On Design, Evaluation and Enhancement of IP-Based Routing Solutions for Low Power and Lossy Networks

Joydeep Tripathi

Electrical & Computer Engineering Dept

Drexel University

Advisor: Jaudelice C. de Oliveira

Thesis submitted for the partial fulfillment for the degree of

PhilosophiæDoctor (PhD)

2014 June

This thesis is dedicated to

To My parents, Bidyanath and Anindita Tripathi for their love and unconditional support,

To my sisters, Rinku and Mou for their encouragement,

To my wife, Priyanka for all her love and patience,

And to my thesis advisor, Dr. Jaudelice C. de Oliveira for her vision, encouragement, support and guidance.

Acknowledgements

My education at Drexel University has been a journey to remember, and I can not thank enough to the number of people who contributed to the culmination of my degree in so many ways.

First, I would like to thank my advisor, Dr. Jaudelice C. de Oliveira, who has been more than a guide to me in this foreign country - a guardian, and a friend. At the same time, I would like to express my gratitude towards Dr. Jean-Philippe Vasseur, who has been our collaborator at Cisco Systems. They have continuously mentored through the ups and downs, and taught me how to apply theoretical knowledge in real-life perspective. They have provided me with the vision that would undoubtedly help me to make a better decision, always.

I am highly grateful to my committee chair, Dr. Steven Weber for his priceless inputs to my works, for the group discussions, and for the fact, that his suggestion improved the quality of this thesis. I thank Dr. Nagarajan Kandasamy and Dr. Spiros Mancoridis for their valuable comments and suggestions. To be honest, no word should be enough to thank the committee member that I luckily had the privilege to work under. I feel lucky that Drexel University has provided me with the opportunity to interact with the best professors I ever came across. I can not thank enough Dr. Harish Sethu for the enormous insight he provided through his classroom lectures. Dr. John Walsh, from whom I learned how to bring enthusiasm into teaching what can be considered most difficult of subjects.

I thank the terrific staffs at the ECE department office to make my life so much easier. I can not imagine how I would survive without the assistance of Chad Morris, Tanita Chapelle, Kathy Bryant, Phyllis D. Watson, Taif Choudhury and Wayne Hill. I will always remember the greatest of friends and sharpest of minds I have came across as my lab members - Sukrit Dasgupta, Zhen Zhao, Anbu Elancheziyan, Fei Bao and Peter Thai. I would miss all the technical and non-technical discussions I had with them, over the blackboard or over a glass of drink.

Last but never the least, I would like to thank my whole family for this achievement. If not for my parents, I could never be the person I am today, academically or personally. I would thank them and my sisters for their continuous support and encouragement in toughest of times. Finally, I acknowledge the support I have received from my wife, Priyanka and thank her for her patience during my long journey of doctoral studies.

Contents

List of Figures **ix**

List of Tables **xiii**

Glossary **xv**

1 Introduction **1**

1.1 Introduction to LLNs: Low-Power Lossy Networks 1

1.2 From Proprietary WSN Solutions to Standardization 4

1.2.1 Using application gateways 6

1.2.2 Using standard protocol suits 6

1.3 Unique Routing Challenges in LLNs 8

1.3.1 Urban LLN routing requirements 9

1.3.2 Industrial automation routing requirements 10

1.3.3 Home automation routing requirements 12

1.3.4 Building automation routing requirements 13

1.4 Contributions of This Thesis 14

2 RPL: Routing Protocol for LLNs **17**

2.1 Overview of RPL 17

2.1.1 RPL - Forming and maintaining the DAG 18

2.1.2 RPL - Routing through the DAG 21

2.2 Summary 22

3 RPL: Performance Evaluation **23**

3.1 Methodology and Simulation Setup 24

3.1.1 Common assumptions 28

3.2 Performance Evaluation in Small Network 28

3.2.1 Path quality 28

3.2.2 Routing table size . . . 32
3.2.3 Delay bound for P2P routing . . . 34
3.2.4 Control packet overhead . . . 35
3.2.5 Loss of connectivity . . . 39
3.3 RPL in a Building Automation Routing Scenario . . . 44
3.3.1 Path quality . . . 46
3.3.2 Delay . . . 48
3.4 RPL in a Large-Scale Network . . . 51
3.4.1 Path quality . . . 51
3.4.2 Delay . . . 54
3.4.3 Control packet overhead . . . 55
3.5 Scaling Property, Parameter Configuration and Routing Stability . . . 57
3.6 Summary . . . 63

4 Proactive versus Reactive in LLNs 66
4.1 LOADng: LLN On-demand Ad-hoc Distance Vector Routing Protocol - Next Generation . . . 71
4.2 Theoretical Comparison: A Balanced Aggregation/ Dissemination Tree Model . . . 73
4.3 Simulation Setup . . . 79
4.3.1 Traffic traces . . . 79
4.3.2 RPL and LOADng parameters . . . 80
4.4 Performance Results for Smart Grid Traffic . . . 80
4.4.1 Control overhead and ability to support P2MP or MP2P traffic . . . 81
4.4.2 Dependency of control overhead on application module . . . 87
4.4.3 Path quality . . . 89
4.4.4 High end-to-end delay . . . 90
4.4.5 Impact on memory requirements . . . 95
4.4.6 Qualitative comparative analysis . . . 97
4.4.6.1 Flooding issues in LLNs . . . 97
4.4.6.2 Impact of flooding in battery operated nodes . . . 98
4.4.6.3 Lack of support for routing based on node capability . . . 99
4.5 Performance Results for P2P Communication . . . 100
4.5.1 Path quality . . . 100
4.5.2 End-to-end delay . . . 102

4.5.3 Memory requirements . . . 102
4.5.4 Packet length . . . 104
4.6 Scaling Properties . . . 105
4.7 Summary . . . 109

5 RPL: Control Plane Congestion Mitigation in Non-Storing Mode 111
5.1 DAO Specific Operation in RPL: Motivation for Optimization . . . 112
5.1.1 Trade-off on designing the value of DelayDAO Timer . . . 113
5.1.2 Bottleneck due to a constant value of DelayDAO Timer . . . 114
5.2 Determining DelayDAO - A Distributed Algorithm . . . 117
5.3 Determining DelayDAO - Centralized Approach . . . 119
5.4 Evaluation of the Algorithms . . . 120
5.4.1 Simulation setup and metrics . . . 120
5.4.2 Simulation results . . . 122
5.5 DelayDAO Controller - A Combined Algorithm Proposal to Improve RPL's Performance . . . 128
5.5.1 Routine followed at LBR or collection point . . . 130
5.5.2 Routine followed at nodes other than LBR . . . 130
5.5.3 Evaluation of proposed approach . . . 133
5.6 Summary . . . 137

6 RPL: DAO Propagation in Storing Mode 138
6.1 DAO Aggregation and Delay Method in Storing Method . . . 138
6.1.1 Aggregated DAO packet format . . . 139
6.1.2 DAO aggregation algorithm . . . 141
6.2 RPL Storing Mode Evaluation . . . 142
6.3 Summary . . . 145

7 Evolution of RPL: Towards Load Balanced LLNs 147
7.1 Related Work . . . 148
7.2 A Metric to Define Load Imbalance . . . 150
7.2.1 Example of balanced collection tree . . . 150
7.2.2 Imbalance factor . . . 151
7.3 How Hard Is It to Minimize Imbalance? . . . 153
7.4 Implementation of Load Balancing in LLNs . . . 154
7.4.1 Runtime analysis of proposed method . . . 156

7.5 Evaluation . . . 157
7.5.1 Simulation results . . . 157
7.6 Summary . . . 163

8 Future Work - RPL Adaptation in IoT via HIPC 165
8.1 Function of Proposed Architecture . . . 166
8.2 Detailed HIPC Architecture . . . 168
8.2.1 Distributed Processor Modules (DPM) . . . 169
8.2.2 Centralized Processor Modules (CPM) . . . 171
8.3 Summary . . . 175

9 Conclusion and Future Directions 176
9.1 Summary of This Thesis . . . 176
9.2 Future Work Items . . . 178
9.3 Summary of Publications . . . 179

References 181

List of Figures

2.1 DIO propagation and DAG formation. 18
2.2 DAO propagation and route establishment. 20
2.3 Data routing in RPL non-storing mode. 21
2.4 Data routing in RPL storing mode. 21

3.1 Outdoor Network Topology with 45 Nodes. 25
3.2 Outdoor Network Topology with 86 Nodes. 26
3.3 Example of Link Characteristics. 27
3.4 CDF of Hop Count versus Hop Count. 30
3.5 CDF of Total ETX Path Cost along Path versus ETX Path Cost. 30
3.6 CDF of Hop Distance Stretch versus Hop Distance Stretch Value. 31
3.7 CDF of ETX Path Stretch versus ETX Path Stretch Value. 32
3.8 CDF of Routing Table Size with Respect to Number of Nodes. 34
3.9 Comparison of Packet Latency, for Different Path Lengths, Expressed in Hop Count. 35
3.10 Amount of Data and Control Packets Transmitted against Node Id Using Link ETX as Routing Metric. 36
3.11 Amount of Data and Control Packets Transmitted for Node 12. 37
3.12 Amount of Data and Control Packets Transmitted for Node 43. 38
3.13 Amount of Data and Control Packets Transmitted for Node 31. 39
3.14 CDF: Loss of Connectivity with Global Repair. 41
3.15 CDF: Loss of Connectivity for Different Global Repair Period, Source Rate 20 Seconds/Packet. 41
3.16 CDF: Loss of Connectivity for Different Global Repair Period, Source Rate 10 Seconds/Packet. 42
3.17 Amount of Control Traffic for Different Global Repair Periods. 45
3.18 CDF: Loss of Connectivity for Different DAG Repair Timer Values for Global+Local Repair, Source Rate 20 Seconds/Packet. 45

3.19 CDF: Loss of Connectivity for Global Repair and Global+Local Repair, Source Rate 10 Seconds/Packet. 46
3.20 CDF: Loss of Connectivity for Global Repair and Global+Local Repair, Source Rate 20 Seconds/Packet. 47
3.21 Number of Control Packets for Different DAG Sequence Number Period, for Both Global Repair and Global+Local Repair. 47
3.22 CDF of End-to-End Hop Count for RPL and Ideal Shortest Path in Home Routing. 48
3.23 CDF of ETX Path Cost Metric for RPL and Ideal Shortest Path in Home Routing. 49
3.24 CDF of Hop Distance Stretch from Ideal Shortest Path. 49
3.25 CDF of ETX Metric Stretch from Ideal Shortest Path. 50
3.26 Packet Latency for Different Hop Counts in RPL. 50
3.27 CDF of Total ETX Path Cost versus ETX Path Cost. 52
3.28 CDF of ETX Fractional Stretch versus ETX Fractional Stretch Value. . . . 53
3.29 CDF of Fractional Hop Count Stretch. 53
3.30 End-to-End Packet Delivery Latency for Different Hop Counts. 54
3.31 Data and Control Packet Comparison. 55
3.32 Data and Control Packets over Time for Node 1. 56
3.33 Data and Control Packets over Time for Node 78. 56
3.34 Data and Control Packets over Time for Node 300. 57
3.35 Scaling Property of Maximum Control Packets Processed by Any Node over Time. 58
3.36 Comparison of Distribution of Fraction of Path Change. 59
3.37 Unreachability time to DAG root for different DAG repair periods (RPL). . 59
3.38 Numbers of Control Packets for Different Global Repair Timer Periods. . . 60
3.39 ETX Fractional Stretch Factor for Different Tolerance Levels. 61
3.40 Number of times parents changed across the DAG. 62
3.41 Control overhead for difference tolerance levels. 62
3.42 Distribution of Fraction of Path Change for Network A. 64
3.43 Distribution of Fraction of Path Change for Large Network C. 64

4.1 RREQ forwarding in LOADng. 72
4.2 RREP forwarding in LOADng. 72
4.3 A balanced tree with B = 3. 74

4.4 Maximum control packets processing for RPL and LOAD-ng. 83
4.5 Control overhead for RPL and LOADng. 84
4.6 Control overhead for each node in a large network. 84
4.7 Total control packets processing - RPL vs LOADng in small network. . . . 85
4.8 Total control packets processed in network against time in large network. . 86
4.9 Control overhead vs. node ID for different application rates, LOADng. . . . 88
4.10 Control overhead vs. node ID for different application rates, RPL non-storing. 88
4.11 End-to-end hop distance for RPL and LOAD-ng. 90
4.12 CDF of the total ETX path cost in a large network. 91
4.13 End-to-end delay for RPL and LOADng. 91
4.14 End-to-end delay for RPL and LOADng - large network. 92
4.15 End-to-end delay for RPL and LOADng, zoomed in. 93
4.16 End-to-end delay vs. hop distance for RPL and LOADng. 94
4.17 Maximum RAM occupancy - RPL non-storing vs LOADng. 96
4.18 Maximum RAM occupancy in bytes for each node in a large network. . . . 97
4.19 End-to-end hop distance for RPL and LOADng; P2P application. 101
4.20 ETX path cost for RPL and LOADng; P2P application. 101
4.21 Maximum RAM occupancy for RPL non-storing and LOADng; P2P application. 103
4.22 End-to-end delay comparison : P2P application. 103
4.23 Packet length for LOADng, RPL storing and non-storing modes. 104
4.24 Average control overhead per node with network size. 106
4.25 Total control packets in network with network size. 106
4.26 Maximum RAM occupancy in bytes against network size. 107

5.1 Total DAO transmissions (Tx) and receptions (Rx) versus time and rank. . 115
5.2 Minimum DAO buffer requirement against time for different ranks. 116
5.3 Total DAO Tx + Rx versus time and rank for the 1000-node grid topology. 116
5.4 DAO reach time for distributed algorithm. 123
5.5 DAO reach time for centralized algorithm. 124
5.6 DAO reach time for all mechanisms. 124
5.7 Maximum packet buffer size. 125
5.8 Maximum RAM consumption. 126
5.9 Average control packet overhead. 126
5.10 DAO round-trip time. 127

5.11 Data packet delivery latency. 127
5.12 Stability of 'K' with time. 128
5.13 Routine at LBR - centralized parameter estimation. 131
5.14 Routine followed at nodes - distributed parameter tuning. 132
5.15 CDF of data delivery delay. 134
5.16 Maximum buffer occupancy against network size. 134
5.17 Maximum RAM occupancy against network size. 135
5.18 CDF of DAO round trip time. 136
5.19 CDF of time taken by DAOs to reach the DAG root. 136

6.1 Proposed DAO target option for aggregation. 140
6.2 DAO reach time for both mechanisms. 143
6.3 Maximum RAM consumption. 144
6.4 Average control packet overhead. 144
6.5 Data packet delivery latency. 145

7.1 A DAG created by RPL. 150
7.2 Example of unbalanced parent selection. 150
7.3 Example of balanced parent selection. 151
7.4 Imbalance metric (I) vs Network size. 158
7.5 Top level variance in energy expense. 158
7.6 CDF of energy expense vs % of nodes. 159
7.7 Network Lifetime vs Network size. 160
7.8 Improvement fator by load balancing vs Network size. 161
7.9 Imbalance Factor for Default RPL, Proposed Heuristic, and Maximally Balanced DAG. 162
7.10 Top level Variance in Packet Transmission for Default RPL, Proposed Heuristic, and Maximally Balanced DAG. 162
7.11 Time to run compensate algorithm. 163

8.1 LLNs Connected to an AS. 167
8.2 HIPC in a complete Protocol stack for LLN Devices 167
8.3 Distributed Processor Modules in HIPC. 169
8.4 Centralized Processor Modules in HIPC. 171

List of Tables

1.1 Routing Requirements for different LLNs 15

3.1 Path Quality CDFs. 33
3.2 Loss of Connectivity Time, Data Rate - 10 Seconds / Packet. 43
3.3 Loss of Connectivity Time, Data Rate - 20 Seconds / Packet. 44

4.1 Comparison of Existing Literature and the Contributions of This Study. . . 70
4.2 Simulation settings. 81

Abstract

In early 2008, a new IETF Working Group (WG), namely ROLL, was chartered to investigate the suitability of existing IP routing protocols for Low Power Lossy Networks (LLNs), which at the time were suffering compatibility issues due to the pervasive use of proprietary protocols. Given the vision of the Internet of Things (IoT) and the role LLNs would play in the future Internet, the IETF set out to standardize an IPv6 based routing solution for such networks. After surveying existing protocols and determining their unsuitability, the WG started designing a new distance vector protocol called RPL (recently standardized in IETF RFC 6550) to fulfill their charter. Joining the WG efforts, we developed a very detailed RPL simulator and using link and traffic traces for existing networks, contributed with a performance study of the protocol with respect to several metrics of interest, such as path quality, end-to-end delay, control plane overhead, ability to cope with instability, etc. This work was standardized as IETF Informational RFC 6687.

This detailed study uncovered performance issues for networks of very large scale. In this thesis, we provide an overview of RPL, summarize our findings from the performance study, analysis and comparison with a reactive lightweight protocol and suggest modifications to the protocol that yield significant performance improvements with respect to control overhead and memory consumption in very large scale networks. For future work, we propose a routing technique, named Hybrid Intelligent Path Computation (HIPC), along with modifications to the original RPL protocol standard, that outperforms solely distributed or centralized routing techniques. Finally, we also show how one can facilitate Quality of Service (QoS), load balancing and traffic engineering provision in the IoT without incurring any extra control overhead in number of packets other than that already consumed by the proposed IETF standard, using a combination of centralized and distributed computation.

Glossary

6LowPAN Low Power IPv6 Personal Area Network

AMI Advanced Metering Infrastructure

AODV Ad-hoc Distance Vector Routing Protocol

BAN Body Area Network

CDF Cumulative Distribution Function

CoAP Constrained Application Protocol

DAG Directed Acyclic Graph

DAO Destination Advertisement Option

DIO DODAG Information Option

DIS DODAG Information Solicitation

DODAG Destination Oriented Directed Acyclic Graph

EPC Electronic Product Code

ETX Expected Transmission Count

IETF Internet Engineering Task Force

IoT Internet of Things

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

LBR LLN Border Router

LLN Low-Power Lossy Networks

LOADng LLN On-demand Ad-hoc Distance Vector Routing Protocol - new generation

MAC Medium Access Control

MANET Mobile Ad-hoc NETworks

MP2P MultiPoint to Point

MTU Maximum Transmission Unit

OSPF Open Shortest Path First

P2MP Poiont to MultiPoint

P2P Point to Point

PAN Personal Area Network

PDR Packet Delivery Ratio

PLC Power Line Communication

ROLL Routing Over Low-power and Lossy networks

RPL Routing Protocol for Low-Power Lossy Networks

RREP Route Reply

RREQ Route Request

SRH Source Routing Header

TCP Transport Control Protocol

WSN Wireless Sensor Network

1

Introduction

1.1 Introduction to LLNs: Low-Power Lossy Networks

The Internet is increasingly becoming an inevitable part of our lives. The number of computing and interconnected devices has grown exponentially within last 2 decades, and is expected to grow further. Current day Internet encompasses desktop, data centers, laptop and cellular devices, which generate a huge data traffic. Currently, we have, on an average, 2 connected device per person in this world. However, by the end of 2020, it is estimated that the world will have 6.5 connected devices per person, or an estimated 50 Billion connected devices in total (1). The vision of the future Internet not only includes conventional computing and capable devices, but also everyday objects. Objects such as sensors and actuators in industry, health monitoring, vehicles, electronic devices, AMI (Advanced Metering Infrastructure) meters are expected to be integrated into, and be controlled by the user through the Internet. The number of applications that next generation sensor/actuator networks can cover is endless.

As time progresses, we have more and more sensors for our daily use and in the community around us. Sensors and actuators are an essential part of modern day industrial automation settings and manufacturing plants for the purpose of monitoring, alert reporting including prediction of critical fault and overtaking emergency measures. In health-care industries, patient monitoring includes various sensors that interconnect us-

ing a Body Area Network (BAN) or a Personal Area Network (PAN). Smart homes and intelligent buildings employ sensors to perform tasks ranging from home security, alarm monitoring, climate control, smoke or gas detection to control of electrical appliances and power usage monitoring. At the same time, objects without a sensor/actuator part can be required to be accessed for management, tracking and data collection as well. For example, any object with an intelligent tag can be subjected to monitoring in real time. Intelligent object tags such as Radio-Frequency Identification (RFID) tags or Electronic Product Codes (EPC) (2) can help improve object visibility. Any object with an identifier needs to be traced, and their status and current locations may need to be updated in real time. For smart grids, any AMI meter may need to communicate with the appliances at home or any installed (solar) power generator and communicate to the base stations for power distribution. At the same time, wireless sensor networks are entering the area of smart cities and environment monitoring. Thus the number of applications and use cases is only expected to grow in the future.

In a nutshell, the next generation of the Internet is not going to be constrained by user traffic such as browsing websites, video traffic or voice traffic, but will include a large share of data generated and directed towards 'smart objects'. A smart object is any entity that is uniquely identifiable, and can communicate with a radio or over Power Line Communication (PLC) media and can be accessed by the user to perform specific task or used for data accumulation. Each smart object may contain an intelligent tag such as RFID or EPC, a sensor and/or an actuator. However, the smart objects posses negligible processing horse power with few KBs of RAM and flash storage compared to the routing elements in the Internet. For example, an Atmel 8-bit AVR microcontroller combines 128KB of programmable flash memory, 4KB SRAM, a 4 KB EEPROM working at a maximum of 16 MHz (MIPS). The computing device along with the communication device can be embedded into many objects such as engines, switches, meters, sensors to incorporate the 'smartness' into them. The vision of the Internet of Things (IoT) includes all such smart objects from different contexts, and aim to provide a common platform

connectivity via the Internet. Thus, the Internet of Things can be defined as a world-wide network of interconnected objects and commodities uniquely addressable from anytime, anyplace by anyone (3, 4).

Networks connecting smart objects and sensor nodes, often operate in highly variable link quality conditions. The link speed for these networks are often limited to a few tens of Kbps at maximum. The interconnection of various smart objects, for the link quality connecting them and due to the constraint on size and capacity of the processing units, are classified as Low Power and Lossy Networks (LLNs). LLNs are emerging as a new deployment scenario in many environments, clearly those related to smart home, building or industrial automation, communication among AMI meters in a smart grid and for the most part of the envisioned Internet of Things (IoT). The challenges in these networks include very low device power and memory, highly varying link quality, frequent link outage, etc. Requirements for these deployments such as in smart home, building or industrial automation, communication among AMI (Advanced Metering Infrastructure) meters in a smart grid, etc., relate to delay bound, scalability, strict time bounds on protocol convergence after any change in topology (5), etc. For instance, RFC 5673 (6) requires bounded and guaranteed end-to-end delay for routing in an industrial deployment, while RFC 5548 (7) mandates scalability in terms of protocol performance for a network of size ranging from 10^2 to 10^4 nodes. Link state routing protocols, such as OSPF(8), OLSR (9), IS-IS, and OLSRv2 (10) tend to flood the network with link updates. Since links in LLNs suffer from severe temporal variation and frequent outage, these protocols fail to keep a low control cost overhead (11). Classical distance vector protocols used in the Internet, such as EIGRP (designed by CISCO, (12)), AODV (13), etc., fail to provide quick recovery from link churn, and frequent topology updates.

A suitable protocol for LLNs therefore needs to abide to strict delay constraints, while maintaining a minimum control overhead, and should be capable of providing quick recovery from frequent link outage. The IETF ROLL (Routing Over Low power and Lossy network) working group (14) was chartered to identify an IP routing solution for such net-

works under specific deployment scenarios. After surveying existing solutions, the working group published a number of documents involving specific routing requirements for individual deployment scenarios, and then concentrated its efforts on designing standardized routing protocols suitable for these deployments. As a result, RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks), a distance vector protocol with proactive route creation has been standardized in RFC 6550 (15), which provides mechanisms for multipoint-to-point, point-to-multipoint, as well as point-to-point traffic (for a brief overview of RPL, please refer to Section 2.1).

RPL was designed to meet the requirements spelled out in RFCs 5826 (5), 5673 (6), 5548 (7), and 5867 (16). For example, all the requirement documents mandate parameter constrained routing, and RPL by means of Objective Function (OF) design, can chose parents conforming to the routing constraints. At the same time, RPL supports 'zero configuration' at the set up phase and while new nodes join the network, which means human intervention is not needed. While Home and Building routing requirements (5, 16) prefer an end-to-end delay of less than 500 ms and less than 120 ms respectively, simulation of RPL in such environments provides satisfactory results (17, 18, 19). At the same time, (17) has shown RPL can perform faster repair to converge in case of link churns, which is mandated by Urban (7) and Home (5) LLN routing requirements. We will review the core requirements spelled out in these documents in Section 1.3.

1.2 From Proprietary WSN Solutions to Standardization

For over two decades, research in Wireless Sensor Networks (WSN) field has grown in popularity, and the constrained nature of the WSN system led researchers to believe that layered architecture must be abandoned to make way for cross layer optimizations. As we observe the evolution of TinyOS ((20)), conventional packet header was replaced by the Active Message Dispatch ID ((21, 22)), thus leading away from IP as opposed to the mainstream Internet. As pointed out in (23), several other arguments led to denouncing the Internet architecture. Firstly, it was assumed that individual devices will not require

addressing, or no sensor in particular will be queried from inside or from outside the network; instead, applications will be 'data centric' and will focus on measurements from an area. Secondly, while IP networks are designed to handle various applications with different traffic profile, sensor networks will be limited to sensing tasks only, thus optimization based on application profile is more desired. Finally, localized algorithms will bolster the required scalability and robustness in these networks as "sensor networks have different enough requirements to at least warrant re-considering the overall structure of applications and services", thus leading to introduction of 'Directed Diffusion' (23).

However, where traditional wireless sensor networks were previously presumed to be a separate entity from the mainstream Internet, more and more deployments along with individual components are required to be accessed from outside the network. For example, a patient's family members may require his/her status over the Internet in real time; AMI meters will need to be accessed by their respective owners via the Internet in real time or the owner may need to be notified about any incident, such as an electrical fault or overload in the house. In any such case, each object within the network must be given unique address to be identified and accessed. Based on operation (e.g., alert or regular query or management), within a single network, there will be multiple applications running with different traffic pattern. Thus, LLNs needed to be connected with the existing IP architecture, either by adopting it or by using proper protocol translation gateways.

The authors in (24) proposed an architecture to integrate WSN with IP based external network. It provides platform for achieving mobility, web enablement, time synchronization, and security while accessing the sensor networks as a part of the IoT. The routing protocol used is HiLoW, with mobility management protocol called MARIO (Mobility Management Protocol to Support Intra-PAN and Inter-PAN Handover with Route Optimization for 6LoWPAN (25)). Security is provided through the protocol SSNAIL (26). The architecture is a combination of lightweight protocols at different layers, and a cross layer optimization module which locates vertically from PHY/LNK layer to the adaptation layer to provide a collaboration interface with the upper layers. Thus, this paper

describes how the architecture and protocols in different layers for the sensors / devices and gateways should be, so integration with IP networks is possible.

1.2.1 Using application gateways

Indeed, most of the existing ad-hoc and sensor networks with proprietary solutions (e.g., ZigBee or Wireless-Hart) implements complex application gateways that translate IP world to the non-IP format that their Wireless Sensor Network deployments (WSN) understands to connect to the Internet. However, problems with this approach are multiple to prevent this to be adopted on larger scale, as pointed out in (22) and (27).

- The gateways need to be configured differently for different deployments. A protocol translation, however, will need to understand any application profiles in advance that may be used in the WSN. Subsequent modification or addition in application protocol will require reconfiguring the gateway, which limits the network agnostic innovation and growth that IP architecture currently offers;
- The gateway normally prevents real-time end-to-end connection between the sensor/actuator device and the end-user. Since it needs to store different states and tables for the protocols used within the WSN, seamless transition through the gateway is often obstructed when a query is received;
- These gateways transform HTTP packets contained in IP datagram to the format the proprietary deployment understands, thus requiring reconfiguration and careful synchronization between the gateway and end devices any time there is an update in any of the concerned protocols used. Needing to change or upgrade the border gateway for any future improvement and addition is a hindrance to innovation and scalability.

1.2.2 Using standard protocol suits

The IETF Constrained ReSTful Environments (CoRE) Working Group (28) is in the process of standardizing Constrained Application Protocol (CoAP) (29) as a specialized

web transfer protocol for Machine-to-Machine applications in networks with constrained nodes and in LLNs. If CoAP is used as application layer protocol, the needed translation at any border router will be from HTTP to CoAP, and thus the border router can be agnostic to the set of applications deployed in the network (22). Ubiquity of IP architecture can also provide real time communication with the user, as well as benefits provided by network optimization and debugging tools designed specifically for IP. Clearly, since IPv4, IPv6 and 6LoWPAN header compression formats are standardized, translating between IPv4 to IPv6 or IPv6 to IPv6+6LoWPAN packet format can be done much faster compared to table lookup as provided by the application gateways. The problem, as it stood between the dream of extending IP to everyday objects and reality of implementation, still was the heavy memory footprint of standard IP protocol stack.

With the successful implementation of μIP over low power IEEE 802.15.4 links, the authors in (30, 31) showed that a full RFC compliant TCP-IP stack on 8 bit architecture is possible which does not take advantages of optimizations and mechanisms developed to keep WSNs low resource hungry. However, where 802.15.4 links support MTU of only 127 bytes, IPv6 packets, according to the standard, must support a maximum MTU of 1280 bytes. Clearly, an adaptation layer between the layer 3 and layer 2 was needed to address the issues such as asymmetric routing, fragmentation, unreliable delivery, etc. and consequently, the IETF formed the 6LoWPAN working group. Mechanisms to carry IPv6 datagram over 802.15.4 links, fragmentation and header compression to accommodate as much payload as possible in tiny 127 bytes MTU was standardized in RFC 4944 (32).

The authors in (22) implemented a complete IPv6 based architecture for a wireless sensor network, while employing several existing mechanisms to improve the network performance. The link layer protocol employs a Media Management Control (MMC) that is based on B-MAC (33) and WiseMAC (34). For reliability over the lossy link, each unicast transmission is mandated to have an Acknowledgement back, which is essentially another 802.15.4 data frame with optional payload for scheduling optimization. The approach used header compression in a similar fashion that is described in RFC 4944, and was able to

compress a 48-byte UDP/IPv6 header down to 6 bytes when communicating over link-local unicast, 11 bytes with global unicast and multicast within the WSN, and 25 bytes when communicating with arbitrary IP devices outside the WSN. Routing wise, any forwarder in the network implemented three orthogonal mechanisms: hop-by-hop recovery, streaming, and congestion control. This deployment was shown to achieve a total ROM footprint of 24,038 bytes and a RAM footprint of only 3,598 bytes including the operations for all the stacks, thus proving IP enabled tiny sensors to be a reality much easier to achieve than what one could presume merely a decade back, even while employing security and header compression mechanisms.

1.3 Unique Routing Challenges in LLNs

Though it had been established that IPv6 can be implemented for resource constrained devices, the routing protocol was yet to be finalized. Due to the device capacity and link characteristics, the routing protocol required for proper operation within the LLNs would face significant challenges due to inability of the devices to store detailed routing states or network topology. Also the control overhead would have to be a fraction of what is endured in the Internet, with a network scale still ranging to thousands of nodes. The LLN devices are mostly powered by batteries, with little to no opportunity for energy scavenging and are expected to last a few years without any power source. At the same time different deployment scenarios have various requirement based on their application profile, traffic characteristics and criticality of application data. The ROLL Working Group, thus proposed documents detailing the routing requirements pertaining to four deployments scenarios - Urban LLN deployments in RFC 5548 (7), Industrial Automation LLN deployments in RFC 5673 (6), Home LLN deployments in RFC 5826 (5), and Building LLN deployments in RFC 5867 (16). The common routing challenges are pointed below:

- Parameter constrained routing: Routing requirements for all of the above deployments include a MUST support for node constraint or parameter constraint based routing. For example, the protocol should be able to choose a mains powered node

over a battery operated node while routing. Such constraints can also include battery energy level, CPU speed, RAM size, or any set of constraints placed on the device by the user.

- Auto configuration: Routing protocol for all types of deployments should support 'zero configuration' at the set up phase and while new nodes join the network, which means "a node can obtain an address and join the network on its own, without human intervention" and without allocating device address manually.

- Unreliability of links and network dynamicity: The routing protocol can never assume that all links are reliable or bi-directional. It is inevitable for LLNs that link churn will be frequent due to harsh environment and electro-magnetic interference. The protocol MUST be able to converge within a reasonable amount of time, without spending much control overhead.

1.3.1 Urban LLN routing requirements

As discussed, concepts of smart city or smart locality where sensing and actuating nodes are placed in outdoor environments to improve living conditions as well as to monitor environmental conditions is increasingly making urban LLN more and more imminent. For Urban LLNs, deployment of nodes normally happen in batches of in the order of $100 - 1000$ devices. The deployment may easily span tens of thousands of devices in total, and may consist of $10^2 - 10^7$ devices in future roll-outs. The lifetime of these devices are expected to be $10 - 15$ years where the energy sources for these nodes are mostly non-rechargeable batteries or battery with very limited scope of energy scavenging or harvesting.

The application profile for urban LLNs mostly includes reporting, both periodic measurement and queried measurement reporting. Periodic or regular measurement reporting traffic is usually low in frequency, such as once in few hours or once in a day. Queried measurement traffic, on the other hand, is generated by applications outside the LLN in a random on-demand fashion that can be modeled with a heavy-tail distribution. These

measurement traffics are usually directed from the sensors to the border router, where the routing protocol SHOULD provide the option of delivering the acknowledgement in the reverse direction. The regular measurement traffic is delay tolerant to some extent, and should reach the border router within a small fraction of reporting time. Thus, the regular measurement traffic has a preferable latency requirement of < 1 minute, and queried measurement (e.g., live AMI meter readings) enjoys a latency bound of the order of few seconds. However, the urban LLN system may rarely trigger alarms, which generate 'Alert' traffic. Alert traffic is unicast towards the LBR, and is normally generated by a group of sensors, thus creating a number of directed and simultaneous flows. Alert traffic is highly delay sensitive; independent of number of flows or number of reporting devices, such traffic should reach the border router within a few seconds (< 5 seconds). From the application profile, it is imperative that the routing protocol must support high directional sensed data traffic from sensors to the border router and queries and control traffic from the border router to sensors and actuators. Apart from such unicast traffic, the protocol should also support efficient large-scale messaging (multicast traffic) to groups of actuators and simultaneous MP2P traffic, such as system-wide alerts.

1.3.2 Industrial automation routing requirements

In industrial automation, wired networks are currently being used for the purpose of collecting information, as well as process control, and closed loop control. However, wires provide reliable information propagation at an expense of reduced capacity of remote management and increased difficulty of installation and maintenance. On the other hand, low-power wireless devices provide a plant with the possibility of increased number of collection points and control points that can be remotely managed. Hence it is foreseen that installing sensors and actuators that can communicate wirelessly will significantly improve the productivity of any industrial automation setting. However, given the reliability and low-latency that a cable provides, vendors do not expect the currently existing wired control modules to be completely replaced by a wireless counterpart, rather having

wireless networks to augment the current setting to provide better visibility to the system. RFC 5673 (6) provides the detailed requirement of a routing protocol that may be used to connect such low power wireless field devices to improve process control and factory automation.

