutes a dramatically better strategy for non-DODAG nodes than passively waiting for DIO message reception, in terms of joining quickly a DODAG.

Interestingly, the network convergence time improvement yielded by DIS-Trickle does not always lead to performance degradation in terms of the total number of RPL (i.e. DIO and DIS) messages sent. In dense networks, the RPL message overhead during network convergence does not grow, and even decreases, when DIS-Trickle is used (see Figure 4.13). However, in sparse networks (e.g. when the average node degree is 5), DIS-Trickle requires a greater amount of total RPL message transmissions than that needed in absence of DIS-Trickle; impact of this phenomenon increases with network size. For a deeper analysis of the number of RPL messages sent during network convergence, we next study the number of DIO and DIS messages sent, separately.

Figure 4.14 shows that the average number of DIO messages transmitted decreases when DIS-Trickle is used. This reduction is more significant as network density decreases. The reason is that, as shown in subsection 4.2.1, in a sparse network, when DIS-Trickle is not used, the network convergence time is very high, and during this time a high number of DIO messages are transmitted until a DODAG is formed. However, using DIS-Trickle allows non-DODAG nodes to join a DODAG in a shorter time, and thus fewer DIO messages are required to complete the DODAG construction. On the other hand, when DIS-Trickle is used, the number of DIS message transmissions decreases as the redundancy constant grows (see Figure 4.15). This happens because as k increases, a greater number of DODAG nodes are able to send their DIO messages, therefore non-DODAG nodes have more chances to receive DIO messages, making it unnecessary to send DIS messages. Another important result is that the number of DIS message transmissions increases as network density decreases, since a lower amount of DIO messages are transmitted naturally by DODAG members. However, a large network size exacerbates the number of DIS message transmissions in sparse networks, due to the multiplicative effect of a greater number of end-to-end hops from the root to the most distant nodes. Therefore, in sparse networks, the high number of DIS message

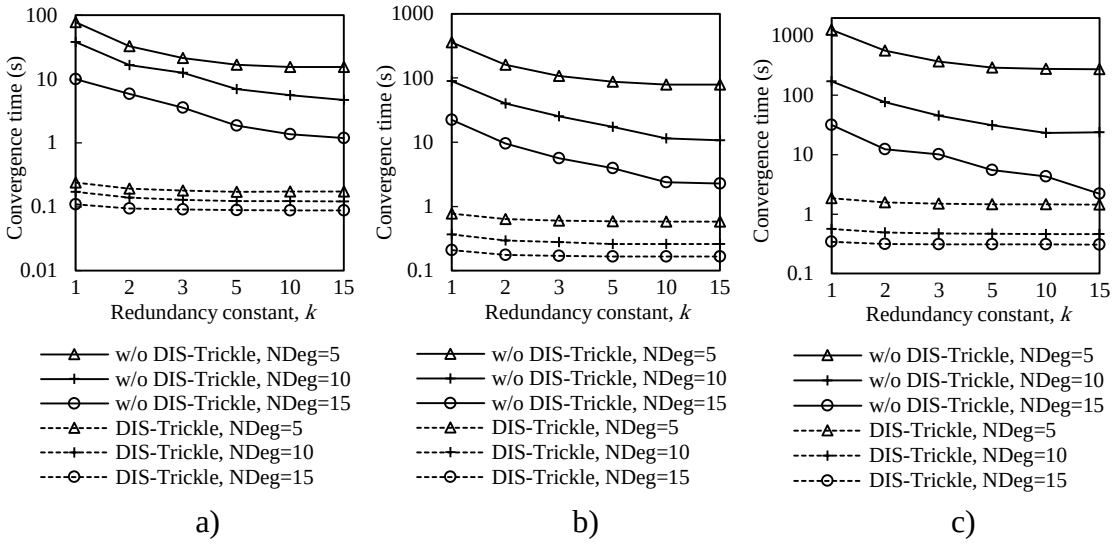


Figure 4.12. Influence of using DIS-Trickle on network convergence time in the small, medium and large network scenarios, for different average node degrees, NDeg, as a function of the redundancy constant, k, when the I_{min} value is set to 8 ms. a) Small network size, b) medium network size, and c) large network size.

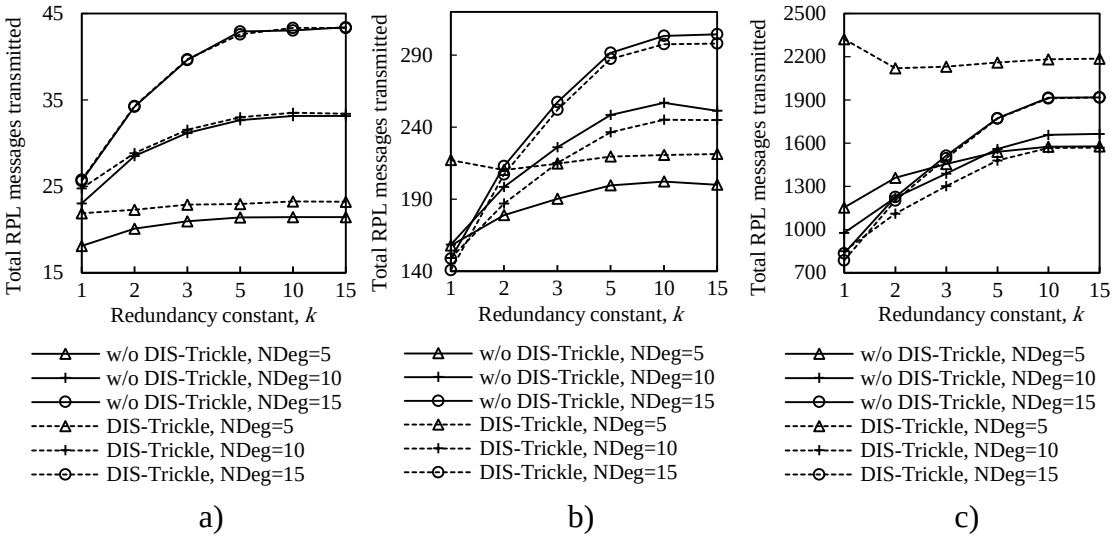


Figure 4.13. Influence of using DIS-Trickle on the total number of RPL messages (i.e. DIO and DIS messages) transmitted in the small, medium and large networks, for different average node degrees as a function of the redundancy constant, k, when the I_{min} value is set to 8 ms. a) Small network size, b) medium network size, and c) large network size.

transmissions dominates the total number of RPL messages transmitted during network convergence, shown in Figure 4.13.

Figure 4.16 illustrates the influence of network density, network size, and the redundancy constant setting on the relative performance of DIS-Trickle (i.e. compared with not using DIS messages) in terms of both network convergence time and number

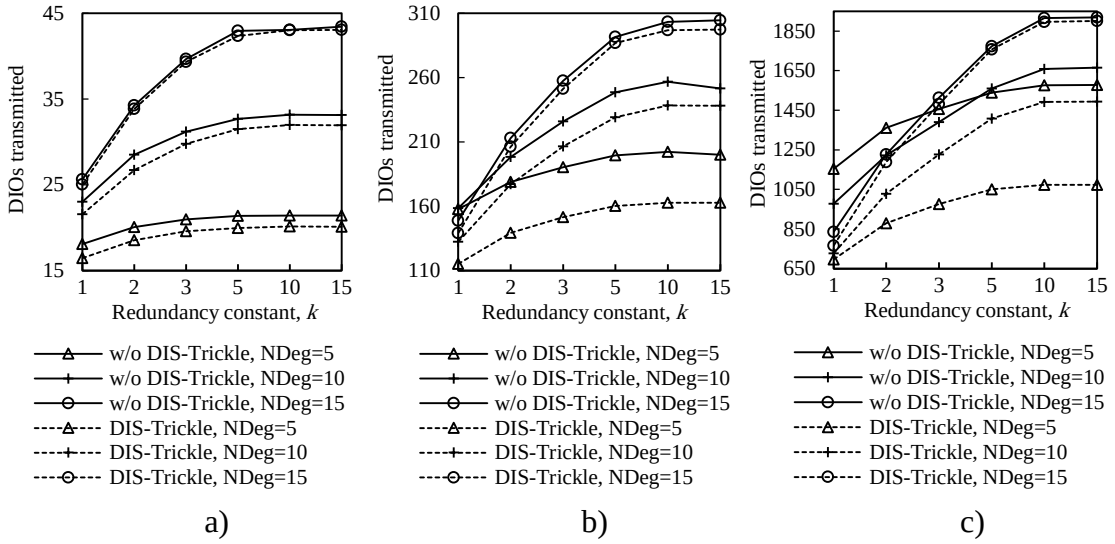


Figure 4.14. Influence of using DIS-Trickle on the number of DIO messages transmitted in the small, medium and large network scenarios, for different average node degrees, NDeg, as a function of the redundancy constant, k, when the I_{min} value is set to 8 ms: a) Small network size, b) medium network size, and c) large network size.

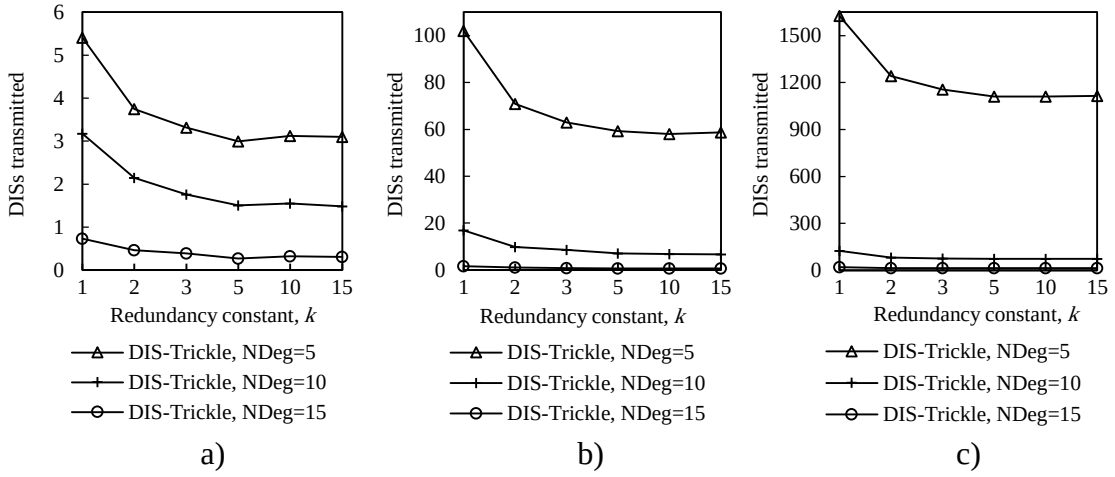


Figure 4.15. Influence of using DIS-Trickle on the number of DIS messages transmitted in the small, medium and large networks, for different average node degrees as a function of the redundancy constant, k. when the I_{min} value is set to 8 ms. a) Small network size, b) medium network size, and c) large network size.

of RPL message transmissions. In sparse networks (e.g. when the average node degree is 5), using DIS-Trickle improves network convergence time, but degrades the total number of RPL messages transmitted, since a high number of DIS message transmissions are required, as aforementioned.

As the network density increases, the network convergence time improvement due to using DIS-Trickle becomes lower, and does not vary significantly with the network size. This occurs because the probability that a node receives DIO messages without the

need to send DIS messages increases with the network density. However, in dense networks, the total number of RPL message transmissions does not grow or even slightly decreases when DIS-Trickle is used. The reason for this result is that, in dense networks, a low number of DIS messages are required, and the DIS messages transmitted help reduce to a similar or even greater extent the number of DIO messages sent. Therefore, in dense networks, DIS-Trickle improves both network convergence time and number of RPL message transmissions, or at least does not degrade this last performance parameter (note that the result obtained in small networks, for a node degree of 10 and $k = 1$ is an exception to this conclusion, whereby the number of RPL message transmissions is slightly degraded when DIS-Trickle is used). Lastly, it can also be observed in Figure 4.16 that the influence of DIS-Trickle decreases as the redundancy constant grows, since a greater amount of DIO messages are naturally transmitted and a lower amount of DIS message transmissions are required.

With regard to the number of collisions, for both DIO and DIS messages, it does not vary significantly by using DIS-Trickle (see Figure 4.17). This occurs because DIS messages are likely to be sent from nodes that have a low number of neighbors, and in addition, DIS-Trickle randomizes DIS message transmission time and suppresses redundant DIS message transmissions. Therefore, when DIS-Trickle is used, the number of message collisions does not significantly increase compared with the one observed in absence of DIS messages.

Finally, in order to understand the influence of I_{DIS} and the DIS initial delay on performance of DIS-Trickle, we carried out simulations with different configurations of these DIS-Trickle parameters. The considered configurations of the aforementioned parameters are shown in Table 4.4.

