

# An Analysis of the RPL Routing Standard for Low Power and Lossy Networks

*Aishwarya Parasuram*  
*David Culler, Ed.*  
*Randy Katz, Ed.*

Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2016-106

<http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-106.html>

May 14, 2016



Copyright © 2016, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

# **An Analysis of the RPL Routing Standard for Low Power and Lossy Networks**

by

Aishwarya Parasuram

A dissertation submitted in partial satisfaction of the  
requirements for the degree of  
Master of Science

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Dr. David Culler, Research Advisor  
Dr. Randy Katz, Chair (EECS, UC Berkeley)

Spring 2016

# **An Analysis of the RPL Routing Standard for Low Power and Lossy Networks**

Copyright 2016  
by  
Aishwarya Parasuram

---

# An Analysis of the RPL Routing Standard for Low Power and Lossy Networks

by Aishwarya Parasuram

---

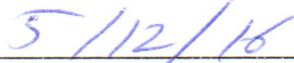
## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences,  
University of California at Berkeley, in partial satisfaction of the requirements for  
the degree of **Master of Science, Plan II.**

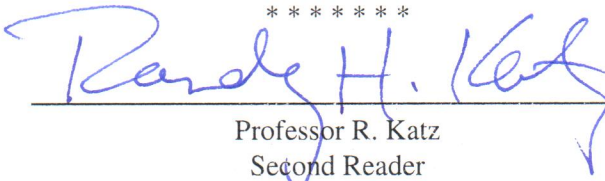
Approval for the Report and Comprehensive Examination:

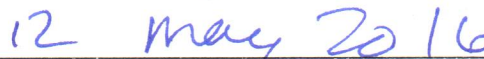
  
Committee:

\_\_\_\_\_  
Professor D. E. Culler  
Research Advisor

  
\_\_\_\_\_  
(Date)

\*\*\*\*\*

  
\_\_\_\_\_  
Professor R. Katz  
Second Reader

  
\_\_\_\_\_  
(Date)

## **Abstract**

An Analysis of the RPL Routing Standard for Low Power and Lossy Networks

by

Aishwarya Parasuram

Master of Science in Computer Science

University of California, Berkeley

Dr. David Culler, Research Advisor

RPL is a distance-vector routing protocol designed by the ROLL Working Group in order to cater to the specific needs of low-power and lossy networks (LLNs). It is specified in the standards document RFC 6550 and is the emerging standard for routing in Wireless Sensor Networks. RPL has been widely criticized for a number of reasons, including underspecification and complexity of implementation. This thesis analyzes the RPL routing standard with regards to specification, performance, comparison with other routing standards, open source and industrial implementations, as well as improvement efforts. It also proposes an alternative to the RPL routing standard, RPL-Lite, that overcomes the shortcomings of the current RPL design. RPL-Lite reduces the feature set by including only the most necessary features required for routing. By doing so, it reduces the implementation complexity and makes it more suitable for deployment on resource constrained nodes.

## Dedication

I would like to dedicate this work to the Almighty, whose grace and kindness has shown me light on the darkest of days.

# Contents

<b>Contents</b>	<b>ii</b>
<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement . . . . .	1
1.2 Motivation . . . . .	1
1.3 Roadmap . . . . .	2
<b>2 Need for RPL</b>	<b>4</b>
2.1 Wireless Sensor Networks (WSNs) . . . . .	4
2.2 Low Power and Lossy Networks (LLNs) . . . . .	4
2.2.1 Introduction to LLNs . . . . .	4
2.2.2 Unique Challenges for Routing in LLNs . . . . .	5
2.3 Routing in LLNs before creation of ROLL . . . . .	5
2.4 Initial Goals of ROLL Working Group . . . . .	6
<b>3 Understanding RPL</b>	<b>7</b>
3.1 Introduction . . . . .	7
3.1.1 What is RPL . . . . .	7
3.1.2 About this Chapter . . . . .	7
3.2 RPL Topology . . . . .	8
3.2.1 DODAG . . . . .	8
3.2.2 Rank . . . . .	8
3.2.3 DODAG Root . . . . .	8
3.2.4 RPL Instances . . . . .	8
3.3 RPL Control Messages . . . . .	9
3.4 RPL Operation . . . . .	9
3.4.1 Supported Traffic Patterns . . . . .	9
3.4.2 Modes of Operation . . . . .	9



3.4.3	Upward Routes . . . . .	10
3.4.4	Downward Routes . . . . .	10
3.4.5	Loop Detection and Avoidance . . . . .	11
3.5	Objective Function . . . . .	11
3.6	Trickle Timer . . . . .	12
3.7	Security Features . . . . .	12
3.8	Features of RPL Necessary for Interoperability . . . . .	13
<b>4</b>	<b>Analysis of RPL</b>	<b>14</b>
4.1	Introduction . . . . .	14
4.2	Performance Evaluation Studies . . . . .	14
4.3	Comparative Studies . . . . .	17
4.3.1	RPL and CTP . . . . .	17
4.3.2	RPL and LOADng . . . . .	18
4.3.3	Proactive vs reactive Routing Protocols for WSNs . . . . .	21
4.4	Summary . . . . .	22
<b>5</b>	<b>RPL Implementations</b>	<b>26</b>
5.1	Open Source Implementations of RPL . . . . .	26
5.1.1	SimpleRPL . . . . .	26
5.1.2	TinyRPL . . . . .	26
5.1.3	ContikiRPL . . . . .	27
5.1.4	RIOT-RPL . . . . .	27
5.2	Industrial Implementations of RPL . . . . .	27
5.3	Issue-by-Issue Analysis . . . . .	28
5.3.1	Incompatible Modes of Operation . . . . .	28
5.3.2	Multiple Instances . . . . .	28
5.3.3	Objective Functions . . . . .	28
5.3.4	Security . . . . .	29
5.3.5	Floating DODAGs and Local DODAGs . . . . .	29
5.3.6	Underspecification of Local and Global Repair Trigger . . . . .	29
5.3.7	General Size of Code Base . . . . .	29
5.4	Summary . . . . .	30
<b>6</b>	<b>Past Work on RPL Improvements</b>	<b>31</b>
6.1	Combined Metrics . . . . .	31
6.1.1	Additive and Lexical Composition of Metrics . . . . .	32
6.1.2	CA-RPL . . . . .	33
6.1.3	QoS-Aware Fuzzy Logic Objective Function . . . . .	34
6.1.4	Per-Hop ETX . . . . .	36
6.1.5	Improved Energy Efficiency . . . . .	36
6.2	Average Delay Metric . . . . .	37

6.3	Multipath forwarding schemes . . . . .	39
6.3.1	Efficient Topology Construction . . . . .	39
6.3.2	Increased Network Lifetime . . . . .	40
6.3.3	Improved Packet Delivery Ratio (LQA-RPL) . . . . .	41
6.3.4	Support for Anycast . . . . .	41
6.4	Broadcast Support . . . . .	42
6.5	Objective Function . . . . .	43
6.6	Summary . . . . .	44
<b>7</b>	<b>Need for a New Standard</b>	<b>47</b>
7.1	Introduction . . . . .	47
7.2	Unnecessary Features of RPL . . . . .	47
7.3	Under-specification of standards document . . . . .	48
7.4	Known Issues . . . . .	50
7.5	Types of applications RPL does not cater to . . . . .	53
7.6	Features that would benefit RPL . . . . .	54
7.7	Has RPL succeeded or failed . . . . .	54
<b>8</b>	<b>RPL-Lite</b>	<b>56</b>
8.1	Introduction . . . . .	56
8.1.1	Design Principles . . . . .	56
8.1.2	Expectations of Link-Layer Type . . . . .	57
8.2	Protocol Overview . . . . .	57
8.2.1	Topologies . . . . .	57
8.2.1.1	Constructing Topologies . . . . .	57
8.2.1.2	RPL-Lite Identifiers . . . . .	58
8.2.1.3	RPL-Lite DAGs . . . . .	58
8.2.2	Upward Routes and DAG Construction . . . . .	58
8.2.2.1	Objective Function . . . . .	58
8.2.2.2	DAG Repair . . . . .	58
8.2.2.3	Security . . . . .	59
8.2.2.4	Administrative Preference . . . . .	59
8.2.2.5	Data-Path Validation and Loop Detection . . . . .	59
8.2.2.6	Distributed Algorithm Operation . . . . .	59
8.2.3	Downward Routes and Destination Advertisement . . . . .	60
8.2.4	Rank Properties . . . . .	61
8.2.5	Loop Detection, Avoidance and Recovery . . . . .	61
8.2.5.1	DAG Loops . . . . .	61
8.2.5.2	DA Loops . . . . .	61
8.3	Traffic Flows Supported by RPL-Lite . . . . .	61
8.3.1	Multipoint-to-Point Traffic . . . . .	62
8.3.2	Point-to-Multipoint Traffic . . . . .	62

8.3.3	Point-to-Point Traffic . . . . .	62
8.4	RPL-Lite Control Messages . . . . .	62
8.4.1	RPL-Lite Equivalents of RPL Control Messages . . . . .	63
8.4.1.1	DAG Information Solicitation Messages . . . . .	63
8.4.1.2	DAG Information Advertisement Messages for Upward Routes . . . . .	63
8.4.1.3	Destination Advertisement Objects for Downward Routes . . . . .	63
8.4.1.4	Destination Advertisement Acknowledgement Messages . . . . .	64
8.4.1.5	Consistency Check Messages . . . . .	64
8.4.2	RPL-Lite Control Message Options . . . . .	64
8.5	Upward Routes . . . . .	66
8.5.1	Upward Route Discovery and Maintenance . . . . .	66
8.5.1.1	Neighbors and Parents . . . . .	66
8.5.1.2	DAG Roots . . . . .	67
8.5.1.3	Poisoning . . . . .	67
8.5.2	Node Advertisement Transmission . . . . .	67
8.5.3	Operation as a Leaf Node . . . . .	68
8.6	Downward Routes . . . . .	68
8.6.1	Destination Advertisement Parents . . . . .	68
8.6.2	Downward Route Discovery and Maintenance . . . . .	69
8.6.3	INDA Transmission Scheduling . . . . .	69
8.6.4	Downward Routing Mechanism . . . . .	70
8.7	Loop Avoidance and Detection . . . . .	70
8.7.1	DAG Inconsistency and Loop Detection . . . . .	71
8.7.2	DA Inconsistency Detection and Recovery . . . . .	71
8.7.3	Global Repairs . . . . .	71
8.7.4	Local Repairs . . . . .	72
<b>9</b>	<b>Conclusion</b>	<b>73</b>
	<b>Bibliography</b>	<b>75</b>

# List of Figures

8.1	IPv6 Neighbor Discovery and IPv6 Inverse Neighbor Discovery Option Format .	62
8.2	IPv6 Neighbor Discovery Option for RPL-Lite DIO . . . . .	63
8.3	IPv6 Inverse Neighbor Discovery Advertisement Message . . . . .	64
8.4	DAG Configuration option for RPL-Lite . . . . .	65
8.5	Solicited Information option for RPL-Lite . . . . .	65

# List of Tables

4.1	Performance Evaluation Studies of RPL . . . . .	23
4.2	Comparative Studies . . . . .	25
6.1	Studies on Improvements to the RPL Routing Standard . . . . .	46

## Acknowledgments

Firstly, I would like to thank my parents Subbalakshmi Parasuram and Tirunelveli Ratnagiri Parasuraman, for their unconditional love and support. Their fierce belief in my potential led me to dream beyond what I am. I would like to thank my sister Sahana Parasuram, who has shown love to me on days when I couldn't love myself. She is my greatest inspiration and has taught me that love and kindness conquers all. I would also like to sincerely thank my research advisor, Dr. David Culler, for giving me the opportunity to pursue graduate studies at my dream university and gain knowledge from some of the best computer scientists in the world. His wisdom and foresight was most inspiring during my journey at Berkeley. I would like to thank all the supporting staff at UC Berkeley. They have been so kind and encouraging throughout and have helped me feel welcome from the very first day. I would like to thank my colleagues and lab-mates. They exemplify self-motivation, and have taught me that success is only secondary to truly enjoying what you do. I would like to thank my grandparents, friends and family members for motivating me throughout. I am very lucky to have you all in my life. I would also like to thank God, for always showering His grace and love upon me and helping me see light on the darkest of days. I am so blessed to be at the receiving end of such priceless gifts, and will forever be grateful to each and every one of you.

# Chapter 1

## Introduction

### 1.1 Problem Statement

In this thesis, we aim to analyze the disadvantages and drawbacks of the Routing Protocol for Low-power and Lossy Networks (RPL) [77] and propose a high-level specification for an interoperable and easy to implement protocol, RPL-Lite.

### 1.2 Motivation

RPL was proposed as a solution for routing in low-power and lossy networks (LLNs) and catered to unique routing challenges. LLNs are typically resource constrained in terms of memory, battery life and processing power. They include Wireless Personal Area Networks (WPANs), low-power Power Line Communication networks (PLC) and Wireless Sensor Networks (WSNs). Standard routing protocols such as OSPF were not suited for the special challenges that LLNs posed. RPL was specially designed to overcome these challenges. It included many specific features such as dynamic rate of control message dispatch based on network consistency and addressing topology changes only when data packets have to be sent[73]. Due to such design considerations, RPL was able to remain conservative in terms of constrained resources. However, the RPL feature set included repetitions of tasks already performed by other IP layers. It also included many unnecessary features that were never used in real deployments. Consequently, the specification proved much too complex to implement in entirety on a single resource constrained node.

Due to this, many current implementations of RPL only implement a subset of the original feature-set, making them non-interoperable. In order to be standards-compliant and thereby interoperable, it is required that the implementations include a certain set of features, which is not always possible given the size and memory limitations of the nodes operating in LLNs. Additionally, the underspecification and ambiguity in the standards document give rise to a large number of implementation choices, many of which adversely impact overall perfor-

mance.

This thesis analyses the shortcomings of RPL, and also proposes a new routing protocol that could potentially serve as a standard built off of RPL with less complexity and a reduced feature-set.

## 1.3 Roadmap

Chapter 2 provides background on the need for a routing standard like RPL. It gives a basic introduction to Wireless Sensor Networks and Low-Power and Lossy networks, and also describes the unique routing challenges in LLNs. A background on routing protocols before the design and standardization of RPL is also provided, and the initial goals of the ROLL Working Group [62] in charge of designing RPL are discussed.

In Chapter 3, the RPL routing standard is analyzed in detail. All information in this chapter is a direct summarization of the standards document RFC6550, and does not include any personal views or opinions. The purpose of this chapter is to provide a broad overview of the RPL routing standard, for further speculation in the following chapters.

Chapter 4 summarizes a number of past studies on RPL. There have been a number of works that focus on quantitatively and qualitatively evaluating RPL's performance with regards to a number of parameters, such as energy efficiency, routing overhead and scale. It was observed that a large majority of such work were tested only on simulators. RPL is also evaluated against routing protocols such as Collection Tree Protocol (CTP) [22] and LOADng [7], which have been designed for similar purposes.

In Chapter 5, a number of open-source and industrial implementations of RPL are studied. Some of the open-source implementations include the popularly deployed ContikiRPL and the recently designed RIOT-RPL. Industrial implementations have not been inspected in as much detail because of proprietary ownership of the code base. However, a study of these implementations points out a number of flaws in the current design, which are presented in this section.

A large number of efforts have been made towards improving the RPL routing standard. Some of these efforts include designing new metrics for better suiting application needs. Others include optimizing RPL's performance by using multipath schemes and providing a higher degree of routing redundancy. Many such works have been analyzed in Chapter 6, and they indicate that the current design is unsuitable for a number of LLN scenarios.

Chapter 7 revisits the need for a new routing standard by succinctly pointing out the unnecessary features in the RPL standards document, the under-specifications that lead to



diverse and non-interoperable implementation choices as well as a number of known issues that RPL faces, in spite of its highly comprehensive feature set. This chapter also looks into the types of applications for which RPL is not fully suited for, and investigates how successful RPL is in catering to the unique challenges posed by LLNs.

Finally, Chapter 8 proposes RPL-Lite, which is a new routing protocol that is built off of the current RPL specification. RPL-Lite is intended to be a lightweight design of RPL that weeds out unnecessary features and modifies the current standard to be less complex and more easily implementable.

# Chapter 2

## Need for RPL

### 2.1 Wireless Sensor Networks (WSNs)

According to [80], WSNs consist of small and cheap nodes with processing, communication and sensing capabilities that cooperatively interact to carry out complex monitoring tasks in a geographical area of interest. WSNs are used in a number of applications today such as industrial monitoring, building automation (HVAC, lighting, access control, fire), connected home, health-care, environmental monitoring, urban sensor networks, assets tracking and refrigeration. They represent a key technology that will revolutionize human life in the upcoming years, providing at the same time new business opportunities [1]. There are many WSN applications that play a crucial role in important domains such as smart-cities, environmental monitoring, distributed sensing in industrial plants and healthcare. However according to [25], WSNs assume an a priori knowledge of the traffic patterns to optimize for, with sensor-to-controller traffic (multipoint-to-point) being predominant, controller-to-sensor traffic (point-to-multipoint) being rare and sensor-to-sensor traffic being somewhat esoteric. Hence a number of routing protocols designed for such networks prioritize MP2P traffic pattern over others [73][22]. It is often acceptable to have longer paths in order to reduce the amount of control traffic flowing within the network.

### 2.2 Low Power and Lossy Networks (LLNs)

#### 2.2.1 Introduction to LLNs

Low power and lossy networks (LLNs) are those in which the routers and their interconnects are highly resource constrained. Routers are usually limited in terms of processing power, battery and memory, and their interconnects are characterized by unstable links with high loss rates, low data rates and low packet delivery rates. The traffic patterns are also varied, and may comprise of point to point (P2P), point to multipoint (P2MP) or multipoint to point (MP2P). They can potentially comprise thousands of nodes [77].

### 2.2.2 Unique Challenges for Routing in LLNs

LLNs are inherently different from standard networks because they are highly resource constrained at the routers as well as the interconnects. Routers are constrained in terms of memory, processing power and battery life. The interconnects are lossy with high packet-drop rate. These networks use a wide variety of communication technologies including both wired and wireless. Furthermore, these networks can potentially comprise of thousands of nodes. They also have to support multiple types of traffic patterns.

An additional consideration is that high data-traffic very easily leads to network congestion [68]. Such scenarios cause large amount of packet loss and delay. Since sensor networks are commonly deployed in environments with potentially high data-traffic and in many cases require a time-sensitive response, it is necessary that such concerns need to be addressed while designing a routing protocol for LLNs.

The existing routing protocols such as Open Shortest Path First (OSPF) [8], Intermediate System to Intermediate System (IS-IS) [12], Ad Hoc on Demand Vector (AODV) [53] and Optimized Link State Routing (OLSR) [67] have been extensively evaluated by the ROLL Working Group (ROLL-WG) [62] and have been found to be unsuccessful satisfying the requirements of LLNs [41]. For example, path selection must be designed to take into consideration the specific power capabilities, attributes and functional characteristics of the links and nodes in the network.

## 2.3 Routing in LLNs before creation of ROLL

Initially, it was thought that the Internet architecture was not suited for sensor networks. This was due to a number of reasons. Firstly, it was assumed that sensor networks are designed to specifically cater to the requirements of a single application domain, hence having a generic architecture that could accommodate a wide range of applications was unnecessary. Also, it was thought that the end-to-end architecture was unhelpful for the localized algorithms and in-network processing required to achieve robustness and scalability in such networks [36][18].

Due to this, the progress and advancement in developing network abstractions and new protocols for sensor networks was a collection of disjoint, non-interoperable and dispersed efforts within the community. Additionally these networks were not able to communicate with each other (due to non-standardization of communication protocols) or with the wider Internet since they were not using IP. They required the use of complex application-layer gateways which added further complexity and overhead of design and coordination.

## 2.4 Initial Goals of ROLL Working Group

The ROLL (Routing over Low-power and lossy networks) Working Group [62] was created to specify a comprehensive routing protocol that could route data efficiently over LLNs. ROLL primarily focused on the determining the routing requirements for the following scenarios while creating RPL: industrial, connected home or building and urban sensor networks. The designed routing protocol must fit the various requirements introduced by the working group's target applications specified in [4], [55], [44] and [15]. This group believed that technology was surely transitioning to IPv6, and hence aimed to provide an IPv6 only routing architectural framework for these application scenarios. According to [25], the unofficial goal of this working group was to prevent fragmentation in the WSN market by providing an IP-based routing standard and solicit broad industrial support behind that standard.

Some of the main features that this group took into consideration were providing high reliability in the presence of time varying loss characteristics and connectivity while permitting low-power operation with very modest memory and CPU pressure in networks potentially comprising a very large number (several thousands) of nodes. The group also explored aspects of mobility within a single LLN (if any) in the routing requirement creation. Routing security and manageability (e.g., Self Configuration) was an important consideration, as were transport characteristics that the control messages will face. The main objective of this protocol is to target networks which comprise up to thousands of routers, where the majority of the routers have very constrained resources, where the network to a large degree is managed by a (single or few) central super routers, and where handling mobility is not an explicit design criteria. Supported traffic patterns include multipoint-to-point, point-to-multipoint and point-to-point traffic. The emphasis among these traffic patterns is to optimize for multipoint-to-point traffic, to reasonably support point-to-multipoint traffic and to provide basic features for point-to-point traffic, in that order [25][77].