The process applications in industrial automation can be categorized into three broad divisions, and further divided into 6 classes of traffic based on their criticality. Safety related applications are always considered as emergency action, and fall into Class 0 traffic. This class of traffic is extremely delay sensitive and requires most reliability, as they indicate critical failure in the process control requiring urgent action and/or human intervention. However, this traffic class should be considered usually dormant, and only appearing once in many years. The second kind of applications are related to Control mechanism, more specifically critical closed-loop regulatory control (Class 1 traffic), non-critical closed-loop supervisory control (Class 2 traffic) and open loop control (Class 3 traffic) where human intervention is present. The control traffic, while are sensitive to latency requirements, they are also sensitive to jitter, as jitter can destabilize a control algorithm. The third type of application normally involves monitoring and asset tracking, which can either be alert or event based maintenance traffic (Class 4) and event/data logging and upload/download traffic (Class 5). Clearly, lower the class number of the traffic, higher is the sensitivity for latency and reliable delivery. Since Class 0 and Class 1 traffic are extremely crucial, wireless networks are expected to augment the existing infrastructure for providing communication to non critical control traffic of Class 2 and 3, and all monitoring traffic of Class 4 and 5.

The topology for majority of industrial automation LLNs consists of $10 - 200$ nodes, which act both as field devices and as forwarders. At this point, the maximum number of hops to reach the border router, or a router connected to the backbone is not envisioned to be more than 20. Where there is no specific requirement on the number and/or physical placement of the border router or backbone network, it is generally assumed that multiple border routers should be deployed in order to keep the LLN subsets to a smaller size. A

routing protocol should support provisioning of available bandwidth according to traffic class and should accommodate occasional burst of traffic arising from alert or monitoring data. Thus, a protocol with QoS provisioning or route computation based on traffic class may often be preferred. Since class 0 and class 1 traffic is not handled via a routing protocol for LLN, the convergence requirement is somewhat loose. RFC 5673 states that the protocol must converge after adding a new field device within order of a few minutes, and delivering a packet via established route, or determining the absence of a functioning one, should be achieved within order of tens of seconds.

1.3.3 Home automation routing requirements

In recent past, smart homes increasingly are using more and more sensing devices and actuators for automation purpose. As in industrial setting, it is argued how wireless communication of smart objects in homes would ease the installation and reduce the cost of maintenance in near future. A home network encompasses both actuators such as light dimmer, heating valve as well as sensors such as switches, leak detectors, blood pressure monitors. The network is supposed to be accessible and controllable via an IP-enabled application, where the devices respond to the queries and controls sent from a web application, cell phone, PDA or a mobile remote control. While these devices previously had been wired through powered lines, they are expected to operate over a wide range of communication media in near future, such as IEEE 802.15.4, Bluetooth, low-Power WiFi and low-power PLC (Power-Line Communication) links. Some typical use cases of home automation applications would include the following:

- Lighting and heating applications, both event driven (e.g. person walking into the kitchen) and query based;
- Appliance regulation based on Demand-Side Management, where depending on available electrical supply, use of heavy appliances such as air conditioning, climate-control systems, washing machines, etc. can be regulated;

- Demand Response based regulation, where appliances are regulated also depending on pricing information from vendor;
- remote video surveillance over the IP, where the video stream is initiated either by query by the end user or as a result of an alarm;
- Healthcare based applications that include monitoring and reporting of patient body temperature, blood pressure, heart rate, ECG, etc. and alarm generation in case of criticality;
- Alarm systems identifying a risk situation.

The home LLNs are subject to a topology size of few tens to few hundreds of nodes, and the requirements are mainly spelled out for networks with fewer than 250 nodes, with a maximum hop distance of 4. The traffic pattern is mainly demand based, for example where wall switches or remote controls generate signal to activate actuators responsible for light/temperature control. These traffics are random and can take place only a few times ($\sim 1 - 10$ times) an hour. The home LLNs application profile also includes query based traffic, such as the user querying for temperature/air pressure/rain sensor reading. These traffics may be both periodic or on demand, and has a frequency similar to the demand based traffics. The routing protocol MUST support mobility for remote control based applications and wearable healthcare devices that are expected to be used within the home network. The round trip latency differs for mobile and stationary nodes. Where the routing protocol must converge within 500 ms if no node has moved, for instances where originator nodes have moved, the protocol must provide a response within 4 seconds.

1.3.4 Building automation routing requirements

Similar to industrial automation, building applications such as Heating, Ventilation and Air Conditioning (HVAC) have been being controlled and monitored via wired infrastructure for decades. Systems to control lighting, elevators and monitor physical security and fire hazards mostly involved human in the loop operation without much scope of automa-

tion. However, with the advent of wireless communication enabled sensors and actuators, gradual replacement of wired components with their wireless counterparts became imminent. In a building automation system, sensor devices can be battery or main powered, but the actuators are almost all the time mains powered. Due to this reason, it is also envisioned that wired and wireless links are likely to coexist in building automation LLNs.

Since Building management systems are often divided into several independent autonomous subsystems, it is predicted and preferred that individual sensors and actuators within the LLN will be exchanging real time traffic. In conjunction, several alarms and event data will need to be reported outside the LLN. Typically, in a building automation LLN, 30% of the traffic will remain inside the LLN as P2P traffic within the devices such as sensed data from sensor to the controller and acknowledgement thereof, control signals from one controller to another controller, etc. The P2P traffic has a typical frequency of 1 packet/minute. On the other hand, 70% of traffic is MP2P, and routed off the LLN. Multicast or P2MP traffic may be present where applications such as turning all lights in the floor on, or during network initialization, but is considered rare in building automation scenario.

As pointed out in (16), routing protocol such an LLN should support a network scale of at least 2000 nodes, where at least 1000 of them will function as routers. Scope of subnetworking should be supported for up to 255 nodes. The protocol, however, may not need to support mobile routers, but only as end hosts. While P2P traffic is not expected to traverse more than 5 hops, a round trip latency (response after sending a query) should be kept less than 120 ms for such a network. Clearly, the latency requirement is most stringent for building automation among all LLN deployment scenarios. The different routing requirements are summarized in Table 1.1 for quick reference of the reader.

1.4 Contributions of This Thesis

Since RPL and routing in LLNs altogether have been a nascent addition to the modern day Internet, before widespread implementation one needs to make sure that the protocol

Criteria	RFC 5548 (Urban)	RFC 5673 (Industrial)	RFC 5826 (Home)	RFC 5867 (Building)
Constrained Routing / Node coloring	MUST	MUST	MUST	MUST
Support of Mobility	MAY NOT be needed	SHOULD be supported for mobile workers	SHOULD be supported for Remote control & Healthcare Apps	SHOULD be supported only for end devices, not routers
Network Scale	$100 - 10^4$	$10 - 200$	$250 - 1000$	Up to A few 1000s
Latency Bound	Fraction of the smallest reporting interval other than alerts ($\sim 4 - 5$s)	10s of seconds	500 ms RTT	120 ms for static nodes
Support of Multicast / Anycast	Both are MUST	Multicast	Multicast	MP2P

Table 1.1: Routing Requirements for different LLNs

behaves as it is intended to. We have designed a detailed RPL simulator, with a GUI to observe the protocol's performance under varying parameter settings, different traffic scenarios and different network topologies. At the same time, our aim has been to find out any probable incident where the protocol might not be suitable for constrained devices and/or large deployments. The detailed and exhaustive simulation work led us to believe that RPL, as proposed in the IETF standard, may not perform very well in large scale networks . We investigated the reason, and provided cases where the standard RPL operation leads to congestion due to control plane traffic that halts normal data plane traffic. We further proposed modifications to RPL in scheduling control traffic to mitigate the mentioned congestion and improving the protocol performance for intended large deployments. We have observed that limited centralized decision making is inevitable for routing in LLNs, as the routing elements are not capable of high performance computing. A hybrid approach rather than adopting a solely distributed or a centralized routing tech-

nique, along with necessary modifications of the IETF standard RPL, can be adopted to the Internet of Things to provide better scalability, performance and handling the control overhead. We propose a routing technique, named Hybrid Intelligent Path Computation (HIPC), along with modifications to the original RPL protocol standard, that outperforms solely distributed or centralized routing techniques. Finally, we also show how HIPC can facilitate Quality of Service (QoS), load balancing and traffic engineering provision in the IoT without incurring any extra overhead other than that already consumed by the proposed IETF standard.

This document is organized as follows: In Chapter 2, we provide a brief overview of the RPL protocol standardized in the IETF, while key performance features stemming from our performance study in our in-house simulator are covered in Chapter 3. In Chapter 4, we compare two main routing techniques, namely proactive and reactive routing, and perform an exhaustive qualitative and quantitative simulation analysis between the proactive protocol RPL, and a recently introduced reactive protocol, namely LOADng. Chapter 5 describes why RPL in its default setting does not scale well for large scale networks, and presents our proposals for distributed and centralized heuristics that fix performance issues in RPL non-storing mode. In Chapter 6, we describe how multiple destination advertisement can be combined for RPL storing mode and, as was the case in Chapter 5, the performance can be enhanced for large scale networks. In Chapter 7, we provide a lightweight load balancing approach to maximize LLN lifetime without incurring extra overhead on top of RPL. Chapter 8 briefly introduces our proposed routing technique, Hybrid Intelligent Path Computation (HIPC), and shows how a deployment can achieve specific traffic engineering tasks such as load balancing, policy enforcement, etc. This is done in broad strokes, with the details left as future work. Finally, Chapter 9 summarizes the work completed and suggests future steps as well as the challenges to be answered.

2

RPL: Routing Protocol for LLNs

Designing a routing protocol for Low-Power and Lossy Networks (LLNs) imposes great challenges, mainly due to low data rates, high probability of packet delivery failure, and strict energy constraints in the nodes. The IETF ROLL Working Group took on this task and specified the Routing Protocol for Low-Power and Lossy Networks (RPL) in (15). In This chapter we provide a brief overview of the protocol.

2.1 Overview of RPL

RPL is an IPv6 distance vector routing protocol (15), where a proactive routing structure is established by the creation of a Destination Oriented Directed Acyclic Graph (DODAG) using an Objective Function (OF) and a set of metrics/constraints. The DODAG (abbreviated as DAG) minimizes the cost of reaching the LLN Border Router (LBR or DAG root) from any node in the network as per the OF in use, which can be defined to minimize a particular metric, such as hop count or the ETX (Expected Transmission count), or any other from the list of metrics as specified in (35). A DAG structure helps restrict the size of routing tables for limited storage capacity nodes, as only a subset of destination addresses are stored at any node other than the DAG root.

2.1.1 RPL - Forming and maintaining the DAG

A node that is connected to a non-RPL implementing node or a backbone network, can act as a DAG root or LBR and has a rank of 1. The DAG root initiates the DAG formation by advertising information about the DAG using the DAG Information Option (DIO), which carries several information regarding the DAG, including the issuing node's distance from the LBR. Nodes receiving DIO, calculates its distance from LBR based on cost received in DIO and its own cost to reach the issuing node. Nodes chose a node as their parent which provides the lowest cost to reach the LBR. Figure 2.1 illustrates the process of broadcasting DIOs to form the DAG and their propagation downward. The solid lines in the figure represent the parent - child relationship in the DODAG, whereas the dotted lines represent other available links. Each node assumes a rank 1 unit greater than its parent's rank.

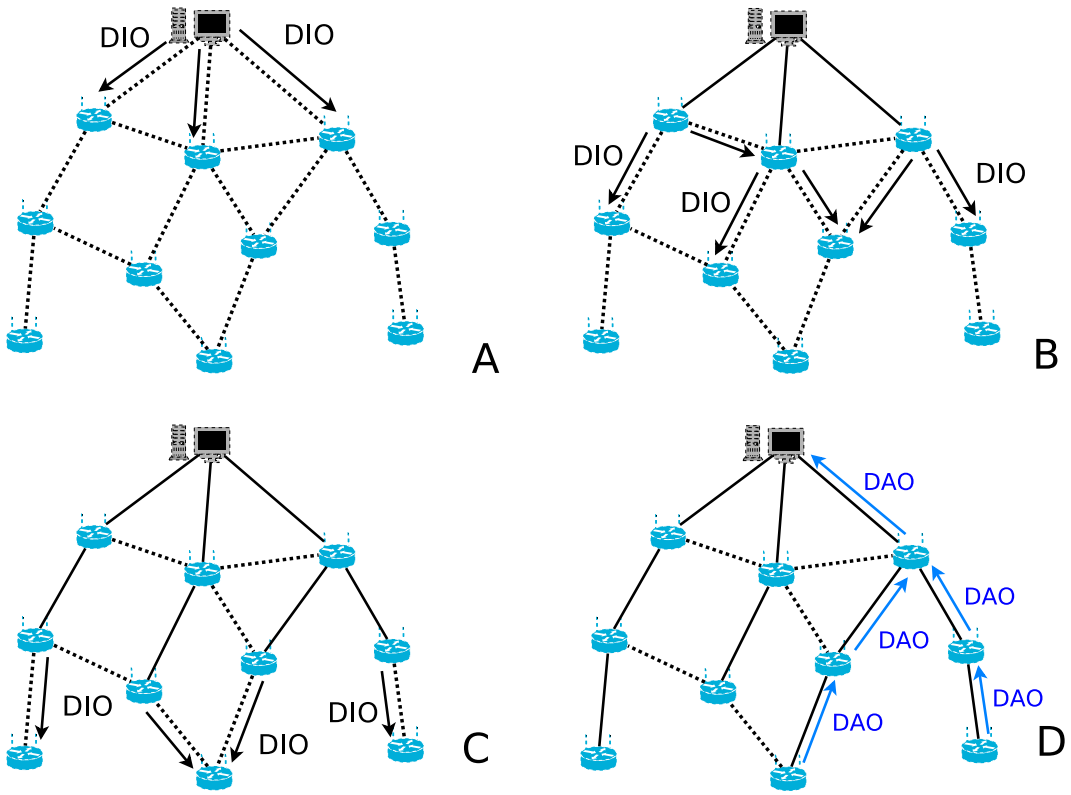

Figure 2.1: DIO propagation and DAG formation.

DIOs are also emitted periodically from each node, triggered by a timer (trickle timer) whose duration increases exponentially (doubled after each time it is fired). The smallest possible interval between two DIOs is denoted by $Imin$, and the number of times $Imin$ can be doubled before maintaining a constant rate is denoted by $DIOIntervaldoublings$, so $Imax = Imin * 2^{DIOIntervaldoublings}$. On any event that causes a change in the DAG structure (such as new parent selection), this timer is reset to the $Imin$ value contained in the DIO. By increasing the duration between two DIOs, the protocol eliminates the need of exchanging neighborhood information prematurely as the network may become stable after a few rounds of information exchange. On one hand, for stable networks where variation in link quality is not significant, the protocol gradually decreases the control plane overhead over time. On the other hand, for a more dynamic topology the protocol helps the network to adapt faster than a protocol implementing only conventional periodic updates. The exponential decay in the frequency of trickle timer, thus increase time duration between periodic updates and perfectly suits the needs of LLNs and, in particular, of large smart meter networks. If a node receives DIOs from multiple potential parents, it may choose to elect a new parent when a better path is found. RPL, however, ignores messages that will incur $x\%$ of change in path cost, where 'x' is a configurable parameter. Hence, a node does not change its parent from the current one, unless the new parent provides a path to the DAG root with a cost less than $(1 - x/100)$ times the current path is found.

A DODAG Information Solicitation (DIS) message may be used by nodes to proactively solicit DAG information (emission of DIO). Nodes who join the DAG also unicast their addresses and reachable prefixes to their parents via Destination Advertisement Option (DAO) messages, which in turn unicast them further up the DAG to advertise destinations reachable through them in support of 'down' traffic. Thus, eventually all DAOs reach the LBR, providing routing information about the whole DAG.

RPL has been standardized to operate on two modes, 'non-storing' mode and 'storing' mode. In storing mode, a node in the LLN is capable of storing routing tables and

next hop information for all the nodes in its subtree. Whenever the node forwards a destination prefix available via itself or any of its children nodes (DAO messages), it creates a corresponding routing entry for the particular destination. Therefore 'downward' route is maintained at every node in the DAG. Hence, when a node 'n' advertises a DAO for a specific destination, the receiving nodes store node 'n' as the next hop for that particular destination. The process of DAO propagation is illustrated in Figure 2.2, where node 3 acts as next hop to destination nodes 6, 7, 9, 10 to nodes 0, 2, etc.

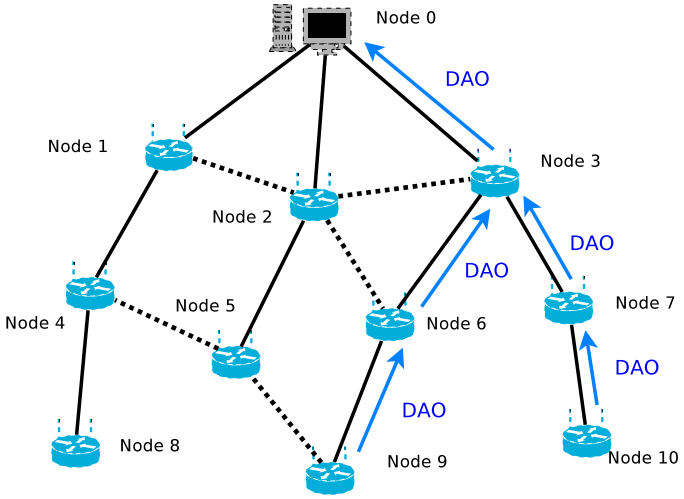

Figure 2.2: DAO propagation and route establishment.

The non-storing mode is common in large networks or networks with extremely restricted nodes. In non-storing mode, nodes do not store routes to any destination other than the DAG root. Instead, any node, on receiving a DAO from its children or other nodes in its subtree, forwards it to its parent. Contrary to storing mode, DAOs in non-storing mode are unicasted to the DAG root. Thus, the DAG root or LBR, which is a more capable device than the other LLN nodes, stores all routes to any node in the network. Figure 2.2 illustrates an instance of routing a packet in non-storing mode. In the figure, node 3 does not store routes to nodes 6, 7, 9, 10, as seen previously. The DAG root

specifies the route through node 3 for destination nodes 6, 7, 9, 10.

The DAG root issues a new DAG sequence number periodically to recompute the DAG for major topology changes. When a new sequence number is received, nodes reset their trickle timer, which results in emitting DIOs after the minimum value *Imin*, and issue a DAO to the root after a certain amount of time, which is implementation specific. For local link churn and anomaly, the nodes use a local repair mechanism by poisoning their sub-DAG, and choosing a backup parent.

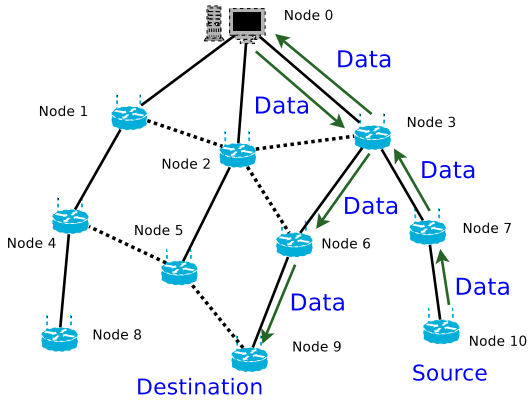

Figure 2.3: Data routing in RPL non-storing mode.

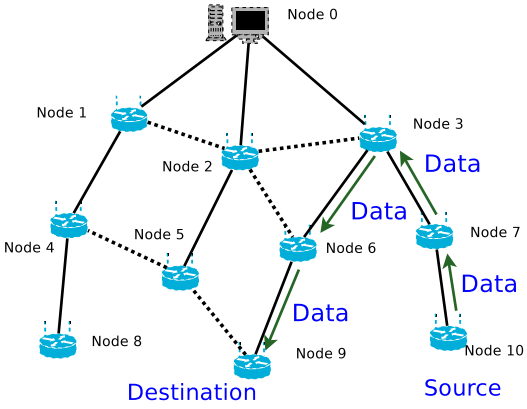

Figure 2.4: Data routing in RPL storing mode.

2.1.2 RPL - Routing through the DAG

MP2P traffic to the LBR, and P2MP traffic from the LBR to nodes follow the parent-child links thus constructed in the DAG. For any traffic destined to the DAG root (LBR), the packet is forwarded to the preferred parent in the DAG, in both storing and non-storing mode. For traffic from DAG root to any other node in LLN, the packet is forwarded to the child which contains the destination prefix in its subtree. In storing mode, a downward route is maintained, and each node 'looks up' the entry for the destination in its subtree and forwards the packet to the next-hop child. This process is explained in Figure 2.4.

In non-storing mode, since downward routes are not maintained, the DAG root con-

structs and inserts into packets a Source Routing Header (SRH), which provides the whole route through the DAG on a hop-by-hop basis. For P2P traffic, where neither source nor destination is the LBR, in storing mode, packets reach up the DAG through node's parents to a common ancestor of both source and destination, and then follow the DAG links down to the destination. In non-storing mode, packets reach the LBR through the parents, and the LBR creates a SRH describing which nodes to traverse downwards the DAG to reach the destination, as illustrated in Figure 2.3. Clearly, RPL defines a very sub-optimal route in terms of path length for P2P traffic. However, since the majority of the traffic in LLN is either to or from the DAG root, this DAG based routing can be viewed as a trade-off between path optimality and memory required to store routing tables in constrained devices.

2.2 Summary

In this chapter we summarize RPL's operation. For a detailed description the interested reader should consult RFC 6550 (15).

3

RPL: Performance Evaluation

This chapter's contribution is to provide details on our performance evaluation study of RPL with respect to several metrics of interest, standardized in IETF RFC 6687 (18). This was accomplished using real data and topologies in a discrete event simulator developed to reproduce the protocol behavior.

After surveying existing protocols and determining their unsuitability, the IETF ROLL WG started designing RPL. Joining the WG efforts, we developed a very detailed RPL simulator and using topology and traffic traces from existing networks, contributed with a performance study of the protocol with respect to several metrics of interest, such as path quality, end-to-end delay, control plane overhead, ability to cope with instability, etc. This work was recently standardized as IETF Informational RFC 6687 (18). Although simulation cannot prove formally that a protocol operates properly in all situations, it can give a good level of confidence in protocol behavior in highly stressful conditions, when real-life data are used. Simulation is particularly useful when theoretical model assumptions may not be applicable to such networks and scenarios. In this evaluation chapter, real deployed network data traces have been used to model link behaviors and network topologies.

3.1 Methodology and Simulation Setup

In the context of this document, RPL has been simulated using OMNeT++ (36), a well-known discrete event-based simulator written in C++ and NEtwork Description (NED). Castalia-2.2 (37) has been used as a Wireless Sensor Network Simulator framework within OMNeT++. The output and events in the simulation are visualized with the help of the Network AniMator, or NAM, which is distributed with the NS (Network Simulator). Note that no versions of the NS itself are used in this simulation study. Only the visualization tool was borrowed for verification purposes. Example of such visualization can be seen in Figure 3.1 and in Figure 3.2 in this section.

In contrast with theoretical models, which may have assumptions not applicable to lossy links, real-life data was used for two aspects of the simulations:

- Link Failure Model: Derived from time-varying real network traces containing packet delivery probability for each link, over all channels, for both indoor network deployment and outdoor network deployment.
- Topology: Gathered from real-life deployment (traces mentioned above) as opposed to random topology simulations.

A 45-node topology, deployed as an outdoor network and shown in Figure 3.1, and a 2442-node topology, gathered from a smart meter network deployment, were used in the simulations. For scalability and comparative analysis in this chapter and the next, another outdoor depoyed network with 86 nodes as shown in Figure 3.2, has been used. In Figure 3.1 and in Figure 3.2, links between a most preferred parent node and child nodes are shown in red. Links that are shown in black are also part of the topology but are not between a preferred parent and child node. Note that this is just a start to validate the simulation before using large-scale networks.

A set of time-varying link quality data was gathered from a real network deployment to form a database used for the simulations. Each link in the topology randomly 'picks up' a link model (trace) from the database. Each link has a Packet Delivery Ratio (PDR)

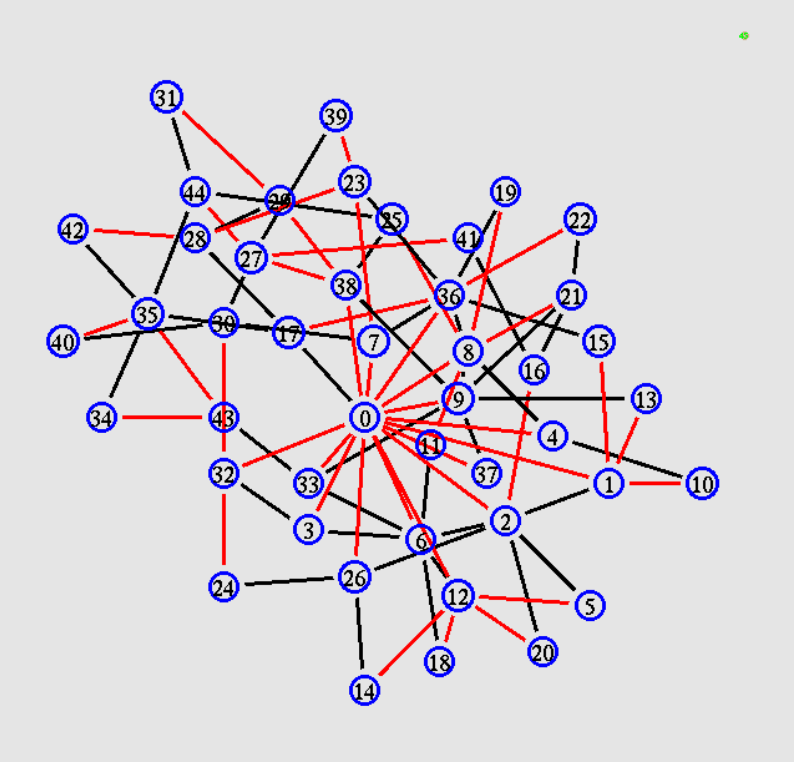

Figure 3.1: Outdoor Network Topology with 45 Nodes.

that varies with time (in the simulation, a new PDR is read from the database every 10 minutes) according to the gathered data. Packets are dropped randomly from that link with probability (1− PDR). Each time a packet is about to be sent, the module generates a random number using the Mersenne Twister random number generation method.

The random number is compared to the PDR to determine whether the packet should be dropped. Note that each link uses a different random number generator to maintain true randomness in the simulator and to avoid correlation between links. Also, the packet drop applies to all kinds of data and control packets (RPL), such as the DIO, DAO, and DIS packets defined in the RPL standard (15). Figure 3.3 shows a typical temporal characteristic of links from the outdoor network shown in Figure 3.2 and is used in the simulations. The figure shows several links with perfect connectivity, some links with a PDR as low as 10%, and several for which the PDR may vary from 30% to 80%, sharply changing back and forth between a high value (strong connectivity) and a low value (weak

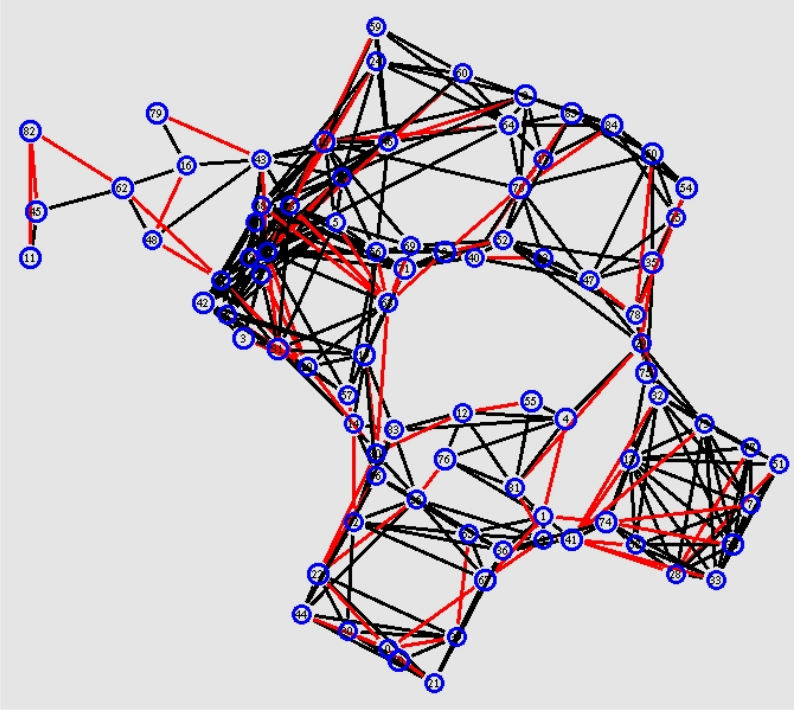

Figure 3.2: Outdoor Network Topology with 86 Nodes.

connectivity). For the large smart grid networks, however, the link temporal variation has been found less than the outdoor network. Nevertheles, we use the link data corresponding only to the particular deployment to preserve authenticity of the simulated network.

In the RPL simulator, the LBR (LLN Border Router) or the Directed Acyclic Graph (DAG) root first initiates sending out DIO messages, and the DAG is gradually constructed. RPL makes use of trickle timers: the protocol sets a minimum time period with which the nodes start re-issuing DAOs, and this minimum period is denoted by the trickle parameter $Imin$. RPL also sets an upper limit on how many times this time period can be doubled; this is denoted by the parameter $DIOIntervalDoublings$, as defined in RFC 6550 (15). For the simulation, $Imin$ is initially set to 1 second and $DIOIntervalDoublings$ is equal to 16, and therefore the maximum time between two consecutive DIO emissions by a node (under a steady network condition) is 18.2 hours. The trickle time interval for emitting DIO messages assumes the initial value of 1 second and then changes over simulation time, as mentioned in RFC 6206 (38). Another objective of this study is to give insight

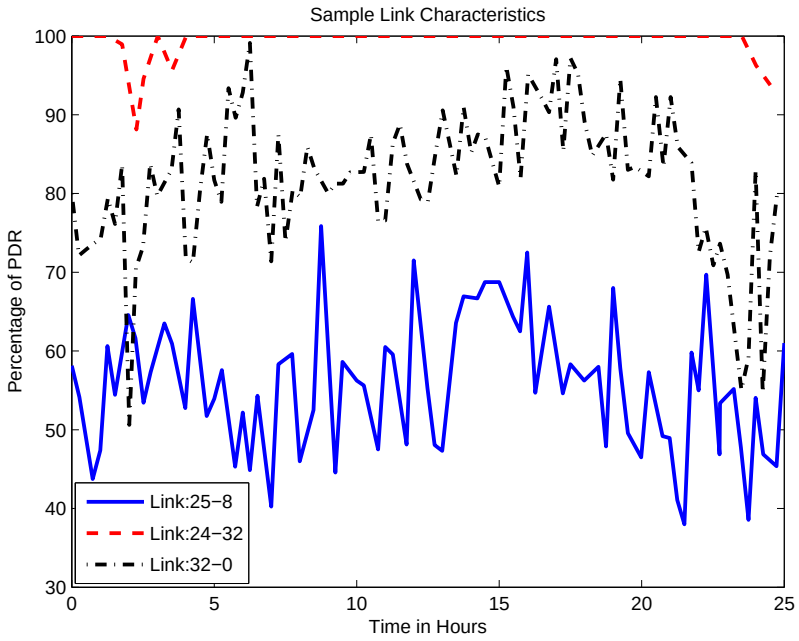

Figure 3.3: Example of Link Characteristics.

to the network administrator on how to tweak the trickle values. These recommendations could then be used in applicability statement documents.

Each node in the network, other than the LBR or DAG root, also emits DAO messages as specified in the RPL standard (15), to initially populate the routing tables with the prefixes received from children via the DAO messages to support Point-to-Point (P2P) and Point-to-Multipoint (P2MP) traffic in the "down" direction. During these simulations, it is assumed that each node is capable of storing route information for other nodes in the network (storing mode of RPL).

For nodes implementing RPL, as expected, the routing table memory requirement varies according to the position in the DODAG (Destination-Oriented DAG). The (worst-case) assumption is made that there is no route summarization (aggregation) in the network. Thus, a node closer to the DAG will have to store more entries in its routing table. It is also assumed that all nodes have equal memory capacity to store the routing states.

For simulations of the indoor network, each node sends traffic according to a Constant Bit Rate (CBR) to all other nodes in the network, over the simulation period. Each node

generates a new data packet every 10 seconds. Each data packet has a size of 127 bytes including 802.15.4 PHY/MAC headers and RPL packet headers. All control packets are also encapsulated with 802.15.4 PHY/MAC headers. To simulate a more realistic scenario, 80% of the packets generated by each node are destined to the root, and the remaining 20% of the packets are uniformly assigned as destined to nodes other than the root. Therefore, the root receives a considerably larger amount of data than other nodes. These values may be revised when studying P2P traffic so as to have a majority of traffic going to all nodes as opposed to the root. In the later part of the simulation, a typical home/building routing scenario is also simulated, and different path quality metrics are computed for that traffic pattern. The packets are routed through the DODAG built by RPL according to the mechanisms specified in RFC 6550 (15).

A number of RPL parameters are varied (such as the packet rate from each source and the time period for emitting a new DAG sequence number) to observe their effect on the performance metric of interest.

3.1.1 Common assumptions

As the DAO messages are used to feed the routing tables in the network, they grow with time and size of the network. Nevertheless, no constraint was imposed on the size of the routing table nor on how much information the node can store. The routing table size is not expressed in terms of Kbytes of memory usage but measured in terms of the number of entries for each node. Each entry has the next-hop node and path cost associated with the destination node. The link ETX (Expected Transmission Count) metric is used to build the DODAG and is specified in RFC 6551 (35).

3.2 Performance Evaluation in Small Network

3.2.1 Path quality

Hop Count: For each source-destination pair, the number of hops for both RPL and shortest path routing is computed. Shortest path routing refers to a hypothetical

ideal routing protocol that would always provide the shortest path in terms of ETX path cost (or whichever metric is used) in the network.

The Cumulative Distribution Function (CDF) of the hop count for all paths ($n*(n-1)$ in an n-node network) in the network with respect to the hop count is plotted in Figure 3.4 for both RPL and shortest path routing. One can observe that the CDF corresponding to 4 hops is around 80% for RPL and 90% for shortest path routing. In other words, for the given topology, 90% of the paths have a path length of 4 hops or less with an ideal shortest path routing methodology, whereas in RPL P2P routing, 90% of the paths will have a length of no more than 5 hops. This result indicates that despite having a non-optimized P2P routing scheme, the path quality of RPL is close to an optimized P2P routing mechanism for the topology and the traffic under consideration. Another reason for this may relate to the fact that the DAG root is at the center of the network shown in Figure 3.1; thus, routing through the DAG root is often close to an optimal (shortest path) routing. This result may be different in a topology where the DAG root is located at one end of the network.

ETX Path Cost: In the simulation, the total ETX path cost (Expected Transmission Count) from source to destination for each packet is computed.

Figure 3.5 shows the CDF of the total ETX path cost, both with RPL and shortest path routing. Here also one can observe that the ETX path cost from all sources to all destinations is close to that of shortest path routing for the network.

Path Stretch: The path stretch metric encompasses the stretch factor for both hop distance and ETX path cost. The hop distance stretch, which is determined as the difference between the number of hops taken by a packet while following a route built via RPL and the number of hops taken by shortest path routing (using link ETX as the metric), is computed. The ETX path cost stretch is also provided.