The evaluation is done in the medium network size scenario, for a node degree of 5. Note that this density corresponds to a sparse network and, as shown in Figure 4.1, the average network convergence time is high in the absence of the DIS-based mechanism, therefore we can observe the maximum influence of DIS-Trickle on network convergence performance within the range of considered network densities. The results obtained for these different configurations are plotted in Figure 4.18, which shows average results for the following performance parameters: network convergence time, number of DIO and DIS messages transmitted, and number of collisions (considering both DIO and DIS messages) during network convergence. Note that Config. 1 (dashed

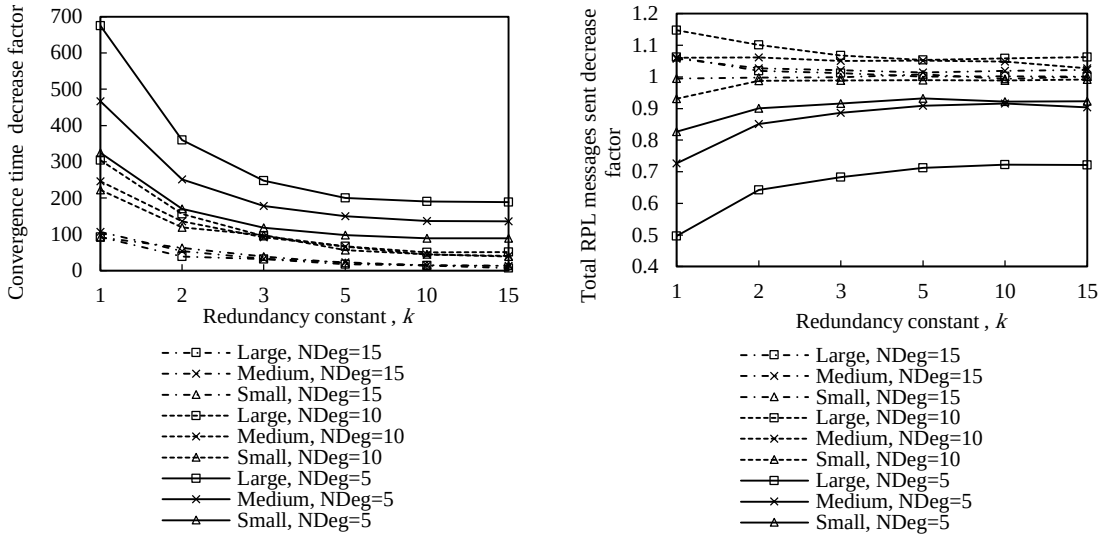


Figure 4.16. Convergence time (left) and number of DIO messages sent (right) improvement by using DIS-Trickle in three network size scenarios, for different average node degrees as a function of the redundancy constant, k.

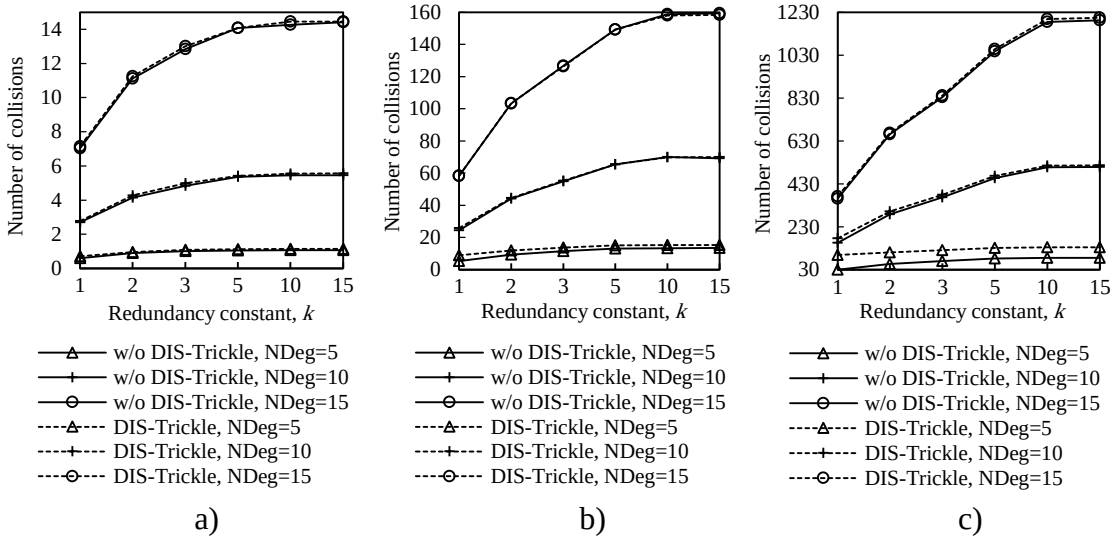


Figure 4.17. Influence of using DIS-Trickle on the number of collisions in the small, medium and large networks for different average node degrees, NDeg, as a function of the redundancy constant, k, when the I_{min} value is set to 8 ms compared with not using DIS messages: a) Small network size, b) medium network size, and c) large network size.

line curve results in Figure 4.18) is the default set of DIS-Trickle parameter values used in our previous simulations.

As shown in Figures 4.18.a) and 4.18.b), when the initial delay is set to 100 ms (Config. 2), the network convergence time and the total number of RPL messages transmitted are greater than those obtained for an initial delay of 200 ms. The main reason is that 100 ms is lower than the time needed by DIO messages to propagate

Table 4.4. DIS-Trickle parameter configurations evaluated.

Parameter	**Configuration**				
	1	2	3	4	5
DIS initial delay (ms)	200	100	300	200	200
Interval length, I_{DIS} (ms)	30	30	30	15	45

through the network. Thus, nodes that are not close to the root may start to prematurely (and unnecessarily) transmit DIS messages while their neighbors are receiving or sending DIO messages, increasing the number of collisions during the network convergence process (Figure 4.18.e)). On the other hand, when the initial delay is set to 300 ms (Config. 3), network convergence time increases compared with using Config. 2, due to the slower rate of DIS message generation (Figure 4.18.d)), while the rest of performance parameters do not vary significantly. As previously mentioned, a suitable value for the initial delay has to be set taking into account network characteristics such as density and size. From the results obtained, an initial delay of 200 ms offers good performance in the scenario considered, since it offers reasonably low network convergence time, low number of DIO and DIS message transmissions and low number of collisions.

Regarding the length of the DIS-Trickle interval, I_{DIS}, we have tested values within the range of the DIS response time values presented in subsection 4.3.1. Figure 4.18 illustrates that, although a small value for I_{DIS} such as 15 ms (Config. 4), reduces slightly the network convergence time compared with the default I_{DIS} value of 30 ms (Config. 1), it causes the rest of performance parameters analyzed to significantly degrade. When I_{DIS} is set to a low value such as 15 ms, many premature DIS message transmissions take place, leading also to a high number of collisions. On the other hand, using a greater value for I_{DIS}, such as 45 ms (Config. 5), has the opposite effect on performance.

Results show that there is a tradeoff between the network convergence time and the rest of performance parameters that depends on I_{DIS}. A medium I_{DIS} value within the DIS response time range yields a good performance tradeoff.

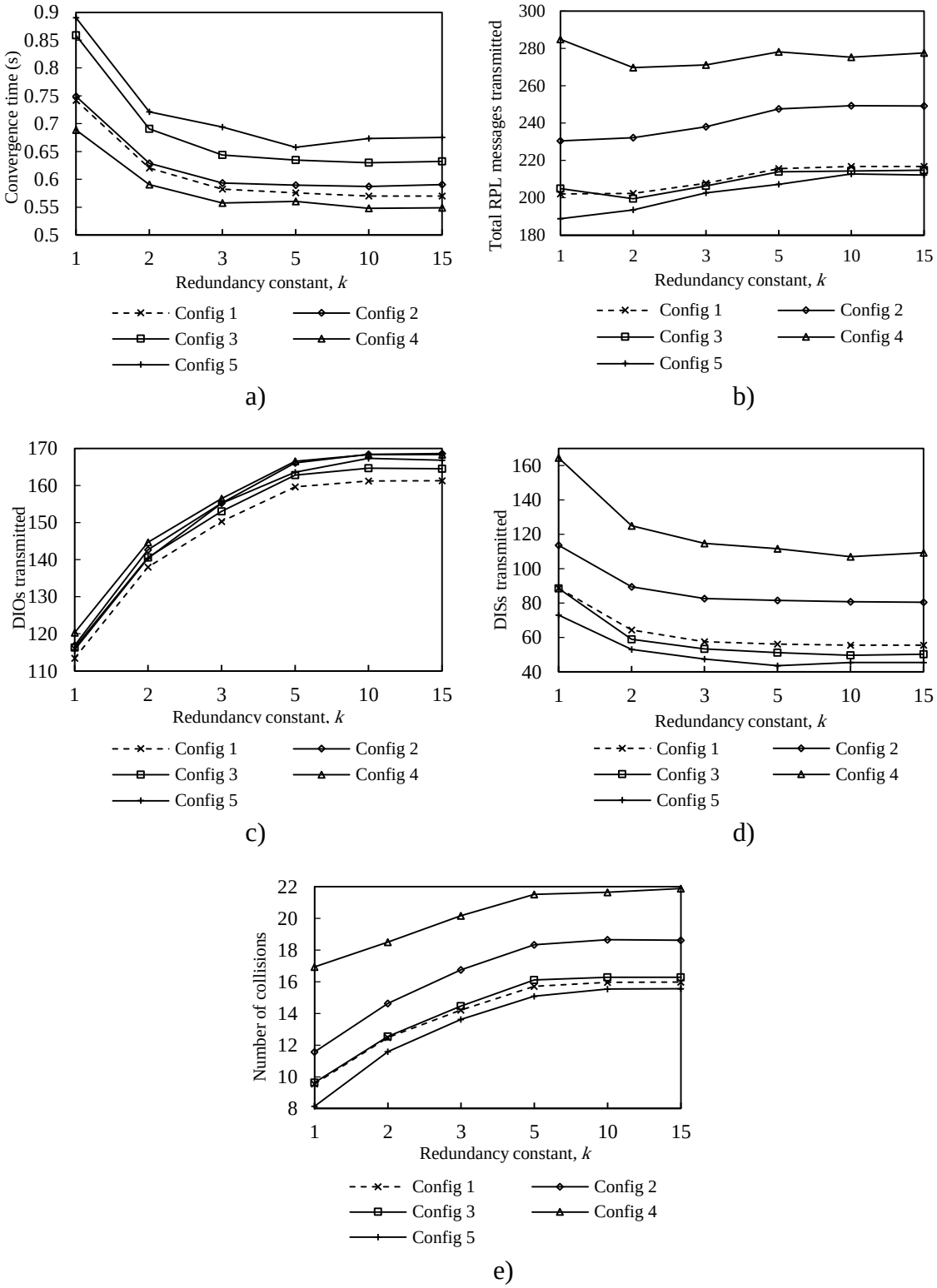


Figure 4.18. Comparing performance of different DIS-Trickle parameters settings (shown in Table 4.4) in medium network scenario when the average nodes degree is 5.

4.4. Related work

Despite the novelty of RPL, it has already been a subject of study in the literature [42-53]. However, only a few research works focus on or consider RPL network convergence. This section reviews work related with this chapter highlighting the differences of such work with the contributions and findings from this chapter.

Authors of [44] present a simulation study of RPL based on the COOJA/ContikiRPL simulator. The main goal of this work is to investigate the impact of network parameters including number of nodes, number of DODAG roots, and packet loss on the performance of the network convergence process. The authors evaluate a range of network size values. Two different network topologies are considered: regular topologies, whereby nodes are quasi-uniformly distributed, and random topologies. A single value is tested for I_{min}, which was set to 1 s. On the other hand, the redundancy constant value is not specified in the paper. Periodical transmission of DIS messages is evaluated in this work, based on a mechanism used in the COOJA/ContikiRPL [30] simulator. An initial delay is applied before the first DIS message transmission. The values used for the initial delay and for the length of the periodic intervals are 5 s and 60 s, respectively.

A survey about the most relevant RPL research efforts, along with an experimental performance evaluation of RPL, is presented in [45]. The performance evaluation of RPL provided by the authors includes network convergence time as a performance parameter, based on a single network size of 30 TelosB motes running ContikiRPL [30]. I_{min} is set to a single value, equal to 4.096 s, which is the default value used in ContikiRPL. However, authors neither indicate which is the value of the redundancy constant nor whether DIS messages are used or not.

Another performance analysis of RPL which is also based on the COOJA/ContikiRPL simulator is presented in [53]. The authors evaluate control message overhead, network throughput and end-to-end packet delay in RPL for fixed values of I_{min} and k set to 4.1 s and 10, respectively. This work considers a single area size and two different numbers of nodes: 20 and 100 nodes. However, the authors do not study the network convergence time. DIS messages are used in this work, but the parameter settings and further details on DIS message scheduling are not given.

A simulation-based performance evaluation of RPL for a single, medium size network is carried out by Tripathi et al. in [48]. Results include that global repair (i.e. a mechanism that allows the root node to form a new DODAG when bad performance is detected) has a significant effect on the control message overhead. However, network convergence time is not studied in this work. I_{min} is set to a single value (equal to 1 s), but the redundancy constant value used is not indicated. DIS-based mechanisms are not used in this work.

Clausen et al. investigate the efficiency of broadcast mechanisms in WSNs using RPL in [42, 49]. The authors use two mechanisms for DIO message transmission: i) the Trickle algorithm, in which I_{min} is set to a single value (equal to 2 s), and ii) periodic DIO message transmission. In both schemes the redundancy constant is set to infinity, i.e. the suppression mechanism is disabled. The authors evaluate the network convergence time and message overhead parameters, based on a fixed node density (50 nodes/km^2), for different network sizes in [42]. Results show that convergence time grows logarithmically with the number of nodes in the network. DIS-based mechanisms are not considered in the paper.