# Chapter 3

## Understanding RPL

### 3.1 Introduction

#### 3.1.1 What is RPL

RPL is a distance-vector and a source routing protocol that is designed to operate on top of several link layer mechanisms including IEEE 802.15.4 PHY and MAC layers [77][20]. These link layers could be constrained, potentially lossy, or typically utilized in conjunction with highly constrained host or router devices, such as but not limited to, low-power wireless or PLC (Power Line Communication) technologies [p10]. RPL mainly targets collection-based networks, where nodes periodically send measurements to a collection point. A key feature of RPL is that it represents a specific routing solution for low power and lossy networks. The protocol was designed to be highly adaptive to network conditions and to provide alternate routes, whenever default routes are inaccessible. RPL provides a mechanism to disseminate information over the dynamically formed network topology. This mechanism uses Trickle [43] to optimize the dissemination of control messages [p9].

#### 3.1.2 About this Chapter

This chapter includes direct references from the standards document [77] released by the ROLL WG. It does not include personal opinions or viewpoints, and is simply intended to be a short summarization of the key features of RPL. It also highlights certain features that draw special attention in the rest of the thesis. Since many lines are directly drawn from the RFC, a special form of referencing is used for referring to a particular line from RFC6550. In particular, the letter 'p' followed by a number implies that the particular line (or a paraphrasing of the line) or set of lines immediately preceding the reference can be found in RFC6550 in the page indicated by the number.

## 3.2 RPL Topology

### 3.2.1 DODAG

RPL organizes its topology into DODAGs or destination-oriented directed acyclic graphs. A DODAG is a DAG rooted at a single destination. The DODAG root has no outgoing edges [p10]. A DODAG is uniquely identified by a combination of RPL Instance ID and DODAG ID. Each DODAG has a DODAG root, which is the DAG root of the DODAG.

### 3.2.2 Rank

A nodes Rank defines the nodes individual position relative to other nodes with respect to a DODAG root. Rank strictly increases in the *Down*<sup>1</sup> direction and strictly decreases in the *Up*<sup>2</sup> direction. The exact way Rank is computed depends on the DAGs Objective Function (OF). The Rank may analogously track a simple topological distance, may be calculated as a function of link metrics, and may consider other properties such as constraints [p11].

### 3.2.3 DODAG Root

The DODAG root is the DAG root of the DODAG. The DODAG root may act as a border router for the DODAG, and aggregate routes in the DODAG and may redistribute DODAG routes into other routing protocols [p11]. The DODAG root is responsible for configuring a number of parameters, which are advertised as options and carried in DIO messages. Examples of such options include:

- Trickle Timer Options (DIOIntervalDoublings, DIOIntervalMin, DIORedundancyConstant)
- Path control size
- MinHopRankIncrease
- DODAGPreference Field

The DODAG root also plays an important role in multicast. It acts as an automatic proxy Rendezvous Point for the RPL network and as a source towards the non-RPL domain for all multicast flows started in the RPL domain.

### 3.2.4 RPL Instances

A RPL instance is a set of one or more DODAGs that share a RPLInstanceID. Each RPL instance operates independently of other RPL instances, and implements a different objective

---

<sup>1</sup>‘Down’ direction implies away from the root.

<sup>2</sup>‘Up’ direction implies towards the root.

function. A network may run multiple instances of RPL simultaneously, when the requirement states that different and possibly countering constraints are to be used for within the same LLN. However, [77] only defines operation of a single instance. A single RPL instance may have multiple DODAGs. A node can only be part of a single DODAG per RPL instance. A single node may be part of multiple RPL instances.

### 3.3 RPL Control Messages

There are four main types of control messages supported by RPL:

- **DIO: DODAG Information Object** This message carries information that allows a node to discover a RPL Instance, learn its configuration parameters, select a DODAG parent set, and maintain the DODAG. This is similar to IPv6 Router Advertisements [45].
- **DIS: DODAG Information Solicitation** These are similar to IPv6 Router Solicitations [45] and are used to solicit DIO from a RPL node.
- **DAO: Destination Advertisement Object** This is used to propagate destination information upward along the DODAG. In storing mode, DAO is unicast to selected parents. In non-storing mode, it is unicast to the DODAG root.
- **DAO-ACK: Destination Advertisement Object Acknowledgement** The DAO-ACK message is sent as a unicast packet by a DAO recipient (a DAO parent or DODAG root) in response to a unicast DAO message.
- **CC: Consistency Check** The CC message is used to check secure message counters and issue challenge-responses. A CC message must be sent as a secured RPL message.

### 3.4 RPL Operation

#### 3.4.1 Supported Traffic Patterns

RPL was designed mainly for optimizing MP2P type of traffic flow which is prevalent in collection-based networks. According to the standards specification, *RPL routes are optimized for traffic to or from one or more roots that act as sinks for the topology* [p13]. It can also be used for P2P and P2MP, but these are less optimized. To implement just the MP2P traffic flow, RPL requires only DIO and DIS control messages. For P2P and P2MP, RPL requires the use of DAO and optionally DAO-ACK messages as well.

#### 3.4.2 Modes of Operation

RPL supports two modes of operation for supporting P2MP and P2P communication: Storing Mode (fully stateful, MOP 0) and non-storing mode (fully source routed, MOP 1). Both

these modes require the usage of DAO messages and the optional usage of DAO-ACK messages. In non-storing mode, a node sends data all the way up to the DODAG root by recursively passing on the messages to DIO parents. At the DODAG root, the packet is source-routed to the required destination. In storing mode, the packet is passed up to the DIO parents until it reaches an ancestor through which the destination prefix is reachable. These two modes of operation are incompatible.

### 3.4.3 Upward Routes

In RPL, upward routes are constructed from each node to the DODAG root to provide MP2P communication. The MP2P communication pattern is a significant feature of collection-based networks, where a number of sensors send their data to a common collection point (which in this case is a DODAG root). This in turn can act as a border router and transmit this data to a time-series data store or other publishing space.

In RPL, these upward routes are by default constructed using a node's preferred DIO parent. Each node has a set of one-hop neighbors, called the candidate neighbor set. Out of these, the node selects a group of nodes which have a rank strictly less than the node's current rank within this particular RPL instance. This set is called the candidate parent set. Out of these, the node chooses one (or possibly more, but by default 1) parent through which it sends packets to the DODAG root. When the node has some data that needs to be sent to the root, it immediately sends this to the preferred parent. The parent node sends it to its own parent and so on until it reaches the DODAG root.

In case the node's immediate parent is not able to participate in routing the data to the DODAG root, the node can select another alternative parent from its candidate parent set, and route through this new parent instead.

### 3.4.4 Downward Routes

In RPL, downward routes are only required for implementing P2MP or P2P communication. It is an optional feature that is supported using different modes of operation. These are accomplished by using DAOs and DAO-ACKs. P2P routes are by default incorporated by having the sensor node transmit the data packet via its preferred parent all the way up to the DODAG root. Once the root receives the message it transmits it to the destination either by appending the source route to the data packet or by simple hop-by-hop routing down the DODAG. This depends on whether the mode of operation (MOP) is set to be storing mode or non-storing mode.



### 3.4.5 Loop Detection and Avoidance

RPL includes a reactive loop detection technique that protects from meltdown and triggers repair of broken paths. The DODAG is inconsistent if the direction of a packet does not match the Rank relationship. A receiver detects an inconsistency if it receives a packet with either the O bit set (to Down) from a node of a higher Rank or the O bit cleared (for Up) from a node of a lower Rank.

## 3.5 Objective Function

In order to be useful in a wide range of LLN application domains, RPL separates packet processing and forwarding from the routing optimization objective. Examples of such objectives include minimizing energy, minimizing latency, or satisfying constraints [p8]. The OF is identified by an Objective Code Point (OCP) within the DIO Configuration option [p17]. The Objective Function (OF) defines how RPL nodes select and optimize routes within a RPL instance. The OF along with a set of metrics are used for the following purposes:

1. The selection of DODAG to join
2. The rank of each node within the DODAG
3. the number of peers in that DODAG as parents and computation of an ordered list of parents
4. The outcome of the process used by a RPL node to select and optimize routes within a RPL instance based on the Information Objects available

As of the date when this document was written, the ROLL WG has defined two objective functions:

- OF0: Objective Function Zero [50]  
Here the routing metric adopted is hop count. OF0 is designed to be the common OF that will allow interoperation between the different implementations of RPL[20].
- MRHOF: Minimum Rank with Hysteresis Objective Function [21]  
MRHOF selects routes that minimize a metric, meanwhile using hysteresis to reduce churn in response to small metric changes. MRHOF works with metrics that are additive along a route. That is the traditional ETX (Expected Transmission Count) metric. ETX of a wireless link is the estimated average number of transmissions of data frames and ACK frames necessary for the successful transmission of a packet [78].

The separation of OFs from the core protocol specification is intended to allow RPL to be adapted to meet the different optimization criteria required by the wide range of deployments, applications, and network designs [50].

## 3.6 Trickle Timer

RPL uses the Trickle timer [43][42] to reduce control message overhead by transmitting updates only when inconsistencies are detected in the network. If a node hears DIO updates from its neighbors that are consistent with its own understanding of the network topology, then a redundancy counter is incremented. If the number of consistent updates heard within a particular time interval exceeds the redundancy count, then the node does not transmit any updates and the listening period is doubled. However, if an inconsistent update is heard, then the timer is reset and an update is rapidly propagated. To save energy, the Trickle timer sends out fewer control messages as the network becomes more stable. However, when an inconsistency is detected, the timer gets reset and starts sending DIO messages more frequently in order to quickly propagate updates through the rest of the network. The Trickle timer has three configuration parameters:

1. The minimum interval size,  $I_{\min}$ , is defined in units of time (e.g., milliseconds, seconds).
2. The maximum interval size,  $I_{\max}$ , is described as a number of doublings of the minimum interval size (the base-2  $\log(\max/\min)$ ).
3. The redundancy constant is a natural number (an integer greater than zero).

In addition to these three parameters, Trickle maintains three variables:

1.  $I$ , the current interval size
2.  $t$ , a time within the current interval, and
3.  $c$ , a counter.

RPL defines default values for Trickle parameters but allows reconfiguration without affecting interoperability.

## 3.7 Security Features

A bit of the RPL message code identifies whether or not a RPL message is secure. RPL includes secure versions of the basic control messages DIO, DIS, DAO and DAO-ACK. In addition, it also includes several messages that are relevant only in networks that are security enabled.

Given the resource constraints in LLNs as well as the size and complexity of RPL, security features defined in the RFC are optional and need not be implemented if unnecessary. RPL has three basic security modes:

- Unsecured Security Mode

The basic versions of control messages are used and security may be implemented using other mechanisms such as link layer security or application layer security.

- **Pre-installed Security Mode**  
In this security mode, RPL uses secure messages. Hosts and routers require a pre-installed key to join RPL instances. These keys provide message confidentiality, integrity, and authenticity.
- **Authenticated Security Mode**  
In this security mode, RPL uses secure messages. Pre-installed keys are used to join a network as a leaf to provide message confidentiality, integrity, and authenticity. To join the network as a router, a second key must be obtained from a key authority.

Since RPL does not support asymmetric algorithms, authenticated security mode cannot be implemented based on current advancements.

### 3.8 Features of RPL Necessary for Interoperability

The list of interoperability criteria specified in the RFC is not exhaustive. The ROLL WG has instead provided some guidelines that may help with interoperability, but have not clearly defined the necessary and optional feature set that is required for interoperability between different implementations. The following is a general list of criteria specified in the RFC:

1. According to the specification, the greatest level of interoperability may be achieved when all of the nodes in a RPL LLN are cooperating to use the same MOP, OF, metrics, and constraints [p109].
2. All RPL implementations need to support the use of RPL Packet Information transported within data packets [p109].
3. RPL implementations will need to support the use of Neighbor Unreachability Detection (NUD), or an equivalent mechanism, to maintain the reachability of neighboring RPL nodes [p109] . The exact mechanism can be implementation specific, but it will lead to more limited functionality.

# Chapter 4

## Analysis of RPL

### 4.1 Introduction

A number of studies have focused on evaluating RPL with regards to various performance metrics such as energy efficiency, network convergence time, protocol control-traffic overhead, path length and packet delay. In this chapter, a few of them are detailed. These works include studies performed on networks that range from 20 nodes to 1000 nodes. All these studies are performed on simulators such as Contiki/Cooja [49], Network Simulator (NS2) [65][70], OMNET++ with Castalia [74][3] and WSNNet [19].

This chapter also discusses the Collection Tree Protocol [22] and LOADng [7], which were proposed as a solution for routing in WSNs and mobile sensor networks alongside RPL. These protocols are evaluated against RPL with respect to a number of metrics such as Packet Reception Ratio (PRR), churn, control-traffic overhead, type of traffic patterns catered to, size of protocol control messages, delay and energy efficiency. A more generic study on the comparison between proactive and reactive protocols for WSNs is discussed in Section 4.3. Of the five comparative studies discussed, four of them have been evaluated on simulators such as Contiki/Cooja and NS2 while one of them [37] has been implemented on a medium size test-bed comprising 51 TelosB motes.

### 4.2 Performance Evaluation Studies

[24] studied the behavior of RPL when deployed in networks that have a predominance of bidirectional traffic. It was found that the maximum and average ranks of participating routers grows logarithmically with the number of routers. Here, rank basically represents the distance of the router from the DODAG root in terms of number of hops. The maximum rank would represent the diameter of the network. The convergence time of the network, i.e, the time that is needed for all the routers that are in the same connected component as the controller (DODAG root) to join the DODAG, also grew logarithmically with the

number of routers in the network. When the density of the network was kept constant, the average number of parents per node grew logarithmically with number of routers. This intuitively seems correct since RPL forms a DODAG structure, which is similar to a tree. The rank would correspond to the depth of the node in the tree, which holds a logarithmic relationship with the number of nodes in the tree. The control traffic overhead was found to grow polynomially with the number of routers in the network. However all of these are simulation studies performed on a Java implementation of RPL on NS2. Hence how much they correlate to the actual hardware implementation is up for speculation.

In [25], the authors analyze the RPL routing protocol and propose a few broadcast mechanisms. The performance evaluation studies are again performed on a Java implementation of RPL on the NS2 simulator. This was a basic version of RPL which had only upward routes and a single RPL instance. There were no global repairs during the simulation hence the DODAG sequence number remained the same. Performance of this implementation was evaluated using two variations, one in which the DIO was transmitted periodically and the other in which the DIO was transmitted according to a Trickle timer. The simulation study comprised a network of 1000 routers. It was found that the maximum and average rank of a node in the network, the average number of parents per node, the convergence time as well as the average path length in terms of number of hops grew logarithmically with the number of routers in the network. Control traffic however, was found to increase linearly with the number of nodes in the network. The variation using a Trickle timer showed much less control traffic and DIO collision ratio.

[1] evaluates RPL's performance using the Contiki Cooja simulator [49]. The network comprised 20 to 100 nodes, and rank is calculated according to the ETX [21] metric. It was found that the sink starts receiving data almost as soon as the network is deployed indicating very low setup time. In steady state, the routing overhead constituted about 25% of the overall traffic in a network with 20 nodes, and this shot up to 75% in a network with 100 nodes. The optional DAO messages, which are used to handle downward traffic, were found to be the dominant cause of control message overhead, as compared to DIO and DIS. They also found that packet delay rate was dependent on distance of the node from the sink and was not very sensitive to Packet Error Rate (PER). The total delay was quite small and less than two seconds for the network with 20 nodes and less than eight seconds for the network with 100 nodes. Hence they concluded that RPL's quick setup time allows it to be deployed in critical scenarios such as rescue or military operations. The network size plays an important factor in responsiveness, as the delay increases with number of nodes. However, it requires further optimization to handle the high protocol overhead.

[72] evaluates RPL's performance using real link data gathered from networks deployed on the fields, in order to make the simulation studies more realistic. This data was used to compute the PDR (Packet Delay Rate) for each link in the simulated network used for testing. RPL was simulated using OMNET++ [74] which is a well known discrete event

based simulator written in C++ and NED [46]. Castalia-2.2 [3] was used as the WSN Simulator framework within OMNET++. Outputs and events were visualized using Network Animator(NAM), which is distributed with Network Simulator (NS). The radio was simulated as TelosB CC2420 radio and 802.15.4 MAC protocol was used. Storing MOP RPL is implemented. Packets were sent by each node according to a Constant Bit Rate (CBR). To depict a more realistic scenario, majority of the traffic (20%) was sent to the DODAG root and the remaining 80% was distributed among the other nodes in an 86 node topology. The results of the simulation showed that most paths generated by RPL had a hop length greater than that produced by a hypothetical ideal routing algorithm. However the authors conclude that this is not drastically worse than the ideal case. While using the ETX metric, RPL showed results very similar to the ideal scenario. When comparing the traffic generated by control and data packets, it was found that nodes closer to the sink have a higher degree of data packets since they participate in the routing. Leaf nodes were found to have a comparable number of control and data packets. This study implemented global repairs through the periodic emission of a new DAG Sequence Number by the DAG root. It was found that as the period to emit a new DAG sequence increases, the amount of control traffic decreases because the trickle interval gets larger for each node. However the smaller amount of control traffic comes at the price of increased time for loss of connectivity between the period when a node loses connectivity to when a global repair is triggered.

In [56], the performance of RPL has been investigated in terms of both the standardized objective functions, OF0 [50] and MRHOF [21]. The network was simulated on Contiki/Cooja [49] and had between 20 and 45 nodes per simulation. For both OF0 and MRHOF, it was found that PDR increased as Packet Reception ratio (RX) increased. Beyond a particular threshold for RX (in this case 60%), PDR was close to 100%. Hence it was concluded that maintaining RX value of 60% was good enough for high performance of RPL with regards to PDR in light-density networks. It was also found that power consumption decreased linearly with increase in RX. At  $RX \geq 60\%$ , the average power consumption was fair, again supporting the claim that maintaining 60% RX was sufficient for RPL. Between OF0 and MRHOF, MRHOF showed slightly better per node power consumption than OF0. This was attributed to MRHOF choosing better routes by taking into account the energy saving, which consequently provides better network lifetime.

[29] analyzes the impact of different routing metrics on the stability and efficiency of RPL. It states that a good metric must reflect the radio link quality of the whole path between the node and the border router, and must not just be a local optimization. The energy efficiency of the path must also be taken into account in order to reduce the total energy consumption. There must be a healthy trade-off between stability and reaction towards changes. RPL was implemented on the Contiki OS, and the WSN simulator [19] was used. The authors propose an OF based on minimum rank with hysteresis using the ETX metric *MIN-HOP*, *ETX* and *LQI*(*Link Quality Indicator*) metrics. PDR was the first evaluation metric, which indicated RPL reliability. It was found that PDR decreased with

increasing distance of the node from the DODAG root. Overall, the PDR was quite low, with LQI having the highest median value. It was concluded that further efforts must be made to improve RPL's reliability. When evaluating end-to-end delay, it was found that MinHop showed the least delay as most packets are dropped, especially those from nodes far from the border router. This was followed by ETX and then LQI. It was also found that energy consumption was more evenly distributed among the nodes in MinHop and ETX, than LQI. This affects the network lifetime. This is because, in MinHop, even nodes that are 200-300 metres away from the root can be one hop away, and thereby get chosen as the preferred parent. In LQI, its most often the nodes closest to the root that are chosen as the preferred parent, leading to a non-negligible portion of the nodes consuming most of the resources. The authors also evaluate the stability of the DODAG by measuring route prevalence, which is the ratio between the number of times the principal route was used and the number of times all routes have been used. It was found that the overall route stability was quite poor. *MinHop* showed the highest stability of 45%, while ETX and LQI performed much poorer. The authors also tested the DODAG stability by measuring the number of times a node changed its preferred parent in a network of 500 nodes. It was found that all three metrics show frequent changes to preferred parents, with ETX having the maximum number of changes. The authors concluded that each metric comes with its own set of trade-offs. *MinHop* metric exhibits the lowest instability, but performs very poorly because it tends to use bad radio links. *LQI* limits the instability but offers larger end-to-end delays. *ETX* balances the load more efficiently among the different nodes but comes at the cost of increased DODAG reconfigurations.

## 4.3 Comparative Studies

### 4.3.1 RPL and CTP

In [69], RPL is compared to the commonly used data collection tree protocol, Collection Tree Protocol (CTP) [22]. CTP is a distance-vector routing algorithm that was developed as a solution to routing in WSNs. It stands as a predecessor to RPL and was considered the de-facto routing standard for TinyOS. It builds a tree-based topology with the root at the sink of the network, similar to RPL's DODAG structure. An adaptive beaconing mechanism is used to broadcast routing control messages. While CTP relied on a specific link-layer technology for topology formation, RPL uses the OF which is more generic and customizable for specific applications. CTP was earlier known for its efficient energy consumption and high Packet Reception Ratio (PRR). Both these protocols were implemented and evaluated on the Contiki/Cooja simulator on a network of 9-49 nodes. A data collection model was tested where the predominant traffic flow pattern was MP2P, as is the case in collection-based networks. It was found that in smaller networks (in this case, a network containing seven nodes), CTP showed a better PRR, but as the network size increased to 49 nodes, RPL showed a better PRR as well as less energy consumption. Upon increasing the data traffic,

CTP's PRR decreased further. The churn, or number of times parents were switched was also significantly less in RPL than CTP. Hence the authors concluded that RPL improved Packet Reception Ratio (PRR) and maintained low levels of energy consumption. Due to the strict rules of DODAG formation and the cooperation between the different control messages (DIO, DIS and DAO), RPL was able to perform better when the network was scaled in terms of network size and growing data traffic.