The CDF of both path stretch metrics is plotted against the value of the corresponding path stretch over all packets in Figures 3.6 and 3.7, for hop distance stretch and

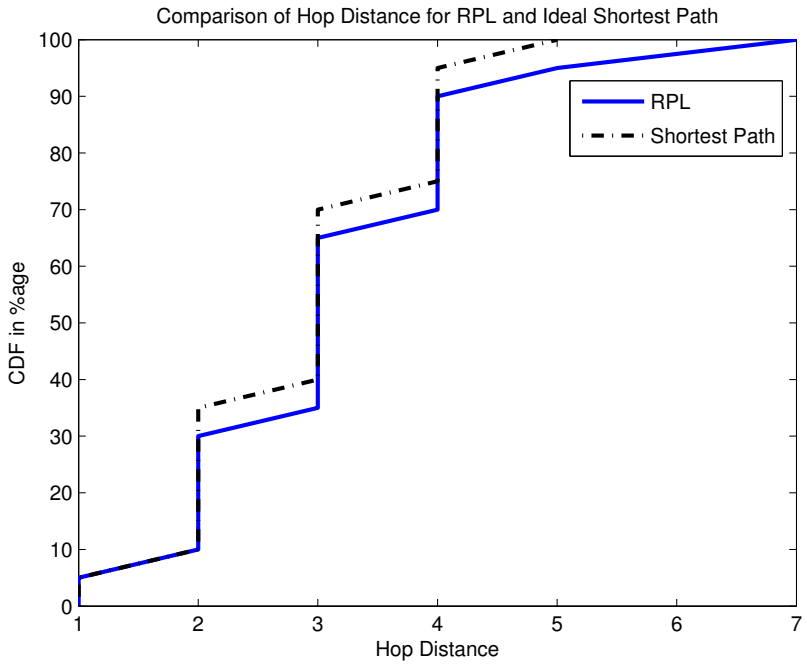

Figure 3.4: CDF of Hop Count versus Hop Count.

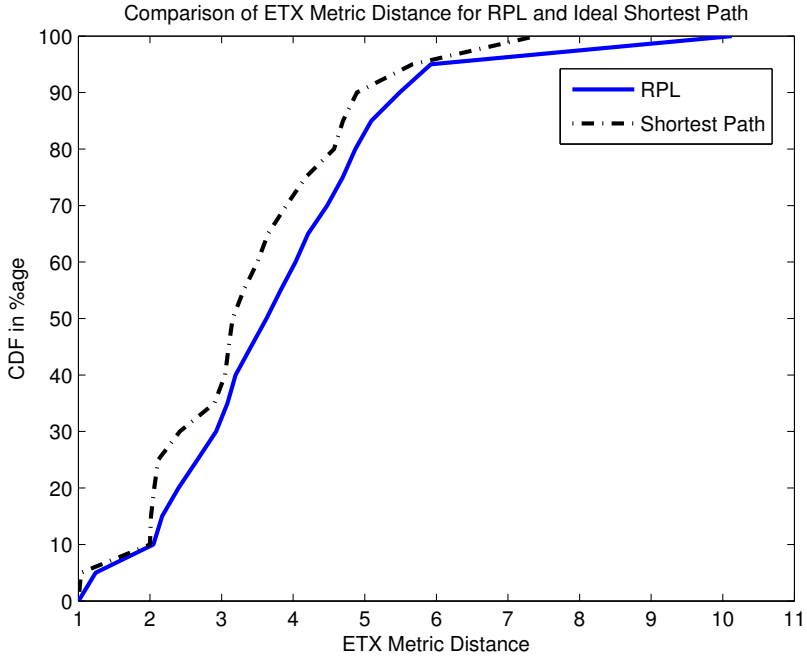

Figure 3.5: CDF of Total ETX Path Cost along Path versus ETX Path Cost.

ETX path stretch, respectively. It can be observed that, for a few packets, the path built via RPL has fewer hops than the ideal shortest path where path ETX is minimized along the DAG. This is because there are a few source-destination pairs where the total ETX path cost is equal to or less than that of the path provided by RPL, when the packet takes a longer hop count for the ideal shortest path. As the RPL implementation ignores a 20% change in total ETX path cost before switching to a new parent or emitting a new DIO, it does not necessarily provide the shortest path in terms of total ETX path cost. Thus, this implementation yields a few paths with smaller hop counts but larger (or equal) total ETX path cost.

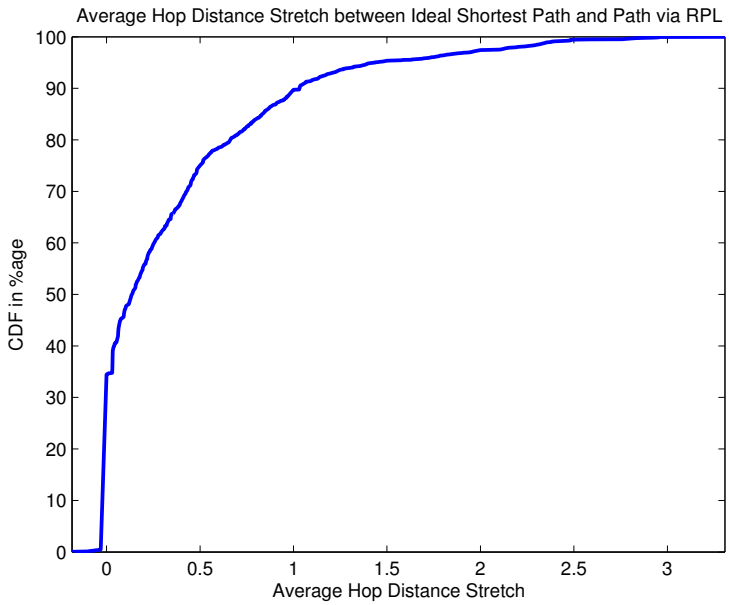

Figure 3.6: CDF of Hop Distance Stretch versus Hop Distance Stretch Value.

The data for the CDF of the hop count and ETX path cost for the ideal shortest path (SP) and a path built via RPL, along with the CDF of the routing table size, is given below in Table 3.1. Figures 3.4 to 3.8 relate to the data in this table.

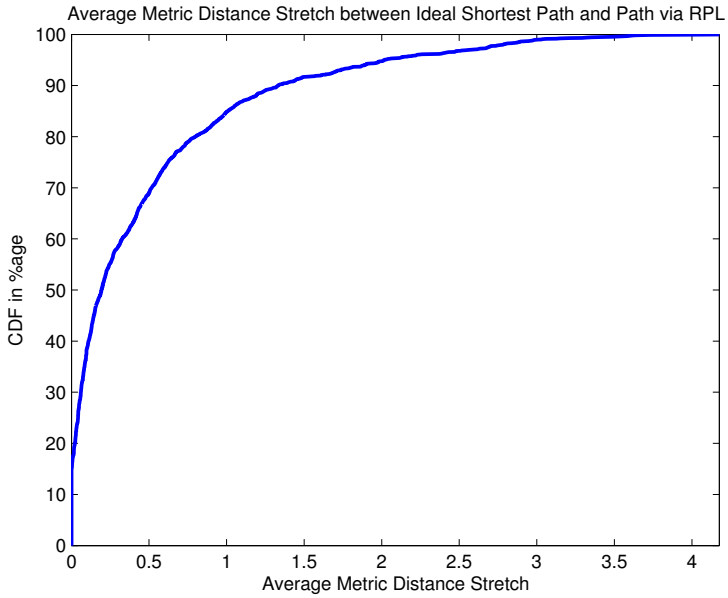

Figure 3.7: CDF of ETX Path Stretch versus ETX Path Stretch Value.

Overall, the path quality metrics give us important information about the protocol's performance when minimizing the ETX path cost is the objective to form the DAG. The protocol, as explained, does not always provide an optimum path, especially for peer-to-peer communication. However, it does end up reducing the control overhead cost, thereby reducing unnecessary parent selection and DIO message forwarding events, by choosing a non-optimized path. Despite this specific implementation technique, around 30% of the packets travel the same number of hops as an ideal shortest path routing mechanism, and 20% of the packets experience the same number of attempted transmissions to reach the destination. On average, this implementation costs only a few extra transmission attempts and saves a large number of control packet transmissions.

3.2.2 Routing table size

The objective of this metric is to observe the distribution of the number of entries per node. Figure 3.8 shows the CDF of the number of routing table entries for all nodes. Note that 90% of the nodes need to store less than 10 entries in their routing table for the topology under study. The LBR does not have the same power or memory constraints as

CDF (%age)	Hop (SP)	Hop (RPL)	ETX Cost (SP)	ETX Cost (RPL)	Routing Table Size
0	1.0	1.0	1	1.0	0
5	1.0	1.03	1	1.242	1
10	2.0	2.0	2	2.048	2
15	2.0	2.01	2	2.171	2
20	2.0	2.06	2	2.400	2
25	2.0	2.11	2	2.662	3
30	2.0	2.42	2	2.925	3
35	2.0	2.90	3	3.082	3
40	3.0	3.06	3	3.194	4
45	3.0	3.1	3	3.41	4
50	3.0	3.15	3	3.626	4
55	3.0	3.31	3	3.823	5
60	3.0	3.50	3	4.032	6
65	3.0	3.66	3	4.208	7
70	3.0	3.92	4	4.474	7
75	4.0	4.16	4	4.694	7
80	4.0	4.55	4	4.868	8
85	4.0	4.70	4	5.091	9
90	4.0	4.89	4	5.488	10
95	4.0	5.65	5	5.923	12
100	5.0	7.19	9	10.125	44

Table 3.1: Path Quality CDFs.

regular nodes do, and hence it can accommodate entries for all the nodes in the network. The requirement to accommodate devices with low storage capacity has been mandated for Industrial, Home and Building Automation LLNs in RFC 5673, RFC 5826, and RFC 5867 (5, 6, 16). However, when RPL is implemented in storing mode, some nodes closer to the LBR or DAG root will require more memory to store larger routing tables. To implement storing mode while deploying RPL, in this case will need to accommodate maximum routing table size for all nodes but the LBR. One can also implement a mixture of storing and non-storing mode of implementation, but it is outside the scope of this chapter to discuss the pathogy arising due to a topology of mixed Mode of Operations (MoP)

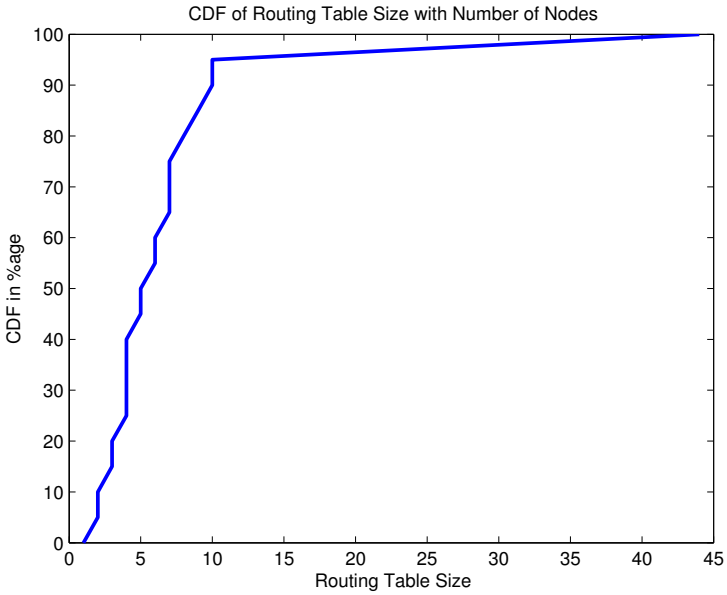

Figure 3.8: CDF of Routing Table Size with Respect to Number of Nodes.

3.2.3 Delay bound for P2P routing

For delay-sensitive applications, such as home and building automation, it is critical to optimize the end-to-end delay. Figure 3.9 shows the upper bound and distributions of delay for paths between any two given nodes for different hop counts between the source and destination. Here, the hop count refers to the number of hops a packet travels to reach the destination when using RPL paths. This hop distance does not correspond to the shortest path distance between two nodes. Note that each packet has a length of 127 bytes, with a 240-kbps radio, which makes the transmission delay approximately 4 milliseconds (ms).

Industrial and Urban LLN requirements in RFCs 5673 and 5548 (6, 7) mention a requirement for the end-to-end delivery delay to remain within a bounded latency. For instance, according to the industrial routing requirement, non-critical closed-loop applications may have a latency requirement that can be as low as 100 ms, whereas monitoring services may tolerate a delay in the order of seconds. The results show that about 99% of the end-to-end communication (where the maximum hop count is 7 hops) is bounded

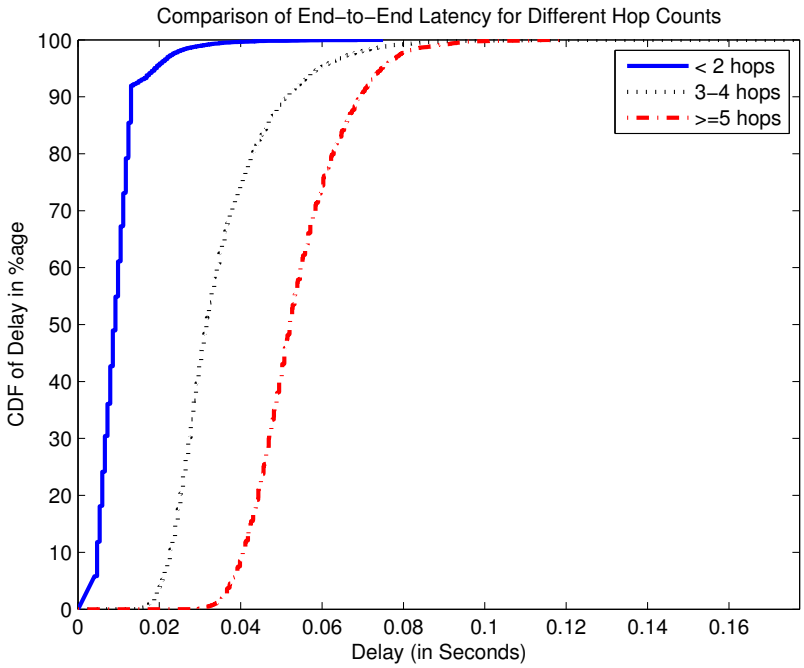

Figure 3.9: Comparison of Packet Latency, for Different Path Lengths, Expressed in Hop Count.

within the 100-ms requirement, for the topology under study. It should be noted that due to poor link condition, there may be packet drops triggering retransmission, which may cause larger end-to-end delivery delays. Nodes in the proximity of the LBR may become congested at high traffic loads, which can also lead to higher end-to-end delay.

3.2.4 Control packet overhead

The control plane overhead is an important routing characteristic in LLNs. It is imperative to bound the control plane overhead. One of the distinctive characteristics of RPL is that it makes use of trickle timers so as to reduce the number of control plane packets by eliminating redundant messages. The aim of this performance metric is thus to analyze the control plane overhead both in stable conditions (no network element failure overhead) and in the presence of failures.

Data and control plane traffic comparison for each node: Figure 3.10 shows the comparison between the amount of data packets transmitted (including forwarded packets)

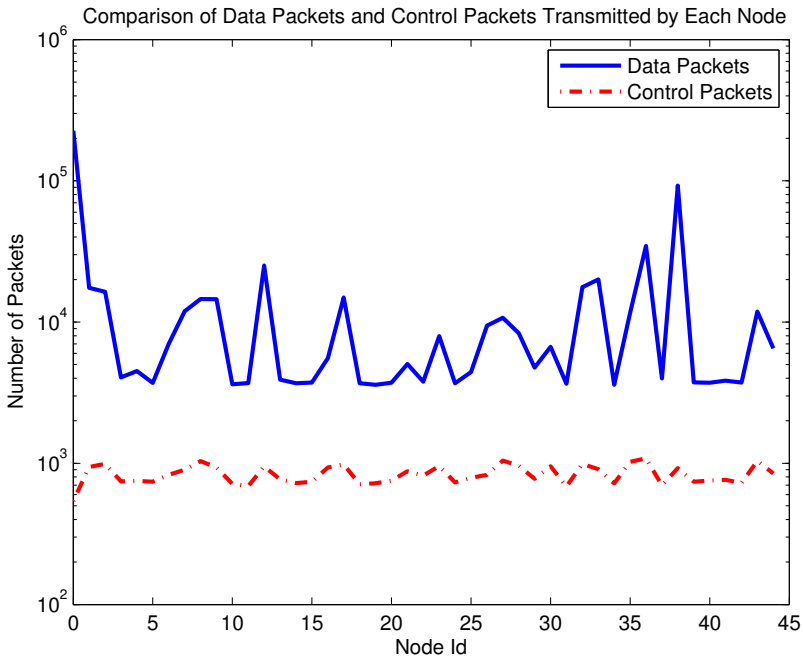

Figure 3.10: Amount of Data and Control Packets Transmitted against Node Id Using Link ETX as Routing Metric.

and control packets (DIO and DAO messages) transmitted for all individual nodes when link ETX is used to optimize the DAG. As mentioned earlier, each node generates a new data packet every 10 seconds. Here one can observe that a considerable amount of traffic is routed through the DAG root itself. The x axis indicates the node ID in the network. Also, as expected, the nodes that are closer to the DAG root and that act as routers (as opposed to leaves) handle much more data traffic than other nodes. Nodes 12, 36, and 38 are examples of nodes next to the DAG root, taking part in routing most of the data packets and hence having many more data packet transmissions than other nodes, as observed in Figure 3.10. We can also observe that the proportion of control traffic is negligible for those nodes. This result also reinforces the fact that the amount of control plane traffic generated by RPL is negligible on these topologies. Leaf nodes have comparable amounts of data and control packet transmissions (they do not take part in routing the data).

Data and control packet transmission with respect to time: In Figures 3.11, 3.12, and

3.13, the amount of data and control packets transmitted for node 12 (low rank in DAG, closer to the root), node 43 (in the middle), and node 31 (leaf node) are shown, respectively. These values stand for the number of data and control packets transmitted for each 10-minute interval for the particular node, to help understand what the ratio is between data and control packets exchanged in the network. One can observe that nodes closer to the DAG root have a higher proportion of data packets (as expected), and the proportion of control traffic is negligible in comparison with the data traffic. Also, the amount of data traffic handled by a node within a given interval varies largely over time for a node closer to the DAG root, because in each interval the destination of the packets from the same source changes, while 20% of the packets are destined to nodes other than the DAG root. As a result, the pattern of the traffic that is handled changes widely in each interval for the nodes closer to the DAG root. For the nodes that are farther away from the DAG root, the ratio of data traffic to control traffic is smaller, since the amount of data traffic is greatly reduced.

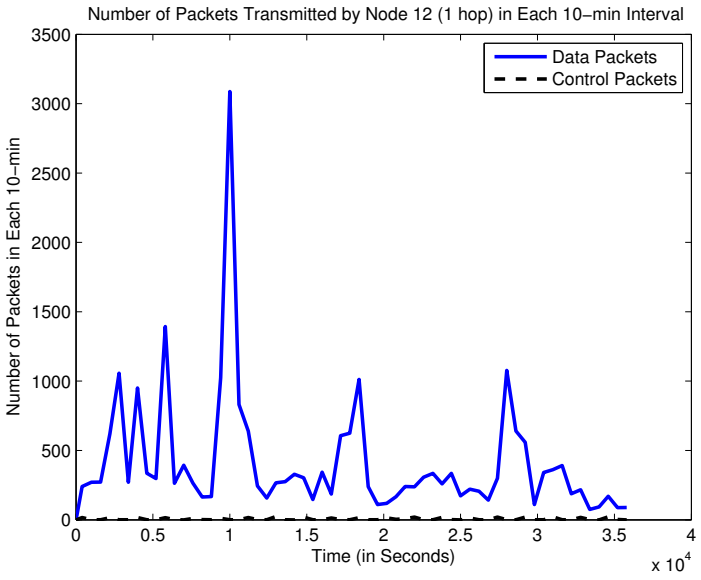

Figure 3.11: Amount of Data and Control Packets Transmitted for Node 12.

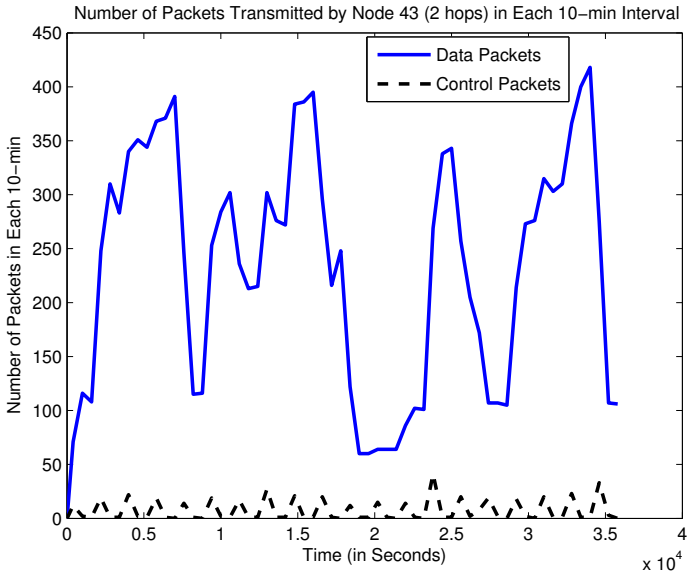

Figure 3.12: Amount of Data and Control Packets Transmitted for Node 43.

The control traffic load exhibits a wave-like pattern with respect to time. The amount of control packets for each node drops quickly as the DODAG stabilizes, due to the effect of trickle timers. However, when a new DODAG sequence is advertised (global repair of the DODAG), the trickle timers are reset and the nodes start emitting DIOs frequently again to rebuild the DODAG. For a node closer to the DAG root, the amount of data packets is much larger than that of control packets and somewhat oscillatory around a mean value. The amount of control packets exhibits a 'saw-tooth' behavior. In the case where the ETX link metric is used, when the PDR changes, the ETX link metric for a node to its child changes, which may lead to choosing a new parent and changing the DAG rank of the child. This event resets the trickle timer and triggers the emission of a new DIO. Also, the issue of a new DODAG sequence number triggers DODAG re-computation and resets the trickle timers. Therefore, one can observe that the number of control packets attains a high value for one interval and comes down to lower values for subsequent intervals. The interval with a high number of control packets denotes the interval where the timers to emit a new DIO are reset more frequently. As the network stabilizes, the control packets

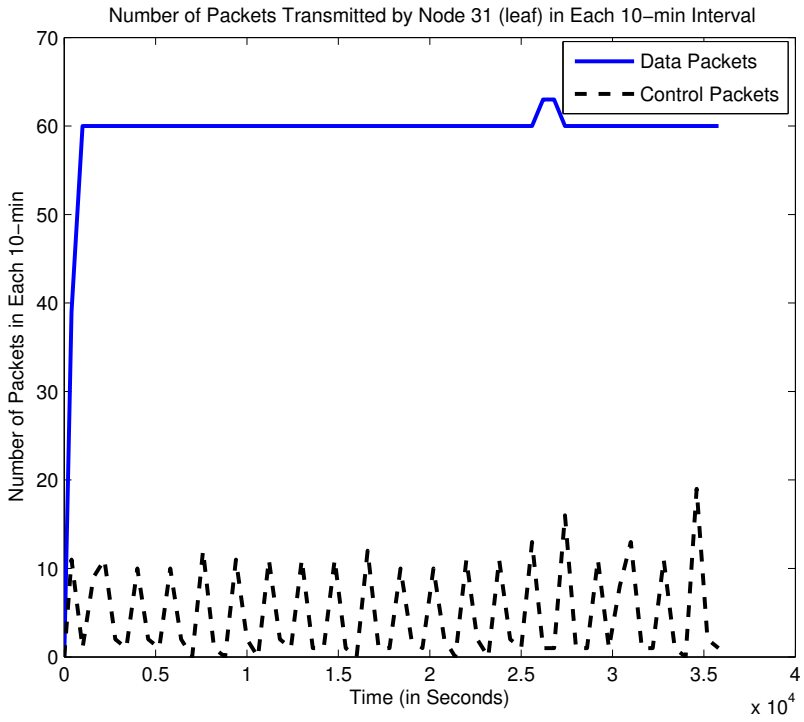

Figure 3.13: Amount of Data and Control Packets Transmitted for Node 31.

are less dense in volume. For leaf nodes, the amount of control packets is comparable to that of data packets, as leaf nodes are more prone to face changes in their DODAG rank as opposed to nodes closer to the DAG root when the link ETX value in the topology changes dynamically. At the same time, the amount of data packet handled by leaf nodes depends only on data generated by themselves, thus smaller in compared to that handled by a node in the middle of the DAG.

3.2.5 Loss of connectivity

Upon link failures, a node may lose its parents – preferred and backup (if any) – thus leading to a loss of connectivity (no path to the DAG root). RPL specifies two mechanisms for DODAG repairs, referred to as global repair and local repair. In this section, simulation results are presented to evaluate the amount of time data packets are dropped due to a loss of connectivity. The following two scenarios of maintenance mechanisms are considered:

a) when only using global repair (i.e., the DODAG is rebuilt thanks to the emission of new DODAG sequence numbers by the DAG root), and b) when using local repair (poisoning the sub-DAG in case of loss of connectivity) in addition to global repair. The idea is to tune the frequency at which new DODAG sequence numbers are generated by the DAG root, and also to observe the effect of varying the frequency for global repair and the concurrent use of global and local repair. It is expected that more frequent increments of DODAG sequence numbers will lead to a shorter duration of connectivity loss at a price of a higher rate of control packets in the network. For the use of both global and local repair, the simulation results show the trade-off in amount of time that a node may remain without service and total number of control packets.

Figure 3.14 shows the CDF of time spent by any node without service, when the data packet rate is one packet every 10 seconds and a new DODAG sequence number is generated every 10 minutes. This plot reflects the property of global repair without any local repair scheme. When all the parents are temporarily unreachable from a node, the time before it hears a DIO from another node with a path to the DAG root is recorded, which gives the time without service. We define the DAG repair timer as the interval at which the LBR increments the DAG sequence number, thus triggering a global re-optimization. In some cases, this value might go up to the DAG repair timer value, because until a DIO is heard, the node does not have a parent and hence no route to the LBR or other nodes not in its own sub-DAG. Clearly, this situation indicates a lack of connectivity and loss of service for the node.

The effect of the DAG repair timer on time without service is plotted in Figure 3.15, where the source rate is 20 seconds/packet and in Figure 3.16, where the source sends a packet every 10 seconds.

The data for Figures 3.14 and 3.16 can be found in Table 3.2. The table shows how the CDF of time without connectivity to the LBR increases while we increase the time period to emit new DAG sequence numbers, when the nodes generate a packet every 10 seconds.

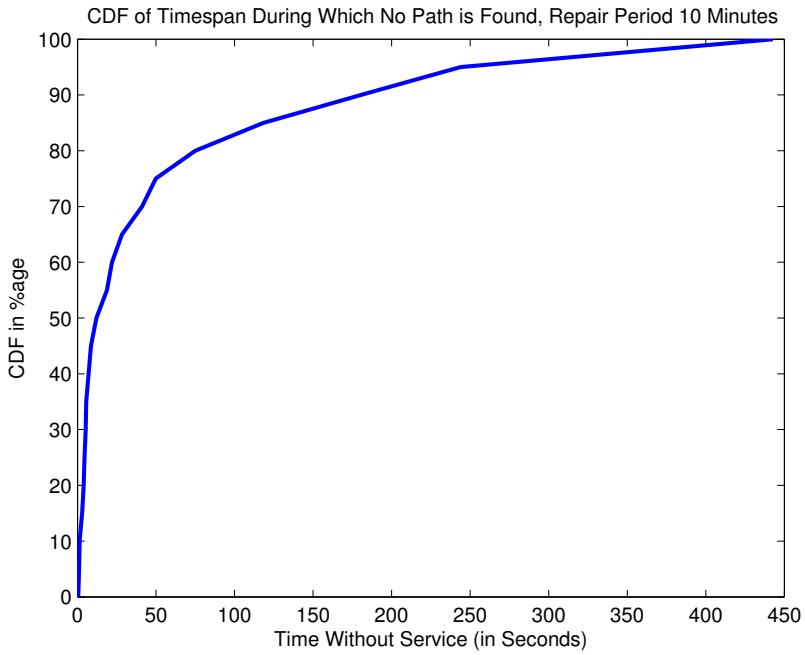

Figure 3.14: CDF: Loss of Connectivity with Global Repair.

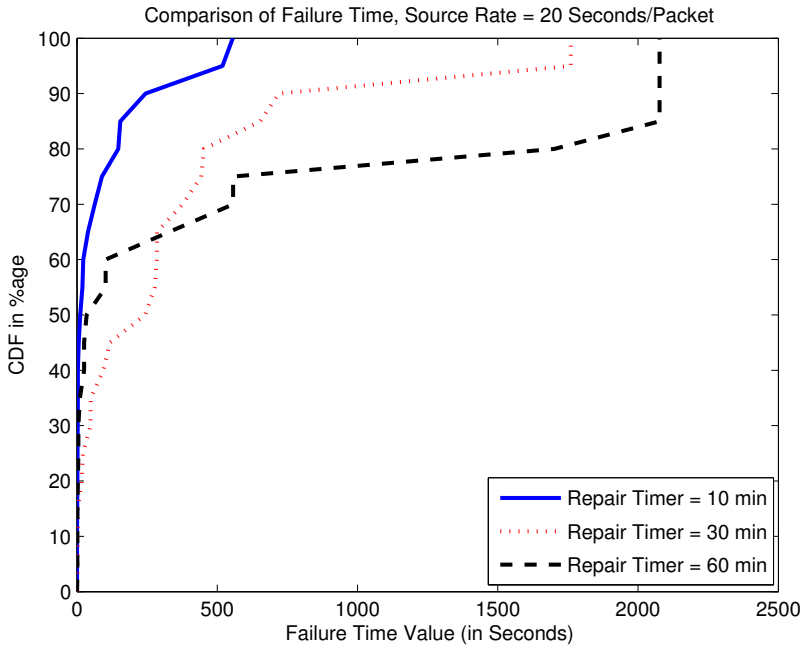

Figure 3.15: CDF: Loss of Connectivity for Different Global Repair Period, Source Rate 20 Seconds/Packet.

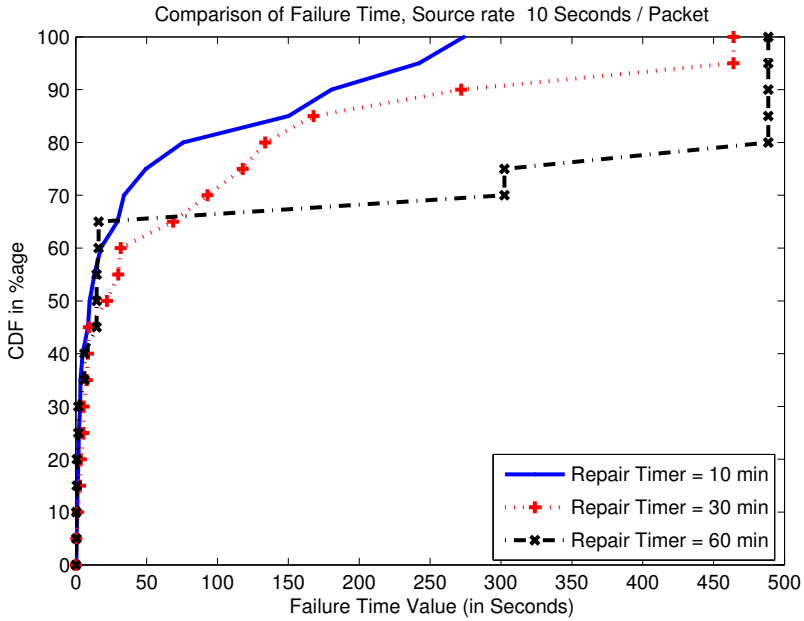

Figure 3.16: CDF: Loss of Connectivity for Different Global Repair Period, Source Rate 10 Seconds/Packet.

The data for Figure 3.15 can be found in Table 3.3. The table shows how the CDF of time without connectivity to the LBR increases while we increase the time period to emit new DAG sequence numbers, when the nodes generate a packet every 20 seconds.

Figure 3.17 shows the effect of the DAG global repair timer period on control traffic. As expected, as the frequency at which new DAG sequence numbers are generated increases, the amount of control traffic decreases because DIO messages are sent less frequently to rebuild the DODAG. However, reducing the control traffic comes at a price of increased loss of connectivity when only global repair is used.

From the above results, it is clear that the time the protocol takes to re-establish routes and to converge, after an unexpected link or device failure happens, is fairly long. Home automation routing requirements in RFC 5826 mandates that "the routing protocol MUST converge within 0.5 seconds if no nodes have moved". Clearly, implementation of a repair mechanism based on new DAG sequence numbers alone would not meet the requirements. Hence, a local repair mechanism, in the form of poisoning the sub-DAG and issuing a DIS, has been adopted.

CDF (%age)	Repair Period 10 Minutes	Repair Period 30 Minutes	Repair Period 60 Minutes
0	0.464	0.045	0.027
5	0.609	0.424	0.396
10	1.040	1.451	0.396
15	1.406	3.035	0.714
20	1.934	3.521	0.714
25	2.113	5.461	1.856
30	3.152	5.555	1.856
35	3.363	7.756	6.173
40	4.9078	8.604	6.173
45	8.575	9.181	14.751
50	9.788	21.974	14.751
55	13.230	30.017	14.751
60	17.681	31.749	16.166
65	29.356	68.709	16.166
70	34.019	92.974	302.459
75	49.444	117.869	302.459
80	75.737	133.653	488.602
85	150.089	167.828	488.602
90	180.505	271.884	488.602
95	242.247	464.047	488.602
100	273.808	464.047	488.602

Table 3.2: Loss of Connectivity Time, Data Rate - 10 Seconds / Packet.

The effect of the DAG repair timer on time without service when local repair is activated is now observed and plotted in Figure 3.18, where the source rate is 20 seconds/packet. A comparison of the CDF of loss of connectivity for the global repair mechanism and the global + local repair mechanism is shown in Figures 3.19 and 3.20 (semi-log plots, x axis in logarithmic scale and y axis in linear scale), where the source generates a packet every 10 seconds and 20 seconds, respectively. For these plots, the x axis shows time in log scale, and the y axis denotes the corresponding CDF in linear scale. One can observe that using local repair (with poisoning of the sub-DAG) greatly reduces loss of connectivity.

A comparison between the amount of control plane overhead used for global repair only and for the global plus local repair mechanism is shown in Figure 3.21, which highlights the

CDF (%age)	Repair Period 10 Minutes	Repair Period 30 Minutes	Repair Period 60 Minutes
0	0.071	0.955	0.167
5	0.126	2.280	1.377
10	0.403	2.926	1.409
15	0.902	3.269	1.409
20	1.281	16.623	3.054
25	2.322	21.438	5.175
30	2.860	48.479	5.175
35	3.316	49.495	10.30
40	3.420	93.700	25.406
45	6.363	117.594	25.406
50	11.500	243.429	34.379
55	19.703	277.039	102.141
60	22.216	284.660	102.141
65	39.211	285.101	328.293
70	63.197	376.549	556.296
75	88.986	443.450	556.296
80	147.509	452.883	1701.52
85	154.26	653.420	2076.41
90	244.241	720.032	2076.41
95	518.835	1760.47	2076.41
100	555.57	1760.47	2076.41

Table 3.3: Loss of Connectivity Time, Data Rate - 20 Seconds / Packet.

improved performance of RPL in terms of convergence time at very little extra overhead. From Figure 3.20, in 85% of the cases the protocol finds connectivity to the LBR for the concerned nodes within a fraction of seconds when local repair is employed. Using only global repair leads to repair periods of 150-154 seconds, as observed in Figures 3.14 and 3.15.