An experimental evaluation of RPL in TinyOS devices is provided in [43]. Regarding RPL parameters, a single value of I_{min} is considered (I_{min} = 128 ms), whereas the paper does not mention the redundancy constant setting. In order to study the impact of the I_{min} value on the network convergence time, the authors present experimental in a single size and single density network of 40 nodes, with perfect and also with lossy channels for three different values of I_{min}: 0.25 s, 1 s and 4 s. Results show that network convergence time increases with I_{min}, which is consistent with results obtained in this paper. However, the influence of I_{min} on other performance parameters such as RPL message overhead or collisions, has not been considered in the work. DIS messages have been used in the work, however details about DIS message scheduling rules or related parameters are not provided by the authors.

Analytical models have also been developed to estimate the network convergence time in a Trickle-based network. Authors in [46] provide such a model for line and grid network scenarios, based on a method to derive the probability density function of the time until network consistency through the use of Laplace transforms. However, the complexity of the analysis increases very quickly with the network diameter. Furthermore, the models presented do not take into account physical and link layer

characteristics. Other network convergence time analytical models for Trickle-based networks are provided for a single-hop, and for a grid network scenario in [47]. These models neither take into account radio interface characteristics, and assume error-free networks.

Despite the fact that RPL has already been evaluated from different perspectives in the literature, to the best of our knowledge there is not any published research study that provides comprehensive insight on the influence of RPL parameters and mechanisms, and network characteristics, on the network convergence process in RPL.

4.4. Discussion

In this chapter we investigated by extensive simulation the influence of two important RPL parameters, the redundancy constant, *k*, and the minimum interval, I_{min}, on the network convergence process, on top of a variety of IEEE 802.15.4 multihop network densities and sizes. We also proposed and evaluated a mechanism called DIS-Trickle for accelerating the network convergence process by exploiting DIS messages.

Results show that using low values for the redundancy constant yields high network convergence time in all network size and density scenarios considered, since a high number of DIO messages are suppressed by the Trickle algorithm. As the redundancy constant increases up to medium values, the network convergence time decreases. However, network convergence times do not vary significantly as the value of *k* increases beyond the values around the node degree. There exists a tradeoff between network convergence time and other performance parameters, such as the number of DIO messages transmitted and number of collisions that depends on *k*. In order to achieve a low network convergence time, the redundancy constant should be set to a value equal or close to the average node degree, since greater values might cause an unnecessary increase in the number of DIO message transmissions and collisions in very dense network areas, which however would not yield a network convergence time decrease. As network density increases, the influence of the redundancy constant grows with the network size. In sparse networks, the relative influence of *k* is independent of the network size.

With regard to the minimum interval, we found that there exists a tradeoff between network convergence time and DIO message overhead (and collisions) that depends on I_{min}. Varying I_{min} in the range of values considered in this study has a greater

quantitative effect on the number of DIO messages sent than on the network convergence time. Fine-tuning I_{min} has a greater impact on the number of DIO messages transmitted during network convergence as both the redundancy constant and network density grow, whereas the impact of I_{min} on network convergence time does not vary significantly with the redundancy constant, network size or network density.

Finally, we proposed and evaluated DIS-Trickle, a mechanism that leverages DIS messages in order to reduce nodes' DODAG join time. Results show that using DIS-Trickle decreases network convergence time by 2-3 orders of magnitude, regardless of the network size or density. The improvement provided by this mechanism grows as network density decreases. Interestingly, in dense networks, DIS-Trickle does not increase or even reduces the total number of RPL messages sent during network convergence. However, in sparse networks, DIS-Trickle creates a trade-off between network convergence time and RPL message overhead. The influence of DIS-Trickle on network convergence performance decreases with the redundancy constant.

The findings and contributions in this work offer a guideline for configuring and selecting adequately RPL parameters and mechanisms for achieving a good network convergence performance tradeoff, on the basis of network characteristics such as size and density.

5. Modeling the network convergence time in RPL in error-prone, IEEE 802.15.4 chain topology multihop network

In the previous chapter we evaluated the network convergence process of RPL over IEEE 802.15.4 by simulation, considering the influence of RPL and network parameters. In this chapter we present an analytical model of the network convergence time in RPL in a chain topology network under realistic assumptions, such as the presence of bit errors and the use of IEEE 802.15.4 at the physical and link layers.

The rest of this chapter is organized as follows. Section 5.1 gives an introduction to this chapter. Section 5.2 presents our analytical model. Section 5.3 describes the simulation environment and methodology used in the chapter, and shows the obtained results. Finally, section 5.4 presents the main conclusions of this chapter.

5.1. Introduction

An important performance aspect of RPL is the network convergence time, which is critical for applications whereby the user expects fast network creation [6]. However, only a few works focus on this performance parameter and, among these, the ones that analytically model the network convergence time in RPL assume ideal links or do not take into account important aspects such as radio interface characteristics.

In this chapter we present an analytical model of the network convergence time in RPL under realistic assumptions, such as the presence of bit errors and the use of IEEE 802.15.4 (i.e. the most popular radio interface for WSNs) at the physical and link layers. The model is developed for a static chain multihop topology, which may be found in various use cases, such as in trains, bridges or roads, and also provides a lower bound on the network convergence time in a generic topology. The model will be useful in the planning, deployment and evaluation of RPL-based networks. The model presented is validated by extensive simulation results, which show the accuracy of the model.

5.2. Network convergence time model

In this section, we present an analytical model of the expected DODAG convergence time in a network of RPL nodes that use IEEE 802.15.4 as the radio

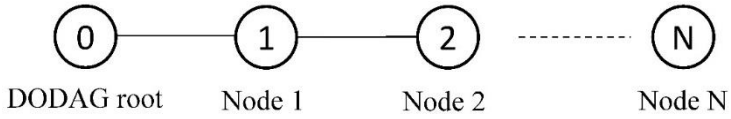


Figure 5.1. A chain network including N+1 nodes where each node is only able to communicate with its immediate neighbors. Node 0 is the root of the DODAG.

interface, and in the presence of uncorrelated bit errors. For simplicity, the model assumes a static *N*-hop chain network (see Figure 5.1) where all links are of equal characteristics and uncorrelated. Nevertheless, the model provides accurate results for the scenario assumed, and also constitutes a lower bound on the DODAG convergence time in any network of a maximum number of hops between the DODAG root and non-root nodes equal to *N*.

The goal of the model is to calculate the expected DODAG convergence time, from the start of the first Trickle interval in which the DODAG root performs its first DIO message transmission until the instant in which all nodes have joined the DODAG. Note that our target scenario is an *N*-hop chain network topology, which consists of *N* non-root nodes plus the root. In this scenario, the time required for the complete DODAG formation, denoted by T_{DODAG}, can be expressed as follows:

$$T_{DODAG} = \sum_{i=1}^{N} t_{join_i}, \tag{5.1}$$

where t_{join_i} is a random variable that indicates the time from the instant in which a DODAG node *i* starts its first Trickle interval until the instant in which a non-DODAG node that is a neighbor of the sender, node *i*+1, receives the DIO message from node *i* and joins the DODAG.

For a realistic approximation of the DODAG formation performance, we consider the presence of bit errors in the communication. In such conditions, t_{join_i} is a random variable that depends on the number of intervals, and DIO message transmissions, required until the node that joins the DODAG receives its first DIO message.

On the basis of (5.1), and since we assume that the links of the *N*-hop chain topology are of equal characteristics and uncorrelated, the expected DODAG convergence time can be calculated as

$$E[T_{DODAG}] = \sum_{i=1}^{N} E[t_{join_i}] = N \cdot E[t_{join}]. \tag{5.2}$$

where t_{join_i} has been substituted by t_{join}.

We next calculate $E[t_{join}]$. Let us define the term BER as the bit error rate and $P_{DIO_{Err}}$ as the probability that a DIO message is affected by bit errors. Assuming that bit errors are not correlated, $P_{DIO_{Err}}$ can be calculated as:

$$P_{DIO_{Err}} = 1 - (1 - BER)^{len} \tag{5.3}$$

where len is the length of a DIO message.

Let us define next α_j as a random variable that indicates the time between the start of the first interval of a node and the instant of its first successful DIO message transmission, which happens in the *j*-th interval (where *j* is an integer greater than 0). Let us also define the term $t_{tx\ DIO}$ as the time needed for a DIO message transmission at the physical layer and the term t_{MAC} as the time incurred by the MAC layer operations needed for transmitting a message in IEEE 802.15.4 (t_{MAC} components comprise backoff time, the receiver setup time, CCA time, and the turnaround time needed to switch the radio from receiver mode to transmitter mode). Finally, let τ_j be the time between the second half of the *j*-th interval and the transmission time for that interval. Based on the above definitions, and neglecting message processing times, the values for α_1, α_2 and α_3 can be expressed as follows (see Figure 5.2):

$$\alpha_1 = \frac{1}{2} \cdot I_{min} + \tau_1 + t_{MAC} + t_{tx\ DIO}, \quad \tau_1 \in \left[0, \frac{1}{2} \cdot I_{min}\right] \tag{5.4}$$

$$\alpha_2 = 2 \cdot I_{min} + \tau_2 + t_{MAC} + t_{tx\ DIO}, \qquad \tau_2 \in [0, I_{min}] \tag{5.5}$$

$$\alpha_3 = 5 \cdot I_{min} + \tau_3 + t_{MAC} + t_{tx\ DIO}, \quad \tau_3 \in [0, 2 \cdot I_{min}] \tag{5.6}$$

And, in fact, α_j can be expressed as follows:

$$\alpha_j = \left(3 \cdot 2^{j-2} - 1\right) \cdot I_{min} + \tau_j + t_{MAC} + t_{tx\ DIO} \tag{5.7}$$

where $\tau_j \in \left[0, 2^{j-2} I_{min}\right]$.

The expected value of α_j, $E[\alpha_j]$ can be obtained as

$$E[\alpha_j] = \left(3 \cdot 2^{j-2} - 1\right) \cdot I_{min} + E\left[\tau_j\right] + E[t_{MAC}] + t_{tx\ DIO} \tag{5.8}$$

We next calculate the expected value of $\tau_j, E[\tau_j]$. Let L_j be the size of the j-th interval. Since, due to the Trickle rules, a DIO message transmission can occur at any moment within the second half of the j-th interval, τ_j can be characterized as a uniformly distributed random variable between $L_j/2$ and L_j. Therefore the expected value for τ_j can be calculated as follows:

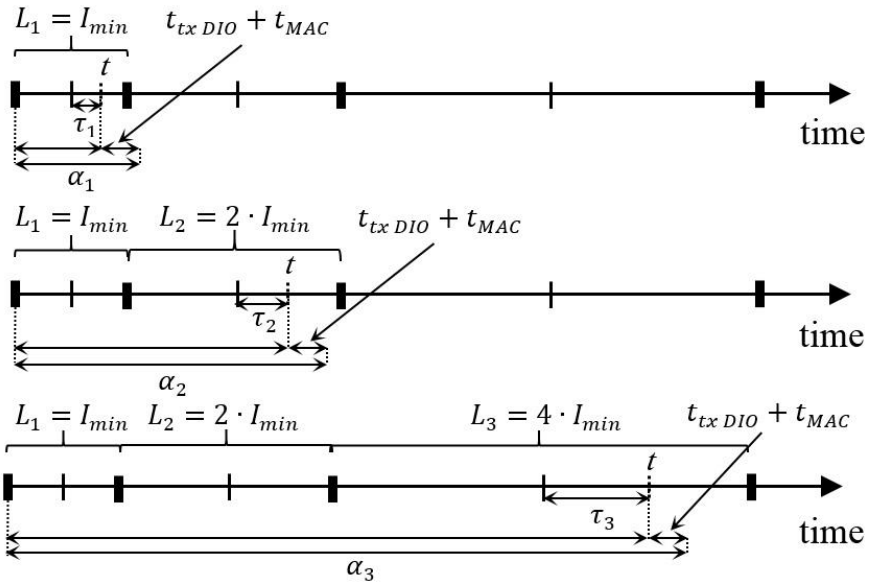


Figure 5.2. Variables used for calculating the time between the start of the first interval and the instant of a DIO message transmission in the first, second and third intervals.

$$E[\tau_j] = \frac{\frac{L_j}{2} + L_j}{2} - \frac{L_j}{2} = \frac{1}{4} \cdot L_j \tag{5.9}$$

Considering the size of the *j*-th interval, L_j, which is $2^{j-1} \cdot I_{min}$, (5.9) can be rewritten as follows

$$E[\tau_j] = \frac{1}{4} \cdot 2^{j-1} \cdot I_{min} = 2^{j-3} \cdot I_{min} \tag{5.10}$$

By substituting (5.10) in (5.8), we obtain

$$E[\alpha_j] = \left(7 \cdot 2^{j-3} - 1\right) \cdot I_{min} + E[t_{MAC}] + t_{tx\ DIO} \tag{5.11}$$

Based on (5.11), the expected time between the start of the first interval of a DODAG member, and the first successful DIO message reception by its next hop can be calculated as follows:

$$E[t_{Join}] = \sum_{j=1}^{\infty} \left(E[\alpha_j] \cdot {P_{DIO_{Err}}}^{j-1} \cdot \left(1 - P_{DIO_{Err}}\right) \right) \tag{5.12}$$

Because the number of interval doublings is limited to 20 by default in the Trickle algorithm, we should rewrite (5.12) expressing the addition in two parts, as shown in (5.13). The first addend is related to the first 20 intervals, whose lengths are doubled in each iteration, and the second part corresponds to the rest of intervals, which have a fixed length equal to I_{max}.