[37] uses the BLIP (Berkeley Low-power IP Stack) and TinyRPL implementations in TinyOS 2.x to evaluate RPL against CTP on a medium-size testbed of 51 TelosB motes distributed over two floors in an office building. This implementation of RPL used the path-ETX metric and MRHOF, and disabled downward routes in order to make a fair comparison with CTP. Upon testing the PRR, it was found that RPL and CTP performed very competitively and both achieved a PRR that was higher than 99.8%. Contrary to results obtained in [69], RPL was found to have a higher churn than CTP. It was also found to have a greater volume of control messages and 15 bytes of overhead in each data packet to achieve the same functionalities of the CTP routing header, though this difference was not considered significant. RPL also had a slightly higher per-hop ETX value and selected marginally longer routes than CTP. However RPL offers a major benefit in terms of catering to a variety of traffic patterns such as P2P and P2MP, which CTP is incapable of doing. It also has the ability to directly connect to Internet nodes by exchanging packets with global IPv6 addresses.

An important conclusion that can be drawn from these two studies is that simulation studies can often yield results that are contrary to real testbed implementations. Hence evaluating RPL using actual hardware is imperative to draw conclusive results.

### 4.3.2 RPL and LOADng

The Lightweight on-demand ad hoc distance-vector routing protocol-next generation or LOADng is a lightweight variation of AODV for LLNs. It is designed based on the idea that LLNs are idle most of the time. Hence instead of adopting a proactive approach would generate unnecessary overhead, LOADng follows a reactive approach in which routes are established towards destinations only when there is some data to send [7].

In [79], the authors perform a detailed comparison of RPL and LOADng, exposing the advantages and disadvantages of both. The proactive RPL was specifically designed for sensor networks where the predominant traffic flow is multi-point to point (MP2P). It provides very fast data collection when there are no loops from the node to the DAG root. However the paper also identifies several disadvantages of RPL. Firstly, since it has been predominantly designed for MP2P traffic flow, it is not optimized for other traffic patterns such as root to sensor (P2MP) or sensor to sensor (P2P). If these traffic patterns have to be included, DAO control messages must be used and this seems to have been included in RFC6550 more as



an afterthought than as an actual design consideration, since the emission interval hasn't been specified nor are the routes optimized. Secondly, [77] places special importance on the existence of a *root* node, which is different from all the other hosts and routers in the network. This node is the central relay for traffic through and between other RPL routers in the networks. This node is also capable of connecting to the outside internet by acting as the border, or it can work as a virtual DODAG root and connect this DODAG with other DODAGs within or outside the current RPL instance. The root is responsible for initiation, configuration and management of the DODAG. However, this root also poses a single point of failure and a very probable congestion point. Additionally, when P2P communication is being carried out, using a root causes route stretch. Thirdly, RPL control messages are quite huge and can be subjected to fragmentation while being routed with a reasonably high probability. Fragmentation of control messages is highly undesirable for LLNs considering they are already lossy. When compared to LOADng which is a reactive routing protocol, LOADng was found to be suitable for a more general traffic pattern. It does not have any node that performs special functions like the root and is hence not subjected to the subsequent problems that arise due to such a consideration. Also, due to its compressed and flexible data format, there is no possibility of fragmentation. It does not impose any strict source routing rules, hence it can accommodate applications which require a fixed MTU. However, LOADng might have a higher delay in the route discovery phase and might have higher control traffic overhead if the traffic flows are predominantly P2P.

[75] also performs a comparative study between LOADng and RPL. However, this study is conducted specifically in the home automation scenario, which poses the special requirement of low latency. This paper identifies a number of issues with LOADng. Primarily there is a route discovery delay since LOADng is a reactive protocol and discovers routes only when there are data packets to send. This was also verified in [79]. Secondly during the discovery process, outgoing packets are buffered which may lead to losses in memory constrained nodes. Also, since LOADng follows a flooding mechanism to discover routes to the destination, the network suffers from energy depletion since flooding is a highly energy inefficient process. Also, during flooding, there maybe collision of control messages leading to unnecessary retransmissions.

The home automation scenario in this paper is studied using a centralized architecture. RPL and LOADng are both implemented on Contiki OS. When comparing the end to end delay in a one-hop scenario, between 26% and 47% of packets sent through LOADng experienced a delay of over 0.5 seconds, which is the maximum latency allowed in home automation scenarios as specified by [4]. The range in percentage arises due to variable Route Hold Time (RHT), which was varied between ten minutes and one hour. RHT indicates the time for which a route remains valid. However, RPL being a proactive protocol has an upper bound of 10% of packets falling outside the 0.5 second latency range. Increasing the number of hops increased the probability of a packet falling outside the allowable latency range, but the increase was felt more sharply in LOADng than RPL. The number of hops was varied

till four, which is again the common maximum as specified in [4]. With regards to number of hops between source and destination, RPL routes were always restricted within four hops whereas LOADng routes sometimes went way over four hops. This is because, when the RHT is small, a packet that reaches a router need not always follow the same route. It could change depending on the results of the latest flooding.

While considering size of routing tables, both LOADng and RPL have disadvantages. In case of LOADng, the size of the routing table increases when RHT is larger, since there are a large number of redundant entries as a result of frequent flooding. In RPL, the size of the routing table varies based on the position of the node in storing mode. A node placed closer to the DODAG root could potentially have a large number of entries causing the device to run out of memory. Also, when the number of nodes is scaled up, RPL performs very poorly due to the limited number of neighbor entries that nodes can store. Hence more routes were needed in the routing table. Finally the control traffic overhead was studied. In case of RPL, there was high volume of control messages during the setup phase but due to Trickle, it reduced and reached a steady state. This also depends greatly on the DAO interval used, which is implementation specific. However, the amount of control traffic in LOADng was inversely proportional to RHT. Hence in case of a steady network, RPL probably performs better but LOADng can be made to perform efficiently under any scenario by varying the RHT. This paper however concludes that RPL is better suited for home automation environments than LOADng.

In [24], the authors compare RPL and LOAD with special focus on how the two protocols perform in environments with bidirectional links. Bi-directional links are an integral part of a number of sensor network applications. While RPL is optimized for MP2P traffic, LOAD is designed for a more general traffic pattern. The two protocols were tested on an NS2 simulator and are compared on the basis of data traffic delivery ratio, control traffic overhead, number of collisions, network convergence time, etc. Since the studies were performed on the simulator, this paper does not perform energy consumption analysis as the amount of energy consumed is directly related to the underlying hardware. The Java implementation of RPL chooses a DAOInterval of 15s and the Java implementation of LOAD chooses a route lifetime of five seconds. It was found that the control traffic overhead of RPL was almost twice as much as LOAD. This was attributed to the periodic DAOs that get sent out. However, [79] argues that this was a result of a poor implementation choice, namely, the DAO transmission interval being 15s. DAO transmission time is definitely a major cause of control traffic overhead in RPL. However, how much of an impact choosing this particular value had over other values is left for future work. A point to note is that the control traffic overhead is directly dependant on data traffic for a reactive protocol like LOAD. For a reactive protocol like RPL, control traffic is independent of data traffic. When comparing path routes, it was again found that LOAD routes were longer than RPL routes. However both were quite close to the routes found by GodRP, or the “God” Routing Protocol, which is a hypothetical protocol that always finds the best routes. It was also noted that since

RPL uses source routing in non-storing mode, the destination can be no more than 8 hops from the source without IPv6 compression, and 64 hops when addresses are compressed [31]. Hence RPL is unsuited for applications with long chain-like topologies. In both cases, the packet delivery rate was unrealistically found to be 100%, which can be attributed to the fact that these studies were performed on simulators and not the real hardware. The delay was higher in LOAD since a data packet is buffered at the source during route discovery. In RPL, routes are formed well ahead and hence delay is much lesser. The number of collisions were also higher in LOAD due to the flooding protocol used, which leads to broadcast storm.

### 4.3.3 Proactive vs reactive Routing Protocols for WSNs

In [58], the authors perform a comparative study of proactive and reactive protocols for WSNs. In particular, they compare the performance of proactive protocols RPL and CTP with reactive protocols AODV [53] and DSR [32] in an emergency monitoring and evacuation scenario, while using CoAP/UDP as the application/transport layer protocol. AODV is the Ad hoc On-Demand Distance Vector Routing Protocol which enables dynamic, self-starting, multihop routing between participating mobile nodes wishing to establish and maintain an ad hoc network. DSR, or the Dynamic Source Routing Protocol is designed specifically for use in multi-hop wireless and allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. Implementations of all these protocols were tested on the SpeckSim simulator, and compared based on the routes generated for humans (nodes) to escape from a fire in a safe manner. The WSN comprised of 24 nodes within a building. The results of the simulation showed that RPL had the lowest latency, since the DODAG root node was strategically positioned close to the exit. RPL also had the least control message overhead, since it is a proactive protocol and uses a Trickle timer for reducing the number of control messages disseminated. However, it must be noted that RPL was only used for MP2P communication in this scenario. DAO messages, which contribute to a majority of protocol overhead, were not involved in this specific application. RPL also showed slightly better per-node power-consumption statistics as compared to all other protocols. RPL however had the least packet delivery ratio among all four protocols, which shows the need to incorporate more reliability into the framework. RPL's response to failures were not evaluated because the standards document does not currently specify any fault tolerance mechanisms. Node failures are neither detected nor propagated through the network due to increased overhead.

Hence out of the five evaluated metrics (PDR, Latency, Overhead, Power Consumption and Fault Tolerance), RPL outperforms the other protocols in three (Latency, Overhead and Power Consumption). However it fares poorly when considering delivery ratio and fault tolerance. The authors claim that in emergency scenarios, the three main metrics to consider are reliability, fault tolerance and latency. Due to RPL performing poorly in two out of these three, the authors recommend AODV as it responds well to failures and has a high PDR.

## 4.4 Summary

This chapter analyzes various performance evaluation studies of RPL and notes the various conclusions drawn. A summary of the results is as follows:

1. According to [24] and [25], the maximum and average rank of a node in the network and the average number of parents per node grows logarithmically with the number of nodes in the network.
2. [24] and [25] also found that control traffic grew linearly with network size. [1] and [72] concluded that DAO traffic was more predominant than DIO or DIS, with DIS being almost negligible. [72] also found that nodes closer to the sink have a higher degree of data traffic than the leaves, since they participate in the routing. The leaves were found to have a higher degree of control traffic.
3. [25] found that the path length also had a logarithmic relationship with network size, while [72] found that the paths produced by RPL were slightly longer than the paths that would be produced by a hypothetical ideal routing protocol.
4. [1] and [29] found that the average packet delay rate (PDR) was inversely proportional to the distance of the node from the sink, or the network size. This led to the conclusion that RPL yielded a longer response time in larger networks, and would require further optimization in order to be used for time-critical applications. [56] found that PDR was also proportional to packet reception ratio.
5. With regards to power consumption, [56] and [29] found that using MRHOF as the objective function shows slightly better per-node power consumption.
6. [29] found that the general stability of RPL was quite poor with a large number of DODAG reconfigurations.

This chapter also discusses comparative studies performed between RPL and similar routing protocols. A summary of the comparative studies is given below:

1. RPL was first compared to CTP in [69] and [37], which was the de-facto routing standard for WSNs before RPL was designed. RPL and CTP perform competitively, with RPL showing a few additional features as benefits such as being link-layer independent and being able to cater to a variety of traffic patterns.
2. Next, RPL was compared with LOADng [79][75][24], which is also a protocol designed for similar scenarios. RPL was found to have larger control messages which lead to increased risk of L2 fragmentation. It also does not cater well to generalized traffic patterns and places special importance on the root node, which could potentially be

---

<sup>1</sup>N is the number of nodes in the network

Citation	Main Results	Size of Study	Objective Function	Simulator	Comments
[24]	$\text{Max Rank} \propto \log(N)^1$ $\text{Avg Rank} \propto \log(N)$ $\text{Convergence Time} \propto \log(N)$ $\text{Control Traffic Overhead} \propto N^x$	63 to 1000	Number of hops	Java implementation of RPL on NS2	Traffic was predominantly bidirectional
[25]	$\text{Max Rank} \propto \log(N)$ $\text{Avg Rank} \propto \log(N)$ $\text{Convergence Time} \propto \log(N)$ $\text{Avg Num of Parents} \propto \log(N)$ $\text{Control Traffic Overhead} \propto N$	1000	Number of hops	Java implementation of RPL on NS2	Broadcast mechanisms proposed
[1]	$\text{Delay} \propto N$ $\text{PDR} \propto \text{Dist from sink}$ Very low setup time Large Routing Overhead mainly due to DAO	20 to 100	ETX	Contiki/Cooja	-
[72]	RPL generates longer paths Nodes closer to sink have high data traffic $\text{Number of global repairs} \propto \text{Control Traffic}$	86	ETX and Number of hops	OMNET++ with Castalia	Evaluates RPL based on real link data
[56]	$\text{PDR} \propto \text{PRR}$ until $\text{PRR} = 60\%$ $\text{Power consumption} \propto \frac{1}{\text{PRR}}$ MRHOF has better power consumption than OF0 MRHOF chooses better routes than OF0	20 to 45	ETX and Number of hops	Contiki/Cooja	Compares OFs in light density networks
[29]	$\text{PDR} \propto N$ Min hop showed least delay Better energy distribution in min-hop and MRHOF than LQI Poor overall route stability	500	Min hop, ETX and LQI	Contiki/WSNet	Efficiency of metrics on route stability and efficiency

Table 4.1: Performance Evaluation Studies of RPL

a single point of failure. However RPL was found to outperform LOADng in terms of path length and volume of control traffic.

3. Finally, [58] analyzed proactive routing protocols versus reactive routing protocols

in the WSN space. RPL showed a few advantages such as low latency and energy consumption. However when used in emergency scenarios, AODV showed a better performance in terms of key metrics such as response to failure and PDR.

All the studies reviewed in this section were performed on simulations (except [37]) and were not tested on actual hardware. This strongly indicates that RPL is too complex to test on real hardware.

Citation	Compared Against	Main Results	Size of Study	Simulator Used
[69]	CTP	<ul style="list-style-type: none"> <li>◦ In smaller networks, CTP showed better PRR. In larger networks, RPL showed better PRR and Energy consumption</li> <li>◦ CTP's <math>PRR \propto \frac{1}{DataTraffic}</math></li> <li>◦ RPL showed lesser churn</li> </ul>	7 to 49	Contiki/Cooja
[37]	CTP	<ul style="list-style-type: none"> <li>◦ Both RPL and CTP showed high PRR</li> <li>◦ RPL had higher churn</li> <li>◦ RPL had higher control-traffic overhead</li> <li>◦ RPL able to cater to variety of traffic patterns, CTP is only collection-based</li> <li>◦ Unlike CTP, RPL is link-layer independent</li> </ul>	51	No, real test-bed evaluation
[79]	LOADng	<ul style="list-style-type: none"> <li>◦ LOADng caters to more general traffic pattern</li> <li>◦ LOADng has flexible and compressible packet format unlike RPL</li> <li>◦ No single point of failure in LOADng like root node in RPL</li> <li>◦ Longer route discovery phase in LOADng</li> <li>◦ More control traffic in LOADng if traffic is predominantly P2P</li> </ul>	63 to 500	Performed on simulator, but details not mentioned
[75]	LOADng	<ul style="list-style-type: none"> <li>◦ Higher route discovery delay in LOADng</li> <li>◦ Flooding in LOADng is energy inefficient</li> <li>◦ Higher proportion of packets show less delay in RPL</li> <li>◦ LOADng generated longer paths</li> <li>◦ Both protocols had large routing table size</li> <li>◦ RPL performs poorly when scaled</li> </ul>	25 to 40	Contiki/Cooja

[24]	LOAD	<ul style="list-style-type: none"> <li>◦ Twice the amount of control traffic in RPL due to DAOs</li> <li>◦ In LOAD, control traffic <math>\propto</math> data traffic</li> <li>◦ LOAD routes longer than RPL routes</li> <li>◦ RPL had cap in route length in storing mode. Unsited for long chain-like topologies</li> <li>◦ Higher delay in LOAD due to buffering during route-discovery</li> <li>◦ More collisions in LOAD due to flooding</li> </ul>	63 to 1000	NS2
[58]	Proactive (RTP,CTP) vs Reactive (AODV, DSR) Protocols	<ul style="list-style-type: none"> <li>◦ RPL had lowest latency by strategic positioning of root node</li> <li>◦ RPL has lower control traffic overhead due to Trickle</li> <li>◦ RPL had better per-node power consumption than all others</li> <li>◦ RPL had least PDR among all four</li> </ul>	24	SpeckSim

Table 4.2: Comparative Studies

# Chapter 5

## RPL Implementations

### 5.1 Open Source Implementations of RPL

There are a number of open-source implementations of RPL, each of which implement only a subset of features of the specification. In this section, we explore some of those implementations.

#### 5.1.1 SimpleRPL

SimpleRPL [64] is Linux-based implementation of RPL. It aims to complete the Linux Wireless Sensor Network ecosystem by bringing a (hopefully) fully-compliant RPL implementation. It has a code base of 3382 lines written in Python. SimpleRPL implements storing mode of operation with no multicast support (MOP value 2). A participating node can act as a DODAG root or as a RPL router. It implements Objective Function zero (RFC 6552). However, the rank increase is always a same fixed value. This is because there is no feedback from the layer 2 or the layer 3 (yet), meaning that there can be no indication on the link quality. SimpleRPL has the capability to store unbounded number of DIO parents, but can only store one DAO parent at time. This indicates that even if a node has multiple candidate parents, it can have only one downward route to itself through any one of the parents.

SimpleRPL however does not implement MRHOF because as of when this implementation was developed, there was no way to retrieve link quality information from IEEE 802.15.4 links. It does not support floating DODAGs, leaf function and only works in unsecured mode. There is also no Path Control support in DAO messages.

#### 5.1.2 TinyRPL

TinyRPL [71] is an implementation of RPL for TinyOS. It is designed to be used with BLIP, the IPv6 stack. TinyOS is an open source, BSD-licensed operating system designed for low-power wireless devices, such as those used in sensor networks, ubiquitous computing,



personal area networks, smart buildings, and smart meters. TinyRPL is currently used by many companies all over the world, such as Motorola, Intel, Arch Rock, Crossbow, and the People Power Company. It has a code base of 3110 lines written in NESC.

TinyRPL implements Objective Function 3 (ETX) by default. It stores a bounded number of DIO parents (20). It does not support secure modes of operation, floating DODAGs or path control. It does however support leaf mode, where a node that does not follow the rules of the DODAG can still participate in routing by acting as a leaf.

### 5.1.3 ContikiRPL

Contiki [10] is an open source operating system for the Internet of Things. Contiki connects tiny low-cost, low-power microcontrollers to the Internet. ContikiRPL [11] is an implementation of RPL made to run on Contiki. It is used by companies such as Texas Instruments, Atmel, SAP, Cisco and Redwire. It has a code base of 5340 lines written in C.

ContikiRPL implements ETX as the default OF, but allows the configuration of different OFs as well. It supports the storage of unlimited number of DIO parents. Additionally, all modes of operation (0-3) are supported. Path control and leaf mode operation are supported but it does not support floating DODAGs or secure modes of operation.

### 5.1.4 RIOT-RPL

RIOT [60] is an open-source microkernel-based operating system, designed to match the requirements of Internet of Things (IoT) devices and other embedded devices. It has been specially designed to work with very low memory footprint (on the order of a few kilobytes), high energy efficiency, real-time capabilities, communication stacks for both wireless and wired networks, and support for a wide range of low-power hardware.

RIOT-RPL [61] supports all modes of operation (MOP 0-3) and is by default configured for OF 0. It supports the storage of unlimited DIO parents but allows only one DAO parent for downward routes. It also supports floating DODAGs. However, security modes are yet to be implemented and it currently works only in unsecured mode. Among all the implementations, RIOT-RPL has the most extensive support for running multiple instances simultaneously. It has a total code base of 1607 lines and is written in C.

## 5.2 Industrial Implementations of RPL

There are a number of companies that have their own custom implementation of RPL including Cisco [6], Ember, Huawei and Samsung. However, due to the implementations being owned by the company, they are proprietary and confidential. Hence not much information

was collected regarding these. It was however, very clear that most companies preferred to use one of the standard open source implementations rather than build it from the scratch.

## 5.3 Issue-by-Issue Analysis

### 5.3.1 Incompatible Modes of Operation

All the implementations discussed in this section have the following choices with regards to modes of operation:

1. Do not support downward routes
2. Support downward routes through storing mode
3. Support downward routes through non-storing mode

This gives rise to a major issue, which is incompatibility between the different modes of operation. Different nodes may have different capabilities, and maybe more suitable for a specific mode of operation. Nodes with more resources such as memory, processing power and battery life may prefer to operate under storing mode whereas nodes with a higher degree of constraint on resources may prefer operating under non-storing mode and handing off routing duties to more capable nodes like the root. An LLN may comprise of a mixture of these two types of nodes, but will be forced to operate under a mode that may or may not suit them. Works like [13] highlight this incompatibility as an issue and propose solutions that allow a hybrid mode of operation.