3.3 RPL in a Building Automation Routing Scenario

Unlike the previous traffic pattern, where a majority (80%) of the total traffic generated by any node is destined to the root, this section considers a different traffic pattern, which is more prominent in a home or building routing scenario. In the simulations shown below, the nodes send 60% of their total generated traffic to the physically 1-hop distant node

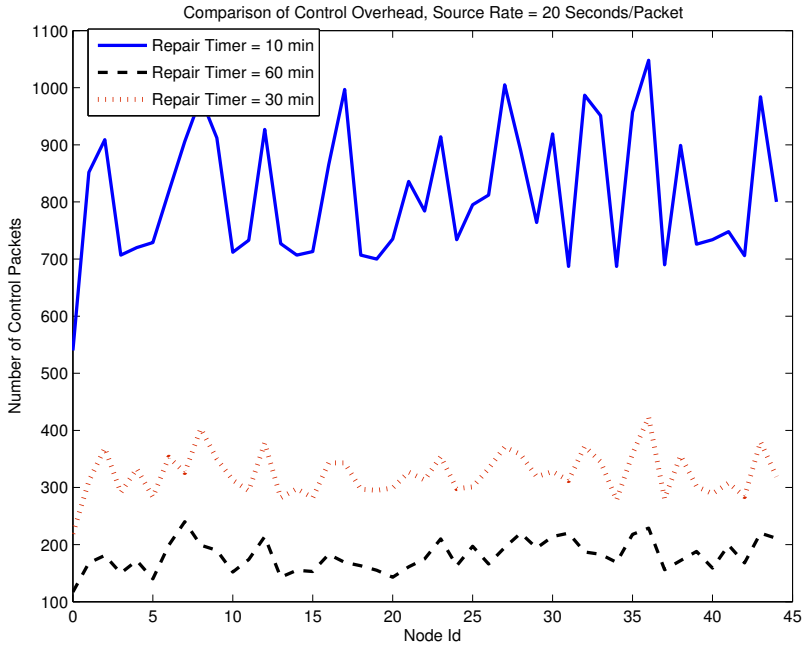

Figure 3.17: Amount of Control Traffic for Different Global Repair Periods.

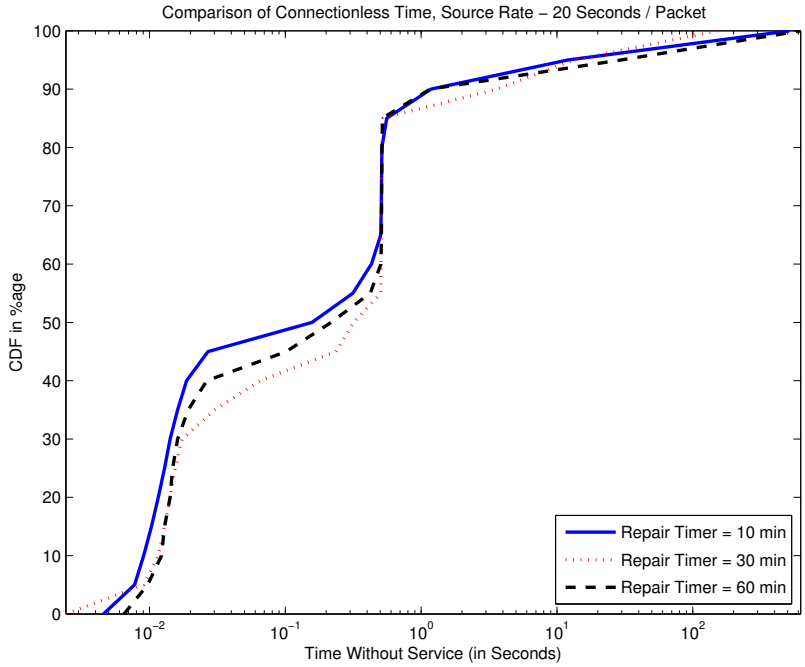

Figure 3.18: CDF: Loss of Connectivity for Different DAG Repair Timer Values for Global+Local Repair, Source Rate 20 Seconds/Packet.

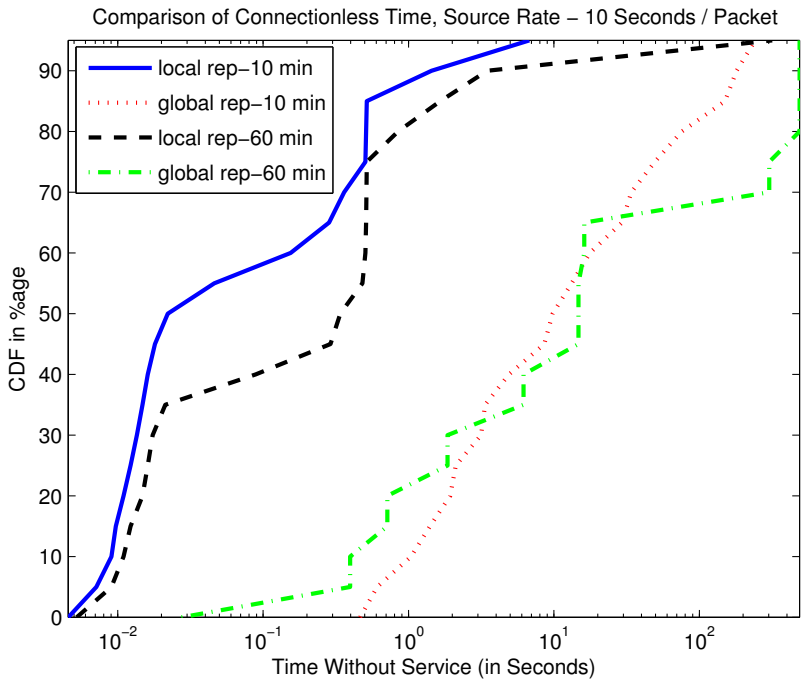

Figure 3.19: CDF: Loss of Connectivity for Global Repair and Global+Local Repair, Source Rate 10 Seconds/Packet.

and 20% of traffic to a 2-hop distant node; the other 20% of traffic is distributed among other nodes in the network. The CDF of path quality metrics such as hop count, ETX path cost, average hop distance stretch, ETX path stretch, and delay for P2P routing for all pairs of nodes is calculated. Maintaining a low delay bound for P2P traffic is of high importance, as applications in home and building routing typically have low delay tolerance.

3.3.1 Path quality

Figure 3.22 shows the CDF of the hop count for both RPL and ideal shortest path routing for the traffic pattern described above. Figure 3.23 shows the CDF of the expected number of transmissions (ETX) for each packet to reach its destination. Figures 3.24 and 3.25 show the CDF of the stretch factor for these two metrics. To illustrate the stretch factor, an example from Figure 3.25 will be given next. For all paths built by RPL, 85% of the time, the path cost is less than the path cost for the ideal shortest path plus one.

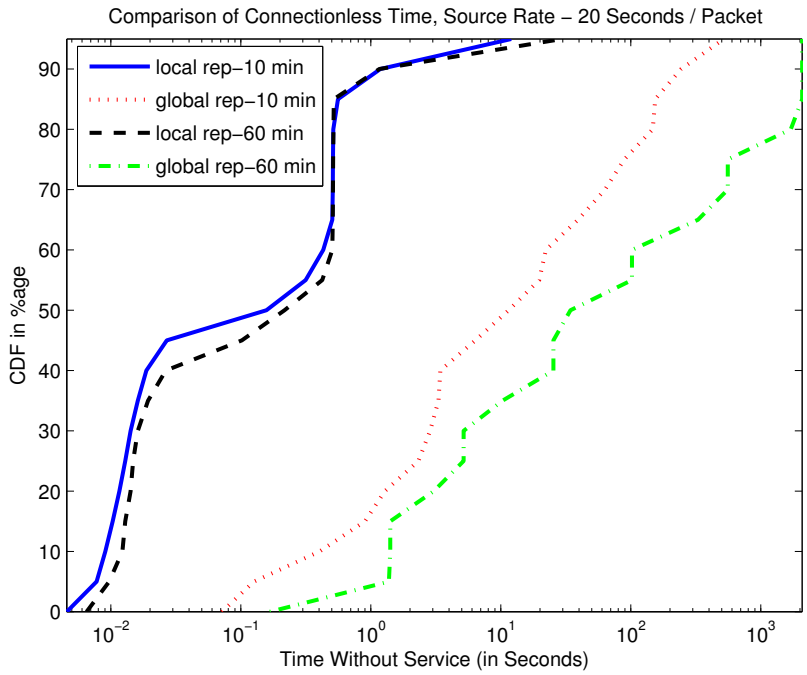

Figure 3.20: CDF: Loss of Connectivity for Global Repair and Global+Local Repair, Source Rate 20 Seconds/Packet.

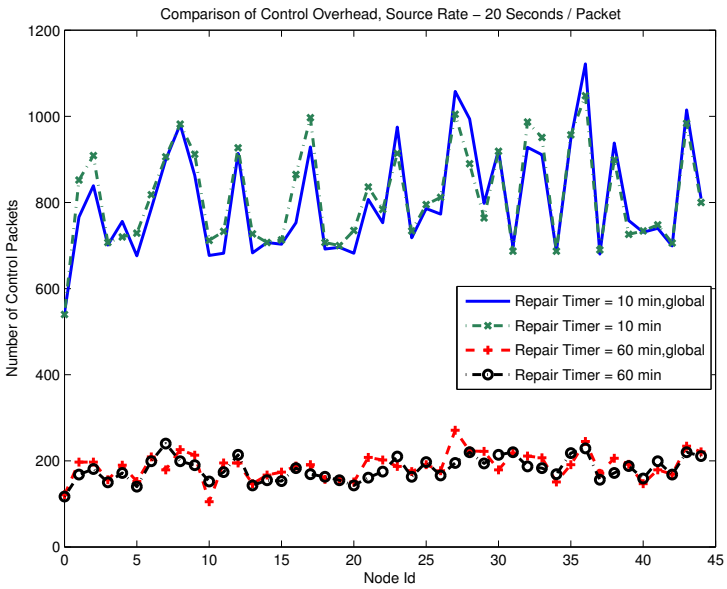

Figure 3.21: Number of Control Packets for Different DAG Sequence Number Period, for Both Global Repair and Global+Local Repair.

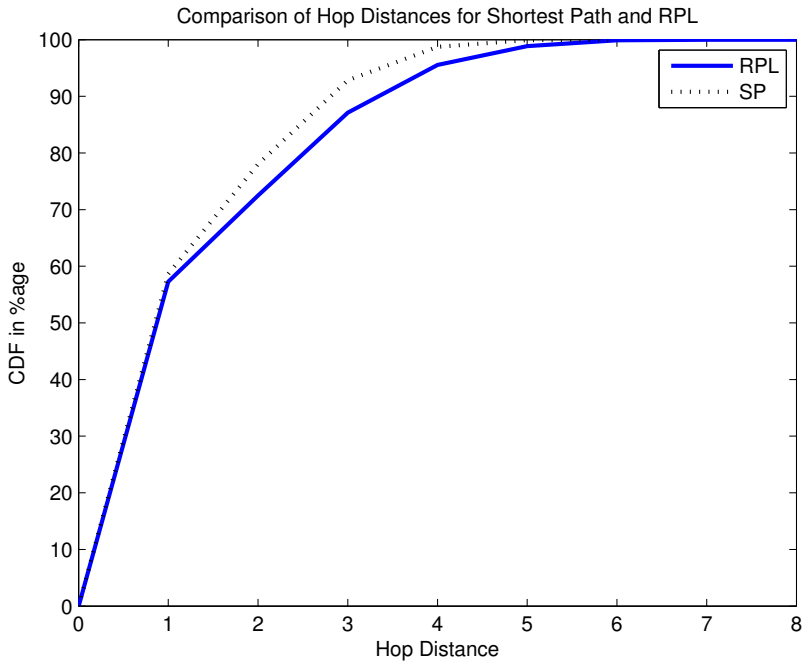

Figure 3.22: CDF of End-to-End Hop Count for RPL and Ideal Shortest Path in Home Routing.

3.3.2 Delay

To get an idea of maximum observable delay in the above-mentioned traffic pattern, the delay for different numbers of hops to the destination for RPL is considered. Figure 3.26 shows how the end-to-end packet latency is distributed for different packets with different hop counts in the network.

For this deployment scenario, 60% of the traffic has been restricted to a 1-hop neighborhood. Hence, intuitively, the protocol is expected to yield path qualities that are close to those of ideal shortest path routing for most of the paths. From the CDF of the hop count and ETX path cost, it is clear that peer-to-peer paths are more often closer to an ideal shortest path. The end-to-end delay for distances within 2 hops is less than 60 ms for 99% of the delivered packets, while packets traversing 5 hops or more are delivered within 100 ms 99% of the time. These results demonstrate that for a normal routing scenario of an LLN deployment in a building, RPL performs fairly well without incurring much control plane overhead, and it can be applied for delay-critical applications as well.

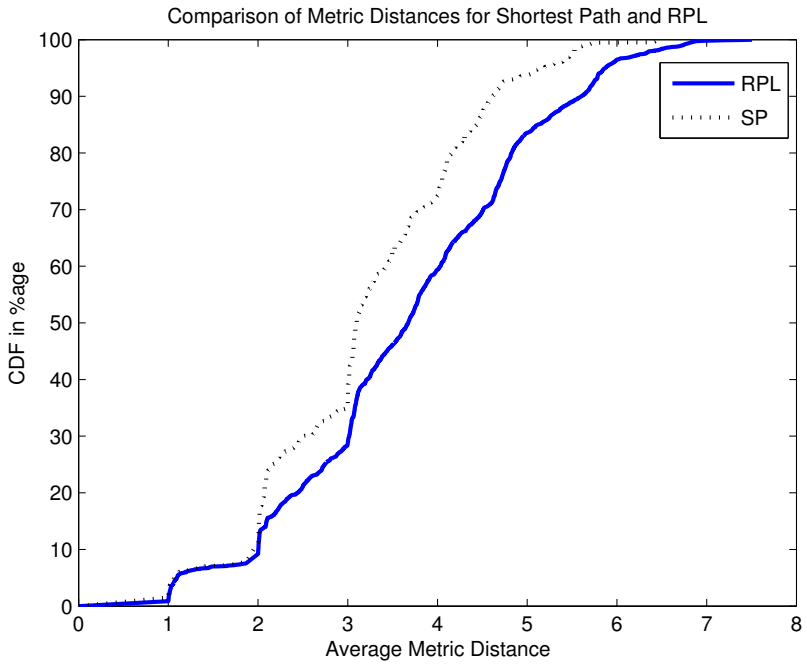

Figure 3.23: CDF of ETX Path Cost Metric for RPL and Ideal Shortest Path in Home Routing.

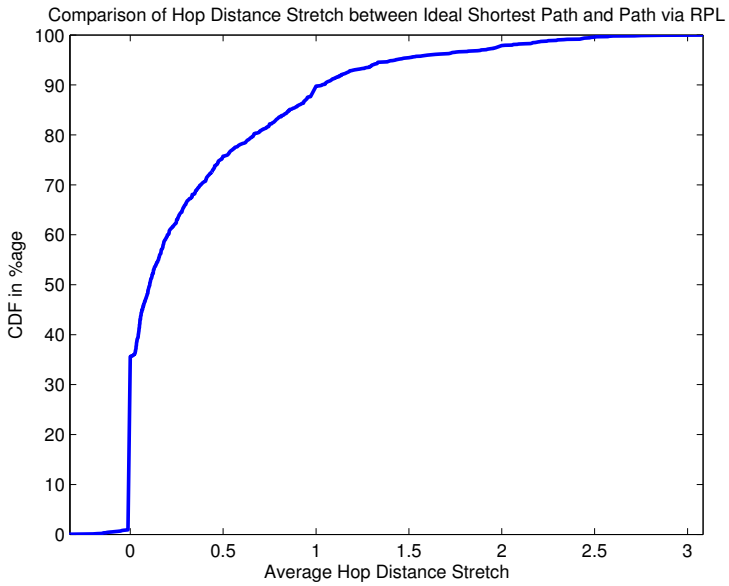

Figure 3.24: CDF of Hop Distance Stretch from Ideal Shortest Path.

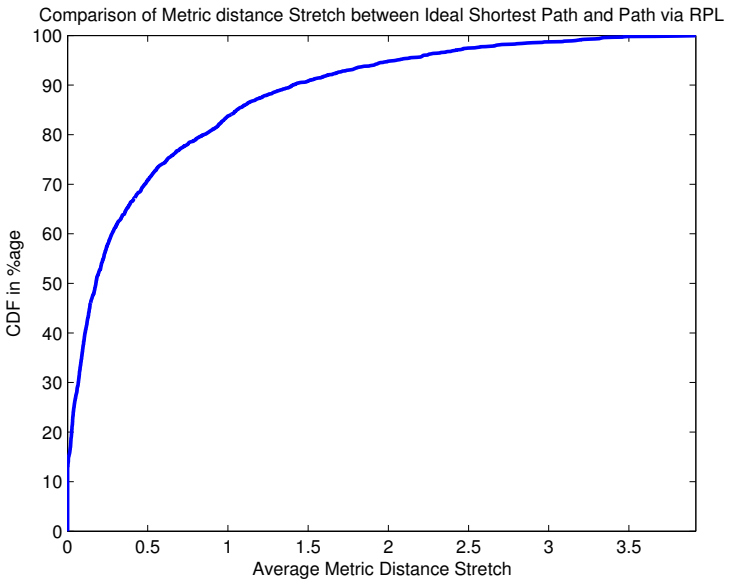

Figure 3.25: CDF of ETX Metric Stretch from Ideal Shortest Path.

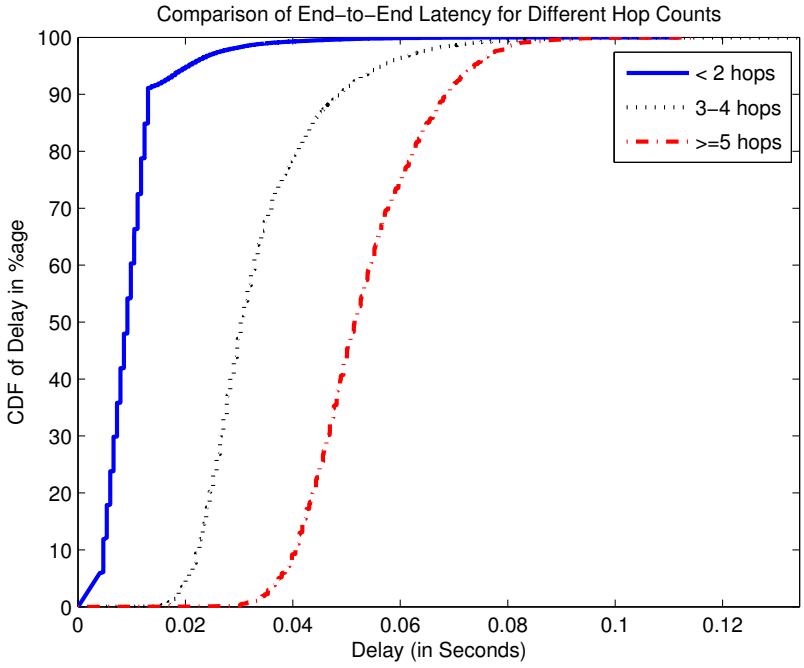

Figure 3.26: Packet Latency for Different Hop Counts in RPL.

3.4 RPL in a Large-Scale Network

In this section, we focus on simulating RPL in a large network and study its scalability by focusing on a few performance metrics: the latency and path cost stretch, and the amount of control packets. The 2442-node smart meter network with its corresponding link traces was used in this scalability study. To simulate a more realistic scenario for a smart meter network, 100% of the packets generated by each node are destined to the root. Therefore, no traffic is destined to nodes other than the root.

3.4.1 Path quality

To investigate RPL's scalability, the CDF of the ETX path cost in the large-scale smart meter network is compared to a hypothetical ideal shortest path routing protocol that minimizes the total ETX path cost (Figure 3.27). In this simulation, the path stretch is also calculated for each packet that traverses the network. The path stretch is determined as the difference between the path cost taken by a packet while following a route built via RPL and a path computed using an ideal shortest path routing protocol. The CDF of the ETX fractional stretch, which is determined as the ETX metric stretch value over the ETX path cost of an ideal shortest path, is plotted in Figure 3.28. The fractional hop distance stretch value, as defined in the Terminology section, is shown in Figure 3.29.

Looking at the path quality plots, it is obvious that RPL works in a non-optimal fashion in this deployment scenario as well. However, on average, for each source-destination pair, the ETX fractional stretch is limited to 30% of the ideal shortest path cost. This fraction is higher for paths with shorter distances and lower for paths where the source and destination are far apart. The negative stretch factor for the hop count is an interesting feature of this deployment and is due to RPL's decision to not switch to another parent where the improvement in path quality is not significant. As mentioned previously, in this implementation, a node will only switch to a new parent if the advertised ETX path cost to the LBR through the new candidate parent is 20% better than the old one. The nodes tend to hear DIOs from a smaller hop count first, and later do not always shift to a larger

hop count and smaller ETX path cost. As the traffic is mostly to the DAG root, some P2P paths built via RPL do yield a smaller hop count from source to destination, albeit at a larger ETX path cost.

As observed in Figure 3.27, 90% of the packets transmitted during the simulation have a (shortest) ETX path cost to destination less than or equal to 12. However, via RPL, 90% of the packets will follow paths that have a total ETX path cost of up to 14. Though all packets are destined to the LBR, it is to be noted that this implementation ignores a change of up to 20% in total ETX path cost. Figures 3.28 and 3.29 indicate that all paths have a very low ETX fractional stretch factor as far as the total ETX path cost is concerned, and some of the paths have lower hop counts to the LBR or DAG root as well when compared to the hop count of the ideal shortest path.

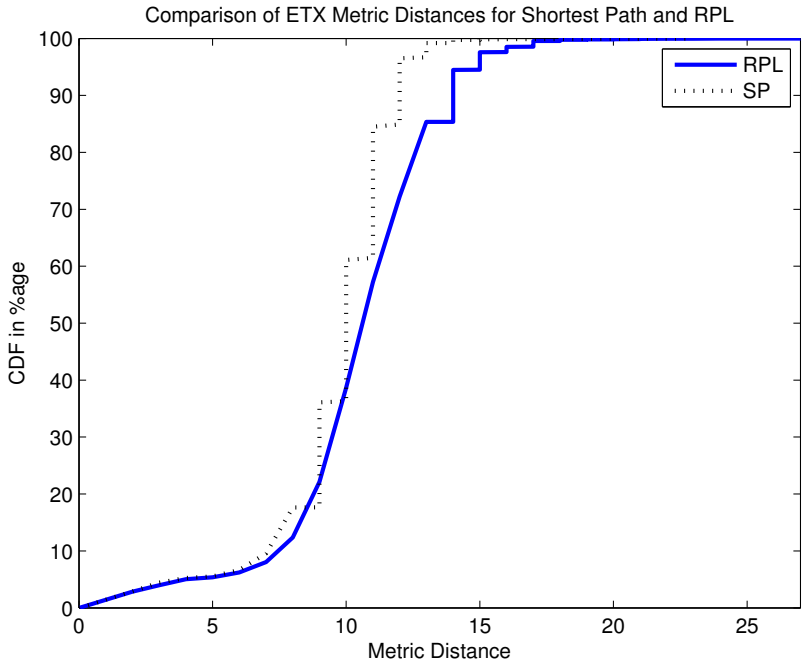

Figure 3.27: CDF of Total ETX Path Cost versus ETX Path Cost.

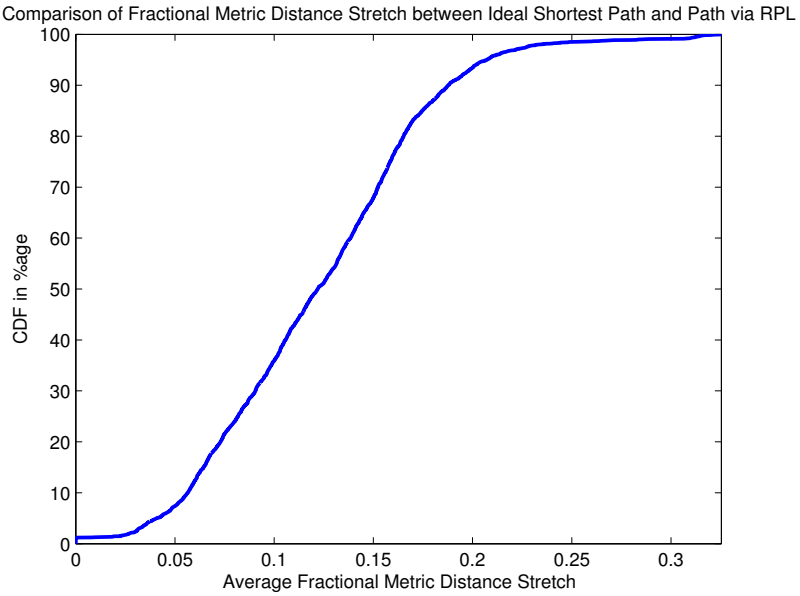

Figure 3.28: CDF of ETX Fractional Stretch versus ETX Fractional Stretch Value.

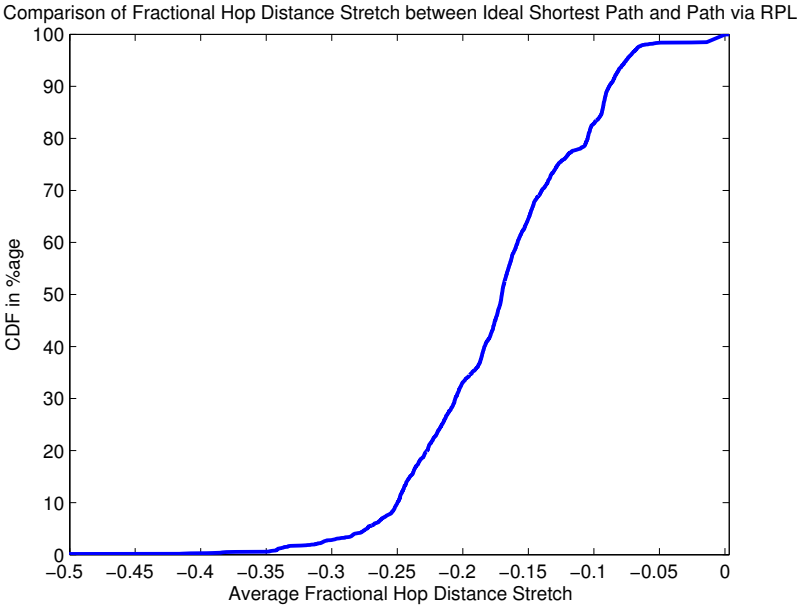

Figure 3.29: CDF of Fractional Hop Count Stretch.

3.4.2 Delay

Figure 3.30 shows how end-to-end packet latency is distributed for different hop counts in the network. According to RFC 5548, Urban LLNs (U-LLNs) are delay tolerant, and the information, except for critical alarms, should arrive within a fraction of the reporting interval (within a few seconds). The packet generation for this deployment has been set higher than usual to incur high traffic volume, and nodes generate data once every 30 seconds. However, the end-to-end latency for most of the packets is condensed between 500 ms and 1 s, where the upper limit corresponds to packets traversing longer (greater than or equal to 6 hops) paths.

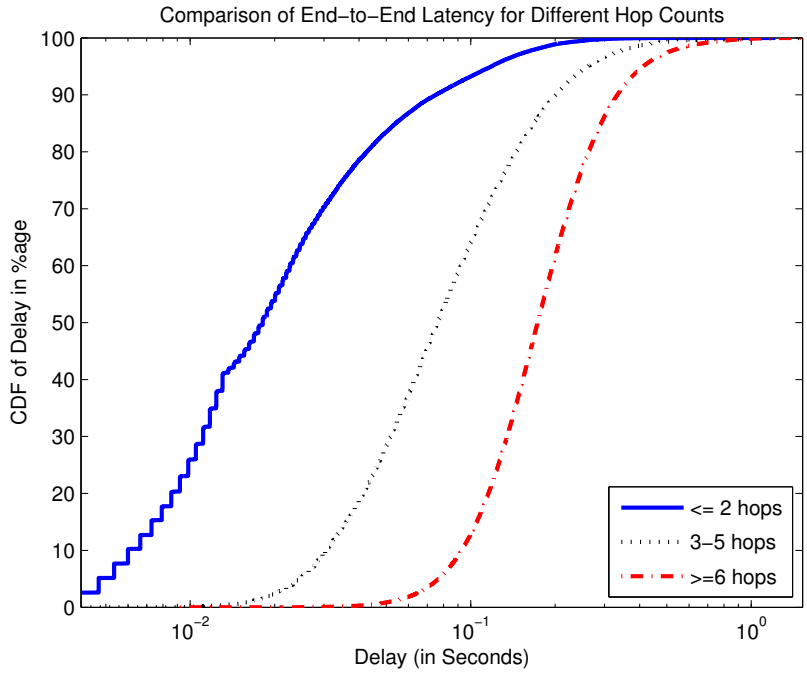

Figure 3.30: End-to-End Packet Delivery Latency for Different Hop Counts.

3.4.3 Control packet overhead

Figure 3.31 shows the comparison between data packets (originated and forwarded) and control packets (DIO and DAO messages) transmitted by each node (link ETX is used as the routing metric). Here one can observe that in spite of the large scale of the network, the amount of control traffic in the protocol is negligible in comparison to data packet transmission. The smaller node ID for this network actually indicates closer proximity to the DAG root, and nodes with high ID numbers are actually farther away from the DAG root. Also, as expected, we can observe in Figures 3.32, 3.33, and 3.34 that the (non-leaf) nodes closer to the DAG root have many more data packet transmissions than other nodes. The leaf nodes have comparable amounts of data and control packet transmissions, as they do not take part in routing the data. As seen before, the data traffic for a child node has much less variation than the nodes that are closer to the DAG root. This variation decreases with increase in DAG depth. In this topology, Nodes 1, 2, and 3, etc., are direct children of the LBR.

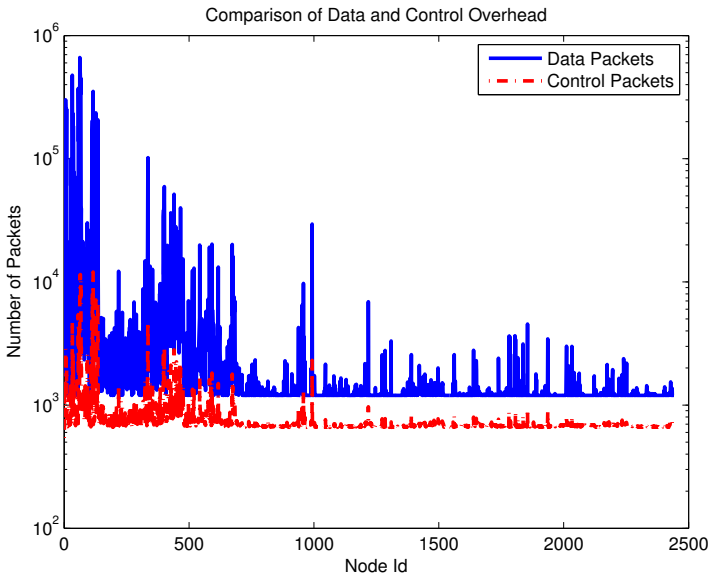

Figure 3.31: Data and Control Packet Comparison.

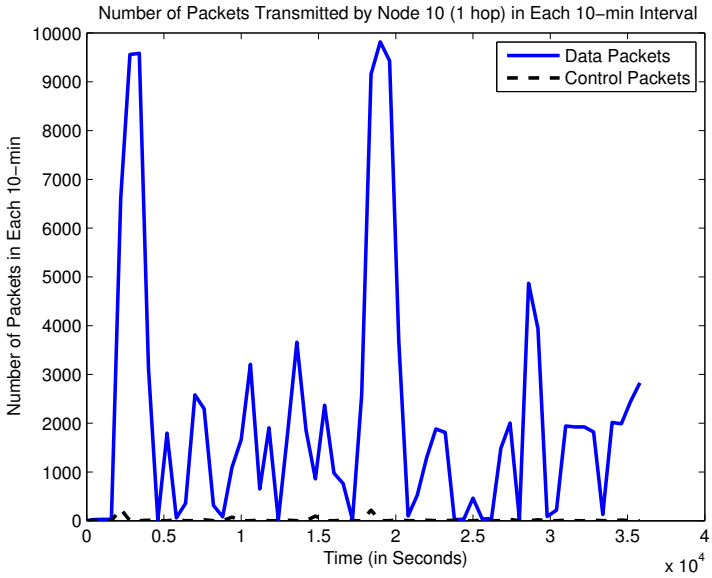

Figure 3.32: Data and Control Packets over Time for Node 1.

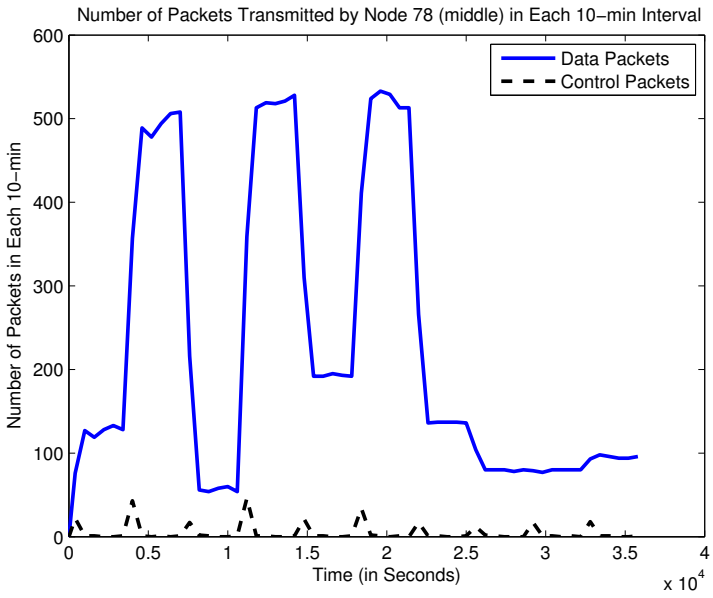

Figure 3.33: Data and Control Packets over Time for Node 78.