Finally, the expected DODAG convergence time in a chain topology network of *N* hops, where the DODAG root is placed at one edge of the chain, T_{DODAG}, can be found by plugging (5.13) into (5.2).

$$E[t_{join}] = \sum_{j=1}^{21}\left(((7\cdot 2^{j-3}-1)\cdot I_{min} + E[t_{MAC}] + t_{tx\,DIO}\ )\cdot {P_{DIO_{Err}}}^{j-1}\cdot\left(1-P_{DIO_{Err}}\right)\right) + \sum_{j=22}^{\infty}\left(\left(((\mathrm{j}-20)\cdot 2^{20}+3\cdot 2^{18}-1)\cdot I_{min} + E[t_{MAC}] + t_{tx\,DIO}\ \right)\cdot {P_{DIO_{Err}}}^{j-1}\cdot\left(1-P_{DIO_{Err}}\right)\right) \quad (5.13)$$

5.3. Evaluation

This section evaluates the analytical model presented and compares the analytical results with results obtained by extensive simulations. The section is organized in two parts. The first one introduces the simulation environment. The second subsection shows both analytical and simulation results.

5.3.1. Simulation environment and methodology

In order to validate the model presented, we implemented RPL [56] in OMNET++, a well-known C++-based discrete event simulator [54]. We also used MiXiM as a WSN simulator framework within OMNET++ [55]. In this framework, we selected the radio interface that simulates the CC2420 radio chip, a popular 2.4 GHz band IEEE 802.15.4 radio interface. We used the beaconless mode and set the receiver setup time, the turnaround time and the CCA duration according to the CC2420 data sheet [57]. The rest of radio interface parameters were set to the default values specified in the IEEE 802.15.4 standard [60]. The I_{min} and I_{max} parameters were set to 8 ms and 2.3 hours, respectively, which are the default values stated by the RPL specification [5]. In order to avoid unnecessary suppression of DIO messages and fast DODAG convergence, we set the redundancy constant, *k*, to 2 (note that greater values of *k* would yield the same results). Table I shows the main parameter settings used in the simulations.

The simulated scenario is a static chain topology network whereby the distance between every two neighboring nodes is the same and is equal to 9.96 m. The DODAG root is placed at one edge of the chain.

Table 5.1. Parameter settings used in simulations and in the analytical model results.

Parameter	Value
I_{min} (ms)	8
I_{max} (hours)	2.3
k	2
IEEE 802.15.4 band (GHz)	2.4
Transmit power (dBm)	-25
CCA (μs)	128
Turnaround time (μs)	192
Receiver setup time (ms)	1.792

We evaluated the model and performed simulations for various BER values, ranging from 0 up to 10^{-3} (note that the latter is the BER for which the receiver sensitivity is defined in IEEE 802.15.4). We also considered a chain topology length between 1 and 15 hops. For each individual configuration, we simulated 1000000 DODAG formation iterations.

To obtain the DODAG convergence time in the simulations, we computed the time from the instant in which the DODAG root starts its first Trickle interval until the instant in which all nodes have joined the DODAG.

5.3.2. Simulation and analytical results

In this subsection we compare the DODAG convergence time, as predicted by the analytical model presented in the previous section, with results obtained by simulation, for different values of both BER and chain topology length.

In order to obtain the expected DODAG convergence time analytically, we applied the same radio interface and RPL parameter values used in the simulation, and solved the model numerically.

For the calculation of the DIO message transmission time, $t_{tx\ DIO}$, we determined the length of a DIO message. The size of a DIO message that carries the necessary parameters for DODAG formation is 52 bytes at the RPL layer. After adding a typical compressed IPv6 header [61], MAC layer and physical layer headers, the size of the DIO message is 88 bytes.

Figures 5.2 and 5.3 compare results obtained by using the analytical model with simulation results. As can be seen in these figures, the analytical model approximates

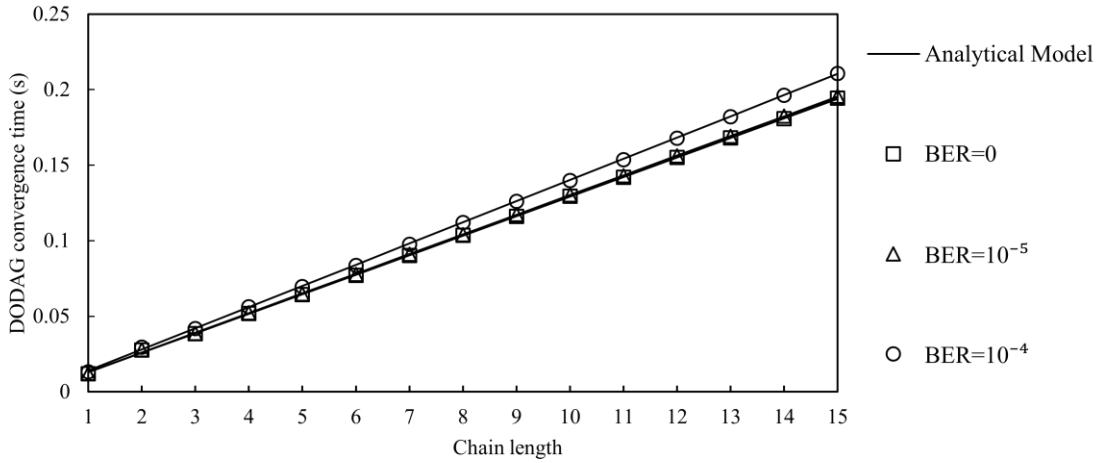


Figure 5.3. Average DODAG convergence time for various chain topology number of hops and BER values: simulation (symbols) vs. analysis (lines) Part I.

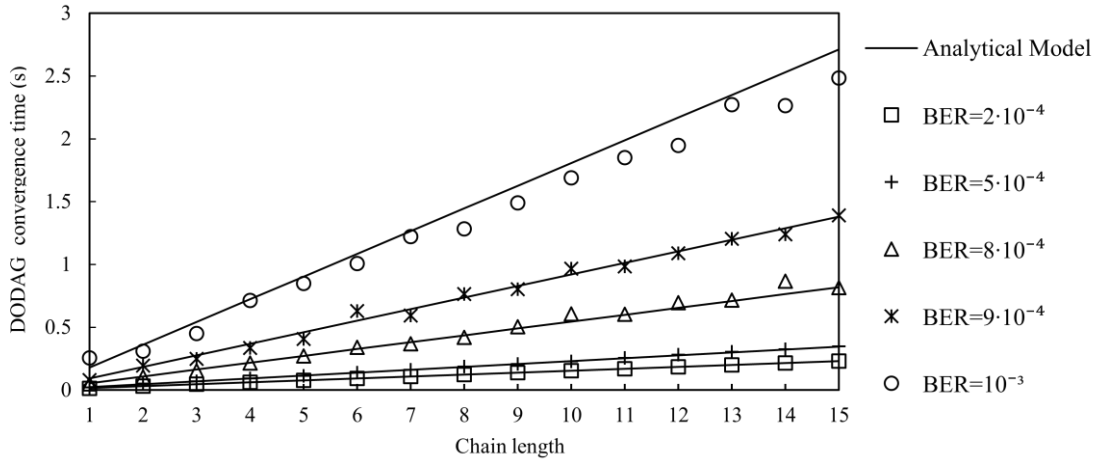


Figure 5.4. Average DODAG convergence time for various chain topology number of hops and BER values: simulation (symbols) vs. analysis (lines) Part II.

well the results obtained by simulation. For BER values up to $5 \cdot 10^{-4}$, the model is highly accurate. For greater BER values, the model accuracy slightly decreases.

Bit errors do not have a significant influence on the DODAG convergence time for BER values up to $5 \cdot 10^{-4}$. However, for greater BER values, the DODAG convergence time may increase by a factor of up to 10 (e.g. for BER=10^{-3}) in comparison with the one obtained when the network is error-free. Note that when a user expects a response after an action, s/he is sensitive to delays greater than 500 ms [6]. This threshold is exceeded for various BER values considered for a chain topology length greater than 3. Furthermore, the evaluation has neglected processing times, which may contribute to the DODAG convergence time in a real network.

On the other hand, the DODAG convergence time linearly increases with the chain topology length (i.e. network diameter), which is consistent with results found in the literature by simulation [44, 45].

5.4. Related work

This section reviews research work that is related with this chapter. Despite the novelty of RPL, many studies have already evaluated performance of this routing protocol and its mechanisms [48-52]. These works generally focus on performance parameters such as packet delivery ratio, control message overhead, power consumption or packet delay in various scenarios. However, only a few works have focused on the network convergence time in RPL, and only a subset of these provide analytical models [46, 47].

A survey on research efforts about RPL, as well as an experimental performance evaluation of RPL using TelosB motes [58] running ContikiRPL [30] is presented in [45]. Among other results, authors claim that network convergence time increases linearly with the number of nodes in the DODAG. However the relationship between the number of nodes and the related topologies is not clear in this work.

A simulation study of RPL with special attention on the network formation process is provided in [44]. The main goal of this paper is to investigate the impact of a set of network parameters and mechanisms including the number of nodes, RPL parameters and mechanisms such as the number of DODAG roots or the objective function, and packet loss on the performance of the network construction process. Results are obtained based on two different network topologies: regular topologies, whereby nodes are quasi-uniformly distributed, and random topologies. Authors show a linear increase of the network convergence time with the network size.

In [46], authors developed analytical models to estimate the duration of a propagation event, which is equivalent to the network convergence time, by using the mechanisms included in RPL for both line and grid network scenarios. These models take into account packet loss ratio, by using the Closest Pattern Matching (CPM) model as the propagation model. The authors provide a method to derive the probability density function of the time until network consistency through the use of Laplace transforms. However, a specific probability tree is required for each specific network, and an equation for the convergence time in terms of the network diameter is not given.

In fact, complexity of the analysis increases very quickly with the network diameter. Furthermore, the models presented do not take into account physical and link layer characteristics, such as transmission time, backoff time, etc.

Finally, authors in [47] presented analytical models to estimate the number of control message transmissions per interval and also the propagation speed, which is related with the network convergence time, by using the mechanisms used in RPL in single-hop and multihop networks. However, these models have not taken into account physical and link layer characteristics and assume error-free networks.

To the best of our knowledge, an analytical model for estimating the network convergence time in RPL that considers the network diameter, the presence of bit errors and radio interface characteristics has not yet been published.

5.5. Discussion

In this chapter we presented an analytical model for computing the expected DODAG convergence time in a static chain multihop network of IEEE 802.15.4 nodes, considering bit errors, when RPL is used as the routing protocol. We validated the model presented by extensive simulation results.

As results show, the analytical model approximates well the results obtained by simulation for a wide range of both BER and chain topology length. We have found a linear increase of the network convergence time with the number of hops in the network, which is consistent with the literature. Bit errors do not have a significant effect on the network convergence time for BER values up to $5 \cdot 10^{-4}$. However, greater BER values may lead to degradation of this performance parameter, and BER values close to 10^{-3} may have a dramatic effect on the user experience for a chain topology length greater than 3 hops.

6. Modeling the Message Count of the Trickle Algorithm in a Steady-State, Static Wireless Sensor Network

The last two chapters focused on RPL performance in transient state, i.e. during network convergence. This chapter considers steady-state networks. As presented in Chapter 3, the Trickle algorithm regulates the transmission of DIO messages in RPL. In this chapter we present an analytical model for the number of messages transmitted in a steady-state, static network that uses the Trickle algorithm, as a function of the redundancy constant and the average node degree.

6.1. Introduction

As mentioned in section 3.2, the Trickle algorithm can be used for different purposes including control traffic timing, multicast propagation and route discovery. Knowing the number of transmitted messages for applications that use the Trickle algorithm can be very useful for network engineers and researchers in the planning, deployment and evaluation of Trickle-based networks, such as RPL networks. To this end, in this chapter we present an analytical model for the number of messages transmitted in a steady-state, static network that uses the Trickle algorithm, as a function of the redundancy constant (i.e. the main parameter of the Trickle suppression mechanism), and the average node degree. The model constitutes a tool that will be useful for network engineers and researchers in the planning, deployment and evaluation of Trickle-based networks. The model presented is validated by simulation results.

6.2. Trickle model

In this section, we present an analytical model for the number of message transmissions allowed by Trickle in a WSN within a given time interval (note: by ‘time interval’ we refer to the time intervals defined by the Trickle algorithm). Our study assumes a static network which is in steady state, where the current interval size of all the nodes is the same and is equal to I_{max}. This assumption is consistent with the characteristics of WSN deployments with very stable links [35]. For more dynamic WSNs, the model provides a lower bound on the number of message transmissions.