### 5.3.2 Multiple Instances

Three out of the four implementations of RPL that have been discussed in this section do not support multiple instances. This can be largely attributed to the fact that the RPL specification itself does not define RPL's operation when multiple instances are present. RIOT is the only implementation that provides full support for multiple instances. Even in the case of RIOT, it is unclear whether there is any existing deployment that actually uses this feature of RPL.

### 5.3.3 Objective Functions

RIOT and SimpleRPL contain implementations for OF0 in the source code, while TinyRPL implements MRHOF. ContikiRPL includes both OF0 and MRHOF in the source code. Although multiple objective functions can be supported, these implementations choose to provide default support for only a single objective function. In practice, all the works reviewed in this thesis used only a single objective function in their deployments. Among those that choose between MRHOF and OF0, a majority use MRHOF due to better energy efficiency.

A number of works also design their own routing metric as discussed in Chapter 6. To achieve a higher degree of interoperability, a single routing metric must be standardized for LLNs, as discussed further in Chapter 7 and Chapter 8.

### 5.3.4 Security

All four open source implementations of RPL do not support secure modes of operation. they operate under the default Unsecured mode and rely on other security mechanisms (such as link layer security) to satisfy application-specific security requirements. Even the industry implementation of RPL by Cisco does not have any support for the pre-installed and authenticated security mode. Since the code base of other industrial implementations was not available for review, there is no information regarding whether they implement security features or not. However, all the works studied in this thesis did not implement a secure RPL. Hence there is evidence to indicate that security as a feature can be removed from the RPL specification and left to other layers.

### 5.3.5 Floating DODAGs and Local DODAGs

Three out of the four implementations discussed above do not provide support for floating DODAGs. RIOT is the only implementation that includes support for this feature, and even then it is unclear whether this has actually been used in real deployments. None of the works studied in this thesis have used floating DODAGs as a useful feature to maintain connectivity during repair (which is stated as a use case in the specification). Local DODAGs are also not explicitly implemented.

### 5.3.6 Underspecification of Local and Global Repair Trigger

The RPL specification does not give an exhaustive list of events that trigger a local or global repair. This ambiguity is clearly reflected in the implementations, as each of them have their own mechanism of triggering a repair (with few overlaps). This implementation choice has a huge impact on performance, as repairs are often costly and unnecessary. RIOT, ContikiRPL, SimpleRPL and TinyRPL trigger local repairs when a non-root node receives a DIO message that has a newer version number than the one they currently have. Additionally, RIOT and Contiki also trigger local repairs when no preferred parents are found in the DODAG that the node is part of. SimpleRPL, ContikiRPL and RIOT trigger global repairs when the root receives a DIO message that has a newer version number than itself. However TinyRPL has periodic global triggers which is maintained by a separate timer.

### 5.3.7 General Size of Code Base

Because of RPL having so many features, a complete (or incomplete) implementation involves a huge code base that is often unsuitable for resource constrained nodes with limited on-board

memory. Among the open source implementations discussed in this chapter, RIOT-RPL has the smallest code base of 1607 lines of C code, while ContikiRPL has the largest with 5340 lines of C code. While RPL is designed with an intent to satisfy a wide-variety of routing requirements posed by LLNs, the size of code could be a new challenge rendering RPL unsuitable for small nodes.

## 5.4 Summary

A fair number of groups have implemented RPL. This chapter explores some of the open source and proprietary implementations of RPL. The most popular opens-source implementations of RPL include TinyRPL, ContikiRPL and the more recent RIOT-RPL. Companies like Cisco have their own custom implementation of this routing standard. From the observable code base, it is clear that the underspecification and ambiguity in the standards document has lead to a wide variety of implementation choices, such as the list of events that trigger a local and global repair. Also, these implementations highlight issues in the protocol's design, such as inclusion of too many features that render the code base too huge to fit into resource constrained nodes, or the incompatible modes of operation.

# Chapter 6

## Past Work on RPL Improvements

A number of efforts have been made to improve the RPL Routing standard as defined in RFC6550. Some of them focus on designing composite routing metrics that cater to specific application needs such as congestion avoidance and improved energy efficiency. There are also a class of improvements that target multi-path routing for efficient topology construction, increased network lifetime and improved PDR. Yet another class of improvements target better support for broadcast, reducing average delay and standardizing routing objectives. A few of these works have been discussed in this chapter. These studies range from

### 6.1 Combined Metrics

A lot of past work in the field of RPL improvements has had to do with combining two or more routing metrics to achieve better optimization. In [34], the authors describe a number of distinct routing metrics and their characteristics:

- Hop Count (HC): This primary routing metric is used to report the number of traversed nodes along a path. This metric increases strictly monotonically from the root towards the leaf nodes.
- Link Latency (LL): This is a primary routing metric that is a positive real number and is minimizable. Paths with lower link latency are better than those with higher values. However this does not imply lower overall energy consumption since all links are not equally loaded or equally lossy.
- Expected Transmission Count (ETX): This link-reliability metric was defined to distinguish lossy and/or congested paths. It is defined as the number of transmissions (including retransmissions) a node expects to make towards a destination in order to successfully deliver a packet, according to [33].

- Link Quality Level (LQL): This is another link-reliability metric. The LQL of a path contains an array of at most seven values. Each value represents the number of links along this path with an LQL value equal to the index.
- Remaining Energy (RE): This is an energy-aware routing metric which reflects the remaining energy of the end node of a direct link. It is expressed as the ratio between the maximum (initial) energy and the current energy value. It is minimizable.
- Path Forwarding Indication (PFI): This trust-metric is defined as the inverse log of the probability of successful forwarding of a packet by the next-hop neighbor of a node. Links with lesser value of PFI are preferred.

As evident, each of these metrics optimizes a single performance aspect. In order to optimize more than one performance aspect, it is ideal to combine multiple primary routing metrics into a composite one.

### 6.1.1 Additive and Lexical Composition of Metrics

In [34], the authors define two ways of combining routing metrics: additive composition and lexical composition. If additive composition is to be used, then the participating routing metrics need to hold the same order relation in order to be valid. However this condition need not hold for combining them lexically. In both cases, positive real numbers can be used as multiplication factors to adjust the relative weights of routing metrics according to the application requirement.

Composite metrics were tested on the JSIM open simulation platform [35] on a 10X10 grid of 100 nodes, on MP2P traffic. The results are as follows:

- When combining HC with PFI, it was found that the lexically combined metric showed better detection of malicious nodes when tested against a pure HC metric. It also showed comparable latency even though it chose longer paths in certain cases, when misbehaving nodes were detected.
- When combining HC with RE, it was found that the composite metric showed better performance in terms of energy distribution among all nodes when compared to pure HC. There was no difference between lexical combination and additive combination. there was also no difference when the parameters of composition were varied.

However, while combining HC and PFI, the composite metric is tested only against pure HC and not against pure PFI. It seems like PFI alone would have helped in singling out malicious nodes and a composite metric would have only helped in finding a shorter path while also avoiding misbehaving routers. A similar approach was followed while combining HC with RE. The composite metric was not tested against pure RE. Also, the authors have described six primary routing metrics in great detail, but have only discussed two possible

combinations out of 720. This, in addition to using just simulated results, seems inefficient to validate whether composition is really beneficial for the WSN scenario or not.

### 6.1.2 CA-RPL

CA-RPL [68] is a congestion avoidance multi-path routing protocol which uses composite routing metrics based on RPL. This is particularly useful in emergency scenarios where there is high data traffic which could possibly lead to network congestion, high packet loss and delay. During emergencies, the sensor network needs to not only collect information from multiple sensors and relay this information to a central controller, it also needs to make rapid responses in case they are needed. If the network is congested due to large amounts of data traffic, the monitoring system becomes unable to detect information in a timely manner and may also lose some important information.

As of now RPL implements only two types of Objective Functions, Objective Function Zero (OF0) [50] and Minimum Rank Hysteresis Objective Function (MRHOF) [21]. In OF0, the routing metric used is the hop count. In MRHOF, the routing metric used is the ETX. In either case, RPL uses only a single metric to make routing decisions and both of these fail in case of emergencies. If OF0 is used, the nodes suffer from uneven energy and if MRHOF is used, there is increased delay. In the event of an emergency, it is ideal that neither of these effects are caused. The basic version of RPL implements ETX as the default OF, which is clearly not suitable for dealing with this kind of scenario.

CA-RPL considers the link reliability, load balance and time factor, and balances the LLN by easing the congestion and reducing the time delay. A new routing metric, DELAY\_ROOT, is proposed that minimizes the average delay towards the DAG root. This metric is based on the ContikiMAC radio duty cycling protocol where a node saves time by sending packets to the parents that are already awake, instead of waiting for the preferred parent to wake up. The other three route computation metrics are:

1. Link quality using ETX
2. Network Load found by measuring the number of packets received by a node  $v$  in a specific time period ( $REC_v$ ). A smaller value indicates that the network is relatively free.
3. Rank of a node, used to check for loops and prioritize sending packets to nodes closer to the DAG root

The respective weights of each of these factors can be varied using weight coefficients according to application requirements. By using multipath routing, CA-RPL is successful in dispersing large amounts of data traffic to different paths, thereby avoiding congestion and reducing delay. The performance evaluation is done through simulation experiments based

on Contiki. The setup was deployed in a chemical factory where a methane leak was anticipated, leading to high volume of data traffic. The results showed that CA-RPL reduces average time delay by about 30% compared to the original RPL when the inter-packet interval is short. It also showed almost 20% reduction in packet loss ratio when the network was congested. The average throughput, or number of packets received by the root per unit time (PRN) was studied, and CA-RPL was shown to improve PRN by 20% to 50% depending on simulation parameters.

However it is worth considering that ContikiMAC already has powerful mechanisms to achieve precise timing, and hence can be used to implement this new routing metric. It is unclear whether this metric can be efficiently implemented using other duty-cycling protocols without adding complexity overhead. ContikiMAC also assumes that all nodes have the same wake-up interval. This may not be true in all WSN scenarios. This method adds a lot of additional fields to the DIO, such as ETX, Rank,  $REC_v$ ,  $RANK_v$ , and  $DELAY\_ROOT$ . There is already a risk of L2 fragmentation due to the large size of RPL control messages so this might only increase the probability of that happening. Adding extra fields to DIO messages might make CA-RPL non-interoperable with standard implementations of RPL. Interoperability studies have not been discussed by the authors. Additionally, the simulation studies were performed on a WSN containing only 21 nodes, hence performance under scale has not been studied. Also, when there is no congestion in the network, CA-RPL shows worse packet loss rate than the original RPL because the original RPL is simpler and operates quickly when the data traffic is small. Time delay for nodes closer to the DAG root have been found to be higher while using CA-RPL than while using the original RPL. This is because of the extra complexity in CA-RPL to find optimized routes, as well as competing for the limited number of congested links to the root in the event of emergencies. Congestion cannot be avoided here because there is possibly only one route to the root.

Hence, though CA-RPL proposes a composite routing metric that is suitable for emergency scenarios in certain ways, it is largely unclear whether this provides any real performance improvement over the existing standard.

### 6.1.3 QoS-Aware Fuzzy Logic Objective Function

In [20], the authors note that relying on a single objective function maybe inefficient and could degrade the performance of the DAG as it may not fully satisfy the application requirements. Choosing the hop count metric might generate shorter paths, but it could also lead to node failure due to battery depletion, as battery level is not considered in the decision making process. In the same manner, ETX might make the network more reliable but it comes at the cost of high routing latency. Hence it would be unsuited for applications with specific timing requirements. They identify a need to design a holistic objective function that combines several representative metrics to be able to characterize the route quality in a more efficient way. A number of advantages of fuzzy logic have been indicated in the paper:



1. It allows an abstract reasoning on values of any range
2. It provides a rigorous algebra for dealing with imprecise information
3. It is a convenient method of combining conflicting objectives and expert human knowledge
4. It can be implemented with low complexity algorithms

A new objective function, OF-FL or the Fuzzy Logic OF has been defined which is a four-input fuzzy controller with three membership functions for each input. OF-FL considers end-to-end delay, hop count, link quality and node energy as the different fuzzy variables. Fuzzy logic is used to assess the best neighbor to be the preferred parent.

The simulation studies are performed on a Contiki/Cooja implementation of RPL on a simulated test-bed of 100 RPL routers. The newly defined OF-FL is compared against the standard OF0 and MRHOF metric and evaluated based on average hop count, end-to-end delay, packet loss ratio, average remaining energy and average number of parent changes. The results are summarized as follows:

- OF-FL has the tendency to reduce the number of hops within a DAG and performs similar to OF0.
- There was higher churn while using OF-FL as compared to using OF0 or MRHOF. Although this indicates more responsiveness, it also indicates instability.
- OF-FL was found to have the lowest average end-to-end delay for nodes farther away from the root. In other cases, the delay was comparable with the standardized OFs.
- The energy is well distributed among the nodes while using OF-FL. OF-FL also delays the battery depletion of the first node.
- OF-FL has a much lower packet loss ratio than that with OF0, and almost the same packet loss as an ETX based network. OF-FL outperforms MRHOF with ETX when the amount of data packets is high, which demonstrates the effectiveness of OF- FL in high throughput.

However, it is worthwhile to note that using fuzzy logic requires extra rule-based condition checking during parent selection, which adds complexity to the already convoluted RPL algorithm. Additionally, OF-FL causes higher churn, which indicates instability. This would generate higher control traffic overhead, an aspect which hasn't been discussed in the paper. Also this study has only been tested on a simulator.

### 6.1.4 Per-Hop ETX

In [78], the authors propose a new metric, *PER-HOP ETX* which combines the minimum hop count metric and the expected transmission count metric. ETX metric was originally proposed to help the destination node choose the best wireless link for data transmission. However in some cases, the cumulative ETX value of paths with lesser number of hops might be relatively smaller than the ETX value of paths with more nodes, even if it has a relatively low transmission rate. When the number of nodes increases, especially in a dense network scenario, a routing path may contains more nodes. The ETX value of the entire path will be larger than that of a long single hop even though it has a much higher transmission rate. Such long single hops will become a bottleneck restricting the whole network. *PER-HOP ETX* metric takes both link state and hops of potential routing path into consideration. When calculating the best routing path with the proposed method, the sum of the ETX values between each node is used in combination with the hop count to get the average ETX value.

This metric was tested on a Contiki implementation of RPL on the Cooja simulator, against the traditional OF0 and MRHOF with ETX metrics proposed in [50] and [21]. In a larger network, *PER-HOP ETX* was found to reduce network latency by about 25 - 50%. It was also found to show a very slight increase in PDR, though it is unclear if this is entirely due to the combined metrics or due to the particulars of the simulation. It was also found to be slightly more energy efficient. However, this paper was unsuccessful in quantitatively analyzing the results, with the graphs being quite unclear about the exact values of evaluated metrics. Also the details of the simulation study were absent from the paper. It is not practical to draw conclusions without knowing the size of the network it was tested on, type of radio simulated, etc.

### 6.1.5 Improved Energy Efficiency

In [5], the authors again attempt to tackle the problem caused by RPL considering only a single metric: reliability or energy. Choosing reliability would lead to nodes suffering from uneven energy distribution and choosing energy would lead to higher packet loss rate. To overcome this, a combination of ETX and other energy metrics is used in order to balance the overall energy consumption and therefore prolong the network lifetime. The ETX metric represents the stability of a link or route. The lower the ETX value, the higher the transmission success rate. Consequently, nodes with higher ETX value may become the bottleneck which will eventually lead to unbalanced traffic distribution, residual energy, network partition and reduction in overall network lifetime.

To combine ETX with energy metrics, the following equation is used for calculating the score  $R$ :

$$R = \alpha \frac{ETX}{MaximumETX} + (1 - \alpha) \left(1 - \frac{RemainingEnergy}{MaximumEnergy}\right)$$

where  $\alpha$  is the relative weight between the ETX value and energy consumption. For the simulation studies it was assigned the value 0.5. The results show that the overall energy is somewhat distributed when using the composite metric, and network lifetime is prolonged slightly (according to the study, by about 12% for one case). The simulation was performed on the Cooja simulator for Contiki OS. Non-storing RPL mode was used with DIOMinInterval set as three seconds. A total of four to six nodes were used for simulation studies lasting 60 minutes.

However, the tests were performed only on a simulated environment, hence their results on actual hardware are inconclusive. Additionally they only use four to six nodes for each simulation which seems terribly inadequate. Network lifetime studies were only conducted on a single network topology comprising of four nodes. This does not seem sufficient to draw a conclusion such as *“The overall network lifetime for our proposed energy-oriented routing algorithm increases about 12% as compared to the RPL mechanism”*.

In [48], the authors propose two energy efficient routing metrics that use both ETX and the remaining energy of parent nodes to determine the preferred parent. This equalizes energy distribution among the nodes and extends network lifetime. The first metric uses the remaining energy of only the one-hop neighbor set, whereas the second metric takes into account the remaining energy of all nodes along the routing path. The metric was tested on the Cooja simulator, on two simulation environments consisting of 34 and 56 nodes respectively. In a dense network, at the time when the first node runs out of energy while using MRHOF, the node with the lowest amount of energy in the runs using both the metrics proposed by this paper had between 14.9% and 22.7% remaining energy. They also maintained a higher PDR for a much longer time, as more nodes remained alive than the standard MRHOF metric.

However, this work might not be fully standards compliant as it requires hard-coding the battery characteristics onto the mote, which is not always easily accomplished considering heterogeneous motes and varying battery specifications. There are also cases when the total energy consumed for routing is greater than RPL using MRHOF, since longer paths are used to avoid depleted nodes. In these cases, MRHOF gave a longer network lifetime than the metrics proposed in this paper, as these metrics only aim to equalize the energy and not minimize it. The work was also only tested on a simulator hence its performance on real hardware is questionable.

## 6.2 Average Delay Metric

[68] assumes that all nodes have the same sleep interval and wake-up cycle. In [23], the authors describe a delay efficient routing metric, Averaged Delay (AVG\_DEL), that aims to reduce the delay between the nodes in a WSN to the DODAG root, while assuming that

nodes run at varying wake-up cycles and have very low duty cycles (under 1%). AVG\_DEL significantly reduced the averaged delay towards the DODAG root, especially for nodes that are located far away from the DODAG root.

The authors identify seven components of delay between when an intermediate router receives a packet to be forwarded, and when the next hop receives this packet. Out of these, only the time spent in waiting for the next hop to wake up can be minimized, and the other factors form a lower bound on total delay. Since the duty cycles vary from node to node, only the average delay can be computed. AVG\_DEL is equal to:

- 0 for the DODAG root
- $\text{AVG\_DEL}_p + D_p$  if  $\text{AVG\_DEL}_p + D_p < D_{\max}$
- $D_{\max}$  if  $\text{AVG\_DEL}_p + D_p > D_{\max}$

Here,  $\text{AVG\_DEL}_p$  is the average delay announced by the candidate parent,  $D_p$  is the forwarding delay between the node and its parent, and  $D_{\max}$  is the maximum threshold. The authors propose an enhancement to the ContikiMAC duty cycling protocol, allowing for each node to have a different sleeping period. The wake up phase of neighbors is learnt by unicast of a MAC layer probe beacon to candidate neighbors.

Simulation studies were carried out by implementing this metric on Contiki 2.6 operating system using Cooja simulator. It was tested against the standard ETX OF. It was found that the average delay reduced by almost 40% in certain cases, where the nodes were far from the DODAG root. However no reduction was observed for nodes which were only one hop away from the DODAG root since it reduces to competing for a single link.

Since this protocol allows each node to have different duty cycles (as opposed to [68]), each node should inform its neighbors about its cycle time within a DIO. This adds more data to an already large DIO, increasing the risk of L2 fragmentation. The authors do not characterize  $D_{\max}$ , and do not show the basis of its calculation. Also it is unclear how the delay reaches a threshold when in reality, it could be infinity. Simulation studies were performed on a WSN containing only 19 motes. Hence it is unclear how this metric performs when the number of nodes is much higher. The comparative studies are performed using ETX as the standard routing metric. However, the packet delivery ratio (PDR) is maintained at 100%, thereby ETX routes through peripheral nodes. It's not very clear how ETX studies are conducted when PDR is 100%. A study for an average decrease in delay has not been performed. The percentage reduction in delay has only been mentioned for two out of nineteen simulated motes. There was also no reduction in delay noted for nodes that are only one hop away from the DODAG root. This bottleneck could cause a lot of congestion in the event of emergencies. Due to these reasons, it is difficult to draw generalized conclusions.

## 6.3 Multipath forwarding schemes

Multipath forwarding schemes have been widely studied in the past in works such as [9], [39], [57] and [2]. They have been found to have a myriad of benefits such as improved reliability [9], fault-tolerance [39], congestion-avoidance through efficient load-balancing [2] and improved QoS [57]. RPL could also benefit from a multi-path strategy where multiple parents are used for forwarding instead of just a single parent.