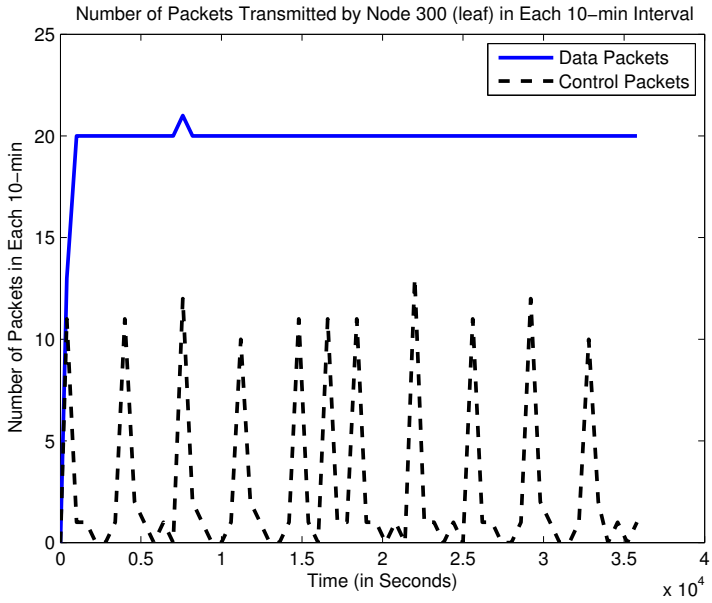

Figure 3.34: Data and Control Packets over Time for Node 300.

3.5 Scaling Property, Parameter Configuration and Routing Stability

An important metric of interest is the maximum load experienced by any node (CPU usage) in terms of the number of control packets transmitted by the node. Also, to get an idea of scaling properties of RPL in large-scale networks, it is also key to analyze the number of packets handled by the RPL nodes for networks of different sizes.

In these simulations, at any given interval, the node with maximum control overhead load is identified. The amount of maximum control overhead processed by that node is plotted against time for three different networks under study. The first one is Network 'A', which has 45 nodes and is shown in Figure 3.1 (Section 3.1); the second is Network 'B', which is another deployed outdoor network with 86 nodes and is shown in Figure 3.2 (Section 3.1); and the third is Network 'C', which is the large deployed smart meter network with 2442 nodes as noted previously in this chapter.

In Figure 3.35, the comparison of maximum control loads is shown for different network sizes. For the network with 45 nodes, the maximum number of control packets in the

network stays within a limit of 50 packets (per 1-minute interval), where for the networks with 86 and 2442 nodes, this limit stretches to 100 and $2 * 10^3$ packets per 1-minute interval, respectively.

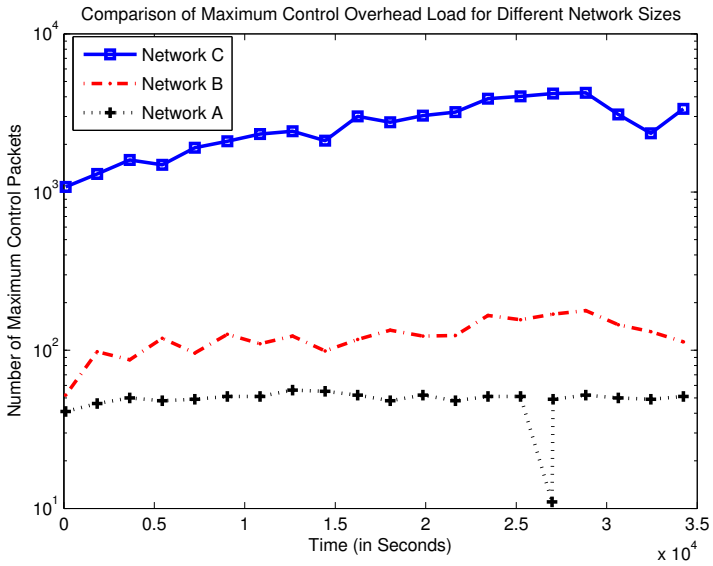

Figure 3.35: Scaling Property of Maximum Control Packets Processed by Any Node over Time.

This section also compares the CDF of the fraction of path change for three different networks – A, B, and C. Figure 3.36 shows how the three networks exhibit a change of P2P path when a 30% change in metric cost to the root is ignored before shifting to a new parent.

It is also important to set an adequate validity time for the routing structure established by the protocols. In RPL, frequent global repair will lead to frequent DAO and DIO message transmissions, increasing the control cost, but keeping the topology up-to-date. However, if the time period between two global repairs (henceforth mentioned as 'DAG repair period') is too large, inconsistencies in the DAG may occur often. To quantify inconsistencies, we measure the time spent by all nodes during which they do not have a parent or path to the DAG root. In Figure 3.37, we plot the CDF of this loss of connectivity

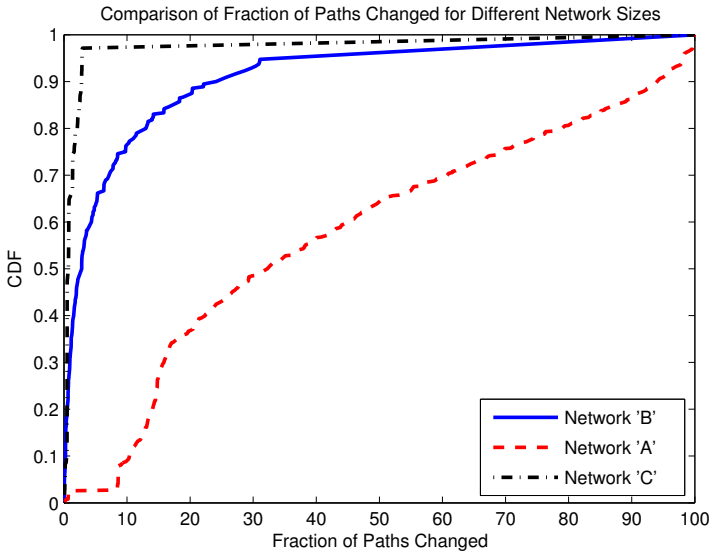

Figure 3.36: Comparison of Distribution of Fraction of Path Change.

time for different values of DAG repair period. Thanks to the local repair mechanism, the repair period has less effect on path unavailability. In Figure 3.38, the effect of the global repair period timer on control packet overhead is shown where the control overhead against the Node ID for different repair period values.

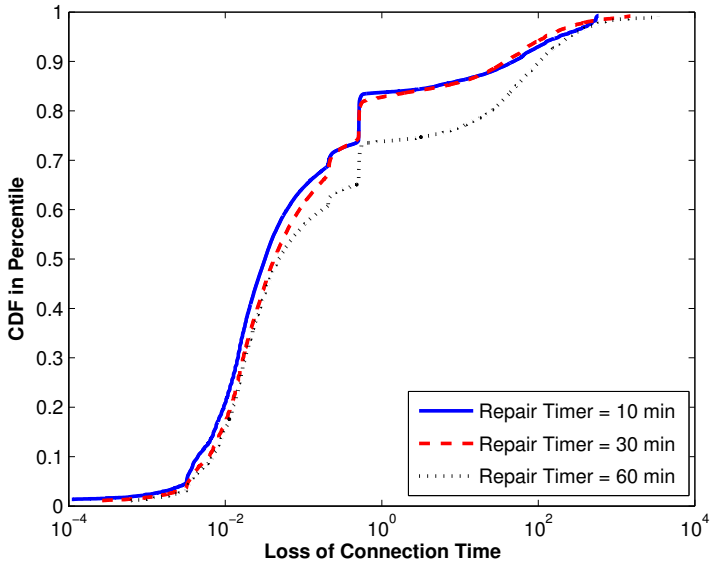

Figure 3.37: Unreachability time to DAG root for different DAG repair periods (RPL).

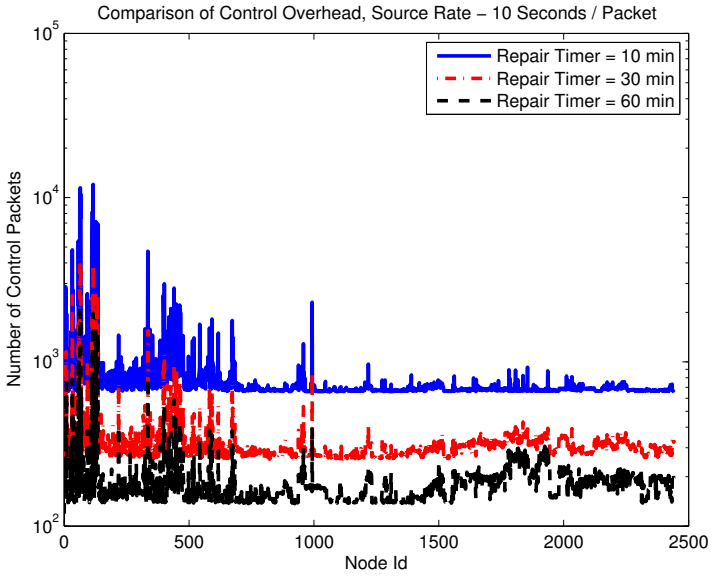

Figure 3.38: Numbers of Control Packets for Different Global Repair Timer Periods.

For a network built with low-power devices interconnected by lossy links, it is of the utmost importance to ensure that routing packets are not flooded in the entire network and that the routing topology stays as stable as possible. Any change in routing information, especially parent-child relationships, would reset the timer, leading to emitting new DIOs, and would hence change the node's path metric to reach the root. This change will trigger a series of control plane messages (RPL packets) in the DODAG. Therefore, it is important to carefully control the triggering of DIO control packets via the use of thresholds.

In this section, the effect of the tolerance value that is considered before emitting a DIO reflecting a new path cost is analyzed. Four cases are considered:

- No change in DAG depth of a node is ignored;
- The implementation ignores a 10% change in the ETX path cost to the DAG root. That is, if the change in total path cost to the root/LBR – due to DIO reception from the most preferred parent or due to shifting to another parent – is less than 10%, the node will not advertise the new metric to the root;
- The implementation ignores a 20% change in ETX path cost to the DAG root for

any node before deciding to advertise a new depth;

- The implementation ignores a 30% change in the total ETX path cost to the DAG root of a node before deciding to advertise a new depth.

This decision does affect the optimum path quality to the DAG root. As observed in Figure 3.39, for 0% tolerance, 95% of paths used have an ETX fractional stretch factor of less than 10%. Similarly, for 10% and 20% tolerance levels, 95% of paths will have a 15% and 20% ETX fractional path stretch. However, the increased routing stability and decreased control overhead are the profit gained from the 10% extra increase in path length or ETX path cost, whichever is used as the metric to optimize the DAG.

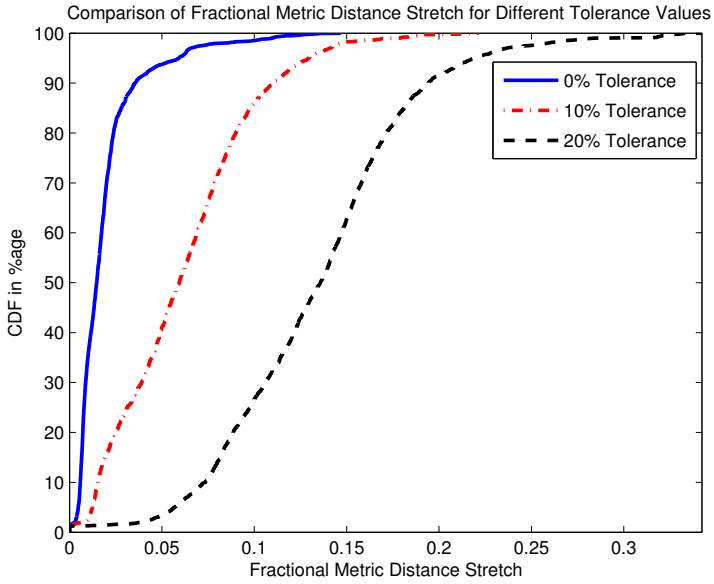

Figure 3.39: ETX Fractional Stretch Factor for Different Tolerance Levels.

Figure 3.40 shows the effect of these cases on the stability of parent-child relationships, where the average number of times the nodes' preferred parent changed against the depth of the node in the DAG is plotted. As expected, responding to all changes in the metric to reach the DAG root results in more frequent change of parent in the middle of the DAG,

which may bring instability of the routing table or instability of the DAG as a whole. This decision does affect the optimum path quality to the root.

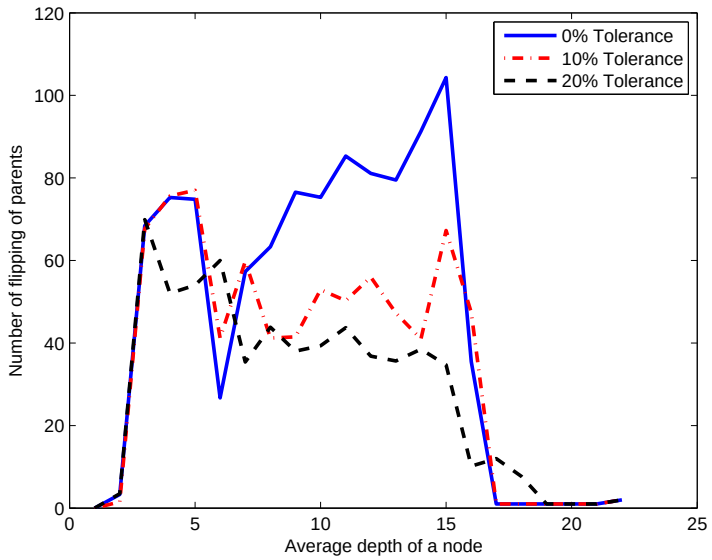

Figure 3.40: Number of times parents changed across the DAG.

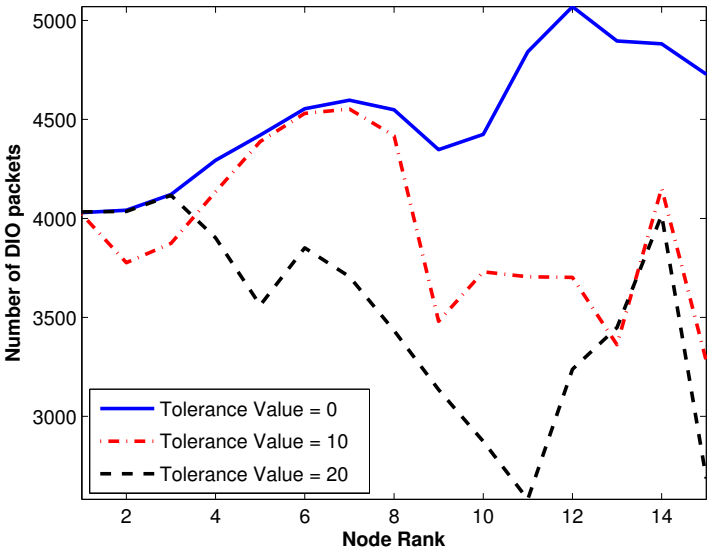

Figure 3.41: Control overhead for difference tolerance levels.

However, the profit gained from the sacrifice in total path cost is increased routing stability and decreased DIO message overhead, as observed in Figure 3.41, which shows the number of DIO packets against node depth (X axis) for the three tolerance values. It can be observed that Figures 3.40 and 3.41 are correlated. In a sense, more flipping of parents have contributed to more DIO generation. In the topology considered for this parameter tuning study, the region of DAG corresponding to rank $10 - 15$ appeared to be most unstable, with a spike in number of parent flippings, and consequent DIO emission for 0% tolerance level. We can observe that with a 20% tolerance level, we are able to remove the 'spike' and decrease the amount of DIO transmission. However, flipping of parents or DIO emission can not be a monotonous function of node rank, for they depend on several factors such as node degree, link variation, link quality between parent and nodes farther upwards, etc. Also, it must be mentioned that DAO packets contribute more significantly to overall control overhead, a fact also observed in (19), so this variation might seem insignificant while considering total overhead.

As the above-mentioned threshold also affects the path taken by a packet, this study also demonstrates the effect of the threshold on routing stability (number of times P2P paths change between a source and a destination). For Network 'A' (shown in Figure 3.1) and the large smart meter network 'C', the CDF of path change is plotted in Figures 3.42 and 3.43, respectively, against the fraction of path change for different thresholds (triggering the emission of a new DIO upon path cost change).

For above graphs, if X packets are transferred from source A to destination B, and out of X times, Y times the path between this source-destination pair is changed, then we compute the fraction of path change as $Y/X * 100\%$. This metric is computed over all source-destination pairs, and the CDF is plotted in the y axis.

3.6 Summary

In this Chapter we presented a very detailed evaluation of RPL unsung a newly developed RPL simulator and topology and traffic data from real deployments. All the simulation

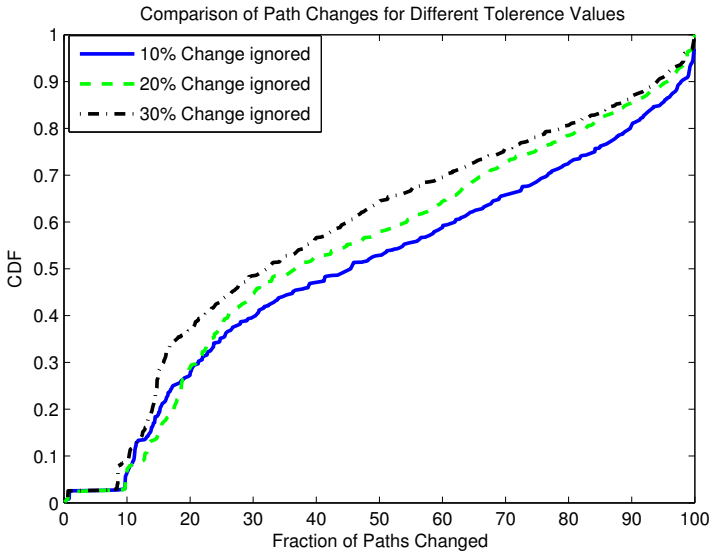

Figure 3.42: Distribution of Fraction of Path Change for Network A.

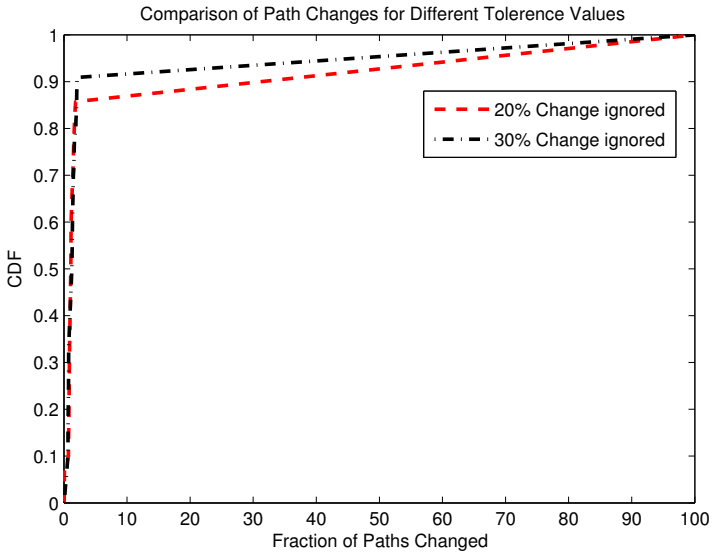

Figure 3.43: Distribution of Fraction of Path Change for Large Network C.

results presented in this document corroborate the expected protocol behavior for the topologies and traffic model used in the study. For the particular discussed scenarios, the protocol is shown to meet the desired delay and convergency requirements and to exhibit self-healing properties without external intervention, incurring negligible control overhead

(only a small fraction of data traffic). RPL provided near optimum path quality for most of the packets in the scenarios considered and is able to trade-off control overhead for path quality as per the application and device requirement through configurable parameters (such as decision on when to switch to new parent), and thus can trade-off routing stability for control overhead as well. Finally, as per the requirement of urban LLN deployments, the protocol is shown to scale to larger topologies (few thousand nodes), for the topologies considered in this implementation.

4

Proactive versus Reactive in LLNs

Despite the existence of a standard track routing protocol for LLN, discussions have been taken place as to whether other routing approaches could be suitable, such as deploying reactive routing protocols in LLNs, such as in smart metering networks, industrial automation, water management networks, etc.. Recently, an alternate protocol called LOADng (LLN On-demand Ad-hoc Distance vector routing protocol - next generation) (39), was proposed at the IETF. LOADng is inspired by the AODV protocol, being reactive to incoming traffic. In (40), the authors of LOADng have compared it to a simplified version of RPL and shown the benefits of the former with respect to control overhead, delay (contrary to what was found in (41) for AODV), frame collisions, etc. In their comparison, however, RPL has been naively configured (e.g., time period for DAO emissions) resulting in an unnecessarily cumbersome implementation, which lead the authors to simulate it for a shorter period of time and for a completely different traffic pattern than that of LOADng's, therefore jeopardizing the value of the conclusions reached in the paper. In the following, we attempt to summarize the main issues in the above study:

- The study considers that DAO messages in RPL are emitted periodically every 15 seconds, quite an unrealistic assumption for any RPL deployment. Unicasts of DAO

packets are event based than periodic with small interval.

- The amount of time RPL and LOADng are run differs and moreover the traffic pattern for each is also different. In RPL, data is sent from the "controller" or collection point every 0.1 seconds, and for LOADng, that duration between successive data transmission is 30 seconds. As we will show in this study, the performance of LOADng depends hugely on data traffic load. It would be practical to show how these protocols behave in a real traffic scenario, which is bi-directional, diverse, and encompasses all nodes in the network.

- The deployed network in (40) does not assume any variation in link quality, or any of the lossy characteristics that a typical LLN posses, leading to 100% delivery of the packets.

This chapter aims at a fair comparison between the two protocols using a realistic simulation study that includes an investigation on appropriate configuration parameters for both protocols. Since many metrics on comparison of proactive and reactive protocols are well studied, we concentrate more on issues and metrics specific to LLNs and have not been offered much attention in existing literature, such as temporal variation of control overhead due to multicast traffic, memory consumption, packet length for source routing, scaling properties, etc. To accomplish this, the RPL simulator that we developed for the study in (18) was extended to include LOADng's implementation described in (39).

It may be argued that the debate of reactive versus proactive protocols has been extensively researched in the context of traditional ad-hoc networks, and it is a well-known fact that reactive protocols perform better in networks with low traffic volume. We however argue that LLNs are different from traditional ad hoc networks not only due to node capacity, but also in the nature of traffic generation and routing requirements, making it worth to revisit the debate in the context of LLNs.

Recently, geographic routing protocols, such as (42), have also gathered attention in solving the data gathering problem in constrained networks. One of the main advantages of geographic routing is its low amount of state required to run the network. Indeed, in (43)

the authors have developed a framework to successfully implement geographic routing in an RF mesh network. However, the idea of appending a Global Positioning System (GPS) to every node in some LLNs has detrimental consequences on low power devices, and thus has been abandoned by the IETF ROLL working group (14). The work in (44) provides performance comparison in terms of delay and hop count between RPL and a geographical routing protocol for a 500 node smart utility network. Geographic routing keeps a state of O(1) regardless of network size, and thus can be effective in large scale urban LLNs. (44) also shows that geographical routing provides a larger delay and hop count than RPL.

Some of the recent literature on reactive versus proactive debate include (45), (46), (47), (48), and (49). (45) compares OSPF and AODV in the context of Mobile Ad-hoc NETworks or MANETs while using grid networks of size 9, 25, 49, 81 on TOSSIM simulator, and concludes in favor of AODV. Clearly, the network scale considered in this study is much smaller than the size envisioned for several instances of LLNs. However, the authors realize, as number of simultaneous flows increases, proactive protocols tend to be more favorable over reactive ones. Similar result had been concluded analytically by the authors of (50). This work compares proactive and reactive approaches from energy consumption aspect, and determines that there exists a cross over point in message duty cycle, beyond which reactive protocols consume more overhead and energy than proactive routing protocols. In (46), the authors considered various QoS metrics qualitatively to judge the merits of several routing protocols in MANET, and concluded that reactive protocols outperform proactive counterparts in terms of scaling, power consumption, control overhead and table size. (47) compares the protocols AODV, DSR (Dynamic Source Routing, (51)) and OLSR in terms of metrics such as throughput, end-to-end delay, etc., for varying traffic loads, number of flows and network sizes up to 100 nodes. A link state protocol was chosen as a proactive routing candidate and the authors conclude that proactive protocols are better suited for MANETs where the traffic is constant and periodic. (48) compares a proactive distance vector protocol, DSDV (52), with a reactive protocol, DSR, and a hybrid protocol, ZRP (53), and concludes that DSDV performs poorly when

compared to the other protocols with respect to metrics such as delay, number of dropped packets, and routing overhead. In (49), the authors have compared different protocols such as OSPF, DSDV, TORA (Temporally-Ordered Routing Algorithm, (54)), DSR, and AODV in terms of throughput, delay and routing load. However, Medium Access Control (MAC) protocol are ignored in the study, thus effects of typical lossiness and congestion of wireless links have been neglected, jeopardizing the results obtained. Almost all of the literature above indicated that proactive protocols have a much lower delay than reactive protocols, but suffer more from high control overhead.

The above mentioned literature fails, however, to reach the network scale of thousands of nodes, which is typical for some LLN deployments. The traffic considered in most cases is CBR, with no multicast traffic analysis. At the same time, no previous comparisons so far consider varying link quality in the simulations. Since lightweight implementations are now available for both proactive and reactive approaches, and classic MANET protocols are not suitable for LLNs, a new routing solution was sought. RPL trades off a huge part of control overhead for sub-optimality of path quality. In (41), Wang et al. presented a comparison between RPL and AODV for smart meter AMI networks. Their results indicated that RPL outperformed standard AODV with respect to delay and packet delivery ratio. The authors in (55) have used a 25-node home automation network scenario and corresponding traffic, to compare LOADng and RPL using the COOJA simulator. It is worthy to note that the authors in (55) also recently pointed out similar drawbacks with the work carried in (40). However, this article carries out simulation on networks of much larger sizes, and encompasses several metrics not investigated in the current literature. For example, the largest network considered in this study, is a 2442 node smart grid Advanced Metering Infrastructure (AMI) network, where each node is an AMI meter, deployed in an urban area. In Table 4.1, we provide a summarized differences of the works mentioned, and in this work in terms of protocols considered, network scale, traffic pattern etc. To be noted, in this study we are not considering many cross layer implementations available for WSNs, such as (56, 57), as we aimed at providing the answer to a more generalized

Works Referenced	Protocols Considered	Network Scale	Traffic Pattern	Simulation Platform	PHY+MAC model	Link Variation	Metrics considered
(45)	AODV, OSPF	$9-81$	CBR, MP2P	TOSSIM	CSMA/CA	Not used	Overhead, time to recover
(47)	AODV, DSR, OLSR	$25-100$	CBR, MP2P	OPNET	802.11b, No packet drop within range	Mobile Nodes	Throughput, Delay, Routing Load
(48)	DSDV, AODV, DSR, ZRP	$10-50$	CBR, P2P	NS-2	802.11, No packet drop within range	Mobile Nodes	Throughput, Aver-age delay, Packets dropped, Overhead
(49)	SPF, EXBF, DSDV, TORA, DSR, AODV,	30	P2P-Poisson	MaRS (Maryland Routing Simulator)	Dedicated links, No packet drop	Mobile Nodes	Packet Delivery Ratio, Delay, Routing Load
(41)	RPL, AODV	1000	CBR-MP2P and Poisson-P2MP	NS-2	No Inter-ference / packet drop	Not used	Average delay, Packet Delivery Ratio
(55)	RPL, LOADng	$15, 25, 40$	P2MP, MP2P, both periodic and random	COOJA / Contiki OS	802.15.4, CC2420	Not used	Delay, Hop Distance, Overhead, Table Entries
This Work	RPL, LOADng	$45-2442$	MP2P and P2MP, both periodic and random	Castalia / OM-NET++	802.15.4, CC2420	Replays gathered link-traces	Delay, Path Cost, Overhead, Buffer Oc-cupancy, RAM con-sumption, Packet Length

Table 4.1: Comparison of Existing Literature and the Contributions of This Study.

question, whether to compute and maintain the route beforehand or not. There are many implementation styles available, and it is not within the scope or context of this work to address every single of them, rather to look at general routing technique. Within the requirements of routing in LLN proposed in IETF ((5, 6, 7, 16)), a protocol candidate needs to be accommodating enough to consider all three major types of communication,

P2P, P2MP and MP2P. Their are many protocols available which may be suitable for converge-casting or multicasting in specific scenarios, but to extend the LLN or future WSNs to be a part of IoT, one needs to avoid cross layer implementations, as they lead to devising suitable gateway protocols and is a hindrance to both the real time IPv6 communication and the seamless network growth that IP has offered (22, 27).

4.1 LOADng: LLN On-demand Ad-hoc Distance Vector Routing Protocol - Next Generation

LOADng is an AODV based protocol, adapted to LLN needs. LOADng works by generating Route Requests or RREQs by a source route (originator) while discovering a route to a destination. RREQs are forwarded by the nodes in the network following a controlled flooding or a classical flooding. A node receiving a RREQ packet creates a routing table entry for the generator of the RREQ, and thus by receiving multiple RREQ from the same generator, it learns the best route back to the source. Route Replies (RREPs) are generated upon receipt of a RREQ by the destination. These RREPs are forwarded towards the generator of RREQ via the path learned when the RREQs were received. Every node in the return path of RREP thus learns about the destination node, and installs a forward route to the destination. Also, each RREP is acknowledged by the recipient; if an RREP is not ACKed within RREP_ACK_TIME - a configurable parameter, the RREP is re-sent to next hop to the originator of the request. Learned routes have an expiration time or R_Valid_Time, after which the nodes have to generate new RREQ if a packet for that destination arrives. If a route is not used within this R_Valid_Time, the route is considered to be expired or obsolete. Figure 4.1 illustrates the process of RREQ forwarding. The data follows the reverse path set up by RREP packets, as shown in Figure 4.2.

Route maintenance in LOADng is performed on a reactive basis. If a node detects a link down, upon failure of a data packet delivery to the next hop, it may initiate route discovery through the RREQ/RREP cycle again. It may also generate a route error (RERR) packet and forward it back to the source of data traffic, forcing the source to

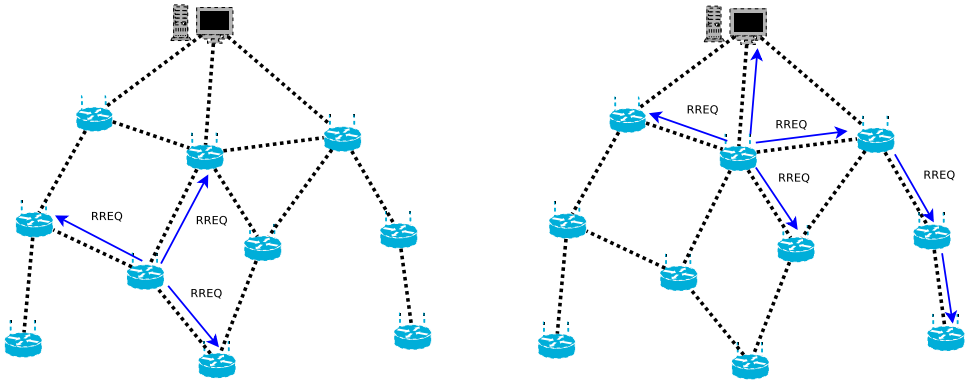

Figure 4.1: RREQ forwarding in LOADng.

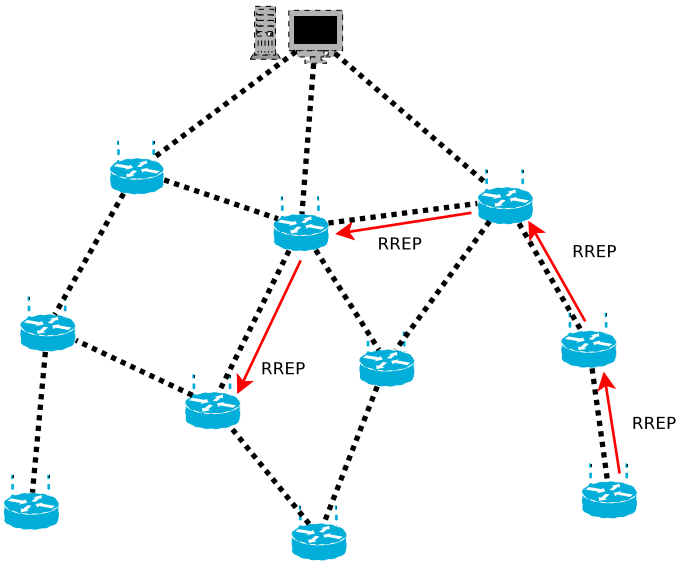

Figure 4.2: RREP forwarding in LOADng.

re-initiate the route discovery procedure. Moreover, when a node wishes to send traffic to a destination for which the route is expired, or a next-hop link is broken, it re-initiates the route discovery process.

LOADng and AODV have however some basic differences. Firstly, LOADng accepts number of weak links in a path to be the path cost to adapt to LLN characteristics. When a node receives an RREQ, it increments the path cost by one if the interface has a weak

link with the previous hop node, otherwise the cost remains same. If two paths are of equal cost, hop count is used for tie-braking. Thus, the protocol adapts itself to the lossy character of the network. Secondly, no node other than the destination responds with an RREP, even if it has a route to the destination. Furthermore, for RREQ messages, all nodes check the link with previous hop for bi-directionality. If the link is unidirectional, the RREQ is discarded. Lastly, no node maintains a precursor list for the nodes that use itself as destination. Hence, when a link breakage is detected at a node, the node does not send route error (RERR) packets to all such neighbors. When a data packet is received at a node which does not have a route to the destination due to link break-down, it simply sends back an RERR to the source of the data packet, forcing the source to initiate route discovery once more.

4.2 Theoretical Comparison: A Balanced Aggregation/ Dissemination Tree Model

Being a reactive protocol, LOADng has its own boon and bane. While it does not proactively disseminates control packets, on demand route discovery needs the packet to be buffered while RREQs are multicasted and RREP is received, leading to higher delay bound than a proactive protocol as RPL, as well as more buffer space. Also, control packet volume scales in proportion to the number of flows in the network. In Smart Grid AMI Meter networks, or in a building/home/industrial automation system, the LBR sends periodic data to every sensor in the system. For a network of size N, this operation needs N different RREQs to be generated by the LBR alone over the period. As each node forwards a particular RREQ for a particular destination at least once, the control overhead scales with $\Omega(N^2)$ for a network of size N. In this section we will show that RPL, by creating a DAG and forwarding DAOs only through the parent-child links in the DAG, results in an overall control overhead that is lower than $O(N^2)$, and reduces to $O(Nlog(N))$ for a balanced tree structure.

Recall that RPL creates a Directed Acyclic Graph (DAG), in which routing takes

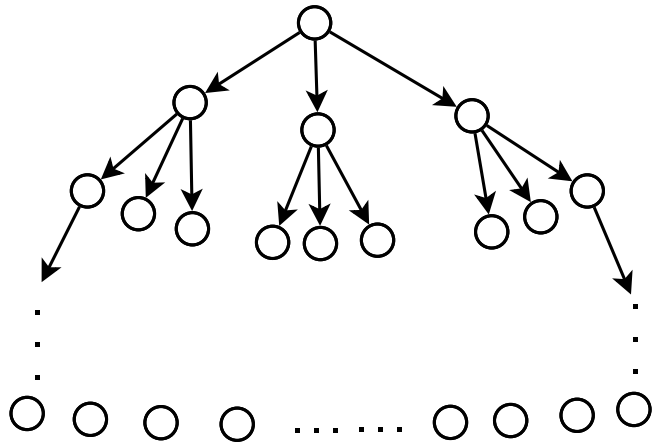

Figure 4.3: A balanced tree with B = 3.

place in an LLN. Since the created graph is acyclic by definition, links between preferred parents and children create a spanning tree of the network through which data aggregation (MP2P), or data dissemination (P2MP) takes place. In this section, we attempt to provide a lower and upper bound on control packets for data aggregation/dissemination traffic for both protocols. For the sake of simplicity, we first assume that the tree created by constructing a DAG is a balanced tree, and show how the control overhead for RPL and LOADng compares to one another.