For simplicity, our model assumes a synchronized network whereby the interval start time is the same for all the network nodes. Nevertheless, as shown in section 6.3.3, the number of transmissions in a network when intervals have arbitrary start times is very close to the one in a synchronized network.

In our model, we assume a uniformly random spatial distribution for the nodes on a two-dimensional area of size *A*, and a total number of nodes equal to *N*. Let the area defined by the coverage range of a node be equal to a. Based on these definitions and assumptions, the probability that a node will be a neighbor of a given node, q, can be calculated as follows:

$$q = \frac{a}{A}. \tag{6.1}$$

The goal of the model is to calculate the total number of messages transmitted within a given interval in the whole network. In order to achieve this goal, in the following we calculate the average probability that a node belonging to the network, denoted by node x, will send a message in a given interval. This average probability is denoted by $P(tx)$.

Recall that *k* denotes the Trickle redundancy constant. In order to calculate $P(tx)$, we consider two possible situations: a) node x has fewer than *k* neighbors; b) node x has at least *k* neighbors.

Let y be the number of neighbors of node x. Also assume that $P(y < k)$ is the probability that the number of neighbors of node x will be smaller than *k*, and $P(tx|y < k)$ is the probability that node x will send a message in the current interval when its number of neighbors is smaller than *k*.

Let us define the term $P(y \geq k)$ as the probability that the number of neighbors of node x will be greater than or equal to *k*, and the term $P(tx|y \geq k)$ as the probability that node x will send its message in the current interval when it has *k* or more neighbors. Then, the average probability that a node will send a message in a given interval, $P(tx)$, can be written as follows:

$$P(tx) = P(tx|y < k) \cdot P(y < k) + P(tx|y \geq k) \cdot P(y \geq k). \tag{6.2}$$

Next we calculate each term of (6.2) separately. First, when the number of neighbors for node x is smaller than *k*, the node will surely transmit at time *t*, i.e. $P(tx|y < k) = 1$, because it will hear at most *k*-1 transmissions during the interval (i.e.

its counter *c* will never reach *k*). The probability that node x will have fewer than *k* neighbors, $P(y < k)$, can be obtained as:

$$P(y < k) = \sum_{i=0}^{k-1} \left(\binom{N-1}{i} \cdot q^i \cdot (1-q)^{N-1-i} \right). \tag{6.3}$$

We next derive the second addend of (6.2). The probability that node x will have at least *k* or more neighbors, $P(y \geq k)$, can be calculated as follows:

$$P(\geq k) = \sum_{i=k}^{N-1} \left(\binom{N-1}{i} \cdot q^i \cdot (1-q)^{N-1-i} \right). \tag{6.4}$$

Note that each node can have at most *N*-1 neighbors.

In order to calculate the probability of message transmission for node x in the current interval when it has at least *k* neighbors, $P(tx|y \geq k)$, we consider two possible cases: i) the time selected by node x for the transmission of its message is one of the first *k* transmission times selected by its neighbors and by itself; and ii) the time selected by node x is not one of those first *k* transmission times. In the first case, node x will surely send its message at the selected time *t*. The probability of selecting one of the first *k* transmission times, denoted by $P(i)$, can be obtained as:

$$P(i) = \frac{k}{y+1}. \tag{6.5}$$

In the second case, node x can send its message if, at time *t*, at most *k*-1 of the subset of neighbors that have selected smaller transmission times than that of node x actually transmit their messages. (Note that the remaining neighbors suppress their transmissions due to the influence of their own neighbors.) If these conditions are satisfied, node x will be able to send its message. The probability that node x will select any of the last $y + 1 - k$ times, denoted by $P(ii)$ is:

$$P(ii) = 1 - \left(\frac{k}{y+1} \right). \tag{6.6}$$

The probability that, in that case, at most *k*-1 nodes will transmit before node x, denoted Γ, can be expressed as:

$$\Gamma = \sum_{j=0}^{k-1} \left(\binom{n-1}{j} \cdot \big(P(tx)\big)^j \cdot \big(1 - P(tx)\big)^{n-1-j} \right), \tag{6.7}$$

where *n* is the positive integer that denotes the position of node x transmission time in the set of increasingly ordered transmission times selected by node x and by its

neighbors. Note that there are n-1 nodes with smaller selected transmission time than the one chosen by node x, and $n > k$.

By plugging (6.1) and (6.3)–(6.7) into (6.2), we obtain (6,8), which gives an expression for $P(tx)$, i.e. the average probability of sending a message for node x during a given time interval. Since (6.8) is a polynomial in $P(tx)$, $P(tx)$ can be obtained from (6.8) by applying a root-finding algorithm.

Finally, the average number of transmissions in a network composed of N nodes in a given time interval, denoted N_{Tx}, can be expressed as follows:

$$N_{Tx} = P(tx) \cdot N. \quad (6.9)$$

6.3. Evaluation

6.3.1. Simulation environment and methodology

In order to validate the model presented, we have developed a simulator using Delphi programming language. In our simulation environment, 100 nodes (i.e. N=100) are placed according to a uniformly random spatial distribution on a square scenario of area 150 m x 150 m. The coverage range of a node is a circular area defined by the transmission radius of the node. Any node contained within the coverage range of another node receives correctly the messages transmitted by the latter. The coverage range (which is the same for all nodes) is varied in order to evaluate the results for different average node degrees. In order to cope with the border effect problem in the simulations, we use the toroidal distance for calculating the distances between nodes [62]. Nodes execute the Trickle algorithm, which has been implemented in the simulator according to the Trickle specification [25]. The simulator allows the network to be configured in two different modes: synchronous mode and asynchronous mode.

In the synchronous mode, all nodes start their first interval (of size I_{min}) at the same time. When the intervals of all nodes reach the I_{max} size (i.e. the network reaches the steady state), the simulator computes the number of transmitted messages for five consecutive intervals.

In the asynchronous mode, each node randomly chooses a start time for its first interval (of size I_{min}) in the range between 0 and I_{max}. Thus, the last Trickle execution start can occur while the node that selected the first start time is already in its first interval of I_{max} size. When the last node starts its second interval of I_{max} size, the network reaches steady state (note that, during the previous interval of I_{max} size, the neighbors of the last node may have a decreased transmission probability, since the last node's interval of $0.5 \cdot I_{max}$ size favored its own transmissions). Then, the simulator computes the number of transmitted messages for five consecutive intervals.

The procedures described for both synchronous and asynchronous modes are repeated 2000 times, using a new randomly generated node spatial distribution instance for each procedure and set of conditions. I_{min} and I_{max} parameters are set to 8 ms and 2.3 hours, respectively, which are the default values established by the RPL specification [5].

6.3.2. Synchronous network simulation results vs analytical model results

In this subsection we compare the number of messages transmitted in a given interval, as predicted by the analytical model presented in the previous section, with results obtained by simulation, for values of both k and average node degree between 1 and 15, respectively. The range of values considered for k includes the default value determined by the RPL specification (i.e. k=10) [5]. This subsection focuses on the synchronous mode, since the model assumes a synchronous network. The slightly different performance of an asynchronous network is discussed in the next subsection.

Figure 6.1 compares simulation results (which show the average results obtained from 10000 intervals for each set of parameter values) with analytical results. The largest 95% confidence interval size obtained for the simulation results is 4.73% of the corresponding average result. For the sake of clarity, Figure 6.1 does not show the confidence intervals.

$$P(tx) = \sum_{i=0}^{k-1} \binom{N-1}{i} \cdot \left(\frac{a}{A}\right)^{i} \cdot \left(1-\frac{a}{A}\right)^{N-i-1} + \sum_{i=k}^{N-1} \left(\frac{k}{i+1}\right) \cdot \left(\binom{N-1}{i} \cdot \left(\frac{a}{A}\right)^{i} \cdot \left(1-\frac{a}{A}\right)^{N-i-1} \right) +$$

$$+ \sum_{i=k}^{N-1} \left[\left(1-\frac{k}{i+1}\right) \cdot \left(\binom{N-1}{i} \cdot \left(\frac{a}{A}\right)^{i} \cdot \left(1-\frac{a}{A}\right)^{N-i-1} \right) \cdot \sum_{j=0}^{k-1} \left(\binom{i}{j} \cdot (P(tx))^{j} \cdot (1-P(tx))^{i-j} \right) \right] \quad (6.8)$$

As shown in Figure 6.1, the analytical model approximates well the results obtained by simulation. When the average node degree is comparable to or smaller than *k*, the model is highly accurate. As the difference between the average node degree and *k* increases, the model accuracy decreases slightly. This occurs because the model assumes in (6.7) that message transmissions of node x neighbors are independent events. However, a transmission done by a node x neighbor within the coverage range of another node x neighbor will lead to an increment of the counter *c* and a decrease in the transmission probability of the latter.

6.3.3. Synchronous vs asynchronous modes

As can be seen in Figure 6.1, for low values of *k* in comparison with the average node degree, the asynchronous mode yields a slightly greater number of message transmissions than the synchronous one. Otherwise, an asynchronous network leads to a message count that is equal to or slightly smaller than that of a synchronous network. We next explain the reasons for the different performance of these two types of network modes.

In an asynchronous network, when the value of *k* is small, results are influenced by the non-negligible probability that more than *k* transmissions will occur within the coverage range of a node during an interval. This happens because the start times of the

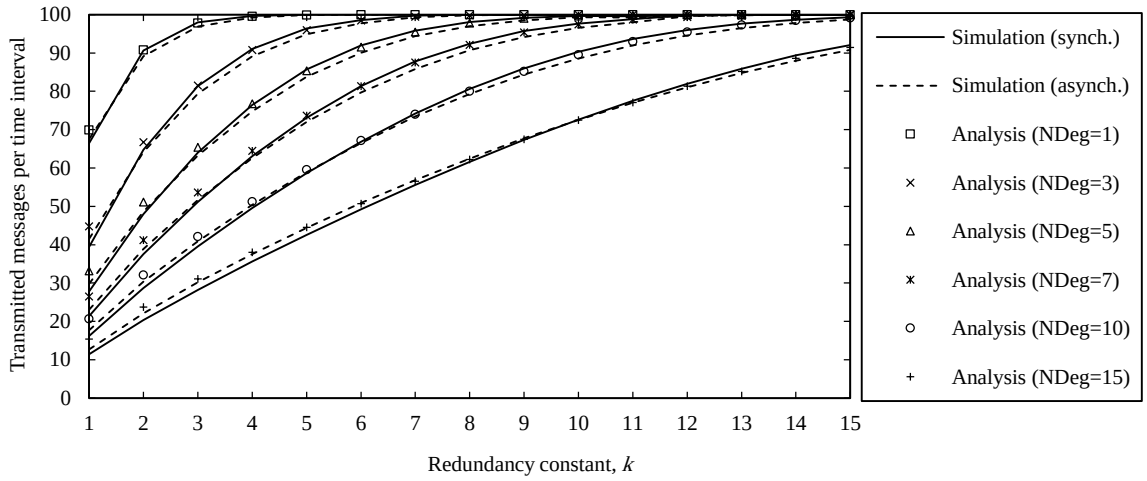


Figure 6.1. Number of messages transmitted per time interval in a network composed of 100 nodes for different values of the redundancy constant, k, and different average node degrees, denoted NDeg: analysis (symbols) vs simulation (lines).

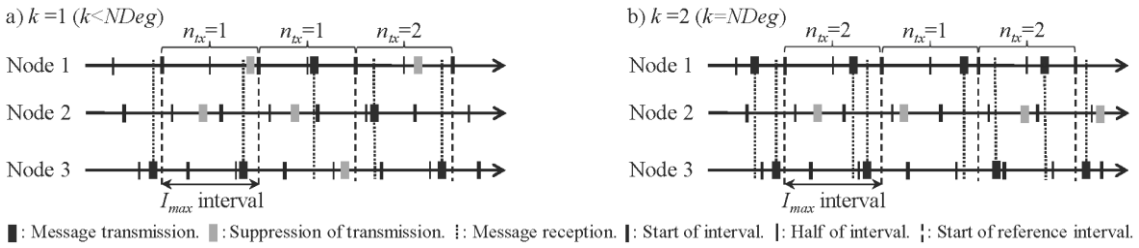


Figure 6.2. Example of Trickle operation in an asynchronous, steady-state network composed of three nodes where all nodes can hear each other's transmissions: a) the value of k is set to 1 (i.e. k is smaller than NDeg); b) k is set to 2 (i.e. k is equal to NDeg). The total number of transmissions per interval is denoted n_{tx}.

nodes' intervals do not coincide. For example, Figure 6.2.a) depicts the behavior of a network composed of three nodes, all of which can listen to each other, whereby $k = 1$. As a consequence of the asynchronous interval start times of each node, 2 (i.e. more than k) transmissions occur in the last interval. As the average node degree grows in comparison with k, this phenomenon is more likely to occur. However, the same network in synchronous mode would yield k transmissions per interval.