### 6.3.1 Efficient Topology Construction

In [52], the authors identify a number of issues with the tree-like structure of the DODAG such as:

- A single node failure can cause large scale disruption and loss, since there is only one path to the sink.
- Forwarding the integrality of the traffic to a single parent could lead to congestion and performance degradation.
- Recent works such as [63] and [2] offer multiple paths that create stable routes. This cannot be achieved in a tree-like topology.
- If beacons are used to save energy, a lot of scheduling is required to avoid collision.

In this paper, the authors propose to shift RPL from its current cluster-tree topology to a DAG topology. This is so that each node has multiple routes to the DAG root and can fully leverage the redundant topology. Currently, RPL does not fully leverage the mesh topology since only a single path exists towards the root. Although the RPL protocol maintains a larger parent set, when the preferred parent is stable, traffic is always forwarded through it. This depletes the node when alternatives are available. [52] uses an opportunistic anycast strategy described in [51] to overcome this issue. Each node picks a packet from its buffer regardless of the current parent. This approach helps to distribute the traffic more evenly and leads to performance improvements such as improved PDR and lesser delay. It was observed that following a new parent required only 0.18% more energy. When following a greedy approach, having multiple parents actually improved overall energy consumption as the load was evenly distributed and number of collisions (in MAC layer) were reduced. Memory requirements increased only linearly per extra parent as very little information about each parent was maintained. The computational complexity remained unchanged for maintaining multiple parents. Connectivity of the network, which was defined as the number of nodes that need to be removed in order to partition the network, was greatly improved. In a cluster tree topology, this value is one (unless the node is a leaf node). Additionally, a reduced delay was experienced. RPL anycast forwarding distributes load among different parents. Hence the average forwarding queue size is reduced and this positively impacts end-to-end delay. It was found that maintaining a single parent in a cluster tree required

20% more control packet transmission than anycast with three parents.

However the studies were only performed on a simulator and not on real hardware. The authors failed to draw a relationship between number of parents and size of the network. This would prove useful for drawing conclusions on network scalability.

### 6.3.2 Increased Network Lifetime

In [30], the authors propose an energy-balancing routing protocol that maximizes the lifetime of the most constrained nodes. In order to maximize lifetime, each node should consume the same amount of energy. The DAG structure is exploited to forward traffic to multiple parents to speed up convergence and increase stability. The authors identify that the current version of RPL can be improved in the following ways:

1. [77] focuses on protocol design. Metrics and mechanisms to make RPL function efficiently are a work in progress, and must be defined based on energy criteria.
2. Rather than using a single parent to forward all traffic, energy balancing mechanisms should be used to distribute traffic among multiple parents.

The main idea is that all paths must consume the same amount of energy and no single path must be overused. They use the *Estimated Lifetime* metric (ELT) [28], which is the time before a node runs out of energy if it keeps on forwarding the same quantity of traffic. This is computed by making a series of assumptions and calculations regarding traffic generated by the node and all its children. Using this value, an energy-balanced topology is constructed using multiple parents. This topology highlights the *bottlenecks*, which are the nodes that will be the first to run out of energy. The traffic is balanced among all neighbors while taking into account the remaining energy on the bottlenecks. This is done by maximizing the minimal lifetime of all bottlenecks. As a result, the number of DAG reconfigurations were reduced and the routing reliability as well as the network lifetime showed an improvement. RPL was simulated on WSNets [19] over 30 simulations. ELT was found to show almost as much reliability as ETX, which is the most reliable metric so far. This new metric also reduced the percentage of nodes that change their preferred parent during one hour of simulation, as well as the number of changes.

However, there is increased risk of L2 fragmentation while using this approach. Since each node sends a list of all known bottlenecks as a part of the DIO message, the DIO message grows larger, and also increases the amount of energy consumed per node. Also, choosing the preferred parent set requires back and forth computation which is time intensive. Selecting which parent to forward traffic through requires computations which are  $n^2$  in time complexity. This introduces a lot of delay in the forwarding plane. Additionally, the ELT metric is not monotonic, hence the authors propose using ETX to construct the DODAG and ELT to compute the rank of each node. This adds further complexity to the

already convoluted algorithm. The algorithm does not handle fragmentation of packets, and the possibility of each fragment taking a different path. It merely proposes some possible solutions but does not implement them. Fragmentation and packet loss is a very real problem in LLNs and is exacerbated when multipath is used.

### 6.3.3 Improved Packet Delivery Ratio (LQA-RPL)

In [66], the authors propose a metric through which each node selects a parent with the maximum remaining energy as the next hop. This multi-parent routing metric is known to improve PDR (Packet Delivery ratio) especially in environments with a high bit-error rate. The proposed protocol is called Link Quality Aware Routing Protocol (LQA-RPL). LQA-RPL calculates rank based on the probability of unsuccessful transmission, which is a metric that can be derived from ETX. The preferred parent is chosen based on the maximum residual energy so as to extend the network lifetime. Based on simulation studies, LQA-RPL was found to outperform standard RPL (which uses hop count as the default metric) in terms of PDR and network lifetime. However, LQA-RPL is only evaluated against the hop count metric. It would be more meaningful to evaluate it against ETX since LQA-RPL also uses a metric that is derived from ETX. Also, the quantitative results of the simulation are not clearly called out and left to the user's interpretation.

### 6.3.4 Support for Anycast

ORPL [17] is an opportunistic routing protocol that supports any-to-any on-demand traffic. In large-scale networks where battery operated nodes communicate wirelessly over multiple hops, a number of extra requirements are imposed on the routing protocol:

1. It must satisfy the energy requirements of the devices, to help operating on battery and to keep maintenance costs low.
2. It must support traffic with any pattern in time and space, to act as an application-agnostic routing infrastructure.
3. It must be reliable and reactive, enabling interactive applications.

The authors note that in today's WSNs, nodes are not only seen as data sources but also as globally addressable actuation end-points. Hence routing must be supported not only towards a sink, but from any node to another. RPL's rooted DODAG topology helps scale to large networks by maintaining reliable and small routing state towards a single destination. However this comes at the cost of increased hop count. ORPL keeps the advantages of this structure while decreasing routing latency by using a multipath approach. It uses the Expected Duty Cycle (EDC) metric [40], which is claimed to be the multipath equivalent of ETX. It also uses a Trickle timer to disseminate routing updates, the same way RPL does. Nodes anycast packets instead of choosing the next hop and unicasting it. P2P routing is

done by forwarding a packet up until a common ancestor is found, and then forwarding it down. This is similar to the storing mode of operation in RPL. In order to reduce the size of storing state, Bloom filters are used.

ORPL was implemented on the Contiki OS and evaluated on the Indriya testbed [14]. The hardware platform consisted of 135 TelosB motes. Since RPL uses routing tables and ORPL stores routing sets, ORPL was found to be more memory efficient even when RPL routing tables were stored using address compression. When considering upward and downward routes respectively, ORPL showed a slightly higher PDR of 99.5% and 99.0% than RPL, which showed a maximum PDR of 97.39% and 91.9%. ORPL was also seen to have lower end-to-end latency since packets are not always sent up to the root. It was also found to be more robust during network outages since the topology need not always be reconfigured.

Although ORPL does provide certain advantages, it is worth considering that it produces a significant number of duplicate control packets, which accounted for 9% to 50% of the total traffic during evaluation studies. This increases as wake-up interval increases. Since each node needs to necessarily store routing sets, the memory requirement of each resource constrained node increases. This is not always possible in WSNs. Non-storing mode of operation cannot be used with ORPL. Since routing sets are broadcasted with trickle updates, the size of the trickle control messages increase, thereby increasing the risk of L2 fragmentation.

## 6.4 Broadcast Support

In the current specification, RPL does not provide support for broadcast. Broadcast is an important requirement for a variety of WSN applications. If RPL has to support broadcast, then the DODAG root would have to unicast data individually to all the nodes in the subtree. This is very inefficient and would cause a lot of redundant messages in the network. [25] proposes a few mechanisms to deal with broadcast, in the event that it is required by a RPL implementation. Although classical flooding is a solution, it causes a lot of duplicate transmissions which is highly energy inefficient, and drains the resource constrained nodes and reduces network lifetime. In this paper, the authors propose mechanisms which leverage the existing DODAG structure that RPL constructs. The first mechanism proposed is classical flooding, wherein a router floods a packet upon first receipt. Subsequent receipts of the same packet are not retransmitted. Through this mechanism, there is no need for control traffic. However, the following conditions are necessary:

1. There must be a method of uniquely identifying a packet to avoid retransmission
2. Each router must maintain state of already received packets, which increases the memory requirements in resource constrained nodes



3. There is a large degree of redundant retransmissions in the network which causes energy drain.

Due to these reasons, classical flooding is not recommended as the flooding mechanism for broadcasting in WSNs, however it is used as a baseline.

The second mechanism suggested is Multiparent Relay Flooding (MPRF) in which each node selects a subset of its neighbors to forward data to so that all the two hop neighbors of the node would receive the packet. This drastically reduces the number of retransmissions but requires each node to also maintain the two hop neighbor table. Each broadcast packet must also be uniquely identified at the router. Another mechanism is Parent Flooding (PF), in which a router retransmits only the broadcast packets received from its parent. An optimization is to only forward packets received from the preferred parent, known as Preferred Parent Flooding (PPF). Here, duplicates received from other routers are ignored for retransmission. They also suggest mechanisms such as Preferred Parent MPR Flooding (PPMPRF) and an optimized version of the same. These might reduce the number of retransmissions but increase the amount of state stored at each router. Upon testing a Java implementation of RPL with broadcast on the NS2 simulator, it was found that MPRF and PPMPRF yield the lowest number of collisions and the highest delivery ratios. The optimized MPR-based broadcast mechanism had the lowest delay with optimal path length.

These mechanisms are certainly useful in heading towards a network wide broadcast. However, these are only add-ons to the existing RPL specification, and come at their own cost while trying to use RPL for something it wasn't designed for.

## 6.5 Objective Function

The Objective Function is a controversial part of the specification. Many argue that it is unnecessary and adds a lot of overhead to the already complex protocol. In [26], a new routing protocol called DMR is proposed as a solution for routing in mobile sensor networks (MSNs). DMR builds off of RPL's framework, but provides path redundancy and allows mobile sensors to find multiple alternative paths in the event of local and global failures. DMR constructs a DODAG by using two routing metrics, rank information obtained from RPL using hop count and LQI. Hence there is no concept of OF. Hop count is used to determine the rank, which is easy to compute and requires no additional measurements even in the situation where nodes are moving randomly. DMR also moves closer to a DAG structure versus a DODAG, by allowing routing through sibling nodes in the event of a local or global network failure. This increases network reliability and leverages the redundancy of a DAG structure to a much greater extent than standard RPL.

## 6.6 Summary

A number of research efforts have been made to improve upon the RPL routing standard, in its existing form as specified in IETF RFC6550. In this section, a few of these efforts have been discussed in detail. A large number of these works [68][23][34][5][66][78][20] focus on developing new routing metrics which might be better suited for specific application requirements. Many of them point out a number of flaws with the existing metrics defined by ROLL WG, rendering them unsuitable for a variety of applications such as emergencies [68] and latency-sensitive scenarios. If OF0 is used, the nodes suffer from uneven energy, battery depletion and network failure, as battery level is not considered in the decision making process. MRHOF in turn leads to increased delay and high routing latency. Many of these works design composite metrics [68][34][5][78][20] which use a combination of different metrics in order to better satisfy application requirements. Some improvements suggest changes to the topology by moving towards a DAG structure [52][30][66][17] while others provide solutions for broadcast [25] and anycast [17].

Once again, it was observed that all except one of these studies have been conducted only on simulators, and hence the results drawn are not conclusive of performance in hardware. Only one of these studies [17] was performed on a real testbed. This again seems to indicate that RPL has a lot of implementation constraints and might not be suitable for real WSNs.

Citation	Main Idea	Improvements	Drawbacks	Number of Nodes	Simulator Used
[34]	Additive and Lexical Composition of Metrics	Lexical Composition performed better than individual metrics	Composition metric not tested against all combinations	100	JSim Open Simulator Platform
[68]	Congestion avoidance using multipath routing	Avoids congestion, increases reliability in high data-traffic scenarios and reduces delay	Assumes usage of Contiki MAC Duty cycling protocol, and that all nodes have the same wake-up cycle. Adds additional fields to DIO message	21	Contiki/Cooja

[20]	QoS aware fuzzy-logic OF	Backwards compatible with RPL, reduces number of hops, packet loss and average delay, distributes energy well	Higher churn, complex to implement	100	Contiki/Cooja
[78]	Composite metric that evaluated Per-hop ETX	Reduces network latency in larger networks, slight increase in energy efficiency	Unsuccessful in clear quantitative analysis as details of simulation studies were absent	10 to 100	Contiki/Cooja
[5]	Composite metric for improved energy efficiency	Overall energy is more distributed, network lifetime is slightly more prolonged.	Very few nodes used for simulation studies	4 to 6	Contiki/Cooja
[48]	Composite metric for improved energy efficiency	More remaining energy per node	Higher PDR, requires hard-coding battery characteristics in nodes	34 to 56	Contiki/Cooja
[23]	Propose an improvement to MAC duty cycling protocol for better energy efficiency	Average delay was reduced for nodes far away from DODAG root	Adds wake-up cycle information to DIO, increases control message size	19	Contiki/Cooja
[52]	Propose to shift RPL from DODAG to DAG topology	Improved overall energy consumption, reduced end-to-end delay	Performance evaluation studies were only conducted on simulators	160	WSNet

[52]	Propose to shift RPL from DODAG to DAG topology	Improved overall energy consumption, reduced end-to-end delay	Performance evaluation studies were only conducted on simulators	160	WSNet
[30]	Use multipath to increase network lifetime	Comparable reliability with ETX, lesser churn	Increased DIO message size, increased time complexity and no proposed solution for packet fragmentation	50	WSNet
[66]	Select node with maximum remaining energy for next hop	Improved PDR and network lifetime	Not evaluated against ETX (only against hop-count)	200 to 600	WSNet
[17]	Nodes anycast packets instead of unicasting it to next hop (ORPL)	More memory efficient, better PDR and latency	Produces significant number of duplicate control messages, cannot be used in non-storing mode	135 TelosB motes	No, ContikiRPL on Indriya Testbed
[25]	Propose broadcast mechanisms for use with RPL	Multiple broadcast mechanisms and their trade-offs detailed	Trying to use RPL for something it wasn't designed for	1000	Java implementation of RPL on NS2
[26]	DMR provides path redundancy, uses single OF	Increased reliability, leverages redundancy of DAG structure	Studies not tested on hardware	100	NS2

Table 6.1: Studies on Improvements to the RPL Routing Standard

# Chapter 7

## Need for a New Standard

### 7.1 Introduction

From the preceding sections, it is very clear that the RPL routing standard has a lot of flaws that need to be addressed. Performance evaluation studies and comparative studies have shown that RPL is not suitable for a number of WSN scenarios. Numerous studies that propose improvements to RPL highlight the fact that the current standard has a lot of issues and would benefit from renovation. Hence this section summarizes the drawbacks of RPL and emphasizes the need for a new routing standard for LLNs and WSNs.

This chapter includes direct references from the standards document[77] released by the ROLL WG. Hence, similar to Chapter 3, a special form of referencing is used for referring to a particular line from RFC6550. In particular, the letter 'p' followed by a number implies that the particular line (or a paraphrasing of the line) or set of lines immediately preceding the reference can be found in RFC6550 in the page indicated by the number.

### 7.2 Unnecessary Features of RPL

1. This thesis studies over 25 past works on RPL and it was found that all of them implement only a single instance of RPL. From studying the open source implementations of RPL in Chapter 5, it was found that only one out of four implementations provided full support for multiple instances. As a matter of fact, [77] itself contains specification for running only a single instance. It is unclear whether any real world deployment successfully supports and uses multiple RPL instances within an LLN application.
2. The Objective Function adds a lot of extra implementation complexity, as it gives rise to each node having multiple ranks (one for each instance that it is part of). Also, among all the works studied in this paper (over 25), none of them used multiple objectives within the same LLN scenario. RPL instances and OFs lead to a number of

extra parameters that need to be configured on every resource constrained router such as:

- a) RPLInstanceID. Though this is configured at the root, in case a node becomes the root of a floating DODAG, this could lead to a single router containing multiple instance IDs.
- b) list of supported Objective Code Points (OCPs)
- c) list of supported metrics

From the analysis, we found that a huge majority of the studies use only the ETX metric. It is unclear whether all the new application-specific metrics proposed show an improvement that is worth the additional complexity [20].

3. DIS mode of operation must be configured at boot-up in order to decide whether the non-root nodes join an existing DODAG in an active or passive manner [p115]. However this feature seems quite unnecessary as it requires the addition of many extra parameters such as the number of DIS messages to be sent and the interval between them. Instead, all implementations could follow either active probing or passive listening to reduce the overall complexity.
4. The control messages sent out by RPL (DIO and DIS) are equivalent to IPv6 Router Advertisements (RA) and IPv6 Router Solicitation (RS) messages defined in IPv6 Neighbor Discovery (RFC4861). In order to reduce the protocol overhead, it would be more efficient to use the existing control messages instead of defining new ones specifically for RPL. This would also increase the degree of interoperability.
5. None of the RPL studies analyzed in this thesis implement Local DODAGs and Floating DODAGs. They are a redundant feature that can be removed from the specification.
6. Although security is an important feature for routing in LLNs, RPL's security mechanisms are not used in any of the discussed RPL implementations. This leads to the conclusion that security as a feature must be given further thought by the ROLL-WG to better suit LLN applications and the capabilities of resource constrained nodes. The security features specified in RPL are largely redundant and unused and therefore can be removed.

### 7.3 Under-specification of standards document

1. RPL does not specify the process by which a router obtains a key from an authentication authority in order to join an authenticated RPL instance to securely operate in the authenticated mode [p17]. It also does not clarify where the key should be obtained from.

2. The details of how an ingress router maps a packet to a particular RPL instance is not specified in the standards document. The document deals with only a single instance [p102].
3. While dealing with DAO inconsistencies, a node that receives a packet with the Forwarding-Error bit set may attempt to send it to an alternate parent after the expiration of a user-defined implementation specific timer. No extra details about this timer are given in the specification [p104].
4. The specification does not provide a full description of the generation of DODAGs specifically designed for multicast or the details of operating RPL for multicast [p104].
5. Exactly when a DODAG root increments the DODAGVersionNumber is implementation dependent and out of scope for this specification [p70]. Hence it is unclear whether a global repair happens periodically, or is triggered by a specific event. This decision affects performance as incrementing the version number often decreases stability of the routing algorithm.
6. The method of coordination between virtual roots is unspecified [p70]. Although the specification allows the existence of virtual DODAG roots, there is no clear method for communication or coordination between them. This would require the specification of new types of control messages, which would only increase the control traffic overhead.
7. The specification states that after hearing a change of DODAG version number from parents through DIO, a node may choose to migrate to this new version at any point in time. However, this implementation decision plays a role in performance. No guidelines are given regarding the value of this variable quantity.
8. The specification also does not state how much preference should be given to staying in the same DODAG once your preferred parent has detached from the DODAG. This is an important consideration since link breakage is a very common scenario in LLNs and nodes must have guidelines about what to do once the parent is no longer a part of the DODAG.
9. The RFC does not give an exhaustive list of events that count as inconsistencies and can cause the Trickle timer to reset [p74]. The RFC only mentions that the timer must be reset when a new DIS message is received (with some restrictions) and when an inconsistency or loop is detected. Additionally, it provides implementers with the flexibility to design more events that could reset the timer. According to [37], it was found that Trickle timer intervals and the number of their resets heavily affected the amount of overhead spent to maintain the DODAG. Hence this is again a case where flexibility is provided at the cost of performance.
10. The selection of DAO parent set is implementation and OF specific [p78]. It is not necessary that the DAO parent has to be the same as the preferred parent. This again

causes extra implementation complexity. The document does not specify how DAO parents are chosen.

11. The number of retransmissions of a DAO message with the ‘K’ flag set (but have not received a DAO-ACK in response) and the interval between these retransmissions is implementation specific [p80]. This again affects performance and needs better definition.
12. DAO transmission scheduling when a DAO with updated information needs to be sent out is implementation dependent. This uses a DelayDAO timer which has a default DAO delay defined in the RFC, but the timer calculation itself is unspecified [p83]. Proper DelayDAO selection can optimize the transmission of DAO messages upwards. The specification suggested that DelayDAO should be inversely proportional to the rank. That way, each node sends only one DAO message upwards during a triggered update, starting from the leaf nodes [p84]
13. Triggering DAO updates by incrementing DAO Trigger Sequence Number (DTSN) field in DIO messages is implementation dependent. In non-storing mode, it is usually the DODAG root that independently increments DTSN, triggering updates from the entire DODAG. In storing mode, a parent may increment its DTSN, triggering DAO updates only from immediate children. However in general, any parent node may increment DTSN at any point in time depending on implementation-specific guidelines [p84]. The exhaustive list of events that trigger a parent to increment the DTSN is not specified, and again affects performance and control traffic.