Total control overhead for RPL with balanced tree

To determine the theoretical bounds, we first consider that the data aggregation tree, as constructed from the DAG structure in RPL, is a balanced tree, with B children under each parent. An example of a balanced tree with 3 children under each node is shown in Figure 4.3. The tree has a height of H and hence, the number of nodes in the network is given by $N = (B^{H+1} - 1)/(B - 1)$. Therefore, for any node at a rank R, $1 \leq R \leq H + 1$, the number of nodes in its subtree including itself is given by

$$N_R = 1 + B + B^2 + ... + B^{H-R+1} = \sum_{i=0}^{H-R+1} B^i \tag{4.1}$$

Note that the root itself has a rank equal to 1. Now, if the time period to emit a new DAG sequence number, as mentioned in Section 2.1, is T, then a new DAO is emitted by each node every T seconds. For simplicity of calculation, we assume that there is no link breakdown or topology change within this T time. For RPL, since a node forwards all DAOs from all nodes in its sub-tree to the DAG root or LBR, the node X would handle N_R number of DAO messages in a period T. We assume the same topology stability during T for LOADng, and therefore the route validity time (R_Valid_Time) for any route in LOADng protocol is kept as T. Also, in LOADng, all N_R nodes under a node X at rank R would route their data to the sink/LBR through the node X and vice-versa. Hence, the node X would be responsible to forward N_R number of RREPs to the LBR for data dissemination. Also, for both RPL and LOADng, the first respective DIO/RREQ received by a node, arrives via the best path and, hence, no further DIO/RREQ updates the parent entry/routing entry that is generated. So, for LOADng, effectively one RREQ is forwarded per destination. This assumption is necessary, because it is not known when the RREQ/DIO corresponding to the best path would arrive in a randomized situation, so we assume that they arrive through the best path first.

Since the number of nodes at a given rank R is given by B^{R-1}, the total number of DAOs to be forwarded by all nodes at a rank R is

$$DAO_R = B^{R-1} * \sum_{i=0}^{H-R+1} B^i \tag{4.2}$$

Since for all nodes in the network excluding the root node, R can vary from 2 to $H+1$, the total DAO packets in a network operating RPL is given by

$$DAO_{Total} = \sum_{R=2}^{H+1} (B^{R-1} * \sum_{i=0}^{H-R+1} B^i) \tag{4.3}$$

Since the inner sum of Equation 4.3 can be written as

$$\sum_{i=0}^{H-R+1} B^i = \frac{B^{H-R+2} - 1}{B - 1} \tag{4.4}$$

Hence,

$$\begin{aligned} DAO_{Total} &= \sum_{R=2}^{H+1} (\frac{B^{H+1} - B^{R-1}}{B-1}) \\ &= \sum_{R=2}^{H+1} (\frac{B^{H+1}}{B-1}) - \sum_{R=2}^{H+1} (\frac{B^{R-1}}{B-1}) \end{aligned} \tag{4.5}$$

As mentioned in Section 2.1, every T seconds, DAO_{Total} control overhead is generated in the network operating RPL due to DAO propagation. Each node issues a DIO every time the trickle timer for DIO fires, and the timer period is doubled after DIO emission. Hence, each node, on average emits $\log_2 T$ DIO messages within T seconds. Assuming no delivery error, no DIS message in RPL, and that DAO_ACK messages are to be acknowledged by the LBR (same in number as DAO messages), the total control overhead in RPL for a time period S is given by

$$C_{T,RPL} = \frac{S}{T}[Nlog_2 T + 2\sum_{R=2}^{H+1} \frac{B^{H+1} - B^{R-1}}{B-1}] \tag{4.6}$$

Clearly, the assumptions here are that S is large enough to ignore the remainder of S divided by T in comparison to S, or S is divisible by T. Normally, in a practical implementation, T would be kept around half an hour, and total operation time S will be in order of days or even years maybe. So this assumption is reasonable. Some simple mathematical manipulation of Equation 4.6, considering $N = (B^{H+1} - 1)/(B - 1)$ and $H = \log_B N$, can yield

$$C_{T,RPL} = \frac{S}{T}[N \log_2 T + 2\sum_{R=2}^{H+1} \frac{B^{H+1} - B^{R-1}}{B-1}]$$

$$= \frac{S}{T}[N \log_2 T + 2H * (N + \frac{1}{B-1}) - \frac{2B}{B-1} * \sum_{i=0}^{H-1} B^i]$$

$$= \frac{S}{T}[N \log_2 T + 2H * (N + \frac{1}{B-1}) - \frac{2B}{B-1} * \frac{B-1}{B}(N + \frac{1}{B-1})]$$

Since $1/(B-1)$ is a constant, and a fraction, when compared to N, we disregard it from the complexity analysis. We assume $(N + 1/(B-1)) \simeq N$, which yields,

$$C_{T,RPL} = \frac{S}{T}[N(log_2 T - 2) + 2N \log_B N] \tag{4.7}$$

Clearly, when RPL creates a balanced tree through the DODAG creation, the total control overhead scales with $\Theta(NlogN)$, where N is the network size.

Total control overhead for LOADng in similar topology

Since LOADng is a reactive protocol and sets up the routing path when data transfer is needed, the control packet overhead will depend on the traffic pattern. Lets assume a simple traffic pattern of the LBR sending data to each node in a round-robin fashion, with a period F. In an LLN, F may range from 30 minutes to few hours. So, the LBR communicates with all N nodes once in a period F. The cache period of considering a route valid is assumed to be T, as mentioned before. Now, if $F \geq T$, each time the LBR wants to send some data to a particular node, the route entry for that node would be invalid. Hence, the LBR needs to set up the routing path by issuing RREQ for each node within F time. Since we assume that one RREQ is broadcasted by each node for each destination, the total RREQ transmission within F time is equal to N^2. If we assume the same topology as in the RPL analysis before, all RREPs would follow the path taken by DAOs in case of RPL. Hence, the number of RREPs in F time is same as the number of DAOs transmitted with RPL in T time, as we assume that RREQ received first corresponds to best path, and each route request generates exactly one RREP. So, similar to Eqn. 4, the total number of RREPs in F time is given by

$$RREP_{Total} = \sum_{R=1}^{H} (B^R * \sum_{i=0}^{H-R} B^i) \tag{4.8}$$

In addition, there will be same number of RREP_ACKs as RREPs. So, the total

control overhead (excluding RERR, given our assumption of no topology change within T or R_Valid_Time) for a run-time of S is given by

$$C_{T,LOAD} = \frac{S}{F}[N^2 + 2\sum_{R=2}^{H+1}(B^{R-1} * \sum_{i=0}^{H-R+1} B^i)] \tag{4.9}$$

By simple manipulations, the total control overhead is

$$C_{T,LOAD} = \frac{S}{F}[N^2 + 2N(\log_B N - 1)] \tag{4.10}$$

Equations 4.7 and 4.10 provide the total control overhead for RPL and LOADng during S runtime, assuming the DAG created by RPL creates a balanced tree for the topology, and LOADng uses a similar path for the same topology. It is also assumed that both DIO and RREQ reach every node via the best path first, and the link quality does not change for time T, where T is the period at which global repair occurs for RPL, and routes are valid in LOADng. However, topologies do not tend to create a balanced tree at the LBR, and when RPL yields a completely unbalanced tree at DAG root, such as for cases in a chain-like topology, the total control overhead scales with $\Theta(N^2)$. To understand why it is so, consider a chain-like topology, where N nodes create a DAG of height equal to N. The leaf node will be responsible for 1 DAO (1 RREP for LOADng), the node above the leaf will forward 2 DAOs (2 RREP for LOADng), and so on. So, with RPL, the number of DAOs in T time will equal to

$$DAO_{Total} = 1 + 2 + \cdots + (N-1) = \frac{N(N-1)}{2} \tag{4.11}$$

The case for RREP messages for LOADng in F time will be similar. Hence both protocols will have a total control overhead complexity of $\Theta(N^2)$ for a chain-like topology. For any other practical topology, the case will be between the two extremes of a balanced tree and a chain-like topology, and in most cases RPL is expected to provide less control overhead depending on the value of T and F, as we would observe in Section 4.4.

4.3 Simulation Setup

As in (18) and Section 3.1, real link layer data gathered from the outdoor and smart meter deployments were collected and used to compute the PDR (Packet Delivery Ratio) for each link in the network. The link's PDR in simulation varies according to the gathered traces of the same link, therefore creating a "pseudo-replica" of the real network. The simulator has been extended for this study, to include LOADng as per described in (39). In the simulator, both RPL and LOADng run over a 802.15.4 MAC/PHY layer on top of a CC2420 radio model for TelosB motes. The simulator was fed with topologies and traffic traces from two real deployment networs (86 nodes outdoor network and a 2442 nodes smart grid AMI network). Random topologies were used only in the scalability study (Section 4.6). The simulation is run for 2 days of simulation for both RPL and LOADng.

4.3.1 Traffic traces

The following types of traffic and patterns, provided by smart grid AMI meter vendors, were used in the simulation:

- Polling traffic: The collection point (LBR) retrieves information (statistic report, status, load curve) from each node in the topology once a day, polling nodes in a round robin fashion.
- On-demand operation traffic: Reading of index (same traffic pattern as above). Time period between two readings from same node = $F1$. Short volume in both directions (50 bytes). $F1$ is normally kept at a rate of once in 2 hours, unless otherwise mentioned.
- Traffic due to multicast parameter modification (e.g. new tariff): 50 bytes sent by the LBR to a node, multicast, with frequency once a day at 2200 seconds after start of simulation.
- Traffic due to alarms: From a meter (node) to the LBR and unsolicited, so unidirectional. 20 bytes of data is sent by each meter to the LBR, once a day.

4.3.2 RPL and LOADng parameters

Based on our observations in Figures 3.37 and 3.38, a new sequence number is emitted by the DAG root to globally re-construct the DAG every 30 minutes (DAG repair period) for rest of the study. This value provided a good balance between control overhead, and repair time (protocol convergence). R_valid_time for LOADng is set to the same value as DAG repair period to bring fairness into the comparison, as both these parameters deal with the trade-off between freshness of routes and control overhead. Hence, a path in LOADng will be consider invalid 30 minutes after a route to destination is setup or a successful packet forwarding to the particular destination takes place. Note that a lower value of this parameter may incur more control overhead for LOADng, as mentioned in 4.2.

In this study, nodes ignore paths advertised via new DIOs if the path cost improvement via the new parent is less than 20%. This value was chosen as it seemed to tone down the amount of parent flipping and resetting DIO trickle timer. The default value of $Imin$ is provided to be 8 ms in RFC 6550, however, existing works prefer to set at in the range of 1 second (18, 58) or 4 seconds as in COOJA simulator by Contiki or (19). In this study, we set $Imin = 1$ second. Every node sends out a DAO message to the DAG root through the preferred parent after a new DAG sequence number is recorded or preferred parent is changed after a time duration proportional to the node's own rank. This is to make sure the reverse path for a DAO-ACK is set when the DAG root receives a DAO. For LOADng, RREP_ACK_TIME is set to 0.2 seconds, after which nodes transmitting a RREP will check their cache for the respective ACKs. For transmission power settings, please refer to (37). The simulation details are summarized in following Table 4.2.

4.4 Performance Results for Smart Grid Traffic

To evaluate these protocols fairly, we consider both small outdoor deployment and the large smart meter AMI network, and evaluate RPL and LOADng under the same traffic profile described in Section 4.3.1. In this section, results for end-to-end delay, path quality, control

Common paramaters	Data rate	240 Kbps
	MAC standard	802.15.4
	Transport protocol	UDP
RPL paramaters	Imin	1s
	MaxDoubling	12
	DAG repair period	30 min
	Path cost tolerance	20%
LOADng paramaters	R_Valid_Time	30 min
	RREP_ACK_TIME	0.2s

Table 4.2: Simulation settings.

overhead, memory requirements, dependency on application data rate, and packet length are discussed and analyzed. Similar metrics are also studied for a P2P communication scenario in Section 4.5. It is evident from our study that collecting reading reports more frequently, or adding new application modules will further deteriorate the performance of LOADng, so we consider a lower frequency of reporting operation.

4.4.1 Control overhead and ability to support P2MP or MP2P traffic

Since nodes in LLN have limited capacity in terms of power storage and/or scavenging, as well as face scarce bandwidth, control overhead is one of the most important considerations in choosing a routing protocol. It is well known that in networks with light traffic load and a small topology, a reactive protocol may be better suited than proactive protocols. The deployments and nature of applications running over a LLN often require a node to send the same data to multiple recipients, requiring multicasting or P2MP support from the routing protocol. These destinations may be several hops away, requiring an efficient dissemination method or multicast traffic (P2MP) support provided by the routing protocol. P2MP traffic includes, but is not limited to:

- management information multicasted to all nodes to a certain region in a landscape, or a certain part of the manufacturing pipeline in an industrial automation setting;
- new tariff notification to a group of users in a smart grid deployment;

- a single node providing sensed data to multiple servers.

Indeed, RFC 5548 (7) necessitates the support of P2MP traffic for a protocol to be suitable for U-LLN deployment. It describes - "the protocol(s) should be optimized to support a large number of unicast flows from the sensing nodes or sensing clusters towards a LBR, or highly directed multicast or anycast flows from the nodes towards multiple LBRs." While AODV can support multicast IP address in RREQ destination address, and an AODV based protocol may be altered to support an on-demand bi-directional route between any two nodes in the network, the protocol does not have provision to support P2MP traffic. LOADng does not have provision for supporting multiple route discovery either. A naive solution, would be to create a copy of the same message to send for each destination. To find the route to each destination, the node may have to create separate route request(RREQ) messages and broadcast them. This broadcast event creates a huge control overhead and the protocol does not scale well with the network size. Hence, AODV or reactive protocols in general may become unsuitable to be deployed in a large scale U-LLN where P2MP traffic needs to be supported, even if for 1-2 times a day. This is the reason we included multicast traffic once a day in simulation, and the following simulation results confirm our intuition.

Figure 4.4 shows the maximum control packets processed by any node over time for RPL non-storing mode and LOAD-ng, with the objective of studying the computational and processing load exerted on the nodes in the network. The 2 days of simulation time have been broken down to small windows of 1 minute each, and we computed the number of control packets processed/transmitted by each node within that window. The maximum processed control packets over all nodes for the window is plotted against time for both RPL and LOAD-ng. The figure shows that RPL exhibits a peak of control message processing when a new sequence number is emitted for global reconstruction of the DAG, such as at $0-60$, $1800-1860$ and $3600-3660$ seconds. Recall thats this implementation considers a DAG repair period of 30 minutes. LOADng exhibits a peak of maximum control message processing when multicast traffic is generated by the LBR. In subsequent

intervals, as the nodes finish broadcasting RREQs and forwarding RREPs, the control volume gradually decreases. We skipped plotting RPL storing mode in this figure and in the next one for the sake of clarity, as we have observed in Figure 4.5 and 4.6 that both storing and non-storing mode create almost similar amount of control overhead for this network. Similar behavior is also observed for the large deployment (omitted for brevity).

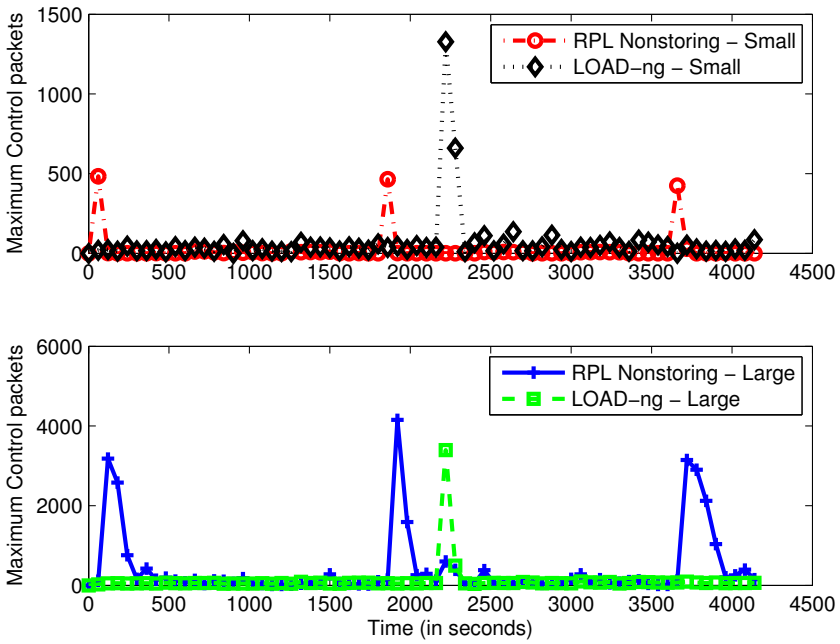

Figure 4.4: Maximum control packets processing for RPL and LOAD-ng.

Likewise P2MP traffic, MP2P traffic needs to be supported by a routing protocol intended to be designed for an LLN. A U-LLN should support occasional large-scale traffic flows from sensing nodes through LBRs (to nodes outside the U-LLN), such as system-wide alerts (7), or all nodes in one area reporting malfunction / emergency in part of the manufacturing plant. This situation may lead to a broadcast storm in the network, similar to the P2MP traffic scenario, leading individual RREP to reach the initiator node much later. Proactive Route to the actuators and periodic update can prevent this broadcast storm.

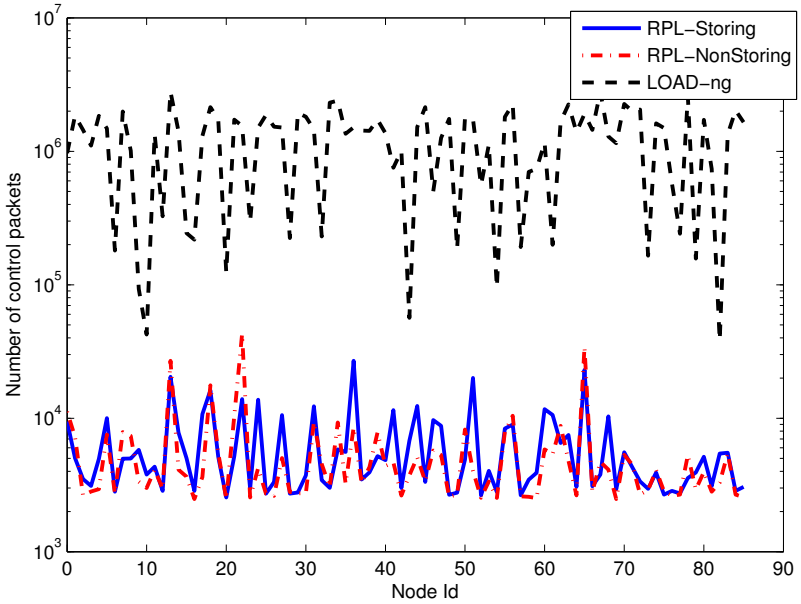

Figure 4.5: Control overhead for RPL and LOADng.

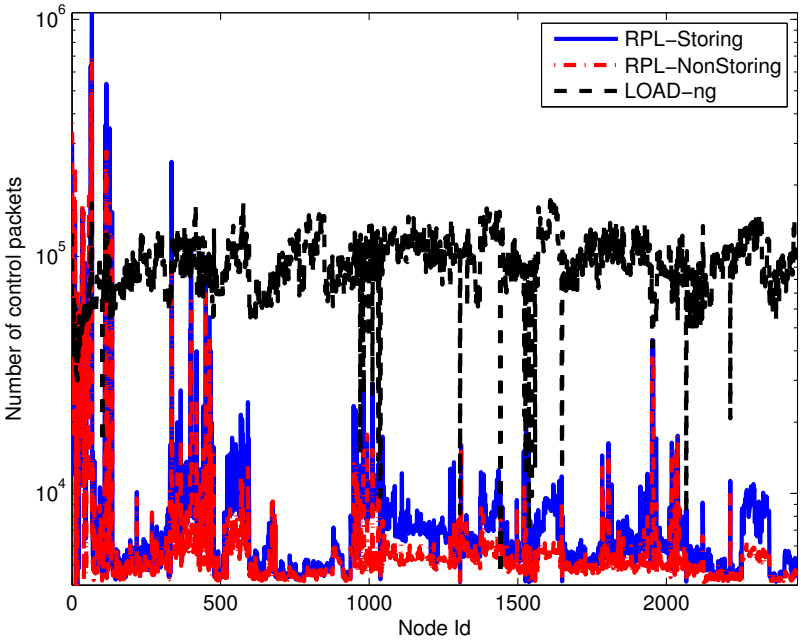

Figure 4.6: Control overhead for each node in a large network.

Figure 4.5 shows the control overhead cost for both LOADng and RPL non-storing mode in the smaller topology, whereas in Figure 4.6, the same is demonstrated for the

larger smart grid deployment. We observe that LOADng yields a control volume a magnitude larger than that of RPL. The control overhead in RPL depends mainly on two types of advertisements, *i*) DIO and *ii*) DAO (including DAO-ACK). While the DIO control overhead exhibits an exponential delay over time, the DAO control overhead for a given node depends on the number of nodes in its sub-tree, and thus it inversely varies with the rank of the node. In the figure, it is worth observing that, for RPL, some nodes have higher amount of control message transmission; these nodes (eg., 22, 66 in the smaller network, and lower ID nodes in the large deployment) are the nodes directly connected to the LBR, and hence are responsible for larger amount of DAO and/or DAO-ACK propagation. Hence, overall, for most nodes (other than a few tens among thousands) LOADng exhibits an order of magnitude more control packet transmissions than either mode of RPL. For LOADng, RREQ and RREP packets travel through nodes closer to the collection point; moreover communication with each destination also incurs RREQ packet forwarding for each node in the network.

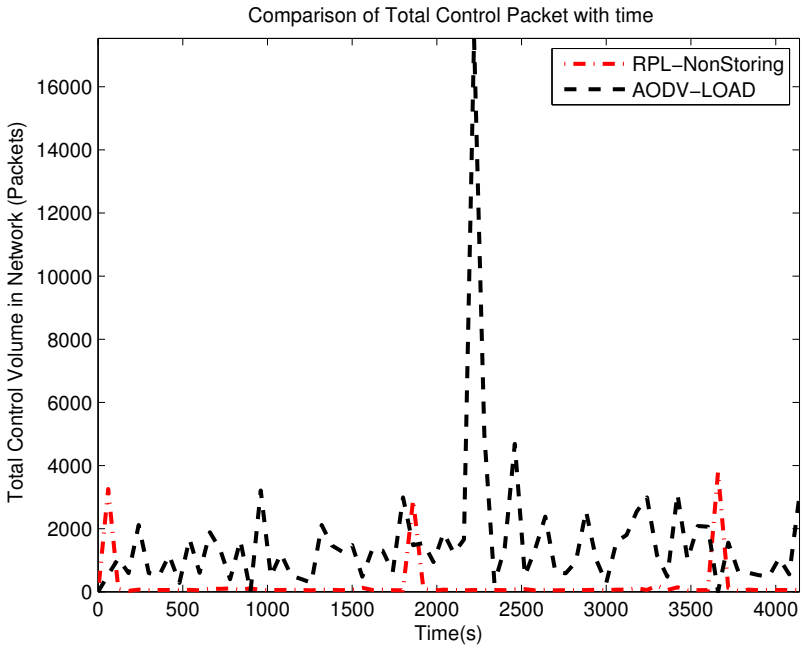

Figure 4.7: Total control packets processing - RPL vs LOADng in small network.

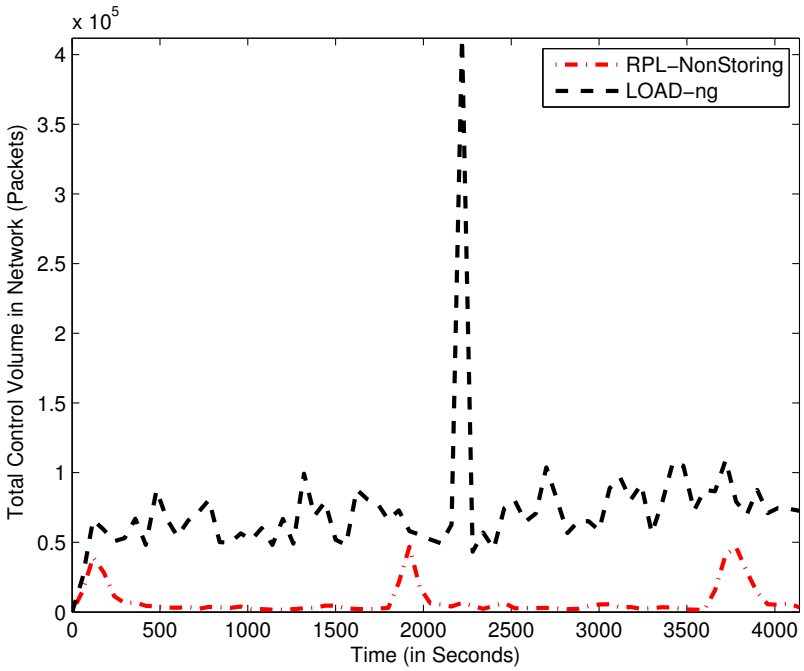

Figure 4.8: Total control packets processed in network against time in large network.

Though RPL periodically reaches a peak of maximum control overhead processing, this peak is only applicable to nodes directly connected to the LBR, therefore processing a large number of DAOs/DAO-ACKs. For LOADng, however, the control overhead does not depend much on the distance from the LBR. From a network congestion point of view, it is more disadvantageous to have all nodes transmitting large number of control packets, than only a few nodes at lower rank doing so. To look into this scenario more carefully, Figure 4.8 shows the total number of control packets processed by all nodes in each 1 minute window for the large network. The plot shows the overall higher volume of control traffic when multicast traffic appears. For LOADng, when a multicast traffic appears, every node takes part in broadcasting the RREQ packets, and the number of broadcasts for each node reaches as many as the number of recipients in the multicast. Hence, the total number of control packets in the network becomes significantly larger than that of RPL. The figures also show that even when global repair is initiated in RPL, the total control traffic volume in the network is still lower than that of LOADng. LOADng's high

control packet flooding may therefore lead to stall of data communication, and/or network crash.

4.4.2 Dependency of control overhead on application module

LLNs are a nascent area, also referred to as IP smart objects networks or the "Internet of Things," constantly growing in importance. Thus, a LLN that is currently provisioned to be used for data gathering purpose only, may include additional application modules in the future. Smart grid deployments may need to implement new modules of management traffic from the base stations to AMI meters, in addition to what is envisioned at present. LLNs are evolving and therefore it is expected that new applications and requirements will be part of its future (an LLN may also be re-purposed).

Reactive protocols, however, discover route on an on-demand basis. Thus, adding a new application module which requires more data communication in addition to the current data traffic pattern will incur additional control overhead. Hence, if a network is designed to operate within bounds in terms of maximum control overhead load, adding new application modules may well force the control overhead to surpass the designed maximum limit. For example, a deployment requiring both MP2P and P2P application may incur more overhead than a deployment which is currently working with only data aggregation. Since LLNs will undoubtedly require more application modules and management modules to be augmented in future, a suitable routing protocol should be able to cope with the added traffic. For the sake of illustration, many smart grid networks, which were originally designed for the purpose of advanced metering, now require a multi-service networks in support of a variety of applications including meter reading, use of meters of alarms, distribution automation and electric vehicles, leading to a variety of traffic patterns each with different quality of service requirements.

To illustrate this, Figure 4.9 shows the control overhead versus node ID for different polling intervals for LOAD-ng: 30 minutes, 1 hour, 2, 6 and 12 hours. One can see that the control overhead increases as the LBR polls the nodes more often. Figure 4.10 shows the

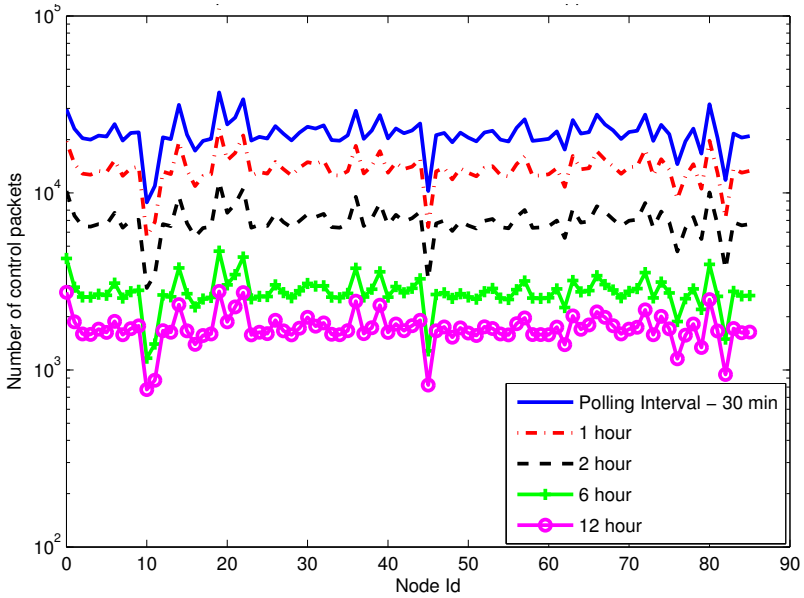

Figure 4.9: Control overhead vs. node ID for different application rates, LOADng.

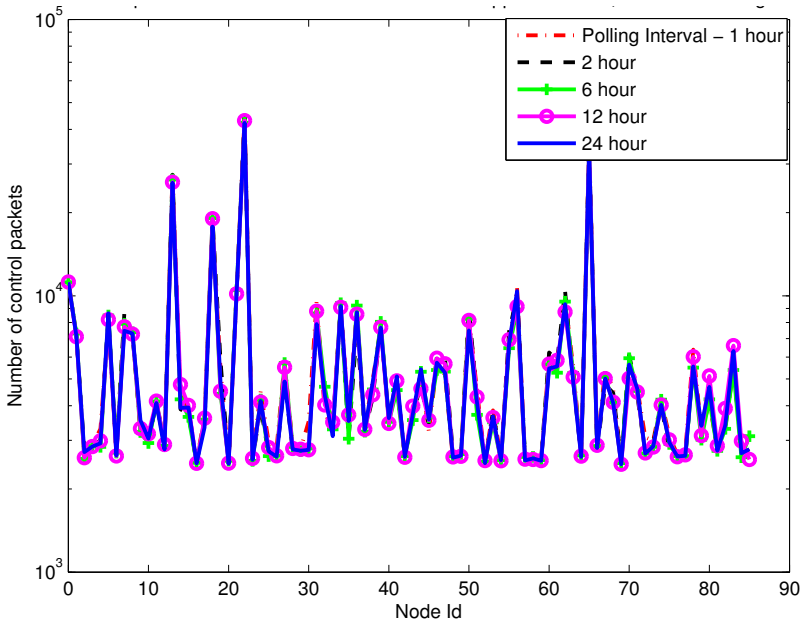

Figure 4.10: Control overhead vs. node ID for different application rates, RPL non-storing.

total control packet volume for RPL non-storing mode of operation for different polling intervals. The LBR is set to probe each meter in a round robin fashion with 1, 2, 6, 12 and

24 hours intervals. Clearly, being a proactive protocol, RPL's performance with respect to control overhead does not depend on which fashion application data is being generated, as shown by the overlapping results.

4.4.3 Path quality

Reactive and proactive protocols do not differ much on the path quality. There are several implementations of reactive protocols where a destination node may or may not wait to release a RREP after receiving a RREQ for itself. In LOADng, even if multiple RREPs are received, the data packet is released when the first RREP reaches the originator, which in most of the cases does not result in the best path selection between two nodes. On the other hand, in RPL, P2P paths may often be very non-optimal. RPL may also not select the best path, as it ignores paths that do not yield a certain percentage (20% as mentioned in Section 4.3.2) of improvement of path cost when it receives a DIO. However, the majority of the data traffic in LLNs flow between the LBR and the meters, the path quality does not reflect the non-optimum path length for RPL protocol in peer-to-peer scenarios. Hop distance or similar path quality metrics are not observed as the most important in LLN deployments.

In LOAD-ng, even if multiple RREPs are received, the data packet is released when the first RREP reaches the originator, which in most cases does not reflect the best path between two nodes. Similarly, RPL may also not select the best path, as it ignores paths that do not yield a certain percentage (20% in this study) of improvement of path cost when it receives a DIO. Figure 4.11 shows the CDF of the end-to-end hop distance for RPL and LOAD-ng for both networks. It is observed that LOAD-ng provides paths with almost the same number of hops as RPL's. For instance, in the smaller network, 85% of end-to-end communications in RPL took paths with length less than or equal to 8 hops, while the same happened with LOAD-ng. For the larger network, 85% of communications are retained within 12 hops for both protocols. The CDF is calculated over thousands of packets to and from the collection point, and we can observe all protocols exhibit similar

performance in terms of hop count. Additionally, Figure 4.12 shows the CDFs of the ETX path cost for all packets in the large network for both modes of RPL and LOADng, which are almost identical. Hence we can conclude that "path-quality-wise", the three protocols exhibit similar behavior even for a large topology for the specified traffic pattern. For fair comparison, such a P2P type application is studied in Section 4.5.

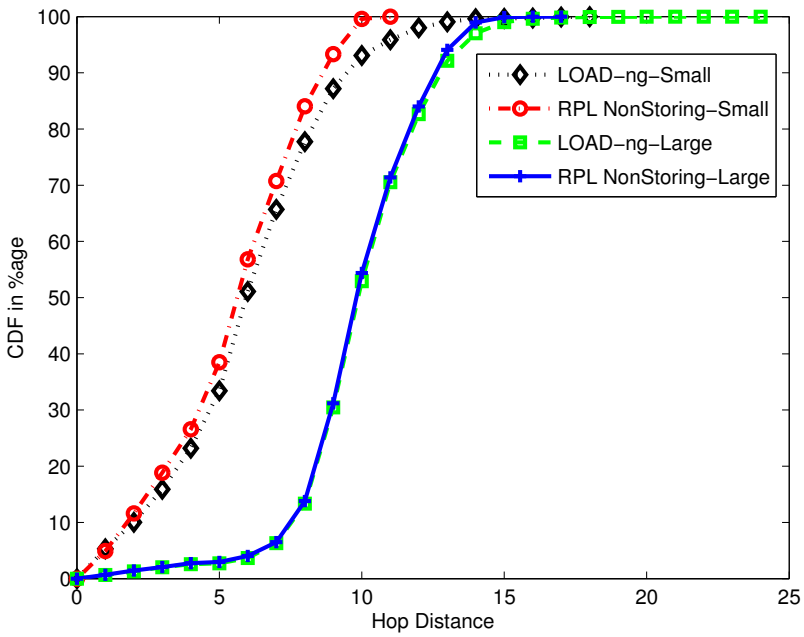

Figure 4.11: End-to-end hop distance for RPL and LOAD-ng.