On the other hand, in an asynchronous network, when k is not small in terms of the average node degree, results are affected by the fact that less than k transmissions can occur within the coverage range of a node during an interval. This also happens as a consequence of the network asynchronicity. Figure 6.2.b) shows an example of the behavior of the same network referred to in Figure 6.2.a), albeit with k set to 2 (i.e. k is equal to the number of neighbors of all the nodes). In Figure 6.2.b), only one message is transmitted within the second interval. In fact, Node 2 suppresses its transmission in that interval since it has already heard k messages in its own interval. On the other hand, Node 3's next transmission is scheduled out of the second interval, while Node 1 is allowed to carry out its transmission. In the second interval, since there are no further neighbors that can transmit, the total number of transmissions is finally smaller than k. (Note that as the node degree increases in comparison with k, more nodes have the opportunity of transmitting.) Remarkably, the same network in synchronous mode would yield k transmissions per interval.

6.4. Discussion

In this chapter we presented an analytical model for predicting the number of transmitted messages in a Trickle-based steady-state, static WSN, as a function of the

redundancy constant and the average node degree. We validated the model by simulation for synchronous and asynchronous networks. Whereas the model provided is highly accurate, the slight differences in the performance of synchronous and asynchronous networks were analyzed. Note that the model assumes that the Trickle algorithm never resets. Therefore, the model offers a lower bound on the message overhead in Trickle-based networks, such as RPL networks, of a greater degree of topology dynamics.

7. Route change latency with RPL and 6LoWPAN Neighbor Discovery

The previous chapter focused on steady-state Trickle-based networks, such as RPL networks, assuming that topology changes do not occur. In this chapter we analyze performance of the mechanisms provided by default in 6LoWPAN ND and the impact of the relevant RPL parameters for detecting link failures. The goal of this chapter is the calculation of Route Change Latency (RCL), the time needed to determine an alternative route when a link or node failure occurs, in RPL, and its influence on end-to-end connectivity, when 6LoWPAN ND is used, as well as the related message overhead.

This chapter is organized as follows. Section 7.1 gives an introduction to this chapter. Section 7.2 is devoted to a theoretical analysis of the RCL in RPL and 6LoWPAN ND. Section 7.3 presents a simple analytical model for evaluating end-to-end path connectivity in a network by using RPL and 6LoWPAN ND. Section 7.4 discusses the range of parameters used in the evaluation and Section 7.5 shows the obtained results. A simple analytical model for calculating the message overhead due to connectivity maintenance is given in section 7.6. Section 7.7 presents the main conclusions of this chapter.

7.1. Introduction

As we mentioned in chapter 3, RPL has been designed taking into account the requirements of control and monitoring applications from many environments, including home and building automation, industrial monitoring, and urban sensor networks. These applications operate in unstable environments, whereby link and node failures may frequently occur (e.g. due to wireless propagation issues, node mobility, changes in the environment, battery depletion, etc.) [63, 64]. In consequence, a radio link may unexpectedly disappear or become unreliable. If that link is being used, then the link failure should be detected and a new route should be used (if any available route exists). This procedure incurs a delay, which has been denoted by RCL [65], that may not be negligible. Nevertheless, timely data transmission is very important for many applications. Some examples are the transmission of medical alarms for users of body

medical sensors or pushing a remote control's button in order to perform a command [6].

RPL may use a variety of mechanisms for detecting a link failure, including layer two and layer three mechanisms. The layer two mechanisms may not always be available, and depend on each particular link layer used and are tied to the implementation of RPL for a particular sensor node platform. Hence, RPL relies by default on the layer three protocol called IPv6 Neighbor Discovery (ND) [27]. However, it is expected that LLNs (in particular, those that use radios compatible with IEEE 802.15.4) exploit an optimized version of ND, which has currently been developed by the IETF IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) WG, denoted 6LoWPAN ND [34].

This chapter presents a theoretical evaluation of: i) the RCL incurred by RPL when 6LoWPAN ND is used, ii) the impact of the relevant 6LoWPAN ND and RPL parameters on path availability and iii) the trade-off between path availability and message overhead.

7.2. RCL with RPL and 6LoWPAN IPv6 ND

In this section we analyze the RCL of RPL and 6LoWPAN ND. For the basis of our study, we consider a simple topology which is shown in Figure 7.1 (the impact of the RCL on more complex topologies is analyzed in Section 7.4).

Figure 7.1.a) illustrates a topology whereby node D is a parent of nodes B and C; node A has selected nodes B and C as its parent set; and node B is the preferred parent of node A. Suppose that node A has registered its address with both of its parents. Then, the AB link fails, which leads to the new network topology depicted in Figure 7.1.b).

In order to analyze the RCL, we study two different scenarios (see Figure 7.2). In both scenarios, the last positive confirmation from node B in response to an NS (or DAO) message sent by node A is received at time *T*. Let us assume that node A wants

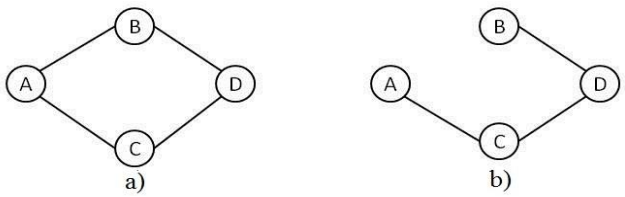


Figure 7.1. a) Node A has selected nodes B and C as its parents and B is its preferred parent; b) Node B becomes unreachable for node A and subsequently node A selects node C as its preferred parent.

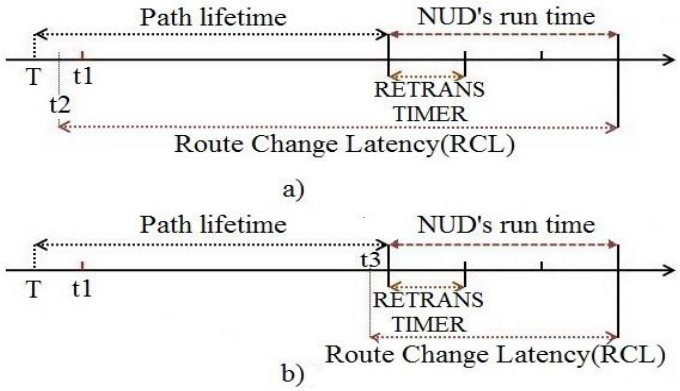


Figure 7.2. Two scenarios for data transmission and router unreachability. In a) router becomes unreachable right after sending an acknowledgement in response of NS message, at time t_2. In b) router unreachability takes place right before the expiration of node's lifetime, at time t_3.

to send a data packet to node D via its preferred parent, node B.

In the first scenario, assume that node B, after sending an acknowledgement to node A, becomes immediately unreachable for node A (see Figure 7.2.a)). In the second scenario, the router unreachability happens right before the registration lifetime expiration (see Figure 7.2.b). Node A may start sending data to node D at time t_1 because it believes its preferred parent is available during a period of time equal to the registration lifetime indicated in the last NS (or DAO) message sent by node A to node B. The node A will continue data transmission within the registration lifetime duration, until it performs re-registration along with NUD by sending a new NS (or DAO) message to its preferred parent. We assume that the registration lifetime used in ND is equal to the default lifetime contained in the RPL DIO message and path lifetime in DAO messages. For simplicity, we will use the term 'path lifetime' for any of these.

The time required to detect the router unreachability, denoted by T_{NUD}, can be expressed as follows:

$$T_{NUD} = MAX\_UNICAST\_SOLICIT * RETRANS\_TIMER \quad (7.1)$$

where the $MAX\_UNICAST\_SOLICIT$ and $RETRANS\_TIMER$ parameters are the ones already presented in section 3.4.2. The default values for these parameters are 3, and 1 second, respectively.

If we denote the lifetime of node A by $T_{lifetime}$, the RCL for the first scenario (see Figure 7.2.a)), can be calculated as follows:

$$RCL = T_{lifetime} + T_{NUD} \quad (7.2)$$

and for the second scenario (see Figure 7.2.b)), RCL is equal to:

$$RCL = T_{NUD} \quad (7.3)$$

Since the router unreachability can occur at any moment within the path lifetime duration, the RCL can be characterized as a uniformly distributed random variable between the two values expressed in equations (7.2) and (7.3). Therefore the expected value for RCL can be expressed as follows:

$$E[RCL] = \frac{T_{lifetime}}{2} + T_{NUD} \quad (7.4)$$

7.3. End-to-End connectivity model

In this section, we present a simple analytical model to evaluate the impact of using RPL with 6LoWPAN ND on the end-to-end connectivity of an LLN.

We denote the average lifetime of a link, from its creation until the instant in which it disappears, as Time to Link Failure (TLF). We assume that the parent set of all nodes in a DODAG has more than one member. Under this assumption, when a node or link failure occurs in an active path, a node can use another parent as preferred parent to continue the data transmission towards the same destination (note that this is an optimistic assumption). Hence, the probability of link unavailability, which we denote by *q*, can be calculated as follows:

$$q = \frac{E[RCL]}{TLF + E[RCL]} \quad (7.5)$$

We next calculate the probability of end-to-end path availability for data transmission in a path composed of *N* hops. We have considered that the length of both paths, i.e. the failed path and the new path, is equal to *N*.

Based on these assumptions, and on equation (7.5), the probability of end-to-end path availability for data transmission in an N-hop path, denoted by *P*, is equal to:

$$P = (1 - q)^N \quad (7.6)$$

7.4. Discussion of TLF and path length values

This section discusses the TLF and path length range of values for the evaluation of link and end-to-end path availability of an LLN, which is shown in section 7.5.

7.4.1. Time to Link Failure

Link failures occur in static scenarios due to phenomena which are intrinsic to radio signal propagation (e.g. multipath, fading, etc.) [66]. Furthermore, changes in the environment that may temporarily attenuate the radio signal (e.g. people moving around, opening and closing a door, rain or snow in an urban LLN) and interference

(e.g. from WiFi equipment or from a microwave oven) may affect the reception of radio signals [6] [34]. Link failures may also happen due to node mobility (e.g. due to the mobility of people using on-body sensors or using a portable remote control) and node failures (e.g. due to battery depletion). Depending on the particular environment, the expected TLF may range from less than one minute to more than one day.

7.4.2. Number of hops

The expected number of hops in a path depends on each particular scenario and application. For example, the number of hops in an industrial application can be up to 20 [9], e.g. when a few hundred nodes are deployed for controlling a very large refinery. In home and building automation [7, 8] the number of hops can be up to 5, whereas for some applications the expected number of hops can be equal to 1 or 2.

7.5. End-to-End connectivity evaluation

In order to evaluate the probability of link unavailability and the probability of end-to-end path availability, we consider different configurations for the RPL and 6LoWPAN ND parameters that affect the RCL. These configurations are shown in Table 7.1.

The value we use for *MAX_UNICAST_SOLICIT* for all of these ten configurations (i.e. #1 to #10) is equal to 3, i.e. the default value indicated in 6LoWPAN ND [34]. We believe this value constitutes an appropriate trade-off between reactivity to link failures and spurious link failure detection in a wireless environment.

Note that the *RETRANS_TIMER* value in configurations #1 to #5 is the default

Table 7.1. Proposed parameter configurations.

Configuration number	Path Lifetime (s)	RETRANS_TIMER (s)
#1	**10**	**1**
#2	**50**	**1**
#3	**100**	**1**
#4	**500**	**1**
#5	**1000**	**1**
#6	**10**	**2**
#7	**50**	**2**
#8	**100**	**2**
#9	**500**	**2**
#10	**1000**	**2**

value as proposed in [27]. In configurations #6 to #10 this value is multiplied by 2, in order to compare the impact of this parameter on the probability of end-to-end path availability. We considered as well a *RETRANS_TIMER* of 0.5 s. However, this setting led to very similar results to those obtained with the default value. Next, we evaluate the probability of link reachability (see Figure 7.3) and the probability of end-to-end path availability (see Figures 7.4, 7.5 and 7.6) by using equations (7.5) and (7.6).

For Figure 7.3, we have considered the range of TLF values mentioned in Section 7.4. As shown in Figure 7.3, all the configurations yield almost 100% link availability for TLF values greater than 5 hours. However, for short-lived links, only the configurations with low path lifetime values offer at least moderate link availability.

For path lifetime greater than or equal to 100 s, results are almost independent of the *RETRANS_TIMER* setting. Figures 7.4, 7.5 and 7.6 illustrate the end-to-end path availability for a range of path lengths that covers the various application requirements mentioned in Section 7.3, and for TLF of 10, 30 and 60 minutes, respectively. For a TLF of 10 minutes, only the configurations that use a path lifetime equal to 10 s offer an end-to-end path availability beyond 70%. When TLF is 30 minutes or 60 minutes, the same end-to-end path availability can be achieved by using a path lifetime of 50 s or 100 s, respectively.

On the other hand, we recommend the use of layer two mechanisms for connectivity maintenance whenever possible, which allow faster reactions to topology changes and better network connectivity. For example, it is possible to detect a link failure in acknowledged IEEE 802.15.4 networks in only 30 ms [67].