## 7.4 Known Issues

1. For loop detection and avoidance, RPL relies on forwarding the RPLInstanceID through the Packet Information Option (PIO) which is inserted into the IPv6 hop-by-hop option header. This adds to the already large size of the control message and increases the risk of L2 fragmentation.
2. According to [77], *candidate neighbors that advertise an OF incompatible with the set of OFs specified by the policy functions are ignored* [p107]. Hence, this leads to a lot of interoperability issues since non-standardized OF metrics can be defined by any RPL user.
3. The current documentation does not support routing outside the current instance. It states that it can be done by putting extra checks and measures in place, such as strict ordering of instance IDs.
4. RPL imposes special requirements on the root node. The root is required to initiate the creation of a DODAG by sending out DIO messages. It is also required for configuration



and management. Additionally, the root is required for P2P routing in both storing and non-storing mode. It might also need to act as the border if the participating nodes use a link-local address. However, the root acts as a single point of failure. Though the RFC gives some specification for floating DODAGs that can continue operating if the root fails, this is not always possible. To function as the root of a floating DODAG, a node has to have extra provisions in terms of processing and memory, which might be unused under normal mode of operation.

### 5. Link-Layer Fragmentation of Control Messages

According to [38], application that run in an Advanced Metering Infrastructure (AMI) network should have small control packets so that it does not lead to the risk of link-layer fragmentation. IEEE 802.15.4 standard specifies a frame size of 127 octets [27]. Out of this, 25 octets maybe used for frame overhead and 21 octets for link layer security. This leaves out 81 octets for the layer-2 payload. If the IPv6 compression mechanisms are used as specified by [31], the compressed IPv6 header occupies a minimum of two octets. This leaves at most 79 octets for L3 payload which includes routing protocols signals as well application data. If it exceeds this, then it will lead to fragmentation.

In order to analyse the size of RPL control messages, a standard DIO was chosen since it is a very basic control message that is an integral part of DODAG construction and maintenance. RPL control messages are carried by ICMPv6. The ICMPv6 header occupies four octets. In the remaining 75 octets, 24 are consumed by the DIO base objects. Link metrics vary in size depending on the type of metric used. However, the very basic hop count metric uses eight octets. Hence as a minimum, link metrics can be assumed to claim eight or more octets from the remaining 51 octets, leaving 43. The DODAG Configuration Object occupies 16 octets, leaving only 27 octets for the optional Prefix Information Object for address configuration, which consumes a minimum of 32 octets. Hence this already exceeds the allowed capacity of 79 octets by five octets and is sure to undergo link-layer fragmentation. The above calculation only includes configuration of DIO. In case data traffic is included, then the probability of fragmentation is increased if non-storing mode is used. In non-storing mode, the DODAG root affixes a source route with a fixed overhead of eight octets to every data packet traveling in the downward direction. Additionally the size of source route overhead increases depending on the number of entries in the source route. The exact amount of increase depends on the address length and the number of hops to be traversed. Apart from this, the data packet also includes the data payload, which again varies in length depending on the nature of information to be communicated. Also, since the root is responsible for adding the source routing header, the routers within the DODAG do not know the size of the message in advance. Hence the applications can't tailor their data packets to fit within the unfragmented frame before sending them to the DODAG root for routing down the DODAG.

## 6. Link Bi-Directionality

The basic and standard operation of RPL requires the presence of bi-directional links. This is explicitly called out in the RFC- *RPL operations require bidirectional links* [p8]. This is because, nodes attach themselves to the DODAG through the receipt of DIO messages from candidate parents. However, these established links are used for sending messages from the node to the root, through the preferred parent hence requiring the link to be bidirectional for the very basic operation. However, in case the link is not bi-directional and is unidirectional instead (which is a common occurrence in WSNs and LLNs), RPL does not specify mechanisms to rectify this scenario. RPL also does not specify any mechanism to check whether a link is unidirectional or bidirectional before a node joins a DODAG. The RFC does mention that Bidirectional Forwarding Detection (BFD) and Neighbor Unreachability Detection (NUD) can be used as solutions [p8]. However, it also goes on to say that BFD is highly undesirable as it uses a proactive approach and hence is unsuited for LLNs. NUD is only used in the event of a transmission failure. Hence if a node tries to reach the DODAG root and detects (through NUD) that the parent cannot be reached, its only option is to wait around for another parent which has bidirectional links. Also, if a NUD indicates “no forward progress” based on an upper-layer protocol, there is no guarantee that the problem stems from the preferred parent being unreachable. RPL does not specify any mechanism to react to such events. Hence in the event of a network containing unidirectional links, RPL does not adopt either of these two methods (BFD or NUD) and the router has no standard way of reacting to this situation. This causes delay and unnecessary energy wastage, as the router continues to try to join the network through a bidirectional link which possibly doesn’t exist at all.

7. RPL handles the detection of loops in a reactive manner. When a loop is found, the data packet is buffered at the router and a local repair operation is triggered. However, since the nodes are resource constrained, there is no guarantee that the given router has enough memory to store the incoming data packet while trying to find a new route to the DODAG root. Hence the data packet might be dropped unless the source retransmits it. RPL does not specify any retransmission mechanisms for this type of packet loss.
8. RPL does not fully leverage the DAG structure as it provides only a single path to root. Even though multiple candidate parents are stored by each node, only a single parent is used for forwarding. This could possibly cause congestion, when forwarding the integrality of traffic from the sub-DODAG through a single parent. Additionally it causes faster energy depletion of the preferred parent.
9. The current specification does not provide any support for broadcasting information. Only unicast mechanisms to and from the root have been specified, which would be inefficient for application scenarios requiring broadcast.

10. According to the specification, the greatest level of interoperability may be achieved when all of the nodes in a RPL LLN are cooperating to use the same MOP, OF, metrics, and constraints [p109]. This renders objective functions and the whole concept of RPL instances useless, as it only leads to more issues with interoperability. One of the main goals of designing RPL was to unify the efforts made towards creating a standardized routing protocol for LLNs. Including OFs and RPL instances seems to be in contradiction with that goal.
11. Among all the studies discussed in this thesis which includes over 25 studies on RPL implementations, only two of them [37][17] acknowledge implementing RPL on an actual testbed and testing on hardware. The rest of the studies only test RPL on simulations, which highlights the implementation complexity of this protocol.

## 7.5 Types of applications RPL does not cater to

1. RPL does not support applications which require a fixed MTU (Maximum Transmission Unit). The control message size can vary greatly and RPL does not give any guarantees on the size. There is not much flexibility with the control message fields the RFC doesn't give any specification for compression.
2. RPL is not fully suited for applications where there is a good percentage of P2P traffic flow. In applications such as remote control in building automation, almost 30% of the traffic flows are P2P [44]. Hence using RPL in such an application would cause high control traffic overhead, and also cause congestion at the DODAG root and route stretching. If non-storing mode is used, it might also impose additional memory requirements on participating routers.
3. RPL does not cater well to a request-response traffic pattern such as utility metering [24]. A utility company may want a utility controller to inquire household meters regarding their consumption. This will require them to send a request and expect a reply. The controller might also use the network to set or change parameters in household meters. This would again require a confirmation stating that the change has been carried out. This is commonly the use case in a load management grid where individual households maybe required to increase or decrease their energy consumption depending on the overall load on the grid.
4. RPL does not cater to emergency scenarios where there is a high data traffic in the network. In case of an emergency, a number of messages might be sent out causing congestion at the DODAG root. It also causes delay and possible packet loss [68]. RPL doesn't specify any mechanism of dealing with data packet loss. Additionally the nodes are resource constrained so even buffering them to wait for the network to become decongested would not be possible.

5. RPL cannot be used for topologies with a long chain-like structure that contains paths of length greater than eight (if raw IPv6 is used) or 64 (while using IPv6 address compression [31]), while being used in non-storing mode. The source routing header can be a maximum of 136 octets which includes an eight octet long header. An IPv6 address is 16 octets long. Hence no more than eight can be accommodated unless address compression is used. If the addresses are compressed, then the path length may not exceed 64.
6. In the current specification there is no support provided for broadcast. RPL only supports unicast to and from the DODAG root. In case broadcast is required, the root has to deliver the data to all nodes, which is very inefficient.
7. RPL is not suitable for WSNs that contain sensor nodes that can harvest ambient energy from the environment. Such networks are a possible solution for sustaining sensor networks for decades without maintenance, since they can recharge on their own using ambient energy. However in [47], it was found that energy-harvesting sensor networks running RPL produce 40-45% lower goodput than battery-operated sensor networks as the harvesters drop thousands of packets due to critical nodes running out of energy at the same time.

## 7.6 Features that would benefit RPL

1. RPL nodes should ideally leverage the DAG topology by maintaining multiple paths to the DAG root, rather than choosing a single preferred parent. Additionally, there could be multiple Downward routes to the node as well, rather than having a single DAO parent. The DAG structure provides more routing redundancy than a DODAG, and is more useful in LLN scenario where links are not reliable and fail often.
2. Nodes within an LLN are not homogeneous in terms of on-board resources. Hence depending on the capability of the node, it should be allowed to operate under storing or non-storing mode, as highlighted in works such as Hydro [13]. Forcing all nodes to operate under non-storing mode because of the constraints of a few nodes within the network unnecessarily leads to route stretch and routing delay. Additionally, nodes may not always have the capability to store local routing tables, as resources can vary over time depending on application requirements.

## 7.7 Has RPL succeeded or failed

From this section, it is clear that RPL as a routing standard for low-power and lossy networks has a lot of standing issues that need to be addressed. As far as the design is concerned, it is evident that there are a lot of redundant features that are neither deployed in real-world

implementations, nor have any real use cases in the WSN scenario. Such additional features only increase the implementation complexity of RPL rendering it unusable for small resource-constrained nodes. Additionally the standards document has a lot of underspecification, which leads to a wide variety of implementation choices. These choices not only adversely affect performance, they also give rise to non-interoperable implementations that go against the very basic design ideology of creating such a routing standard. Finally, RPL is not suited for a number of WSN and LLN scenarios, as highlighted in the specific works in Section 7.3. Overall, RPL was headed in the right direction towards unifying the efforts in developing a standard routing protocol for WSNs and LLNs. It includes a highly comprehensive feature-set. However, the ambitious nature of the specification and inclusion of a number of redundant features render it highly complex and quite unsuitable for resource-constrained environments. Hence as a verdict, RPL is not entirely successful in solving the complex routing challenges posed by LLNs.

# Chapter 8

## RPL-Lite

### 8.1 Introduction

This chapter provides a specification for RPL-Lite, a lightweight routing protocol based on the RPL routing standard.

#### 8.1.1 Design Principles

RPL-Lite was designed with the objective to overcome the drawbacks and eliminate the complexities of the RPL routing standard specified in RFC6550 proposed by the IETF ROLL WG. A network may contain and run only a single instance of RPL-Lite, where all nodes obey the same routing objective. OFs are also eliminated from the RPL-Lite specification. All implementations of RPL-Lite follow the same objective, which is reducing the net energy consumption by following the ETX metric (MRHOF). The quantitative comparison between these two metrics is out of scope for this specification and is left as future work.

RPL-Lite expects an external mechanism to be triggered during the parent selection phase in order to verify link properties and neighbor reachability. It cannot operate in an environment that does not contain bidirectional links. In a general fashion, a detection mechanism that is reactive to traffic is favored in order to minimize the cost of monitoring links that are not being used.

RPL-Lite follows RPL's mechanism of disseminating information over the dynamically formed network topology using a Trickle timer [43]. This dissemination enables minimal configuration in the nodes and allows nodes to operate mostly autonomously.

RPL-Lite uses IPv6 Neighbor Discovery (RFC4861) [45] information such as the Router Advertisements (RA) and Router Solicitations (RS), and IPv6 Inverse Neighbor Discovery (RFC3122) messages such as Inverse Neighbor Discovery Solicitations (INDS) Inverse Neighbor Discovery Advertisements (INDA) for topology formation. Link Layer acknowledgements

are used to indicate receipt of destination advertisement messages. Protocol-specific ACKs are eliminated.

RPL-Lite retains the concept of global repairs, which is basically a reboot operation where each node is disassociated from all neighbors and made to rejoin the network as if it were a new node. RPL-Lite also retains local repairs. However, it does not provide an exhaustive list of events that trigger either of these repairs and leaves it as an implementation choice.

RPL-Lite does not have floating DAGs or local DAGs. Every time a node needs to communicate with another peer, a new route is installed either through the DAG root or through a common ancestor.

RPL-Lite also does not support security mechanisms, and relies on external mechanisms such as header encryption or application layer security for secure applications.

### 8.1.2 Expectations of Link-Layer Type

In compliance with the layered architecture of IP, RPL-Lite does not rely on any particular features of a specific link-layer technology. RPL-Lite is designed to be able to operate over a variety of different link layers, including ones that are constrained, potentially lossy, or typically utilized in conjunction with highly constrained host or router devices, such as but not limited to, low-power wireless or PLC (Power Line Communication) technologies.

## 8.2 Protocol Overview

The aim of this section is to describe RPL-Lite in the spirit of RFC4101 [59]. Protocol details can be found in the further sections.

### 8.2.1 Topologies

This section describes the basic RPL-Lite topologies that maybe formed, and the rules by which these are constructed, i.e., the rules governing the DAG formation.

#### 8.2.1.1 Constructing Topologies

RPL-Lite organizes a topology as a Directed Acyclic Graph (DAG) where there is one DAG per sink. A RPL-Lite network comprises a single DAG with its components connected by a transit network. If there are multiple sinks per DAG, then it is expected that the roots are federated by a common backbone, such as a transit link. The details of how the roots communicate among each other or how they are federated is out of scope for this specification.

### 8.2.1.2 RPL-Lite Identifiers

Unlike RPL, RPL-Lite does not use DAGIDs. A node in a network has no necessity to know which DAG it is part of, as long as there is a neighbor through which it can send packets. Hence RPL-Lite eliminates the use of InstanceID and DAG ID. The only identifiers used are Rank and Version Number. Rank establishes a partial order over a DAG, defining individual node positions with respect to any one of the DAG roots. Version Number specifies the version of the DAG that the node is a part of, in order to maintain consistency during global repairs. The Version Number is incremented every time a global repair takes place.

### 8.2.1.3 RPL-Lite DAGs

A network running RPL-Lite contains one or more DAG roots, each of which provide routes to certain destination prefixes. These roots may operate independently, or they may coordinate over a network that is not necessarily as constrained as an LLN. A network running RPL-Lite comprises a single DAG with a single or multiple roots. If there are multiple roots, then these roots must be federated by a common back-bone such as a transit link.

## 8.2.2 Upward Routes and DAG Construction

RPL-Lite provisions routes Up towards DAG roots, forming a DAG optimized according to the MRHOF Objective Function. RPL-Lite nodes construct and maintain these DAGs through IPv6 Router Advertisement (RA) messages.

### 8.2.2.1 Objective Function

The Objective Function (OF) defines how RPL-Lite nodes select and optimize routes. An OF defines how nodes translate one or more metrics and constraints, which are themselves defined in RFC6551 [33], into a value called Rank, which also approximates the node's distance from the DAG root. An OF also defines how nodes select their parents. Based on past studies, it was found that majority of RPL implementations chose MRHOF over OF0, because ETX was able to take link metrics into account instead of choosing a path purely based on hop-count. Although min-hop metric was simplistic and eliminated a lot of implementation complexities as well as simplified loop avoidance, ETX was chosen as it suited the LLN scenario better. ETX was able to minimize energy consumption by choosing paths with lower retransmissions. Since there is only one OF used, the Objective Code Point (OCP) field is eliminated from node advertisements.

### 8.2.2.2 DAG Repair

In RPL-Lite, A DODAG root institutes a global repair operation by incrementing the DAGVersionNumber. This initiates a new DAG Version. Nodes in the new DAG version can choose a new position whose rank is not constrained by their Rank within the old DAG



version. However, since global repairs are costly and often unnecessary, Global Repairs are retained as an optional feature in RPL-Lite. Implementations can choose to disregard this feature by always maintaining a DAGVersionNumber of zero. During a repair, nodes are disconnected and cannot perform routing until they reattach to a DAG.

RPL-Lite also supports mechanisms that may be used for local repair within the DAG. Local repairs may be triggered when a routing loop or inconsistency is discovered by a non-root node. This is fixed by poisoning the one-hop and optionally two-hop neighbors, and de-attaching from the DAG.

An exhaustive list of events that trigger local and global repairs is out of scope for this specification.

### 8.2.2.3 Security

RPL-Lite does not support security mechanisms, and relies on external mechanisms such as application-layer security or header encryption for the specific needs of secure applications. None of the implementations of RPL studied in this thesis include security features. Although security is an important aspect of routing in LLNs, it is not necessary in a majority of LLN applications and must be further researched for ease of implementation and simplicity.

### 8.2.2.4 Administrative Preference

An implementation/deployment may specify that some DAG roots should be used over others through an administrative preference. Administrative preference offers a way to control traffic and engineer DAG formation in order to better support application requirements or needs.

### 8.2.2.5 Data-Path Validation and Loop Detection

The loop-detection mechanism used by RPL-Lite is the same passive strategy used by RPL, wherein loops are not detected until there is data to transmit. The rank of a node is transmitted along with a data packet in the RPL Packet Information. An inconsistency between the routing decision for a packet (Upward or Downward) and the Rank relationship between the two nodes indicates a possible loop. On receiving such a packet, a node institutes a local repair operation.

### 8.2.2.6 Distributed Algorithm Operation

A high-level overview of the distributed algorithm, which constructs the DAG, is as follows:

- Some nodes are configured to be DAG roots, with associated DAG configurations.

- Nodes advertise their presence, routing cost and related metrics by sending a link-local multicast RA message to all-RPL-nodes.
- Nodes listen for RAs and use their information to select new DAG parents, or to maintain existing parents, according to the Rank of their neighbors.
- Nodes provision routing table entries, for the destinations specified by the RA message, via their DAG parents in the DAG version. Nodes that decide to join a DAG can provision one or more DAG parents as the next hop for the default route and a number of external routes.

### 8.2.3 Downward Routes and Destination Advertisement

RPL-Lite uses IPv6 Inverse Neighbor Discovery Advertisement messages (INDA) messages to establish Downward routes. INDA messages are an optional feature that can be used for applications that require point-to-multipoint (P2MP) or point-to-point (P2P) traffic. RPL-Lite supports three modes of Downward traffic: Storing (fully stateful), Non-Storing (fully source routed) and Hybrid (a combination of stateful and source-routed). Any given DAG running RPL-Lite has the ability to support both storing and non-storing mode through the hybrid mode, depending on the capacity of each node. In the pure Non-Storing only case, the packet will travel all the way to a DAG root before traveling Down. In the Storing only case, the packet may be directed Down towards the destination by a common ancestor of the source and the destination prior to reaching a DAG root. In the hybrid case, each node can operate in either storing or non-storing mode depending on its capacity. The packet travels up the DAG until one of two scenarios occur:

- The packet reaches the DAG root because:
  1. Either all the nodes along the way were operating under non-storing mode and kept forwarding the packet to the parent or
  2. The nodes along the way that were operating under storing mode did not have the destination their sub-DAG

Once the root is reached, a source route is attached and the packet gets forwarded in the DOWN direction until the destination is reached.

- The packet never reaches the DAG root because:
  1. All the nodes along the way from the source to the DAG root were operating under non-storing mode but one of them happened to be the desired destination, hence the packet forwarding was stopped or
  2. On its way to the DAG root, the packet encountered a node that operated under storing mode and the desired destination happened to be in the sub-DAG of that node, and hence the packet got forwarded to the sub-DAG of that node instead of UP towards the DAG root.

### 8.2.4 Rank Properties

The Rank of a node is a scalar representation of the location of that node within a DAG Version. The generic properties it must exhibit remain the same as that defined in RFC6550 for RPL. Rank comparisons and Rank relationships also remain the same for RPL-Lite. Since the entire network follows a single OF, each node has only a single rank, as opposed to RPL where each node has multiple ranks depending on the number of RPL instances it is a part of.

### 8.2.5 Loop Detection, Avoidance and Recovery

RPL-Lite tries to avoid creating loops when undergoing topology changes by maintaining strict ordering of Ranks. It includes Rank-based data-path validation mechanisms for detecting loops when they do occur. In practice, this means that RPL-Lite guarantees neither loop-free path selection nor tight delay convergence times, but it can detect and repair a loop as soon as it is used. RPL-Lite uses this loop detection to ensure that packets make forward progress within the DAG and trigger local repairs when necessary. Once a node has joined a DAG, RPL-Lite disallows greediness in order to prevent resulting instabilities in the DAG. To recover from count-to-infinity scenarios, RPL-Lite allows DAG roots to initiate Global Repairs by incrementing the DAG Version Number.

#### 8.2.5.1 DAG Loops

A DAG loop may occur when a node detaches from the DAG and reattaches to a device in its prior sub-DAG. In particular, this may happen when RA messages are missed. This type of loop may possibly be encountered when using some local repair mechanisms. When a loop is detected by a non-root node, a local repair is triggered. When a loop is detected by a root node, a global repair is optionally triggered.

#### 8.2.5.2 DA Loops

A DA inconsistency happens when a router has a Downward route that was previously learned from an INDA message via a child, but that Downward route is not longer valid in the child. With DA inconsistency loop recovery, a packet can be used to recursively explore and clean up the obsolete DA states along a sub-DAG.

## 8.3 Traffic Flows Supported by RPL-Lite

Just like RPL, RPL-Lite supports three basic traffic flows: multipoint-to-point (MP2P), point-to-multipoint (P2MP), and point-to-point (P2P).