4.4.4 High end-to-end delay

Data in LLNs can be of different types, origins and may require different Quality of Service (QoS) requirements. Some data, such as periodic reports, are delay tolerant up to even few tens of seconds, whereas some are very delay sensitive, for example emergency alarms, fault notification and alert packets. According to RFC5673 (6), in industrial setting, "non-critical closed-loop applications have a latency requirement that can be as low as 100 milliseconds". Clearly, these types of alert packets need a path to the destination

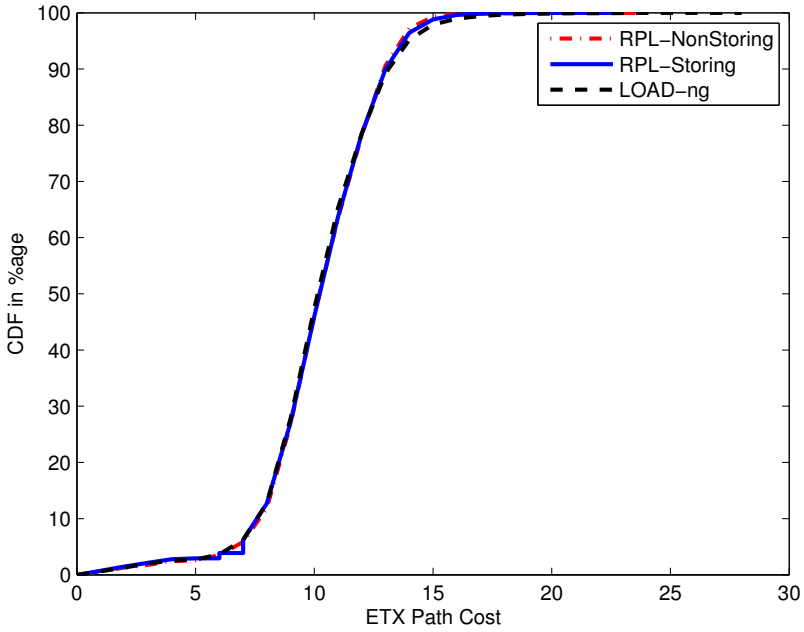

Figure 4.12: CDF of the total ETX path cost in a large network.

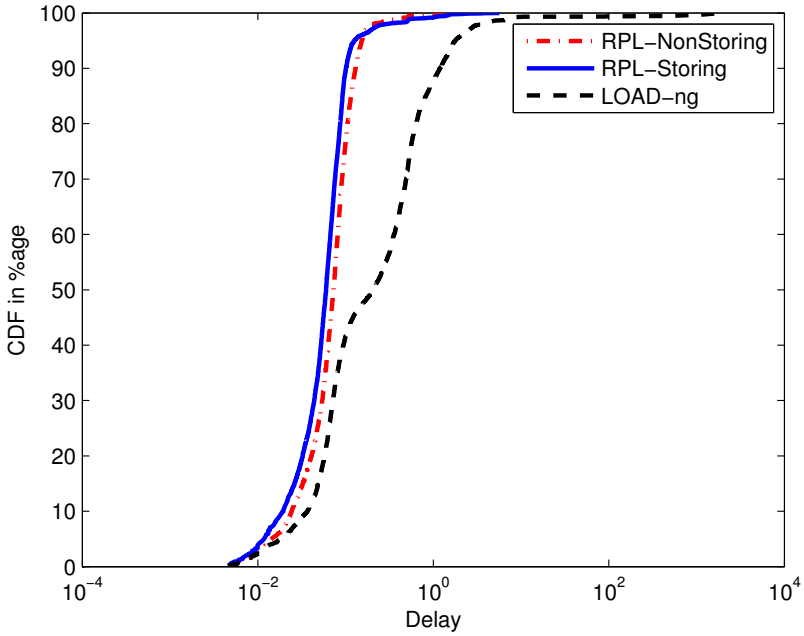

Figure 4.13: End-to-end delay for RPL and LOADng.

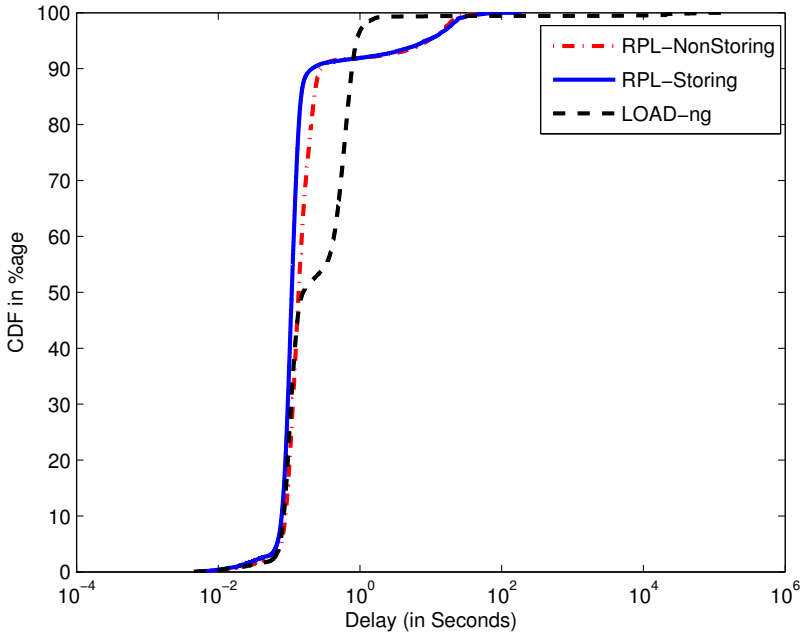

Figure 4.14: End-to-end delay for RPL and LOADng - large network.

as soon as they are generated. This section of the simulation results intends to compare end-to-end delay effectiveness of the protocols under study. For a reactive protocol, it is well established that the delay is larger due to on-demand path computation. This study not only confirms the same, but also points out pathological cases which may further affect the delay bound provided by the protocols. Figure 4.13 and Figure 4.14 show the Cumulative Distribution Function (CDF) of the end-to-end delay for both modes of RPL and LOADng for the small and large deployments respectively, where the X axis denotes the delay in seconds (in logarithmic scale) and the Y axis demonstrates the corresponding CDF value. For instance, it can be seen in the figure that 90% of the packets have been delivered within 0.1 seconds or less with RPL non-storing mode, while only 65% of the packets have been delivered within the same time-frame with LOADng.

This behavior stems from the fact that LOADng, and reactive protocols in general, first set up the path before sending the data. Since data communication in LLNs between any two peers is not very frequent (2 hours as mentioned in Section 4.3.1), the established path to a particular destination may be invalid next time data is generated for that

particular destination. Also, the next hop might be unreachable, given the fluctuating link conditions in LLNs. Hence, practically every time that peers need to communicate, they need to flood the network with RREQs (preferably in a controlled manner), wait for RREP to be received, and then release the data packet, incurring a larger end-to-end delay. The results for LOADng also show that some data packets may suffer a delay of a few tens of seconds to reach the destination. This is due to the loss of RREQ, RREP and/or RERR packets. Most of these data packets, which take few tens of seconds or more to get delivered, result from multicast traffic that clogs the network with control traffic, as we will observe in Section 4.4.1.

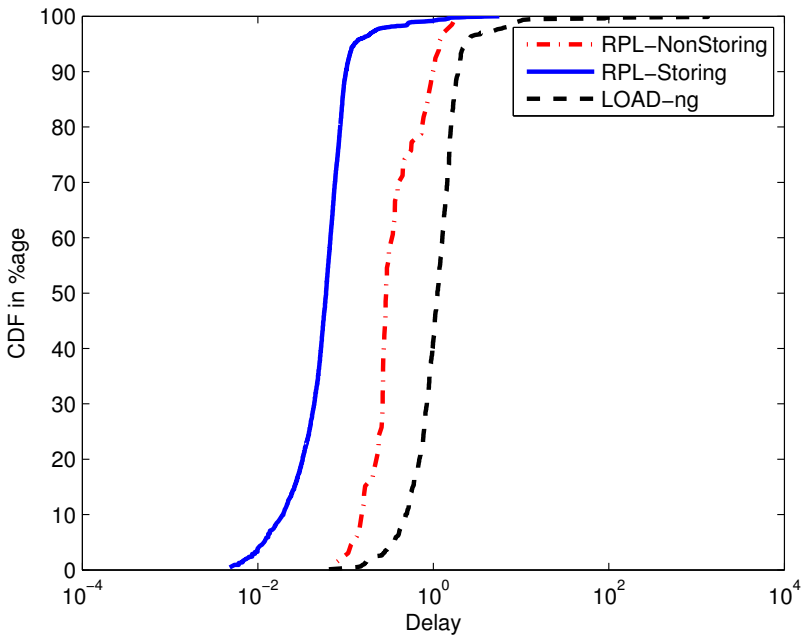

Figure 4.15: End-to-end delay for RPL and LOADng, zoomed in.

We then consider only cases where the data packet has been lost, or any RREP/RREQ has been lost due to collision, link conditions, etc. Figure 4.15 zooms into such pathological cases. Again, we plot the CDF of delay versus the delay value (in seconds). Since RPL implements backup parents and backup routes (at the DAG root or collection point), the

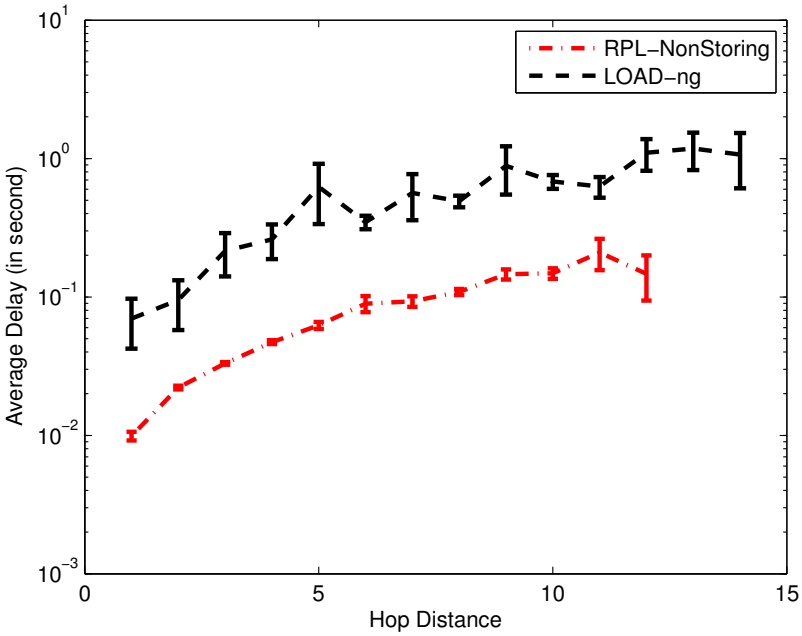

Figure 4.16: End-to-end delay vs. hop distance for RPL and LOADng.

alternate route can be chosen rather quickly upon any delivery failure of a data packet from the lower layer. For LOADng, however, a new route discovery must be initiated. Moreover, on the loss of RREP packets, the data packet will only be transmitted when another RREP for the same destination arrives at the source through another path.

In Figure 4.16, we plot the average delay against hop distance for LOADng and RPL non-storing mode, along with their 95% confidence interval. The delay statistics for each hop distance x is gathered from the time each packet takes while traversing a distance of x hops. For each hop distance x, we calculate the average as well as the confidence interval from the delay statistics generated by all such packets. We observe that not only RPL has a lower average delay for any given hop distance between two peers, but also the variance is much lower, yielding a more reliable delay bound. Since both modes of RPL showed similar delay bound in Figure 4.13, for the sake of clarity, this plot does not show RPL storing mode.

The unreliability in delay bound for LOADng, statistically shown as a larger 95% confidence interval is also attributed to the reactive nature of the protocol. Since new

communication requires a route request broadcast, such communications may result in higher delay due to broadcasting, RREP loss, etc. On the other hand, some communications from the same originator to destination happen right away, as the path is established. Note also that communications in reverse direction between the same peers (as happens in query-based communication) do not require route discovery, and may take place as instantly as the application layer packet reaches the network layer. For these reasons, the end-to-end delay for the same pair of peers, or over the same number of hops, may vary widely over simulation time.

4.4.5 Impact on memory requirements

Memory constrained devices with low power are the pivotal components of LLNs, therefore it is important to study the memory requirements of each protocol. Reactive routing protocols usually rely on route caching for discovered destination. With LOADng, nodes build their routing table based on RREQs and RREP packets received, therefore if any node participates in multiple active flows in the network, the node needs to store next hop and validity information for each source and destination node. Thus, depending on the user traffic, some nodes tend to increase their routing table size proportional to number of flows passing through themselves. However, characteristics of LLNs never guarantee enough storage space in any node for storing routing tables. Destination oriented flooding in LLN, tends to worsen this situation. Multiple route requests may reach the same node at the same time for different destinations. Even though the destination may never be reached through the concerned node, the nodes still have to process and re-broadcast each request. On the other hand, RPL non-storing mode does require nodes to store any routing information. In this section, we will demonstrate maximum memory requirement for the protocols such that no packet or routing entry is dropped. To be noted, plots which deal with maximum buffer or RAM occupancy, maximum value is of more importance than any other statistical information such as average value with confidence interval. If a single node in the network in a single scenario needs 'M' KB of RAM, then all nodes for that

network need to be equipped with 'M' KB of RAM.

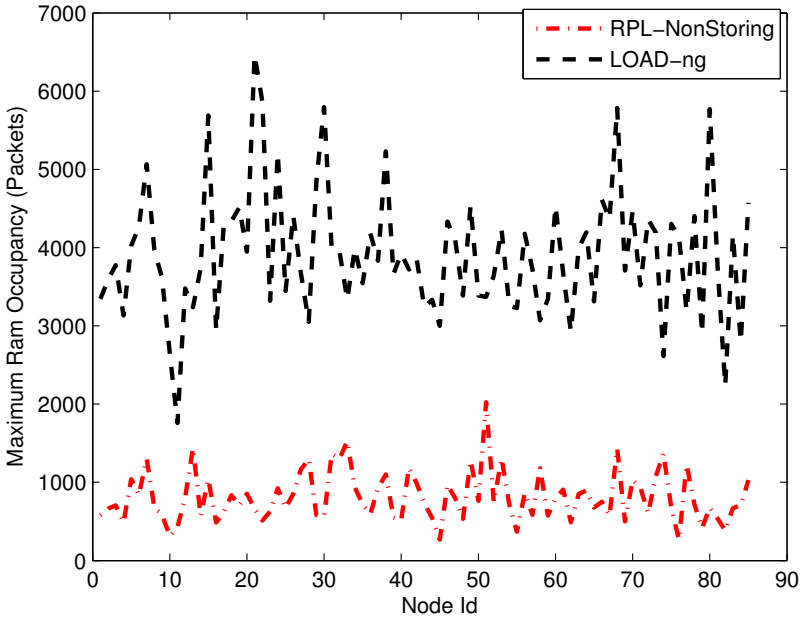

Figure 4.17: Maximum RAM occupancy - RPL non-storing vs LOADng.

Figure 4.17 shows the maximum RAM that has been occupied (in Bytes) against the Node ID for RPL non-storing mode and LOADng for the smaller deployment. It should be noted that this RAM occupancy is only a result of storing routing tables (as in LOADng), parent tables (as in RPL), queued packets for transmission, any kind of data structures, etc., and does not include the amount of RAM space the protocol code itself would occupy for implementation. Note also that the collection point has been excluded from this analysis, as the collection point is supposed to be a computationally resourceful device, irrespective of the routing protocol. The maximum RAM requirement for RPL has been found to be 2 KB, and that for LOADng is 6 KB. However, it should be pointed out that LOADng's code would occupy less memory than RPL's; however in a large network that advantage may well be lost due to high buffer and routing table requirements for LOADng. Figure 4.18 shows the maximum RAM occupancy (in Bytes) for LOADng and

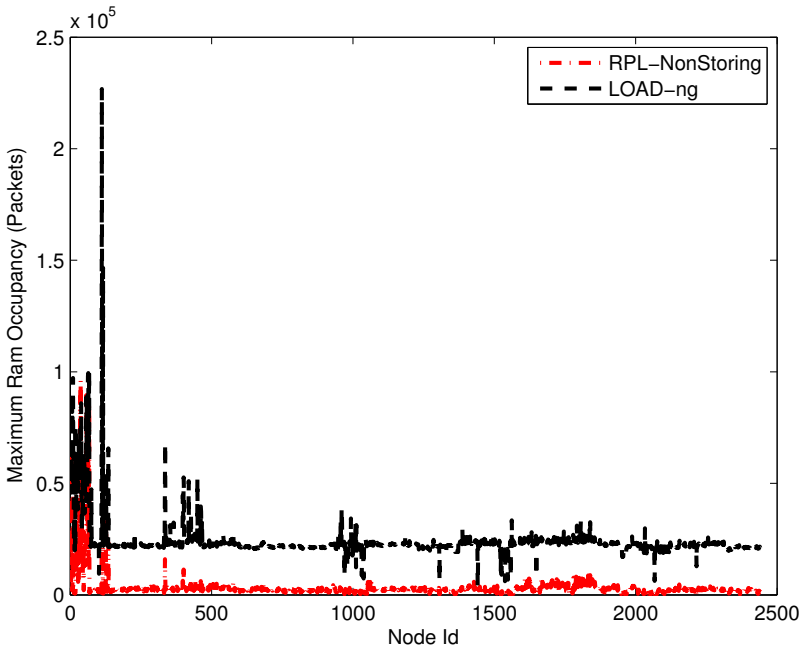

Figure 4.18: Maximum RAM occupancy in bytes for each node in a large network.

RPL non-storing mode for the large smart grid deployment.

4.4.6 Qualitative comparative analysis

4.4.6.1 Flooding issues in LLNs

A reactive protocol is well-suited for a traffic pattern where data transfer is not very frequent. However, if the traffic pattern includes periodic data reporting, even as low as a few times in a day, the traffic pattern will induce periodic broadcast of route request throughout the network. A simple example scenario can clarify this: assume an application in a U-LLN requires periodic data reporting every 6 hours or 4 times in a day - morning, noon, evening and night. If the network consists of 2000 nodes, which is a very conservative number in a typical U-LLN, the application alone will create a route request broadcast for each sensor node every 11 seconds, on average. Thus, over the life of the sensor network, a reactive protocol will use more control overhead than a proactive protocol.

The amount of flooding to discover routes may also be controlled via tweaking the route

expiry time or route validity time. If a route is active, nodes should not waste network resources trying to find out the route to the same destination. Keeping a high expiry time for the routes, on the other hand may prevent flooding the next time data is generated for the same destination. However, the path may well have been invalid by the route expiry time. Considering LLN link characteristics, link flapping is a very frequent event. Hence, high route expiry time may lead a node to find out invalidity of a path, thus forcing to flood the network again for route discovery. Thus, increasing route expiry time or route validity time for an AODV-based reactive protocol may not prove to control flooding in LLN. Proactively choosing a back-up path proves to be an effective way to ensure valid routing path in presence of link flaps, whereas reactive routing approaches are not able to cope with link dynamics, thus preventing its usage for LLNs. Furthermore, if traffic is sent along a broken path, a new request would consequently be generated, thus increasing the control traffic load, in addition to incurring additional delays for the user data.

4.4.6.2 Impact of flooding in battery operated nodes

Note that there is a lot of evidences supporting the claim that using flooding or scoped flooding to discover routes is ill-suited to power constrained Low-power and Lossy Networks (LLNs) in general. This is due to the low-power requirement. In low-power wireless networks, broadcast packets usually cost much more energy by the hardware to transmit than unicast ones due to implementations of sleeping mechanisms. As pointed out in (59), supporting broadcast transmission is difficult while implementing Low Power Listening (LPL) due to asynchronous duty-cycles of nodes. As the wake up time for each node in a neighborhood is independent, often multiple transmissions of a single frame are required to emulate one broadcast transmission. These multiple transmissions may consist of unicast transmissions to each of the neighbor (RI-MAC, see (60)), or repeated transmition of the frame during the whole sleep interval as done in X-MAC-UPMA ((61)). Otherwise a really long preamble would be required, thus increasing power consumption more for broadcasts packets in either case. Ad-hoc networks which are normally always on, will

not face this problem. Hence, reactive routing methods that use flooding mechanism for route discovery often, are more suited for Mobile Ad-hoc networks, but not for LLNs in general. LLNs should limit use of flooding or broadcasting packets as much as possible via algorithms such as Trickle (38). However, at this point, we would exclude consideration of such hardware dependent energy expense from our discussion.

4.4.6.3 Lack of support for routing based on node capability

Apart from providing a route between any two nodes in the network, a routing protocol suitable for LLN should be able to handle additional constraints. An LLN mainly consists of constrained devices, both functionality and memory-wise, and inherently heterogeneous in nature. Hence, any routing protocol suitable for LLNs should support node constraint based routing. This requirement is mandated in RFC5548 (7) as follows: "the routing protocol MUST be able to advertise node capabilities that will be exclusively used by the routing protocol engine for routing decision". For example, the routing protocol should avoid a node with less battery power while routing to reach a server. Similarly, for industrial automation requirements, RFC5673 (6) also needs a routing protocol to provide device-aware routing, as it describes "The routing algorithm MUST support node-constrained routing (e.g., taking into account the existing energy state as a node constraint). Node constraints include power and memory, as well as constraints placed on the device by the user, such as battery life". For home routing automation, RFC 5826 (5) specifies, "The routing protocol SHOULD route via mains-powered nodes if possible. The routing protocol MUST support constraint-based routing taking into account node properties (CPU, memory, level of energy, sleep intervals, safety/convenience of changing battery)".

Clearly, recognizing a node's capability and routing accordingly is an important aspect for any routing protocol designed to be suitable for LLNs. However, any AODV-based protocol (such as AODVv2, formerly DYMO (62) and LOADng), in their current specification, fail to provide routes based on any such constraint. Currently known reactive

routing protocols do not have any provision to determine whether the next-hop node in a route has enough battery power to sustain the route, or whether the next-hop node is main powered or provides a particular functionality. Thus, these protocols fail to provide requirements mandated by (5, 6, 7) for routing in an LLN.

4.5 Performance Results for P2P Communication

Clearly, the performance of LOADng depends on the application characteristics and in what manner nodes communicate with each other. If every node talks to a single meter within a short time duration, a single RREQ broadcast is sufficient for many (however, not all) nodes in the network to gather route information about the destination. At the same time, a single RREQ broadcast provides all nodes in the network with the route to the originator. Thus, while some applications may consume little control overhead, others may create a broadcast storm.

P2P traffic in this section is simulated as follows: every meter communicates with another meter in the network other than the LBR. Each node generates a packet every 60 minutes, and communicates with a different node in each interval. Therefore, no two nodes communicate with the same node in any given 60 minutes interval.

4.5.1 Path quality

Figure 4.19 shows the CDF of the path length (in number of hops) for RPL storing mode, RPL non-storing mode and LOADng, with the above P2P traffic profile. It is observed that RPL non-storing mode has a large path length for true P2P application, as all communication is directed via the LBR. LOADng and RPL storing mode result in very close path lengths, even if storing mode does not yield optimum path quality in terms of path length.

Figure 4.20 shows the CDF of the ETX path cost (plotted in Y axis) against the ETX path cost value (plotted in X axis) for both modes of RPL and LOADng. As before, the ETX path cost for RPL storing mode and LOADng are very similar, but LOADng

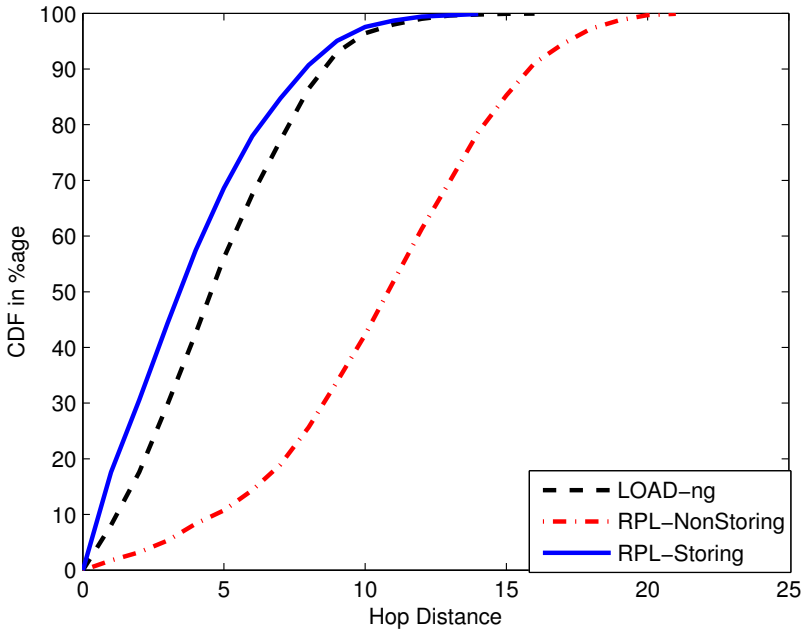

Figure 4.19: End-to-end hop distance for RPL and LOADng; P2P application.

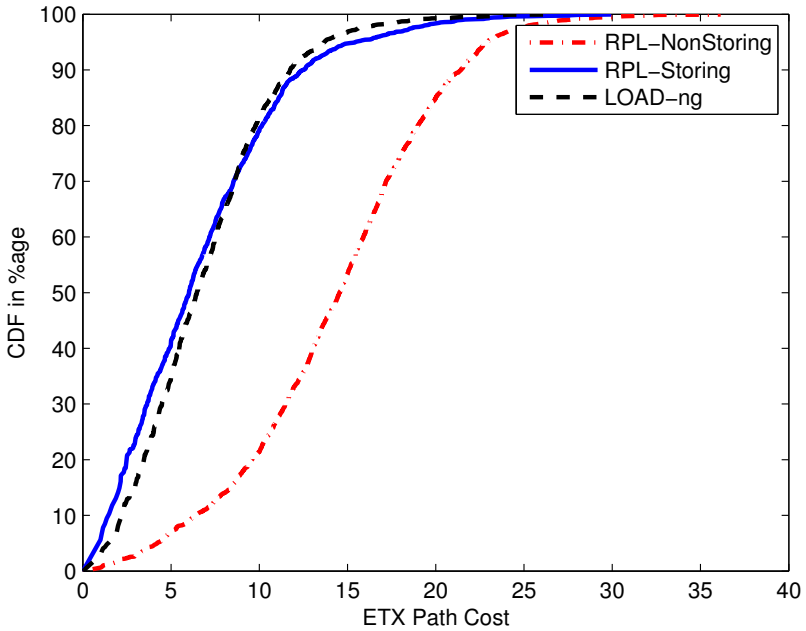

Figure 4.20: ETX path cost for RPL and LOADng; P2P application.

sometimes produces a path with lower cost. RPL non-storing mode is not optimized for P2P applications and therefore provides path with much larger ETX path costs than both

storing mode and LOADng.

4.5.2 End-to-end delay

Figure 4.22 shows the CDF of delay in seconds for both modes of RPL and for LOADng for the P2P application scenario. Interestingly, even though RPL non-storing mode incurs a high cost to reach the destination in terms of hop count and ETX path cost, for many paths it provided lower delay than LOADng. This phenomenon can be explained by the reactive protocol's reaction time for a non-existent routing table entry or a route that is no longer valid. But once the route is established, and the path is valid, LOADng may provide lower delay than RPL non-storing mode, as observed in the figure, for up to 55% of the received packets. RPL storing mode, however, outperforms both, providing overall lower delay.

4.5.3 Memory requirements

We also compare the maximum RAM requirement for RPL non-storing mode and LOADng, shown in Figure 4.21. RAM occupancy in RPL storing mode is topology dependent and, for the topology in use, it is very similar to that of LOADng. It is observed that RPL has a maximum RAM occupancy of around 3 KB, and LOADng has a maximum RAM occupancy of around 10 KB. However, these results depend on traffic pattern, frequency, etc. As before the LBR or collection point of the network has been excluded form the memory analysis, as it should be a computationally resourceful device. The memory footprint is calculated only for resource constrained LLN nodes. Note that, as mentioned in Section 4.4.5, the amount of RAM space that would be occupied by the code for implementation of each protocol is not included in this simulation, and it would be smaller for LOADng.

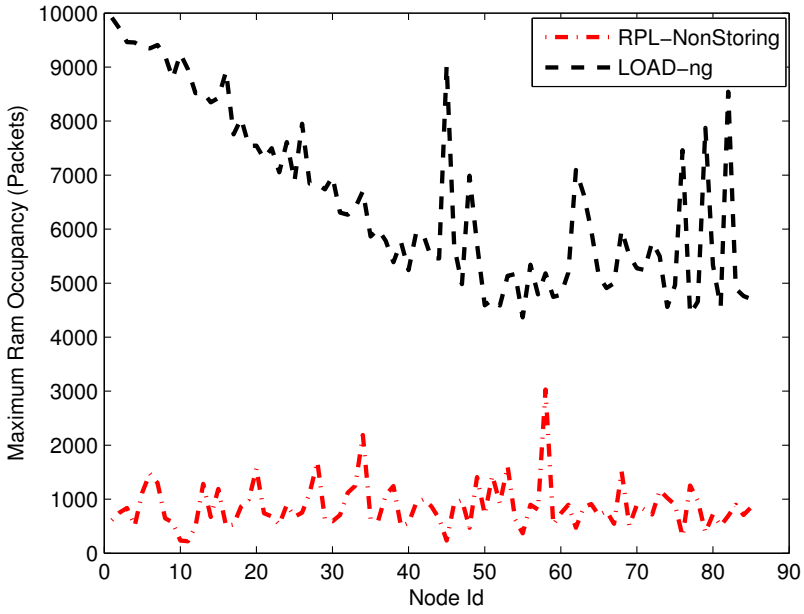

Figure 4.21: Maximum RAM occupancy for RPL non-storing and LOADng; P2P application.

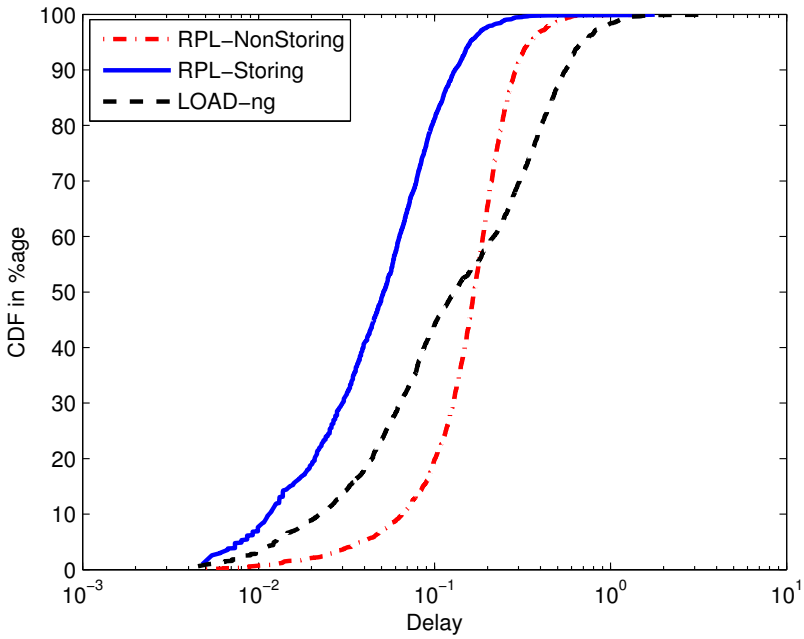

Figure 4.22: End-to-end delay comparison : P2P application.

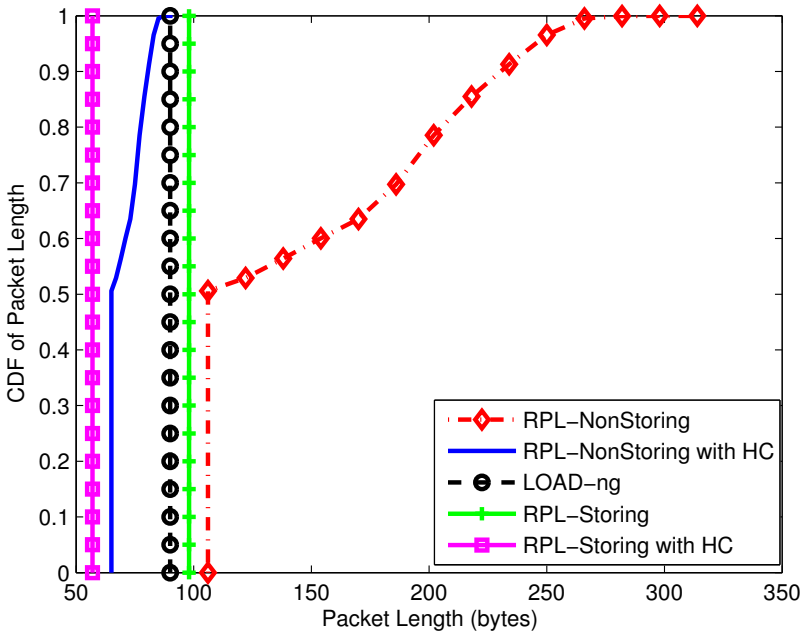

Figure 4.23: Packet length for LOADng, RPL storing and non-storing modes.

4.5.4 Packet length

RPL non-storing mode does not store any route information. While this approach makes RPL viable in very low memory equipped devices, it has its own disadvantages as the protocol has to perform source routing via the LBR. We have already seen how it affects path length in P2P communication. However, another disadvantage may arise from the source routing header (SRH): the farther away a node is from the LBR, the larger the data packet SRH needs to be. Figure 4.23 shows the CDF of data packet lengths for all packets that travel the network. The X axis corresponds to data packet length in bytes, and the Y axis indicates the corresponding CDF. The application layer data is 50 bytes in size, as indicated in Section 4.3.1, under traffic pattern. When header compression is not performed, all the packets also bear a 40 bytes IPv6 header. From the figure we can clearly see that RPL non-storing mode has a much larger packet length than LOADng. This can be problematic when RPL operates over low MTU links, such as IEEE 802.15.4 links with an MTU of 127 bytes. RPL storing mode normally yields packet lengths comparable to (though slightly larger than) LOADng. However, when header

compression is performed with RPL, in accordance to the compression format for IPv6 datagrams over IEEE 802.15.4-based networks in RFC 6282 (63), all packets are well under the MTU requirement, and thus source routing is possible for LLNs operating with low MTU links.

It should be noted that this result has been obtained from a set of P2P communications over the network. However, the variable packet length only arises due to SRH in non-storing mode from LBR downward the DAG to other nodes. Hence, this result can be obtained if the communication includes any P2MP traffic.

4.6 Scaling Properties

It is hard to estimate the general behavior of a protocol with respect to network size when only two topologies have been simulated. To hint at the results not being completely topology dependent, it is necessary to study a varied range of network sizes. Hence, in this section we study LOADng's and RPL's performance in topologies of different sizes: 45 nodes, 86 nodes, 200 nodes, 500 nodes, 1000 nodes and 2442 nodes.

Network scale has most severe impact on two metrics: control overhead and resource utilization. Figure 4.24 shows the average control overhead per node for all three protocols, with varying topology size. The network size is plotted in the X axis in logarithmic scale, whereas the average control packet is plotted in the Y axis, also in logarithmic scale. One can observe that the average control packet overhead increases more sharply for LOADng as the size of the topology grows. Note that although some os these topologies were created randomly, all link characteristics were gathered from the database created from a real deployment. Some networks have links that are more stable than others and this explains why the largest network is found to be most stable one, with the least link variation, and hence we observe the decrease in average control overhead per node for the largest network to be less than that of the network with 1000 nodes. Figure 4.25 shows the total control overhead of all nodes in the network for these three protocols with varying topology size.