7.6. NUD message overhead

Using NUD incurs message overhead. A simple analytical model to calculate the rate of NS messages transmitted by a node is as follows:

$$NS\ rate = \frac{(1-q)*1 + q*MAX\_UNICAST\_SOLICIT}{Path\ lifetime} \tag{7.7}$$

where q is the probability of link unavailability (see equation (7.5)). Figure 7.7 illustrates the NS message rate for different path lifetime configurations and for different TLF values. We have assumed a *RETRANS_TIMER* of 1 s (i.e. the default value). As it can be seen, there is a trade-off between the path lifetime and NS message overhead. The path lifetime should be set to a small value in order to increase path connectivity, but on the other hand this will cause a message overhead increase. For

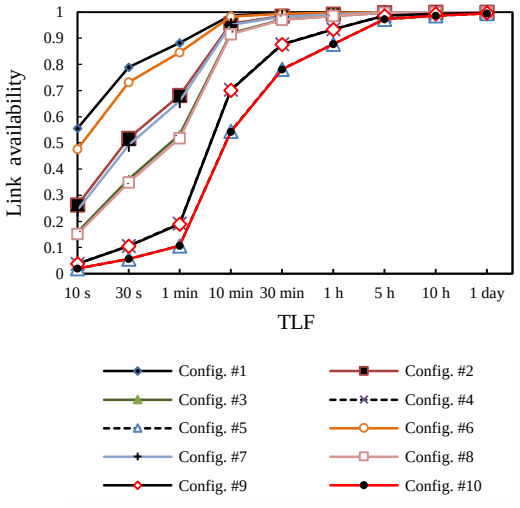


Figure 7.3. Probability of link availability, using various parameter configurations, for RPL and 6LoWPAN ND.

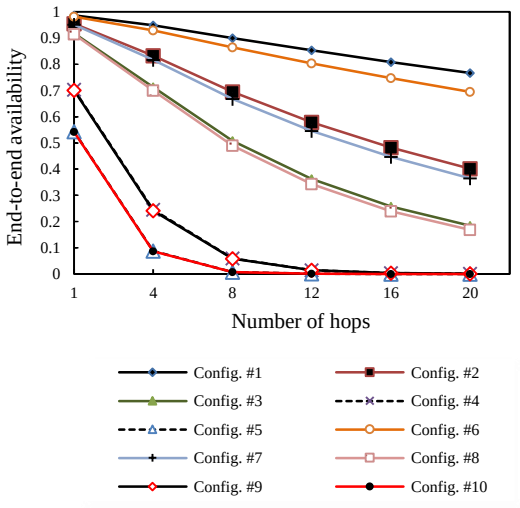


Figure 7.4. Probability of end-to-end path availability using various parameter settings, for RPL and 6LoWPAN ND, for TLF= 10 Minutes.

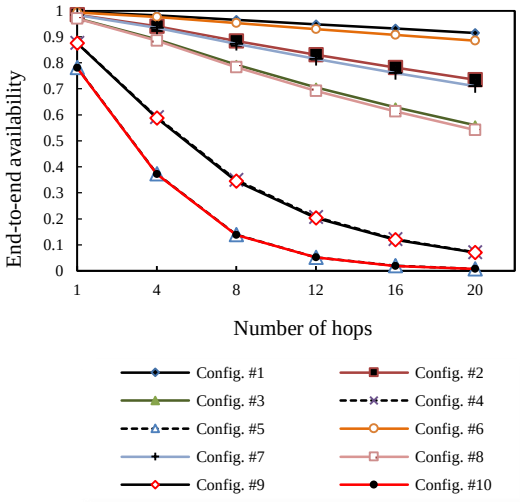


Figure 7.5. Probability of end-to-end path availability using various parameter settings, for RPL and 6LoWPAN ND, for TLF= 30 Minutes.

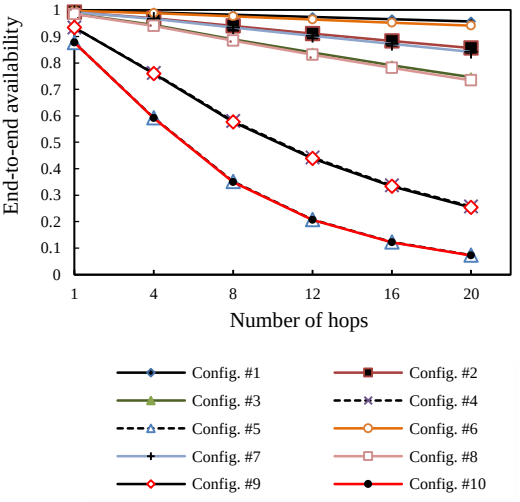


Figure 7.6. Probability of end-to-end path availability using various parameter settings, for RPL and 6LoWPAN ND, for TLF= 60 Minutes.

TLF values greater than one minute, the NS message overhead curves provide similar values because the probability of link availability is high.

7.7. Discussion

This chapter provided an analysis of the impact of various RPL and 6LoWPAN ND parameter settings on the link availability, the end-to-end path availability, and the message overhead incurred by RPL and 6LoWPAN ND for connectivity maintenance. Remarkably, important parameters such as default lifetime, path lifetime and

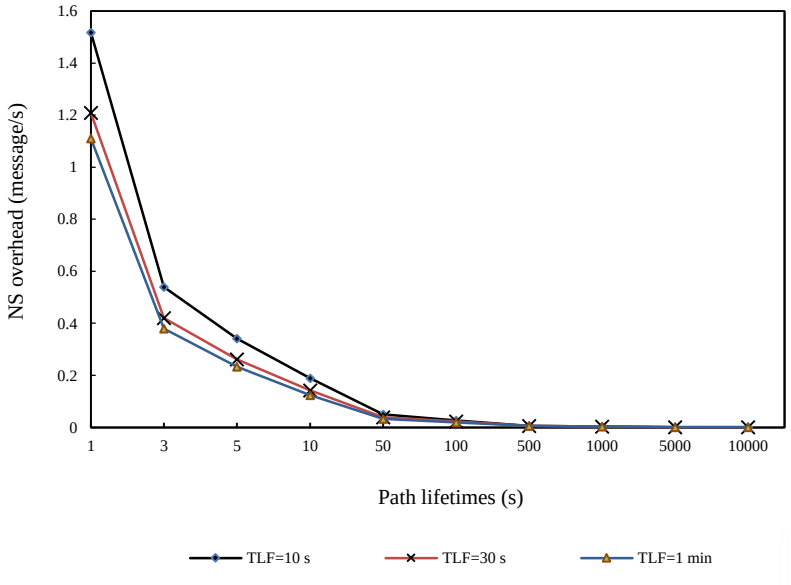


Figure 7.7. NS message overhead for various path lifetimes.

registration lifetime do not account with a default proposed value in the related specifications. Results show that careful tuning of the relevant parameters is critical for obtaining good network performance. There is a trade-off between connectivity maintenance and message overhead that depends on the path lifetime parameter. An appropriate configuration of the parameters considered depends on each particular LLN and application. In particular, the parameter choice has to be carried out depending on the expected path length and link lifetime of a scenario.

On the other hand, we recommend the use of layer two mechanisms for detecting link failures whenever possible, since they can be various orders of magnitude faster than layer-three based mechanisms.

8. Conclusions and future work

This chapter summarizes the main concluding remarks from this Ph.D. The chapter is divided in two parts as follows. Conclusions are presented in section 8.1 and future work is pointed out in section 8.2.

8.1. Conclusions

This thesis has presented a set of research contributions to the evaluation and eventual improvement of RPL, an IP-based routing protocol for operating over low power and lossy networks which has recently been developed by the IETF ROLL working group. Since RPL has been defined following a flexible approach (thus including many parameters and options), and given the key role that RPL plays in the Internet of Things, there is a need to evaluate and eventually propose improvement mechanisms for this protocol.

The remainder of this subsection is organized in four parts as follows. The first one presents conclusions on network convergence process in RPL over IEEE 802.15.4 multihop networks. The second one shows the main remarks from the analytical model presented for estimaing the expected DODAG convergence time in a network of RPL nodes that use IEEE 802.15.4. Third, the main results regarding the analytical model presented for the number of message transmissions allowed by Trickle in a WSN within a given time interval are provided. Finally, the fourth part summarizes the performance evaluation of the neighbor unreachability detection mechanisms in RPL when 6LoWPAN ND is used.

8.1.1. Network convergence process in RPL over IEEE 802.15.4 multihop networks: improvement and trade-offs

In the first contribution of this thesis we investigated by simulation the influence of the two most important RPL parameters, the redundancy constant parameter, *k*, and the minimum interval, I_{min}, on the network convergence process in IEEE 802.15.4 networks. We also proposed and evaluated a mechanism called DIS-Trickle for accelerating the DODAG convergence process by leveraging DIS messages.

Results show that increasing the value of *k* up to a value equal or close to the average node degree leads to a reduction of the network convergence time in all

network size and density scenarios. However, increasing the redundancy constant causes the number of DIO messages transmitted and collisions to increase as well. Therefore, there is a tradeoff between network convergence time and number of DIO messages transmitted and collisions. Results also show that as the network density increases, the influence of the redundancy constant grows with the network size. In sparse networks, the relative influence of *k* is independent of the network size.

Regarding the minimum interval value, results show that decreasing the I_{min} value decreases the network convergence time regardless of the network size, the network density or the redundancy constant. On the other hand, decreasing the I_{min} value causes the number of DIO message transmissions and collisions to increase.

Finally, results show that using DIS-Trickle decreases network convergence time in 2-3 orders of magnitude, regardless of the network size or density. The improvement provided by this mechanism grows as network density decreases. Interestingly, in dense networks, DIS-Trickle does not increase or even reduces the total number of RPL messages sent during network convergence. However, in sparse networks, DIS-Trickle creates a trade-off between network convergence time and RPL message overhead. The influence of DIS-Trickle on network convergence performance decreases with the redundancy constant.

8.1.2. Network convergence process in RPL over IEEE 802.15.4 multihop networks: improvement and trade-offs

In chapter 5, we presented an analytical model to estimate the expected DODAG convergence time in a network of RPL nodes that use IEEE 802.15.4, and in the presence of uncorrelated bit errors. The model assumes a static *N*-hop chain network where all links are of equal characteristics and uncorrelated, and constitutes a lower bound on the DODAG convergence time in any network of a maximum number of hops between the DODAG root and non-root nodes equal to N. Results show a linear increase of the network convergence time with the number of hops in the network. Results also show that bit errors do not have a significant effect on the network convergence time for BER values up to $5 \cdot 10^{-4}$. However, greater values may lead to degradation of this performance parameter, and BER values close to 10^{-3}, may have a dramatic effect on the user experience for a chain topology length greater than 3 hops.

8.1.3. Modeling the Message Count of the Trickle Algorithm in a Steady-State, Static Wireless Sensor Network

Chapter 6 presents an analytical model for the number of message transmissions allowed by Trickle in a steady-state WSN within a given time interval. Although the model assumes a synchronous network, results show the model can also be used in asynchronous networks. Since Trickle operation is critical for performance parameters such as energy consumption and available bandwidth, the model can be useful for network engineers and researchers in the planning, deployment and evaluation of Trickle-based networks. As expected, message overhead grows with the redundancy constant, thus leading to a tradeoff between the message overhead (in steady state and also during network convergence) and fast network convergence.

8.1.4. Route change latency with RPL and 6LoWPAN Neighbor Discovery

Chapter 7 evaluates the performance of the neighbor unreachability detection mechanisms in RPL when 6LoWPAN ND is used. We studied the effect of relevant parameters on performance in terms of link availability, end-to-end path availability, and message overhead. Results show that careful tuning of the relevant RPL and 6LoWPAN ND parameters, such as default lifetime, path lifetime and registration lifetime, is critical for obtaining good network performance. However the RPL and 6LoWPAN ND specifications have not proposed any default value for these parameters. Results show a trade-off between connectivity maintenance and message overhead which depends on the path lifetime parameter. In order to achieve high path connectivity, the path lifetime value should be set to a small value, at the expense of a message overhead increase.

8.2. Future work

This section presents several future work directions that can be derived from the research presented in this document. Some specific areas are listed below where future work can be carried out to extend the work presented in this PhD:

- Designing an adaptive mechanism to determine a value for the initial delay in RPL when DIS-Trickle is used. As we mentioned in chapter 4, the value of this delay has to be set carefully on the basis of the network size and density.

- Developing an adaptive mechanism for Trickle redundancy constant autoconfiguration, based on the number of neighbors of a node. As the network density can change over the network lifetime due to battery depletion, hardware breakdown or operating system misbehavior, or due to the presence of new nodes, using such an adaptive mechanism will help nodes in a network to configure their redundancy constant based on their own local density measurements.
- Extending the analytical model presented for estimating the DODAG convergence time for a random topology, IEEE 802.15.4 multihop network. The model should capture the influence of the redundancy constant on the DODAG convergence time.
- The evaluation and potential improvement of the recently created "Multicast Protocol for Low power and Lossy Networks (MPL)", which is based on the Trickle algorithm.
- Analysis of P2P-RPL. This protocol is based on the reactive discovery of routes which comprises two phases: i) the dissemination of request messages, by using the Trickle algorithm, and ii) a response from the target node. The work in this PhD about network convergence time mostly applies to the first phase of the P2P-RPL route discovery. However, for a complete P2P-RPL study, the second phase should also be added.

References

Contributions

Journal papers

[P1] H. Kermajani, C. Gomez, M. H. Arshad, “Modeling the message count of the Trickle algorithm in a steady-state, static wireless sensor network", IEEE Communications Letters, Vol. 16, Issue 12, 2012, pp. 1960-1963.