### 8.3.1 Multipoint-to-Point Traffic

The destinations of MP2P flows are designated nodes that have some application significance, such as providing connectivity to the larger Internet or core private IP network. RPL-Lite supports MP2P traffic by allowing MP2P destinations to be reached via DAG roots.

### 8.3.2 Point-to-Multipoint Traffic

RPL-Lite supports P2MP traffic by providing mechanisms that provisions Down routes toward destinations (prefixes, addresses, or multicast groups), and away from roots. These mechanisms in combination with link layer acknowledgements can update routing tables as the underlying DAG topology changes.

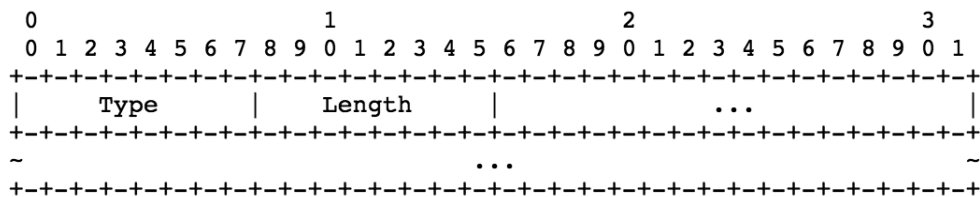
### 8.3.3 Point-to-Point Traffic

RPL-Lite DAGs provide a basic structure for point-to-point (P2P) traffic. For a RPL-Lite network to support P2P traffic, a root must be able to route packets to a destination. Nodes within the network may also have routing tables towards destinations. A packet flows towards a root until it reaches an ancestor that has a known route to the destination. In the most constrained case (when nodes cannot store routes), that common ancestor may be the DAG root. In other cases, it may be a node closer to both the source and destination. RPL-Lite also supports the case where a P2P destination is a ‘one-hop’ neighbor. An optional optimization involves storing ‘two-hop’ neighbors to reduce routing overhead when P2P communication is the dominant traffic pattern within the network.

## 8.4 RPL-Lite Control Messages

RPL-Lite defines control message structures in a slightly different manner than RPL. As specified earlier in this document, RPL’s control messages are ICMPv6 messages identified by a code and composed of a base that depends on that code, as well as a series of options. RPL-Lite also uses ICMPv6 messages. However the specific messages are structured as options to IPv6 Neighbor Discovery Base objects and IPv6 Inverse Neighbor Discovery Base objects. The basic format of the option is shown in Figure 8.1. The ‘type’ field must be set

Figure 8.1: IPv6 Neighbor Discovery and IPv6 Inverse Neighbor Discovery Option Format



to a special ‘RPL’ type, that is yet undefined. In this document, this type is referred to as RPL-TYPE. The ‘length’ field must be set to the length of the entire options field. The rest of the fields can be used for specific RPL fields.

### 8.4.1 RPL-Lite Equivalents of RPL Control Messages

The RPL-Lite equivalents for RPL control messages are detailed in this section.

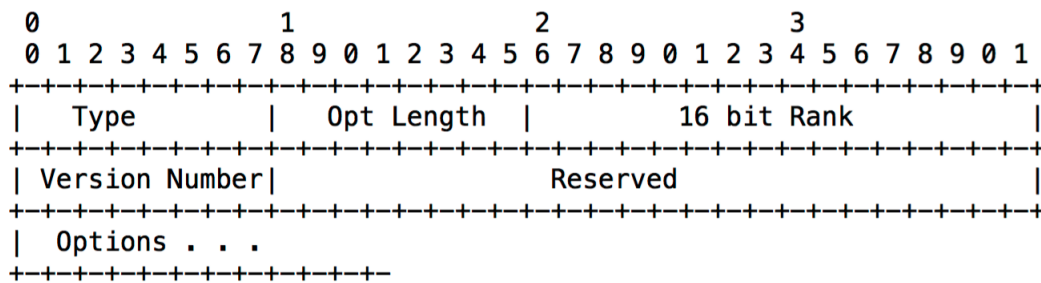
#### 8.4.1.1 DAG Information Solicitation Messages

DIS messages are completely eliminated. Instead, RPL-Lite uses IPv6 Router Solicitations with a RPL option set in order to solicit RPL control information from nodes that are part of the RPL network.

#### 8.4.1.2 DAG Information Advertisement Messages for Upward Routes

A more compact form of DIO messages are added as an option to RAs and perform the equivalent function of RPL DIOs. Mainly, the RPLInstanceID, DODAGID, MOP, DTSN and Grounded/Floating fields are removed from the RPL DIO Base object since they are unused in RPL-Lite. The message format used in RPL-Lite can be found in Figure 8.2. Here

Figure 8.2: IPv6 Neighbor Discovery Option for RPL-Lite DIO



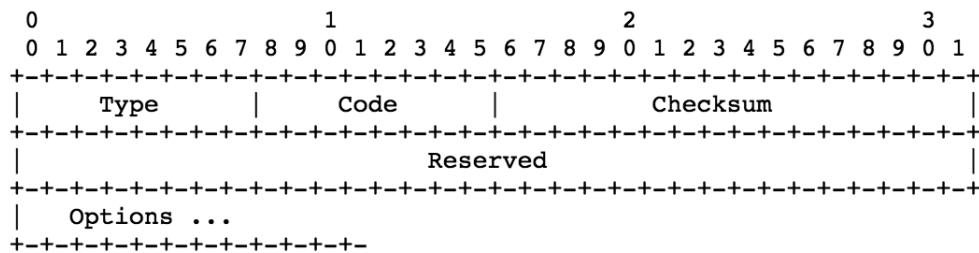
the type field corresponds to RPL-TYPE. The length field must contain a value of 4. The rank is the 16-bit rank of the node that sends the RA, which is calculated according to the OF. In case the node is not part of a RPL-Lite network, then the rank field is set to infinity, which in this case corresponds to 0xFFFF. The Version Number is an 8-bit field indicating the current version of the DAG. In the event of a Global Repair, a DAG root increments the version number and immediately propagates this update through an IPv6 RA with the RPL-Lite DIO option.

#### 8.4.1.3 Destination Advertisement Objects for Downward Routes

DAO messages are also eliminated from RPL-Lite. The IPv6 Inverse Neighbor Discovery Advertisement Message is used instead, to propagate destination information Upward along

the DAG. Irrespective of whether a node is using storing mode or non-storing mode, this message is unicast by the child to the preferred parent. If the preferred parent is using non-storing mode, then the parent forwards this message up along the DAG. If the parent is using storing mode, then in addition to forwarding the packet upwards, it also records a route in its own routing table, towards this destination. The IPv6 Inverse Neighbor Discovery Advertisement Message can be found in Figure 8.3.

Figure 8.3: IPv6 Inverse Neighbor Discovery Advertisement Message



#### 8.4.1.4 Destination Advertisement Acknowledgement Messages

DAO-ACK messages are also eliminated from RPL-Lite. Instead, RPL-Lite uses link-layer acknowledgements to interpret whether a DAO has been received by the parent or not.

#### 8.4.1.5 Consistency Check Messages

Since RPL-Lite does not define any security primitives, Consistency Check (CC) messages are also eliminated from the RPL-Lite specification.

### 8.4.2 RPL-Lite Control Message Options

Similar to RPL, RPL-Lite too allows control messages to carry options. In specific, the DIO option for ICMPv6 RA messages and the IPv6 Inverse Neighbor Discovery messages (similar to DAO in RPL) are allowed to carry protocol-specific options. RPL-Lite uses many of the options defined by RPL, which includes the following:

1. Pad1
2. PadN
3. DAG Metric Container
4. DAG Configuration Option

This option is slightly modified from the RPL DODAG Configuration Option. Many fields that are used in RPL but not in RPL-Lite are removed. In particular, the following fields have been removed:

- Flags
- A
- OCP

The modified version of the Configuration option can be found in Figure 8.4. The included fields have the same meaning as in the RPL specification in RFC6550.

Figure 8.4: DAG Configuration option for RPL-Lite

0								1								2								3															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Type = 0x04								Opt Length = 14								PCS								DIOIntDoubl.								Reserved							
DIOIntMin.								DIORedun.								MaxRankIncrease																							
MinHopRankIncrease								Lifetime Unit																															
Def. Lifetime								Reserved																															

5. RPL Target Option
6. RPL Target Descriptor
7. Transit Information

This option is also slightly altered to always include the Parent Address subfield, since the network may comprise of both storing and non-storing modes.

- ## 8. Solicited Information

This option is modified RPL-Lite because the RPLInstanceID and DODAGIDsub-fields are not a part of RPL-Lite specification. The modified version of the Solicited Information option can be found in Figure 8.5.