[P2] H. Kermajani, C. Gomez, “On the network convergence process in RPL over IEEE 802.15.4 multihop networks: improvement and trade-offs”, submitted to Sensors (MDPI), Current state: “minor revisions”

Conference papers

[P3] Kermajani, H.R; Gomez, C. “Route Change Latency in Low-Power and Lossy Wireless Networks using RPL and 6LoWPAN Neighbor Discovery”, in proceedings of the 16th IEEE Symposium on Computers and Communications (ISCC’11), Kerkyra (Corfu), Greece, June 28 - July 1, 2011.

[P4] Kermajani, H.R; Gomez, C. “Modeling the network convergence time in RPL in error-prone, IEEE 802.15.4 chain topology multihop network”, submitted to the 11th IEEE International Symposium on Wireless Communication Systems (ISWCS’2014), Current state: “Accepted for publication”

Implementations

[P5] H. Kermajani, RPL simulation code for OMNeT++: https://sites.google.com/site/carlesgomez/home/code.

Projects

This PhD has been supported by (and has contributed to) the projects listed below:

[P6] “Nuevas Arquitecturas para Internet Ubicua: Diseño y Evaluación”, TEC2009-11453. Ministerio de Ciencia e Innovación (Spanish Government). Project leader: Dr. Anna Calveras Augé.

[P7] "Arquitectura de Internet basada en servicios atómicos para redes restringidas y ubicuas", TEC2012-32531. Ministerio de Ciencia e Innovación (Spanish Government). Project leader: Dr. Carlos Gómez Montenegro.

Full list of references

[1] C. Gomez, J. Paradells, J.E. Caballero, "Sensors everywhere: wireless network technologies and solutions", Fundación Vodafone España, 2010.

[2] JP. Vasseur," Terms Used in Routing for Low-Power and Lossy Networks", RFC 7102, January 2014.

[3] 6LoWPAN charter: http://datatracker.ietf.org/wg/6lowpan/charter/

[4] ROLL charter: http://datatracker.ietf.org/wg/roll/charter/

[5] T. Winter (Ed.) et al., "RPL: IPv6 Routing Protocol for Low power and Lossy Networks," RFC 6550, March 2012.

[6] A. Brandt, J. Buron, G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5826, April 2010.

[7] J. Martocci (Ed.), P. De Mil, N. Riou, W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5867, June 2010.

[8] K. Pister (Ed.), P. Thubert (Ed.), S. Dwars, T. Phinney, " Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, October 2009.

[9] M. Dohler (Ed.), T. Watteyne (Ed.), T. Winter (Ed.), D. Barthel (Ed.), "Routing Requirements for Urban Low-Power and Lossy Networks", RFC 5548, May 2009.

[10] E-mail from IETF ROLL Working Group cochair, J. P. Vasseur. December 2013. (Available at: http://www.ietf.org/mail-archive/web/roll/current/msg08376.html).

[11] D. Culler, D. Estrin, M. Srivastava, "Guest Editors' Introduction: Overview of Sensor Networks", IEEE Journals, Vol. 37 Issue: 8, August 2004, Pages: 41 – 49.

[12] C. Gomez, J. Paradells, "Wireless home automation networks: A survey of architectures and technologies", IEEE Communications Magazine, Vol. 48 Issue: 6, June 2010, pages: 92 – 101.

[13] M. Tubaishat, S. Madria, "Sensor Networks: An Overview", IEEE Potentials, Vol. 22, No. 2, April/May 2003.

[14] IEEE 802.15.4, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)", May 2003.

[15] P. Levis, A. Tavakoli, S. Dawson-Haggerty, ”Overview of Existing Routing Protocols for Low Power and Lossy Networks”, Internet Draft, draft-ietf-roll-protocols-survey-07, April 2009.

[16] J. Moy, “OSPF Version 2”, RFC 2328, April 1998.

[17] R. Coltun, D. Ferguson, J. Moy, “OSPF for IPv6”, RFC 2740, December 1999.

[18] D. Oran (Ed.), “OSI ISIS Intra-domain Routing Protocol”, RFC 1142, February 1990.

[19] T. Clausen (Ed.), P. Jacquet (Ed.), “Optimized Link State Routing Protocol (OLSR)”, RFC 3626, October 2003.

[20] R. Ogier, F. Templin, M. Lewis, “Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)”, RFC 3684, February 2004.

[21] G. Malkin, “RIP Version 2”, RFC2453, November 1998.

[22] C. Perkins, E. Belding-Royer, S. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing”, RFC 3561, July 2003.

[23] S. Harnedy, R. Cole, I. Chakeres, “Definition of Managed Objects for the DYMO Manet Routing Protocol”, Internet Draft, draft-ietf-manet-dymo-mib-04, January 2011 (work in progress).

[24] D. Johnson, Y. Hu, D. Maltz, “The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4”, RFC 4728, February 2007.

[25] P. Levis, T. Clausen, J. Hui, O. Gnawali, J. Ko, “The Trickle Algorithm,” RFC 6206, March 2011.

[26] M. Goyal (Ed.) et al, “Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks”, RFC 6997, August 2013.

[27] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[28] J. Hui, JP. Vasseur, “The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams”, RFC 6553, March 2012.

[29] P. Levis, N. Patel, D. Culler, S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks", in Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation, NSDI 2004, March 2004.

[30] “COOJA/ContikiRPL simulator” [Online]. Available: http://www.contiki-os.org/start.html.

[31] JP. Vasseur (Ed.), M. Kim (Ed.), K. Pister, N. Dejean, D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, March 2012.

[32] P. Thubert (Ed.)," Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, March 2012.

[33] O. Gnawali, P. Levis, "The Minimum Rank with Hysteresis Objective Function", RFC , September 2012.

[34] Z. Shelby (Ed.), S. Chakrabarti, E. Nordmark, C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

[35] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, P. Levis, "Collection Tree Protocol", In Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys 2009), Berkeley, CA, USA, November 2009.

[36] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, A. Woo, "TEP 123: The Collection Tree Protocol", Aug. 2006.

[37] J.W. Hui, D. E. Culler, "IP is dead, long live IP for wireless sensor networks", In Proc. of the SenSys Conf., New York, NY, USA, November 2008, pages 15–28.

[38] "ZigBee specification", r17, ZigBee Alliance, October 2007.

[39] "Z-Wave protocol overview", Version 4, May 2007.

[40] "INSTEON. The Details", August 2005.

[41] Wavenis Open Standard Alliance website: www.wavenis-osa.org (accessed in December 2009).

[42] T. Clausen and U. Herberg, "Multipoint-to-point and broadcast in RPL," INRIA, Technical Report No. 7244, 2010.

[43] J . Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis, "Evaluating the performance of RPL and 6LoWPAN in TinyOS," in IPSN, 2011.

[44] O. Gaddour, A. Koub, S. Chaudhry, M. Tezeghdanti, R. Chaari, M. Abid, "Simulation and Performance Evaluation of DAG Construction with RPL", Third International Conference on Communications and Networking (ComNet), March-April 2012.

[45] O. Gaddour and A. Koub´ıA, "RPL in a nutshell: A survey", Comput. Netw., vol. 56, no. 14, pp. 3163–3178, Sep. 2012.

[46] M. Becker, K. Kuladinithi, C. Görg, "Modelling and simulating the Trickle algorithm", Mobile Networks and Management, 2012, pp. 135-144.

[47] T.M.M. Meyfroyt, “Modeling and analyzing the Trickle algorithm”, Master thesis, August 2013.

[48] A. Tripathi, J. de Oliveira, and J. Vasseur, “Performance evaluation of routing protocol for low power and lossy networks,” IETF, Draft, 2012.

[49] T. Clausen and U. Herberg, “Comparative study of RPL-enabled optimized broadcast in wireless sensor networks,” in Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), dec 2010, pp. 7 –12.

[50] I. Emilian Radoi, A. Shenoy, D.K. Arvind, “Evaluation of Routing Protocols for Internet-Enabled Wireless Sensor Networks”, in ICCGI 2012.

[51] L. Ben Saad, C. Chauvenet, and B. Tourancheau, “Simulation of the RPL Routing Protocol for IPv6 Sensor Networks: two cases studies,” in SENSORCOMM. IARIA, Aug. 2011.

[52] H. Kermajani, C. Gomez, M. H. Arshad, “Modeling the message count of the Trickle algorithm in a steady-state, static wireless sensor network", IEEE Communications Letters, Vol. 16, Issue 12, 2012, pp. 1960-1963.

[53] N. Accettura, L. A. Grieco, G. Boggia, P. Camarda “Performance Analysis of the RPL Routing Protocol”, in proc. of ICM 2011, Istanbul, Turkey, April 2011.

[54] A. Varga, "The omnet++ discrete event simulation systems," in European Simulation Multiconference (ESM '200l), June 2001.

[55] MiXiM simulator for wireless and mobile networks using OMNeT++. Available: http://mixim.sourceforge.net/.

[56] H. Kermajani, RPL simulation code for OMNeT++: https://sites.google.com/site/carlesgomez/home/code.

[57] Texas Instruments Incorporated, “2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver,” http://www.ti.com/lit/gpn/cc2420.

[58] TELOSB MOTE PLATFORM, http://www.willow.co.uk/TelosB_Datasheet.pdf.

[59] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, D. Culler, “TinyOS: An Operating System for Wireless Sensor Networks”, pp. 115–148 (2005).
http://link.springer.com/chapter/10.1007%2F3-540-27139-2_7.

[60] IEEE Std. 802.15.4-2006, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specic requirements Part 15.4: Wireless Medium

Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE, Inc., September 2006.

[61] J. Hui (Ed.) et al. ," Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.

[62] C. Bettstetter, "On the minimum node degree and connectivity of a wireless multihop network", in Proc. of ACM MobiHoc'02, Lausanne, Switzerland, June 2002.

[63] A. Woo, T. Tong, and D. Culler,"Taming the underlying challenges of reliable multihop routing in sensor networks", in proc. of SenSys '03, Los Angeles, CA, USA, November 2003.

[64] C. Gomez, A. Boix, J. Paradells, "Impact of LQI on the Performance of a One-to-One Routing Protocol for IEEE 802.15.4 Multihop Networks", EURASIP Journal on Wireless Communications and Networking, Volume 2010, Article ID 205407, October 2010.

[65] C. Gomez, D. Garcia, J. Paradells, "Improving Performance of a Real Ad-hoc Network by Tuning OLSR", in proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005), 2005.

[66] C. Gomez, A. Boix, J. Paradells, "Impact of LQI-Based Routing Metrics on the Performance of a One-to-One Routing Protocol for IEEE 802.15.4Multihop Networks", EURASIP Journal on Wireless Communications and Networking, Vol. 2010, February 2010.

[67] C. Gomez, P. Salvatella, O. Alonso, J. Paradells, "Adapting AODV for IEEE 802.15.4 Mesh Sensor Networks: Theoretical Discussion and Performance Evaluation in a Real Environment", in proc. of WoWMoM'06, Niagara Falls, June 2006.

Acronyms

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ADC	Analogue to Digital Converter
AODV	Ad hoc On-Demand Distance Vector
API	Application Programming Interface
APL	Application
BER	Bit Error Rate
BPSK	Binary PSK
CCA	Clear Channel Assessment
CDF	Cumulative Distribution Function
CPM	Closest Pattern Matching
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTP	Collection Tree Protocol
DAO	Destination Advertisement Object
DAO-ACK	Destination Advertisement Object Acknowledgment
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination Oriented Directed Acyclic Graphs
DRO	Discovery Reply Object
DSR	Dynamic Source Routing Protocol
DSSS	Direct Sequence Spread Spectrum
DYMO	Dynamic Mobile Ad-hoc Networks On-demand
ETX	Expected Transmission count
FCF	Frame Control Field
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IS-IS	Intermediate System-to-Intermediate System
LLN	Low power and Lossy Network
LQL	Link Quality Level
MAC	Medium Access Control
MFR	MAC footer

MHR	MAC header
MRHOF	Minimum Rank with Hysteresis Objective Function
MSDU	MAC Service Data Unit
NA	Neighbor Advertisement
ND	Neighbor Discovery
NS	Neighbor Solicitation
NUD	Neighbor Unreachability Detection
NWK	Network
O-QPSK	Offset-Quadrature PSK
OF	Objective Function
OLSRv2	Optimized Link State Routing Protocol version 2
OSPF	Open Shortest Path First
P2P	Point-to-Point
PHR	PHY header
PHY	Physical
PLC	Power Line Communication
PPDU	PHY protocol data unit
PSDU	PHY Service Data Unit
PSK	Phase-Shift-Key
QoS	Quality of Service
RCL	Route Change Latency
RDO	Route Discovery Option
RF	Radio Frequency
RIP	Routing Information Protocol
ROLL WG	Routing Over Low power and Lossy networks Working Group
RPL	IPv6 Routing Protocol for Low power and Lossy Networks
RSSI	Received Signal Strength Indicator
SFD	Start-of-Frame Delimiter
SHR	Synchronization header
TBRPF	Topology Dissemination Based on Reverse-Path Forwarding
TLF	Time to Link Failure
WSN	Wireless Sensor Network