Figure 8.5: Solicited Information option for RPL-Lite

```

0          1          2          3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type           |   Opt Length   | Version Number   |   Reserved       |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Options . . .    |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The following options are not used:

1. Route Information: This is not used because RFC6550 specifically states [p50] that this option carries the same information as the IPv6 Neighbor Discovery (ND) RIO as defined in RFC4191 [16].
2. Prefix Information Option: This option is also removed, because it is the same as IPv6 ND PIO, defined in RFC4861.

## 8.5 Upward Routes

This section describes how RPL-Lite discovers and maintains upward routes. It describes the use of IPv6 Router Advertisement messages (RAs) with a special RPL option, which are used to discover and maintain these routes. It specifies how RPL-Lite generates and responds to RAs containing the RPL option. It also describes the IPv6 Router Information Solicitation messages (RSs) which are used to trigger RA transmissions.

Nodes that decide to join a DAG provision at least one DAG parent as a default route towards the DAG root. This default route enables a packet to be forwarded Upward until it eventually hits a common ancestor from which it will be routed Downward to the destination. However, the default route is not always used for upward routes. Traffic flow is spread in a round-robin manner among all members of the candidate parent set for reliability and energy balance. This draws from ideas presented in [76]. If the destination is not in the DAG, then the DAG root may be able to forward the packet using connectivity to the outside of the DAG. If the DAG root is unable to forward the packet outside, then it has to drop it. A node that receives an RA message with RPL option from a parent must update the Rank field prior to forwarding it to its children.

### 8.5.1 Upward Route Discovery and Maintenance

Upward route discovery allows a node to join a DAG by discovering neighbors that are members of the DAG and identifying a set of parents. This process is very similar to RPL's Upward route discovery and maintenance. The exact policies for selecting neighbors and parents is implementation dependent and driven by the OF. This section specifies the set of rules those policies must follow for interoperability.

#### 8.5.1.1 Neighbors and Parents

RPL-Lite processes four logical sets of link-local nodes.

1. Candidate Neighbor Set: The set of nodes that can be reached via link-local multicast.
2. Parent Set: The restricted subset of candidate neighbor set that has a rank strictly lesser than the node's rank.



3. Preferred Parent Set: Member (or members) of the parent set that are the preferred next hop in Upward routes.
4. RPL-Lite optionally includes two-hop neighbors for applications that have a predominance of P2P flows.

A node must not advertise a rank less than or equal to any member of its parent set.

#### 8.5.1.2 DAG Roots

A DAG root advertises a rank of `ROOT_RANK`. In a deployment that uses non-LLN links to federate a number of LLN roots, it is possible to run RPL-Lite over those non-RPL links and use one router as a “backbone root”. The backbone root is the virtual root of the DAG and exposes a Rank of `BASE_RANK` over the backbone. All the LLN roots that are parented to that backbone root, including the backbone root if it also serves as the LLN root itself, expose a Rank of `ROOT_RANK` to the LLN. These virtual roots are part of the same DAG and coordinate other DAG parameters with the virtual root over the backbone. The method of coordination is out of scope for this specification.

#### 8.5.1.3 Poisoning

The rules for poisoning in RPL-Lite are the same as RPL, and are summarized below:

1. A node poisons routes by advertising a Rank of `INFINITE_RANK`.
2. A node must not have any nodes with a Rank of `INFINITE_RANK` in its parent set.

A node that is unable to retain a non-empty parent set must detach from the DAG and should immediately advertise this new situation in an RA with Rank set to `INFINITE_RANK`. If a node receives an RA from one of its DAG parents indicating that the parent has left the DAG, then that node should also detach and advertise a Rank of `INFINITE_RANK`. The parent that left the DAG must be removed from the node’s parent set.

### 8.5.2 Node Advertisement Transmission

Nodes transmit RA messages with RPL option using a Trickle timer (RFC6206) in the same manner as RPL. An RA from a sender with a lesser DAGRank that causes no changes to the recipient’s parent set, preferred parent, Rank or Version Number should be considered consistent with respect to the Trickle timer. Inconsistent events include, but are not limited to forwarding inconsistencies and a node newly joining a DAG.

### 8.5.3 Operation as a Leaf Node

A node may attach to a DAG as a leaf node only in situations when a node does not understand or does not support (policy) the OF. The node may either join the DAG as a leaf node or may not join the DODAG. A leaf node does not extend DAG connectivity; however, in some cases, the leaf node may still need to transmit RAs on occasion, in particular, when the leaf node may not have always been acting as a leaf node and an inconsistency is detected. A node operating as a leaf node must obey the following rules:

1. Its RAs must advertise a DAGRank of INFINITE\_RANK.
2. It must suppress RA transmission, unless the RA transmission has been triggered due to detection of inconsistency when a packet is being forwarded or in response to a unicast RS message, in which case the RA transmission must not be suppressed.
3. It may transmit unicast INDA messages and multicast INDAs to the ‘one-hop’ or ‘two-hop’ neighborhood.

## 8.6 Downward Routes

This section describes how RPL-Lite discovers and maintains downward routes. RPL-Lite constructs and maintains downward routes with IPv6 Inverse Neighbor Discovery Advertisement messages (INDAs). Downward routes support P2MP flows, from the DAG roots towards the leaves. Downward routes also support P2P flows. P2P messages can flow towards a DAG root or a common ancestor through an Upward route, then away from the DAG root to a destination through a Downward route. This section describes the three modes that RPL-Lite may choose for maintaining Downward routes.

### 8.6.1 Destination Advertisement Parents

To establish Downward routes, nodes send INDA messages Upward. The next-hop destinations of these INDA messages are called “DA parents”. The collection of a node’s DA parents is called the “DA parent set”. The selection of DA parents is implementation and Objective Function specific.

1. A node MAY send INDA messages using the all-RPL-nodes multicast address, which is an optimization to provision one-hop routing.
2. A node’s DA parent set must be a subset of its DAG parent set.
3. The IPv6 source and destination addresses of the INDA message must be a unique-local or a global address.

### 8.6.2 Downward Route Discovery and Maintenance

Destination Advertisement may be configured to be entirely disabled, or operate in either a Storing, Non-Storing or Hybrid mode, as configured on each individual node. If a node receives a data packet, the following actions can be taken:

1. If the packet is moving in the ‘Down’ direction and has an attached source route, then the packet must be forwarded along that route.
2. If the packet is moving in the ‘Down’ direction and the destination has an entry in the node’s local routing table and no source route is found, then the packet may be forwarded along that route.
3. If the packet is traveling in the ‘Down’ direction and there is no source route or entry in the node’s local routing table for that destination then the packet must be dropped. A ‘Destination Unreachable’ ICMP message may be sent out.
4. If the packet was traveling in the ‘Up’ direction and no entry for the packet’s destination was found in the node’s local routing table, then the packet is forwarded to the DAG parent.
5. If the packet was traveling in the ‘Up’ direction and an entry for the packet’s destination was found in the node’s local routing table, then the packet direction is changed to indicate ‘Down’ direction, and the node is forwarded in the ‘Down’ direction towards the destination.
6. If the packet is traveling in the ‘Up’ direction and the node does not find an entry for the destination in its local routing table and also does not have any DAG parents, then the packet should be dropped. An ICMP error message may be sent out.

All nodes that operate under storing mode store routing table entries for destinations learned from INDAs. If a packet reaches the DAG root but the root failed to store source routing information for that destination, then that destination is presumed to be unreachable and the packet is dropped. An ICMP ‘Host Unreachable’ message may be sent to the sender. A node that sends a unicast INDA message but does not receive a link-layer acknowledgement in response may reschedule the INDA message transmission for another attempt, up until an implementation-specific number of retries. Nodes that operate under non-storing mode must forward all incoming INDAs towards the DAG root. Nodes that operate in storing mode may suppress INDA transmissions in the Upwards direction if the incoming INDA message contains pre-existing routing information.

### 8.6.3 INDA Transmission Scheduling

Because INDAs flow Upward, receiving a unicast INDA can trigger sending a unicast INDA to a DA parent. On receiving a unicast INDA message with updated information, a node should

send a INDA. It should not send this INDA message immediately. It should delay sending the INDA message in order to aggregate INDA information from other nodes for which it is a DA parent. A node should delay sending an INDA message with a timer DelayDA. Receiving an INDA message starts the DelayDA timer. INDA messages received while the DelayDA timer is active do not reset the timer. When the DelayDA timer expires, the node sends an INDA. When a node adds another node to its DA parent set, it should schedule an INDA message transmission. DelayDA's value and calculation is implementation dependent. A default value of `DEFAULT_DA_DELAY` is configured at boot time is equivalent to the `DEFAULT_DAO_DELAY` value specified in RFC6550. INDAs can be triggered by sending Inverse Neighbor Discovery Solicitation (INDA) messages, as specified in RFC3122.

### 8.6.4 Downward Routing Mechanism

In all modes of operation, INDAs are used to report a node's DA parents to the DAG root. The DA Parent address is always included in the Transit Information Option, since the network may comprise of nodes operating under both storing and non-storing modes. The DAG root as well as non-root nodes operating in storing mode can piece together a Downward route to a node by using DA parent sets from each node in the route. Nodes pack INDAs by sending a single INDA message with multiple RPL Target options. Each RPL Target option has its own, immediately following, Transit Information options.

In the fully Non-Storing mode, RPL-Lite routes messages Downward using IP source routing. In the fully Storing mode, RPL-Lite routes messages Downward by the IPv6 destination address. In the Hybrid mode, a combination of these is used to route packets Downwards.

The following actions take place upon receipt of an INDA message:

1. If the node is operating under non-storing mode, then no changes are made. The message is directly forwarded to the DA Parent.
2. If the node operates under storing mode, then it first checks whether the incoming INDA message causes any change to the set of prefixes advertised by the node. If yes, then the node propagates this INDA message Upward along the DAG, after making the required changes in its local routing table. Additionally, it also sends out its own INDA message advertising the new prefixes.

Path Control is implemented in the same manner as RPL, and is not separately explained here.

## 8.7 Loop Avoidance and Detection

Loops may form for a number of reasons, e.g., control packet loss. RPL-Lite includes a reactive loop detection technique that protects from meltdown and triggers repair of broken

paths. RPL-Lite loop detection uses RPL Packet Information that is transported within the data packets, relying on an external mechanism such as RFC6553 that places the RPL Packet Information in an IPv6 Hop-by-Hop option header. The content of RPL Packet Information is almost the same as that defined for RPL. However, the RPLInstanceID field is removed. All other fields (Down, Rank-Error, Forwarding-Error, SenderRank) are retained and hold the same functionality.

### 8.7.1 DAG Inconsistency and Loop Detection

Similar to RPL's operation, the DAG is inconsistent if the direction of a packet does not match the Rank relationship. A receiver detects an inconsistency if it receives a packet with either:

- the 'O' bit set (to Down) from a node of a higher Rank
- the 'O' bit cleared (for Up) from a node of a lower Rank

One inconsistency along the path is not considered a critical error and the packet may continue. However, a second detection along the path of the same packet should not occur and the packet must be dropped. This process is controlled by the Rank-Error bit associated with the packet.

### 8.7.2 DA Inconsistency Detection and Recovery

A DA inconsistency happens when a router has a Downward route that was previously learned from an INDA message via a child, but that Downward route is not longer valid in the child. With DA inconsistency loop recovery, a packet can be used to recursively explore and clean up the obsolete DA states along a sub-DAG. In a general manner, a packet that goes Down should never go Up again. If DA inconsistency loop recovery is applied, then the router should send the packet back to the parent that passed it with the Forwarding-Error 'F' bit set and the 'O' bit left untouched. Otherwise, the router must silently discard the packet. Upon receiving a packet with a Forwarding-Error bit set, the node must remove the routing states that caused forwarding to that neighbor, clear the Forwarding-Error bit, and attempt to send the packet again. The packet may be sent to an alternate neighbor, after the expiration of a user-configurable implementation-specific timer. If that alternate neighbor still has an inconsistent DA state via this node, the process will recurse. This node will set the Forwarding-Error 'F' bit, and the routing state in the alternate neighbor will be cleaned up as well.

### 8.7.3 Global Repairs

A Global Repair is a network reboot operation where all nodes are detached from the DAG by incrementing the DAG Version Number. It can be triggered to recover from count to

infinity scenarios. Only a DAG root may trigger a global repair. In case there are multiple roots, the root that triggers the repair must propagate the new version number to the other roots through the backbone transit link prior to disseminating this update to non-root nodes. Since the Version Number is an 8-bit value, the overflow is handled in a lollipop fashion [54]. Details of this mechanism are described in RFC6550. When a Global Repair takes place, a node detaches itself from the DAG and discards all routing state associated with the DAG. All known routes are discarded and the node advertises a rank of `INFINITE_RANK`. This node now listens for new RA messages and can join a RPL-Lite network as if it were a new node. Previous rank relationships are invalid. An exhaustive list of actions to be taken upon triggering a Global Repair are out of scope for this specification. If an implementation does not support Global Repair, then the root always configures the network with a DAG Version Number of 0.

#### 8.7.4 Local Repairs

A Local Repair maybe triggered when a non-root node encounters a DAG loop through Rank-based data path validation mechanisms. These mechanisms are described in detail in RFC6550. A Local Repair may also be triggered by a root node if a specific implementation prefers to do so over triggering a Global Repair. In the event of a Local Repair, the node that triggers the repair detaches itself from the DAG by advertising a Rank of `INFINITE_RANK`. It also sends route poisoning updates to one-hop neighbors. If two-hop neighbors are stored, then poisoning updates are sent to them as well. The node may re-attach to the same DAG if it receives a new RA message, as if it were joining the DAG as a new node. All previous DAG specific data and routing state is discarded. An exhaustive list of actions to be taken upon triggering a Local Repair are out of scope for this specification.

# Chapter 9

## Conclusion

This thesis studies and analyses the RPL Routing Standard as defined in RFC6550. RPL is the emerging routing standard for low-power and lossy networks, and was designed by the IETF ROLL Working Group to cater to the unique routing challenges posed by LLNs. RPL was designed in order to standardize the various independent and non-interoperable efforts towards creating a routing protocol that would suit a wide variety of LLN scenarios. A detailed specification of this routing standard can be found in the IETF document RFC6550 [77].

We summarize the RPL routing standard in Chapter 3. A number of research works have been focused on evaluating this routing protocol against a number of metrics, as we detail in Chapter 4. RPL has also been compared against similar routing protocols such as Collection Tree Protocol (CTP) and LOAD, which we have described in further detail in Chapter 4. Many research groups have attempted to implement and use RPL for specific WSN scenarios. We discuss open-source and industrial implementations of the RPL routing standard in Chapter 5. These highlight a few issues with the current routing standard, such as implementation choices causing non-interoperability. Many research works have been targeted towards improving RPL to make it more compliant for the WSN scenario. Such works include developing new routing metrics, supporting broadcast and leveraging the DAG structure by including multiple routes for redundancy. We discuss these works in Chapter 6.

In Chapter 7, we summarize the drawbacks of the RPL routing standard by dividing the issues into distinct categories. These categories include redundant and unused features of RPL, known issues, applications that RPL does not cater to and beneficial features that were excluded from the specification. Finally in Chapter 8, we provide a comprehensive specification of RPL-Lite, a new routing standard that addresses many of RPL's flaws and provides a simpler and easier to implement routing protocol. RPL-Lite is designed to be lightweight and easily deployable on real hardware.

As a conclusion, we have proposed an idea for a better routing protocol for LLNs in

this thesis. In order for RPL-Lite to become a routing standard, it would require further theoretical and experimental analysis, technical development and implementation studies, which is future work. We believe that the conceptualization and design overview of RPL-Lite presented in this thesis is a good starting point for developing a new routing standard that is suited for LLNs and WSNs.



# Bibliography

- [1] N. Accettura et al. “Performance Analysis of the RPL Routing Protocol”. In: *IEEE International Conference on Mechatronics* (2011).
- [2] O. Banimelhem and S. Khasawneh. “Grid-based multi-path with congestion avoidance routing (GMCAR) protocol for wireless sensor networks”. In: *International Conference on Telecommunications (ICT)* (2009).
- [3] Athanassios Boulis. “Castalia: revealing pitfalls in designing distributed algorithms in WSN”. In: *Proceedings of the 5th international conference on Embedded networked sensor systems (SenSys)* (2007).
- [4] A. Brandt, J. Buron, and G. Porcu. “Home Automation Routing Requirements in Low-Power and Lossy Networks”. In: *Internet Engineering Task Force (IETF)* 5826 (2010).
- [5] Lin-Huang Chang et al. “Energy-Efficient Oriented Routing Algorithm in Wireless Sensor Networks”. In: *IEEE International Conference on Systems, Man and Cybernetics* (2013).
- [6] “Cisco’s Implementation of RPL”. In: <http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/rpl/configuration/15-mt/rpl-15-mt-book.pdf> ().
- [7] T. Clausen et al. “The Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)”. In: *Internet Engineering Task Force (IETF) Draft* (2016).
- [8] R. Coltun et al. “OSPF for IPv6”. In: *Internet Engineering Task Force (IETF) RFC5340* (2008).
- [9] Marco Conti, Enrico Gregori, and Gaia Maselli. “Reliable and efficient forwarding in ad hoc networks”. In: *Elsevier Ad Hoc Networks* (2006).
- [10] “Contiki OS”. In: <http://www.contiki-os.org/> ().
- [11] “ContikiRPL”. In: <https://github.com/contiki-os/contiki/tree/master/core/net/rpl> ().
- [12] Editor D. Oran. “OSI IS-IS Intra-domain Routing Protocol”. In: *Internet Engineering Task Force (IETF) RFC1142* (1990).

- [13] Stephen Dawson-Haggerty, Arsalan Tavakoli, and David Culler. “Hydro: A Hybrid Routing Protocol for Low-Power and Lossy Networks”. In: *Smart Grid Communications (SmartGridComm)* (2010).
- [14] Manjunath Doddavenkatappa, Mun Choon Chan, and Ananda A.L. “Indriya: A Low-Cost, 3D Wireless Sensor Network Testbed”. In: *TRIDENTCOM* (2011).
- [15] M. Dohler et al. “Routing Requirements for Urban Low-Power and Lossy Networks”. In: *Internet Engineering Task Force (IETF)* 5548 (2009).
- [16] R. Draves and D. Thaler. “Default Router Preferences and More-Specific Routes”. In: *Internet Engineering Task Force (IETF)* RFC4191 (2005).
- [17] Simon Duquennoy, Olaf Landsiedel, and Thiemo Voigt. “Let the Tree Bloom: Scalable Opportunistic Routing with ORPL”. In: *SenSys* (2013).
- [18] Deborah Estrin, Ramesh Govindan, and John Heidemann. “Embedding the Internet: Introduction”. In: *Communications of the ACM* 43.5 (May 2000). (Special issue guest editors), pp. 39–41. DOI: <http://dx.doi.org/10.1145/332833.332836>. URL: <http://www.isi.edu/%7ejohnh/PAPERS/Estrin00a.html>.
- [19] Antoine Fraboulet, Guillaume Chelius, and Eric Fleury. “Worldsens: Development and Prototyping Tools for Application Specific Wireless Sensors Networks”. In: *International Symposium on Information Processing in Sensor Networks(IPSIN)* (2007).
- [20] Olfa Gaddour et al. “OF-FL: QoS-Aware Fuzzy Logic Objective Function for the RPL Routing Protocol”. In: *12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)* (2014).
- [21] O. Gnawali. “The Minimum Rank with Hysteresis Objective Function”. In: *Internet Engineering Task Force (IETF)* 6719 (2012).
- [22] O. Gnawali et al. “Collection Tree Protocol”. In: *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems* (2009).
- [23] Pietro Gonizzi, Riccardo Monica, and Gianluigi Ferrari. “Design and Evaluation of a Delay-Efficient RPL Routing Metric”. In: *Wireless Communications and Mobile Computing Conference (IWCMC)* (2013).
- [24] Ulrich Herberg and Thomas Clausen. “A Comparative Performance Study of the Routing Protocols LOAD and RPL with Bi-Directional Traffic in Low-power and Lossy Networks (LLN)”. In: *Proceedings of the 8th ACM Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks(PE-WASUN '11)* (2011).
- [25] Ulrich Herberg and Thomas Clausen. “Study of Multipoint-to-Point and Broadcast Traffic Performance in the IPv6 Routing Protocol for Low Power and Lossy Networks”. In: *Journal of Ambient Intelligence and Humanized Computing* 2 (2011).
- [26] Ki-Sup Hong and Lynn Choi. “DAG-based Multipath Routing for Mobile Sensor Networks”. In: *International Conference on ICT Convergence (ICTC)* (2011).

- [27] “IEEE Std. 802.15.4-2003”. In: *Computer Society, IEEE* (2003).
- [28] Oana Iova, Fabrice Theoleyre, and Thomas Noel. “Improving the Network Lifetime with Energy-Balancing Routing: Application to RPL”. In: *Ad Hoc Networks (Elsevier)* 29 (2015).
- [29] Oana Iova, Fabrice Theoleyre, and Thomas Noel. “Stability and Efficiency of RPL under Realistic Conditions in Wireless Sensor Networks”. In: *IEEE 24th International Symposium on Personal, Indoor and Mobile Radio Communications: Mobile and Wireless Networks* (2013).
- [30] Oana Iova, Fabrice Theoleyre, and Thomas Noel. “Using Multiparent Routing in RPL to Increase the Stability and the Lifetime of the Network”. In: *Elsevier Ad Hoc Networks* (2015).
- [31] Ed. J. Hui and P. Thubert. “Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks”. In: *Internet Engineering Task Force (IETF)* 4944 (2011).
- [32] D. Johnson, Y. Hu, and D. Maltz. “The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4”. In: *IETF* 4728 (2007).
- [33] Ed. JP. Vasseur et al. “Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks”. In: *Internet Engineering Task Force (IETF)* 6551 (2012).
- [34] Panagiotis Karkazis et al. “Design of Primary and Composite Routing Metrics for RPL-Compliant Wireless Sensor Networks”. In: *IEEE International Conference on Telecommunications and Multimedia (TEMU)* (2012).
- [35] P. Karkazis et al. “RPL modeling in J-Sim platform”. In: *Ninth International Conference on Networked Sensing Systems (INSS)* (2012).
- [36] JeongGil Ko et al. “Connecting Low-Power and Lossy Networks to the Internet”. In: *IEEE Communications Magazine* (2011).
- [37] JeongGil Ko et al. “Evaluating the Performance of RPL and 6LoWPAN in TinyOS”. In: *IPSN* (2011).
- [38] N. Kushalnagar, G. Montenegro, and C. Scumacher. “6LoWPANs: Overview, Assumptions, Problem Statement, and Goals”. In: *Internet Engineering Task Force (IETF)* RFC4919 (2007).
- [39] Reddeppa L. Reddy and S. V. Raghavan. “SMORT: Scalable multipath on-demand routing for mobile ad hoc networks”. In: *Elsevier Ad Hoc Networks* (2007).
- [40] Olaf Landsiedel et al. “Low Power, Low Delay: Opportunistic Routing meets Duty Cycling”. In: *IPSN* (2012).
- [41] P. Levis, A. Tavakoli, and S. Dawson-Haggerty. “Overview of Existing Routing Protocols for Low Power and Lossy Networks”. In: *IETF ROLL Working Group* (2009).
- [42] Philip Levis et al. “Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks”. In: *NSDI* (2004).

- [43] P. Levis et al. “The Trickle Algorithm”. In: *Internet Engineering Task Force (IETF)* 6206 (2011).
- [44] Ed Martocci et al. “Building Automation Routing Requirements in Low-Power and Lossy Networks”. In: *Internet Engineering Task Force (IETF)* 5867 (2010).
- [45] T. Narten et al. “Neighbor Discovery for IP version 6 (IPv6)”. In: *Internet Engineering Task Force (IETF)* RFC4861 (2007).
- [46] “NED Language Overview”. In: <http://www.ewh.ieee.org/soc/es/Nov1999/18/ned.htm> ().
- [47] Wilbert Nestor Michael C. Tiglao, China Kimberly Paige S. Yu, and Carl C. Dizon. “Performance Analysis of RPL in an Ambient Energy Harvesting Wireless Sensor Network”. In: *The 4th International Conference on Internet Applications, Protocols and Services (NETAPPS)* (2015).
- [48] Juuso Nurmio, Ethiopia Nigussie, and Christian Poellabauer. “Equalizing Energy Distribution in Sensor Nodes through Optimization of RPL”. In: *IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing* (2015).
- [49] Fredrik Osterlind et al. “Cross-Level Sensor Network Simulation with COOJA”. In: *Proceedings of the 31st IEEE Conference on Local Computer Networks* (2006).
- [50] Ed P. Thubert. “Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)”. In: *Internet Engineering Task Force (IETF)* 6552 (2012).
- [51] Bogdan Pavkovic, Fabrice Theoleyre, and Andrzej Duda. “Multipath Opportunistic RPL Routing over IEEE 802.15.4”. In: *14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (2011).
- [52] Bogdan Pavkovic et al. “Efficient topology construction for RPL over IEEE 802.15.4 in wireless sensor networks”. In: *Elsevier Ad Hoc Networks* (2014).
- [53] C. Perkins, E. Belding-Royer, and S. Das. “Ad hoc On-Demand Distance Vector (AODV) Routing”. In: *Internet Engineering Task Force (IETF)* RFC3561 (2003).
- [54] R. Perlman. “Fault-Tolerant Broadcasting of Routing Information.” In: *Computer Networks* 7 (1983).
- [55] K. Pister et al. “Industrial Routing Requirements in Low-Power and Lossy Networks”. In: *Internet Engineering Task Force (IETF)* 5673 (2009).
- [56] Mamoun Qasem et al. “Performance Evaluation of RPL Objective Functions”. In: *IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing* (2015).
- [57] Marjan Radi et al. “Multipath Routing in Wireless Sensor Networks: Survey and Research Challenges”. In: *Sensors* (2012).

- [58] Ion Emilian Radoi, Aditi Shenoy, and D. K. Arvind. "Evaluation of Routing Protocols for Internet-Enabled Wireless Sensor Networks". In: *The Eighth International Conference on Wireless and Mobile Communications (ICWMC)* (2012).
- [59] E. Rescorla. "Writing Protocol Models". In: *Internet Engineering Task Force (IETF) RFC4101* (2005).
- [60] "RIOT OS". In: [http://riot-os.org/api/group\\_\\_net\\_\\_gnrc\\_\\_rpl.html](http://riot-os.org/api/group__net__gnrc__rpl.html) ().
- [61] "RiotRPL". In: <https://github.com/RIOT-OS/RIOT/tree/master/sys/net/gnrc/routing/rpl> ().
- [62] "Routing Over Low-Power and Lossy Networks (ROLL) Working Group(ROLL-WG)". In: <https://datatracker.ietf.org/wg/roll/documents/> (2008).
- [63] T. Shu, M. Krunz, and S. Liu. "Secure Data Collection in Wireless Sensor Networks Using Randomized Dispersive Routes". In: *IEEE Transactions on Mobile Computing* 9 (2014).
- [64] "SimpleRPL". In: <https://github.com/tcheneau/simpleRPL> ().
- [65] Neha Singh, Rajeshwar Lal Dua, and Vinita Mathur. "Network Simulator NS2-2.35". In: *International Journal of Advanced Research in Computer Science and Software Engineering* 2 (2012).
- [66] Gurram Srinivasarao and Mangamma Dharavatu. "An Improved PDR (Packet Delivery ratio) Defined Routing Protocol in Low-power and Lossy Networks". In: *International Journal for Development of Computer Science and Technology (IJDCST)* (2015).
- [67] Ed. T. Clausen and Ed. P. Jacquet. "Optimized Link State Routing Protocol (OLSR)". In: *Internet Engineering Task Force (IETF) RFC3626* (2003).
- [68] Weisheng Tang et al. "Toward Improved RPL: A Congestion Avoidance Multipath Routing Protocol with Time Factor for Wireless Sensor Networks". In: *Journal of Sensors* 2016 (2015).
- [69] Nguyen Thanh Long et al. "Comparative Performance Study of RPL in Wireless Sensor Networks". In: *IEEE* (2012).
- [70] "The Network Simulator - ns-2". In: <http://nsnam.sourceforge.net/wiki/> (1989).
- [71] "TinyRPL". In: <https://github.com/tinyos/tinyos-main/tree/master/tos/lib/net/rpl> ().
- [72] J. Tripathi, J. C. de Oliveira, and J. P. Vasseur. "A Performance Evaluation Study of RPL: Routing Protocol for Low Power and Lossy Networks". In: *44th Annual Conference on Information Sciences and Systems (CISS)* (2010).
- [73] Tsvetko Tsvetkov. "RPL: IPv6 Routing Protocol for Low Power and Lossy Networks". In: *Network Architectures and Services* (2011).
- [74] A. Varga. "The omnet++ discrete event simulation systems". In: *European Simulation Multiconference (ESM'2001)* (2001).

- [75] Malisa Vucinic, Bernard Tourancheau, and Andrzej Duda. “Performance Comparison of the RPL and LOADng Routing Protocols in a Home Automation Scenario”. In: *IEEE Wireless Communications and Networking Conference (WCNC): NETWORKS* (2013).
- [76] Jonathan W. Hui and David Culler. “IP is dead, long live IP for wireless sensor networks”. In: *Proceedings of the 6th ACM conference on Embedded network sensor systems* (2008).
- [77] Ed Winter et al. “RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks”. In: *Internet Engineering Task Force (IETF)* 6550 (2012).
- [78] Wei Xiao et al. “An Optimization of the Object Function for Routing Protocol of Low-Power and Lossy Networks”. In: *2nd International Conference on Systems and Informatics (ICSAI 2014)* (2014).
- [79] Jiazi Yi, Thomas Clausen, and Yuichi Igarashi. “Evaluation of Routing Protocol for Low Power and Lossy Networks: LOADng and RPL”. In: *IEEE Conference on Wireless Sensors (ICWiSe2013)* (2013).
- [80] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. “Wireless sensor network survey”. In: *Elsevier Computer Networks* (2008).

## List of Abbreviations

**ACK** Acknowledgement

**AMI** Advanced Metering Infrastructure

**AODV** Ad Hoc on Demand Vector

**AVG\_DEL** Averaged Delay Metric

**BFD** Bidirectional Forwarding Detection

**BLIP** Berkeley Low-Power IP Stack

**BSD** Berkeley Software Detection

**CA-RPL** Congestion Avoidance Multipath Routing Algorithm

**CBR** Constant Bit-Rate

**CC** Consistency Check

**CoAP** Constrained Application Protocol

**CPU** Central Processing Unit

**CTP** Collection Tree Protocol

**DA** Destination Advertisement

**DAG** Directed Acyclic Graph

**DAO** Destination Advertisement Object

**DIO** DODAG Information Object

**DMR** DAG-based Multipath Routing Protocol

**DODAG** Destination-Oriented Directed Acyclic Graph

**DSR** Dynamic Source Routing Protocol

**DTSN** Destination Trigger Sequence Number

**EDC** Expected Duty Cycle

**ELT** Expected Lifetime Metric

**ETX** Expected Transmission Count

**HC** Hop Count

**HVAC** Heating Ventilation and Air Conditioning

**ICMP** Internet Control Message Protocol

**ICMPv6** ICMP for IPv6

**IEEE** Institute of Electrical and Electronics Engineers

**IETF** Internet Engineering Task Force

**INDA** IPv6 Inverse Neighbor Discovery Advertisement

**INDA** IPv6 Inverse Neighbor Discovery Solicitation

**IoT** Internet of Things

**IP** Internet Protocol

**IPv6** Internet Protocol version 6

**IS-IS** Intermediate System to Intermediate System Routing Protocol

**L2** Layer 2

**LL** Link Latency

**LLN** Low-Power and Lossy Network

**LOAD** Lightweight on-demand ad hoc distance-vector routing protocol

**LOADng** Lightweight on-demand ad hoc distance-vector routing protocol - Next Generation

**LQA-RPL** Link Quality Aware RPL

**LQI** Link Quality Indicator

**LQL** Link Quality Level

**MAC** Media Access Control

**MIN-HOP** Minimum Number of Hops

**MOP** Mode of Operation

**MP2P** Multi-Point to Point

**MPRF** Multiparent Relay Flooding



**MRHOF** Minimum Rank with Hysteresis Objective Function

**MTU** Maximum Transmission Unit

**NAM** Network AniMator

**ND** Neighbor Discovery

**NED** NEtwork Description

**NESC** Network Embedded Systems C

**NS** Network Simulator

**NS2** Network Simulator 2

**NUD** Neighbor Unreachability Detection

**OCP** Objective Code Point

**OF** Objective Function

**OF0** Objective Function Zero

**OLSR** Optimized Link State Routing

**OMNET** Object-oriented Modular Discrete Event Network Simulation Framework

**ORPL** Opportunistic Routing Protocol for LLNs

**OSPF** Open Shortest Path First

**P2MP** Point to Multi-Point

**P2P** Point to Point

**PDR** Packet Delay Rate

**PER** Packet Error Rate

**PFI** Path Forwarding Indication

**PIO** Packet Information Option

**PLC** Power Line Communication

**PPF** Preferred Parent Flooding

**PPMPRF** Preferred Parent MPR Flooding

**PRN** Number of Packets Received per Unit Time

**PRR** Packet Reception Rate

**RA** Router Advertisement

**RE** Remaining Energy

**RFC** Request for Comments

**RHT** Route Hold Time

**RIO** Route Information Option

**RIOT** Real-time OS for IoT

**ROLL** Routing over Low-Power and Lossy Networks

**ROLL-WG** ROLL Working Group

**RPL** Routing Protocol for Low-Power and Lossy Networks

**RPL-Lite** Lightweight version of RPL

**RS** Router Solicitation

**RX** Packet Reception Ratio

**UDP** User Datagram Protocol

**WPAN** Wireless Personal Area Network

**WSN** Wireless Sensor Network