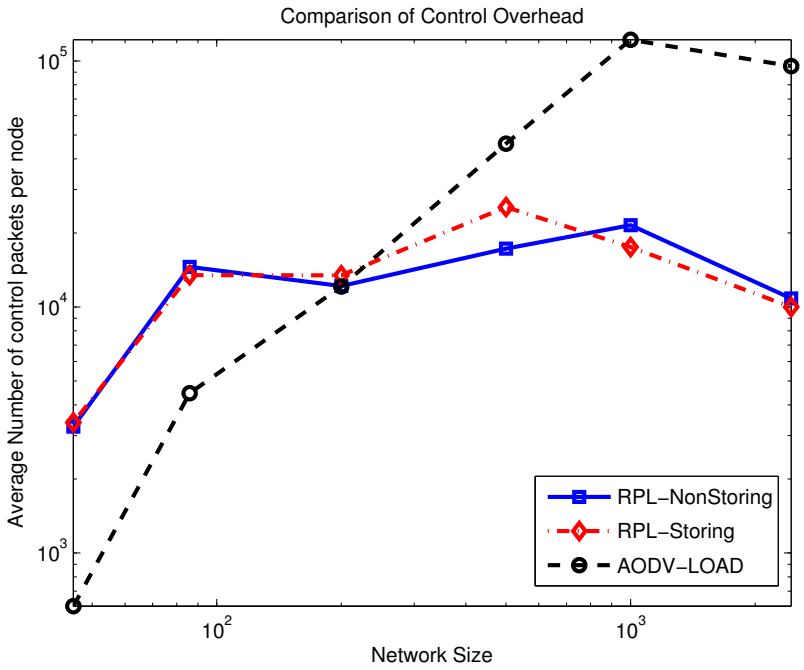

Figure 4.24: Average control overhead per node with network size.

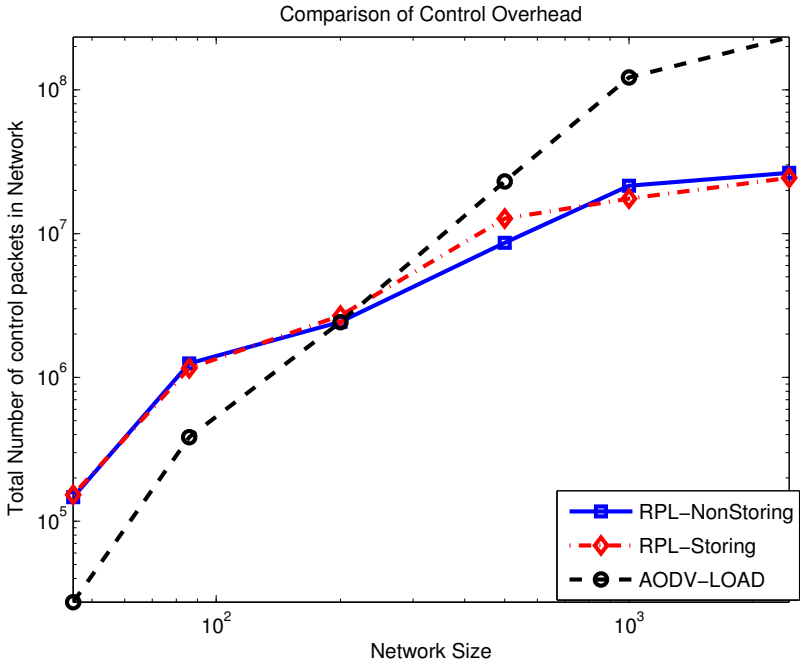

Figure 4.25: Total control packets in network with network size.

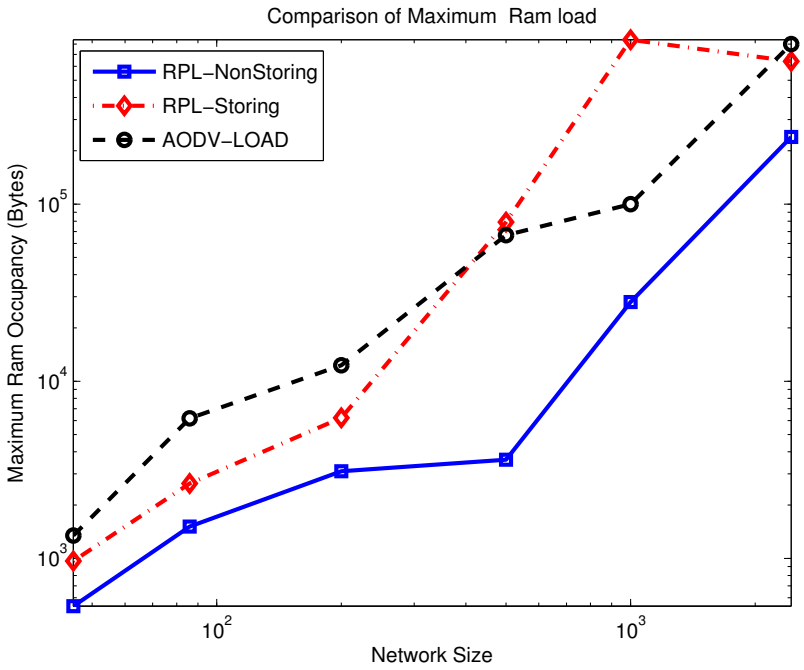

Figure 4.26: Maximum RAM occupancy in bytes against network size.

As expected, LOADng is more efficient for smaller networks with light traffic load. The average control traffic volume for these protocols is very similar when the network size is around 200 nodes, for the case where periodic data polling occurs for each node every 6 hours. This can also be observed with more frequent statistics reporting for a smaller topology.

This study also recognizes the scarcity of resources while deploying an LLN, and hence, it is impossible to overlook the resource utilization while discussing the scalability of the protocols. With increased number of nodes, LOADng and RPL storing mode will need to store more entries. However, the difference between the two is that for LOADng all nodes in the network store the route to an RREQ advertising node, while the routing table size in RPL storing mode decreases with distance from the collection point. On the other hand, RPL non-storing mode does not store any route in any node other than the DAG root. The DAG root or collection point is a much more capable device and therefore it is not resource constrained, thus we limit the calculation of maximum RAM or packet

buffer occupancy to all LLN nodes other than the collection point. Hence, if we consider the maximum RAM utilization, RPL non-storing mode is the least resource consuming. Figure 4.26 shows the maximum RAM occupancy in bytes in the Y axis, against the network scale in a logarithmic X axis.

It might be surprising that despite the fact that RPL non-storing mode does not store route information, the maximum RAM requirement does increase with the network size. The reason for this is that a large portion of RAM is utilized to buffer packets, in either modes of RPL, during each global repair, where DAO packets are propagated upward to the DAG root, and DAO-ACK packets flow downward to the nodes. This leads to a congestion near the collection point: nodes with lower rank. This explains the increasing trend for maximum RAM utilization as observed in Figure 4.26. Note that the two RPL parameters, DAO_ACK_TIME and DAO_LATENCY, can be tweaked to achieve less congestion and less buffer requirement. The first parameter describes how long a node should wait to emit a new DAO for each global repair and/or parent change, and the second moderates how long to wait for an acknowledgment of emitted DAO packet before a new one is sent out.

Motivated by the scaling property study, we set out to improve DAO packet emissions in RPL. Next in Section 5.5, we propose a combination of distributed and centralized algorithms to control DAO packet emissions by adaptively tweaking the parameter DAO_LATENCY in each node to greatly limit DAO congestion and thus restrict the requirement of larger packet buffers. A preliminary study of our proposed algorithms was presented in (64) and is here extended and improved by considering a combination of the previously proposed algorithms.

It should be noted that the two RPL parameters, DAO_ ACK_TIME and DAO_LATENCY, can be tweaked to achieve less congestion and buffer requirement. The first parameter describes how long a node should wait to emit a new DAO for each global repair and/or a parent change, and the second moderates how long to wait for an acknowledgment of emitted DAO packet before a new one is sent out. In next chapter, we provide a combination

of distributed and centralized algorithms to control DAO packet emission by adaptively tweaking the parameter DAO_LATENCY in each node to greatly limit DAO congestion and thus restrict the requirement of larger packet buffers.

4.7 Summary

This chapter presented a detailed performance comparison study between RPL (storing and non-storing node) and LOADng, for several topologies of interest. In particular, results were collected for a small deployment topology with 86 nodes and a large smart meter deployment with 2442 nodes. Other topology sizes were also considered in the scalability study. n the course of this investigation, we also uncovered non-optimal protocol behavior for the case of large networks and proposed new mechanisms that are shown to improve control plane congestion, as a result improving network lifetime for large scale LLNs. Some of the important observations drawn include:

- In terms of control overhead, RPL scales well with the network size. In particular, for the large deployment of smart grid AMI network with 2442 nodes, RPL provided connectivity with the border router with much less control cost;
- Control overhead is a function of application data rate for LOADng, where it is independent for RPL.
- Path quality in terms of hop distance and total ETX path cost is very close for P2MP and MP2P traffic. However, for P2P traffic, RPL non-storing mode yields a much longer and more costly (in terms of total ETX path cost) path;
- End-to-end delay is comparable between the two protocols for the topologies studied. However, in some cases, LOADng may result in a high end-to-end delay between nodes. This is explained by the reactiveness of the protocol as well as control plane floods;

- Due to control packet flooding and buffering of data packets, LOADng has higher buffer size requirement.
- RPL tends to consume more buffer space with increase in network size, but proper scheduling of destination advertisements make RPL scale much better with increasing topology size.

In general, RPL outperformed LOADng for several critical metrics, taking into account the traffic profile that is typical in LLNs, and only in a few cases, both protocols had similar performance. It is true that LOADng's complexity is lower and if run in a topology with very light traffic load, the protocol will thrive. Such a scenario is however becoming further and further away from the reality of current LLNs.

5

RPL: Control Plane Congestion Mitigation in Non-Storing Mode

Urban Low-Power and Lossy Networks (U-LLN) often span a vast geographical region and consist of thousands of nodes. These networks are characterized by highly time variant nature of the links, with nodes having a mere few KB of flash memory, and large-scale deployments. RPL is envisioned as the routing protocol to interconnect smart objects in the Internet of Things (IoT), in smart grid AMI networks, etc. RPL has been designed with mainly two modes of operation: storing and non-storing, with the former implicating nodes' ability to store routing table information. Non-storing mode relaxes that requirement, and therefore is deemed more appropriate for large deployments of nodes with limited memory resources. We have however observed in Section 4.6 that contrary to our intuition, for very large networks, RPL non-storing mode operation actually requires large memory and buffer space. Proportional scaling of memory requirements with network size is disadvantageous for any routing protocol intended to be implemented in vast U-LLNs or Smart Grid AMI networks.

Investigating the causes for such an non-intuitive behavior from a mode that was actually designed to cope with the large deployments of nodes with limited capacity, we determined the culprit: congestion caused by the Destination Advertisement Option

(DAO) packets from every node to the DAG root during a global repair, or after the DAG root triggers a network-wide address accumulation using a new DAO Trigger Sequence Number (DTSN). This congestion is lethal due to a number of reasons: Firstly, upon congestion, the buffer requirement will increase, leading to high memory requirement to store packets. Secondly, if high buffer space is not available, DAOs will be lost and the nodes will be forced to send duplicate DAO packets, which may worsen the situation. Finally, important and time-sensitive data or alert packets may be lost as well. Therefore, in order to control the congestion in large deployments of memory constrained nodes, it is essential to design a suitable approach for DAO message emission. In this chapter, we investigate the reason why the buffer and memory requirement increases for a protocol that was actually designed to cope with the large deployments of nodes with limited capacity.

We propose two adaptive algorithms to control DAO emissions in RPL non-storing mode (one centralized, or controlled by the DAG root, and one distributed). We show that instead of using a fixed universal timer to control DAO emissions, as recommended in the standard, making use of an adaptive timer at each node allows the network to adjust itself to account for topological changes, and adjust each timer in order to avoid congestion and packet drops, specially near the DAG root, which would also impact data delivery delay. Sections 5.2 and 5.3 propose distributed and centralized (run at the DAG root) algorithms respectively to determine the time duration between receiving a global repair and issuing a DAO packet while no address aggregation is performed. To the best of our knowledge, this is the first work to reveal the DAO emission issue, and to shed light on how congestion may occur in a large scale data aggregation network, and further, how one may avoid it, thus setting up the groundwork for further improvement in the protocol.

5.1 DAO Specific Operation in RPL: Motivation for Optimization

A node generates a DAO packet in the following cases:

- The parent set or most preferred parent of the node has changed;
- The node has received a new DODAG sequence number, indicating a global repair of the network;
- The node has received an increased value of DAO Trigger Sequence Number (DTSN) to refresh downward routes.

As specified in (15), when a new DODAG sequence number or a DAO message from its own sub-DAG is received, a node schedules a new DAO to be propagated upwards. In both cases, the node delays the emission of the new DAO with a timer named DelayDAO Timer. The value of this timer is constant, and its default value is defined by the parameter DEFAULT_DAO_DELAY, which is set as 1 second. Once the DAG root receives information about all the destinations, only then, it will be able to direct P2P or P2MP packets to the proper route. If it does not have the route to the particular direction, it must drop the packet for that particular destination.

5.1.1 Trade-off on designing the value of DelayDAO Timer

On one hand, if the value of the DelayDAO Timer is fixed and low, the nodes will quickly emit DAOs after entering global repair (LBR emitted a DIO with a new sequence number) and therefore the whole DAG information should take less time to reach the DAG root. However, within the same DODAG iteration, a node may receive DIOs with better cost to the DAG root through other nodes. Hence, it may switch parents, forcing yet another DAO transmission. On the other hand, a large value of the timer will save on the control plane overhead, but waiting longer at each level of the DAG to generate and/or to forward a DAO to the DAG root will ultimately lead to much delay for the DAG root to construct a full routing table of the DAG. Clearly, there is a trade-off between the number of DAOs transmitted, and the time required by the DAG root to have a full view of the topology.

We have further observed that the effect of the value of DelayDAO Timer, for large scale networks, becomes more severe than a few extra DAOs being issued or the DAOs

reaching the DAg root late: intermediate nodes need to store a DAO from their sub-DAG before relaying it to their parents. In an urban LLN implementation consisting of thousands of nodes, nodes closer to the DAG root will need large buffer space to hold DAO packets for later transmission, while these LLN nodes may have only a few KBs of flash memory. Also, as we will observe later in this section, a constant value of DelayDAO Timer for all nodes of different ranks, leads to huge amount of DAOs to be transmitted within a small time duration, therefore creating congestion and further increasing buffer requirement for nodes closer to the DAG root.

5.1.2 Bottleneck due to a constant value of DelayDAO Timer

Recall that RPL creates a Directed Acyclic Graph (DAG), where links between preferred parents and children create a spanning tree of the network through which data aggregation (MP2P) or dissemination (P2MP) takes place. However, the tree thus created, rooted at the DAG root, is not guaranteed to be a balanced tree, since the number of children under each parent may not be the same. In this section, in the interest of deriving theoretical lower bounds, we assume that the tree created by constructing a DAG is a balanced tree, where every node in the network has B children, and height of the tree is H.

We assume the value of DelayDAO Timer is T_{DD}. The number of DAOs that are generated from the ith level below a node is B^i. These DAOs arrive after $i * T_{DD}$ time, with some random jitter that depends on value of $Imin$. Clearly, in each successive interval of T_{DD}, the node needs to receive B^i DAO packets, and transmit B^{i-1} DAOs. At the same time, during the next interval of T_{DD}, it needs to buffer B^i DAO packets from its own sub-DAG. Hence the minimum buffer requirement would increase exponentially for a node over successive intervals. In Figure 5.1, we show how the time a node's radio is busy varies with rank and time after it receives a DIO with DTSN increased or DODAG sequence number increased. In Figure 5.2, the minimum DAO buffer requirement without considering the effect of congestion for nodes of different ranks at different times is plotted. The average number of children is assumed as $B = 3$, and height $H = 8$. Note that this

analysis only provides a theoretical lower bound on the required number of DAO packets to be buffered. Clearly, in a real deployment, the radio will find other nodes transmitting at the same time, thus increasing required buffer space.

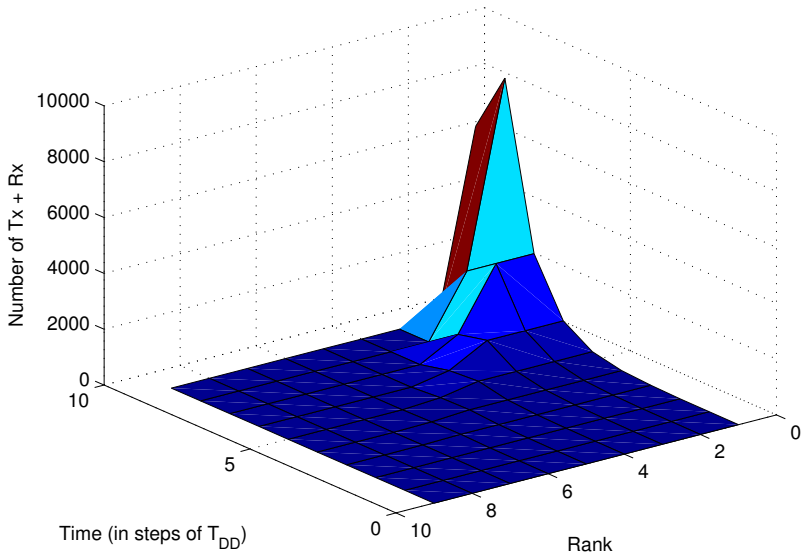

Figure 5.1: Total DAO transmissions (Tx) and receptions (Rx) versus time and rank.

We also consider a grid like topology of 1000 nodes with disc model for radio propagation, and assume the DAG constructed by RPL is the Breadth First Search (BFS) tree for the topology. In general, the number of DAO packets received by a node at the ith T_{DD} interval will be the same as the number of nodes in the ith level of it's sub-DAG. In Figure 5.3, we plot the number of times a node's radio is busy transmitting or receiving DAO messages against rank and time interval. Note also that this analysis does not consider the DAO-ACK or acknowledgement packets that traverse down the DAG, while some DAOs are being forwarded up the DAG. Clearly, the results presented in this section are optimistic and provide a lower bound.

From the above results, it is clear that in large networks, during simultaneous route refresh or global repair, the described mechanism to delay DAO packets may fail due to high congestion and excessive buffer requirements. Intuitively, the high buffer requirement

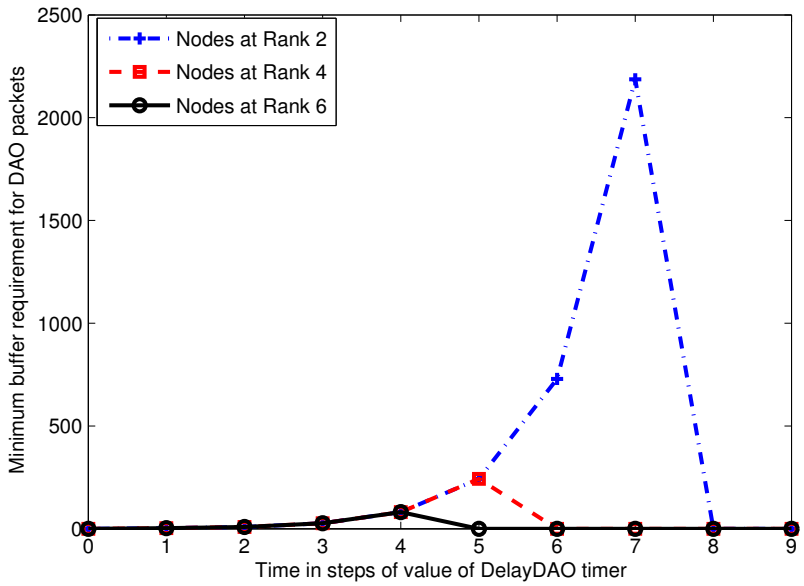

Figure 5.2: Minimum DAO buffer requirement against time for different ranks.

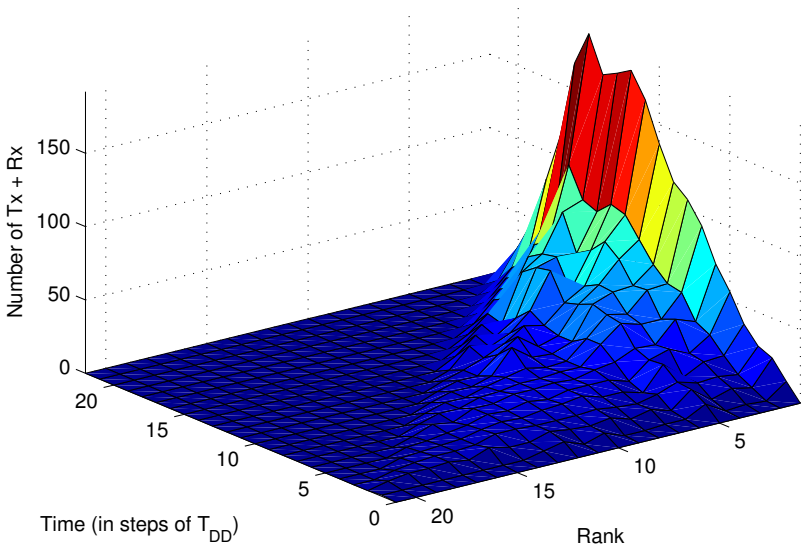

Figure 5.3: Total DAO Tx + Rx versus time and rank for the 1000-node grid topology.

stems from the fact that each node needs to store the DAO packets from its sub-DAG for possible aggregation of DAO routes. However, while implementing non-storing mode, route aggregation is not performed. Hence, a node should forward a DAO from its sub-DAG immediately to its parent, as opposite to what is pointed out in (15). Secondly, in

a particular level in the DAG, the generation time for DAO packets should be distributed within a time duration to prevent propagation of a high number of DAO packets at the same time. The time duration should increase exponentially with rank as we have observed that the number of nodes tends to increase exponentially with rank. This is illustrated in Algorithm 1, which determines the DelayDAO Timer duration. It assumes that each node generates a random number amongst an interval that increases exponentially with the rank of the node.

Algorithm 1 Generation of DelayDAO Timer Value

```
seed = <EUI_64>/MAC address or node ID
T_DD ← random(K * Base^(Rank-1), K * Base^Rank);
DAOStartTime := CURRENT_TIME;
Arm DelayDAO Timer with value = T_DD
Issue a DAO when DelayDAO Timer fires.
```

The parameter '$Base$' is intended to coarsely adjust the timing of DAO releases and to space out the destination advertisements well enough in time, so the packet buffer does not suffer high increase in leangth. A parameter 'K' is used to fine tune the DAO release timing and better accommodate the available bandwidth. This approach intends to trade off the delay by which the root node receives destination advertisements for nodes at higher rank, with in node packet buffer. In the next sections, we will present mechanisms that will be used to estimate values of the two parameters in this algorithm, 'K' and '$Base$'.

5.2 Determining DelayDAO - A Distributed Algorithm

In each node, the algorithm starts with an initial value of the two parameters, 'K' and '$Base$', and uses adaptive filtering to correctly estimate these parameters. The goal of this algorithm is for the nodes to obtain a value of DelayDAO Timer that minimizes buffer occupancy in nodes near the DAG root. No node, however, is aware of buffer size in

any other node but itself. Hence, this problem is different than a classic control theory problem, as the output and feedback are not directly related. Nodes far away from the DAG root can only speculate about a possible congestion near the DAG root by examining the round-trip time of a packet it transmits. In LLNs, not all the data packets need an acknowledgement back to the sender. Hence, this algorithm uses the round-trip time of the DAO packets to estimate probable congestion or large buffer size incident. If this round-trip time is too high, the constant K value is increased by a factor δ_K, and if it is lower than a certain time limit, the value of K is decreased by the same factor. If DAO-ACK-TIME, or the threshold to wait for a DAO-ACK packet, is exceeded, a severe congestion is surmised, and the value of $Base$ is increased by a factor δ_B. Clearly, the base of exponent $Base$ is used for coarse tuning of the value of DelayDAO Timer, where the constant K is used for fine tuning. The routine upon reception of each DAO-ACK is described in Algorithm 2.

Algorithm 2 Distributed Parameter Estimator

$RoundTripTime \leftarrow Current\_Time - DAOStartTime$;
if $RoundTripTime > T_U \times Rank$ **then**
  $K \leftarrow K \times (1 + \delta_K)$;
end if
if $RoundTripTime < T_L \times Rank$ **then**
  $K \leftarrow K \times (1 - \delta_K)$;
end if
if $RoundTripTime >$ DAO-ACK-TIME **then**
  $Base \leftarrow Base \times (1 + \delta_B)$;
end if
if $RoundTripTime < T_B \times Rank$ **then**
  $Base \leftarrow Base \times (1 - \delta_B)$;
end if

The constants T_U, T_L and T_B depend on the transmission time T_{Tx} of the DAO packets. If the DAO packets have a length of L_{DAO}, and the data rate of the radio is given by B_R, we chose these constants as :

$T_{Tx} = \frac{L_{DAO}}{B_R}$, $T_U = 8 * T_{Tx}$, $T_L = 4 * T_{Tx}$, $T_B = 2 * T_{Tx}$.

5.3 Determining DelayDAO - Centralized Approach

The DAG roots or border routers in LLNs posses much more computational power than any other node in the network, and also more memory space to buffer packets or store routing tables. Since all nodes send their DAO message to the DAG root, the root has an overall view of the whole network. Hence it would be advantageous to outsource the computations related to the network to the DAG root. In this section, we will present an algorithm that computes the *Base* and K parameters upon receiving DAOs from the network. After computation, these values are distributed in the network during the next global repair or increased DTSN via a new DIO packet. The values of *Base* and K can be added as objects in the DIO packet via a TLV base object.

We define the node rank set as L, which contains for each rank $R, (1 \leq R \leq H + 1)$ the nodes that are at rank R. Also, the DAG root maintains a parent list P to contain the parent for each node n in N. We define the function Find_Rank to intake a node ID, and returns the rank of the node as illustrated in Algorithm 7.

Algorithm 3 Rank Finder Algorithm

```
int Find_Rank(node n)
{
    int Rank = 0;
    P := Parent(node n);
if P == DAG_Root then
       return(1);
end if
    Rank := Rank + Find_Rank(node P);
    return(Rank);
};
```

Upon receiving each DAO, the data structures are updated as in Algorithm 4.

Algorithm 4 Construction of Node Rank Set

$n \leftarrow$ `Source of DAO message;`
$R_n \leftarrow$ `Find_Rank(n);`
$L[R_n] \leftarrow L[R_n] \setminus \{n\}$;
`Update Route_Table with new parent for n;`
$\hat{R_n} \leftarrow$ `Find_Rank(n);`
$L[\hat{R_n}] \leftarrow L[\hat{R_n}] \cup \{n\}$;

Before each global repair or DTSN increase, the DAG root estimates the parameters to determine the value of DelayDAO Timer for all nodes in the network as shown in Algorithm 5.

Algorithm 5 Centralized Parameter Calculator

$W \leftarrow \max(\{L[i]\})$, $1 \leq i \leq H$;
$R_W \leftarrow R$ s.t. $\{L[R] = \max(\{L[i]\})$, $1 \leq i \leq H+1;\}$
$Base \leftarrow \exp(\frac{\ln W}{R_W})$;
$K \leftarrow \max(2 * T_{Tx} \times \frac{i * \ln i}{Base^i})$, $1 \leq i \leq H+1$;

Of course, the centralized and distributed algorithms can be combined to achieve better convergence. The DAG root via the centralized algorithm can provide the initial parameter values for $Base$ and K, which can be further tuned by the nodes via Algorithm 2.

5.4 Evaluation of the Algorithms

5.4.1 Simulation setup and metrics

To study the behavior of RPL in different networks in (65), (17)) and ((18), the authors designed a detailed RPL simulator. The simulator was developed using OMNET++ (36) discrete-event simulator engine. Since urban LLNs posses highly time dependent attributes, a fixed probabilistic packet delivery or link condition does not represent typical challenges in an urban network. Hence, a database to model how link quality varies in

practically deployed LLNs was created by gathering real link layer data from outdoor and smart meter deployments. A topology of 2442 nodes deployed in a city was considered in this study. The deployment is replicated in the simulation by replicating the network topology and simulating identical temporal variation of the links. Each link in the topology 'picks up' the corresponding link quality model between the same neighbor pairs in the original deployment. Therefore, the link's PDR in simulation varies according to the gathered traces of the same link with respect to time, creating a "pseudo-replica" of the real network.

To analyze topology dependency of results, we have also considered random topologies of 200, 500 and 1000 nodes, where the nodes are distributed in a grid fashion. In all these networks, each link quality is time-dependent and modeled after temporal variation of a link in the created database, and the link's PDR varies with time in the same manner the link quality varies with time in a real deployed network. All these networks employ a 802.15.4 MAC/PHY model (66), and a CC2420 radio model for TelosB motes. To compute maximum required buffer space or memory, no buffer drop is simulated. Each simulation is run for a day (simulation time). The traffic pattern, as provided by smart grid AMI meter vendors, is described below.

- On-demand operation traffic: The collection point requests reading of index from each meter in a round robin fashion, which is reported back. Time period between two readings from same node = $F1$ which is normally 2 hours, unless otherwise noted.

- Polling traffic: The collection point retrieves information (statistic report, status, load curve) from each meter once a day in same fashion as above.

- Traffic due to multicast parameter modification (e.g. new tariff): 50 bytes sent by the LBR to a set of nodes, multicast, with frequency once a day.

- Traffic due to alarms: From a meter (node) to the LBR and unsolicited, unidirectional and random. 20 bytes of data is sent by each meter to the LBR, once a

day.

We mainly concentrate on three metrics which are closely related to performance of RPL when congestion or memory requirement is concerned.

- DAO reach time: Denotes the time required (in seconds) for the DAO of a node to reach the DAG root after the global repair or new DTSN is issued.
- Packet buffer size: Denotes the number of packets in the node, waiting to be transmitted.
- RAM consumption: Denotes flash memory consumption in bytes to store different state variables, data structures, parent tables, buffered packets, etc. but not RPL's code itself, as it may differ among implementations. However, the results do not consider the memory consumption of the DAG root since it is not a constrained device.

We also consider the round trip time of a DAO packet, which is calculated by the time difference between issuing a DAO packet, and receiving the corresponding acknowledgement.

5.4.2 Simulation results

Figures 5.4 and 5.5 show how the CDFs of DAO reach time vary with time for the distributed algorithm and the centralize algorithm, respectively, run in the 2442 node network. These two plots demonstrate that the proposed approaches have the ability to decrease the DAO reach time, helping the protocol perform better as the time progresses. Figure 5.6 shows the CDFs of DAO reach time for the proposed algorithms and the default mechanism described in (15). The default mechanism assumes a DelayDAO Timer value of 6 seconds. The default value of 1 second for DelayDAO Timer proved to be too low for the large network, and incurred congestion that inhibits normal operation of the network, hence not used for default mechanism simulation. The centralized approach delivers DAOs to the DAG root faster than the default mechanism in 95% of the cases, where

the distributed approach delivers 80% of the DAO packets before the default mechanism. Clearly, the proposed mechanisms outperform the default one in most cases, besides also winning in memory consumption, as described next.

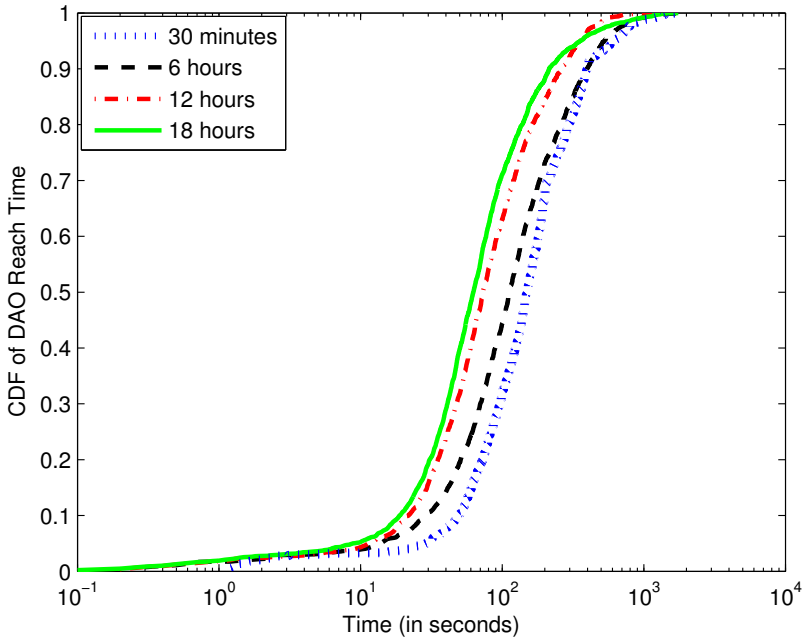

Figure 5.4: DAO reach time for distributed algorithm.

To study memory consumption, we consider different network sizes of 200, 500, 1000 and 2442 nodes. In Figure 5.7, we show how the maximum buffer size in the nodes varies as the network size grows larger. In Figure 5.8, the maximum amount of memory consumed is plotted for the three mechanisms, showing the savings brought on by the centralized and further, the distributed approach. The savings are more significant as the network scale grows; for the 2442-node network, the default mechanism consumes around 800 KB of memory, whereas only 90 and 30 KB of memory are consumed by the centralized and distributed approach, respectively. In Figure 5.9, we also observe that the proposed mechanisms decrease the average control overhead by a small fraction. The small amount of control overhead savings is a direct result of the reduced number of DAO packets being

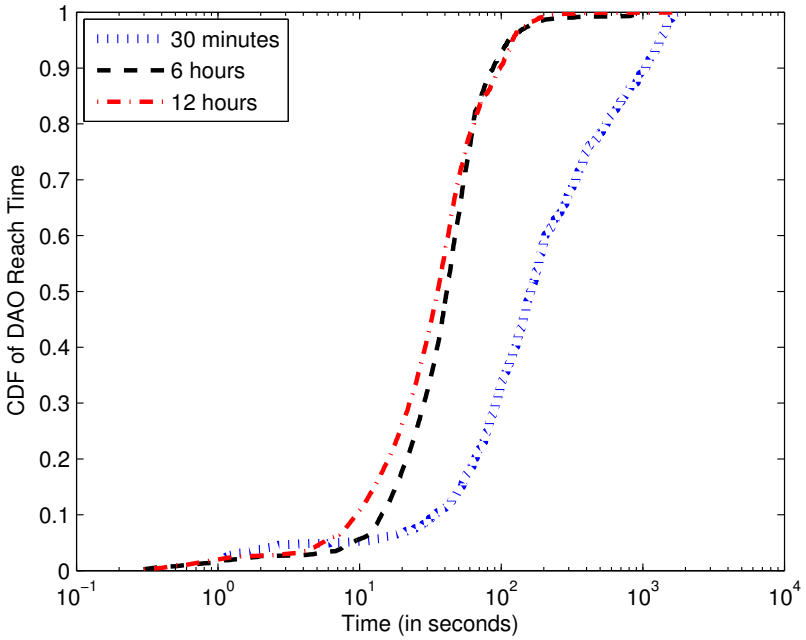

Figure 5.5: DAO reach time for centralized algorithm.

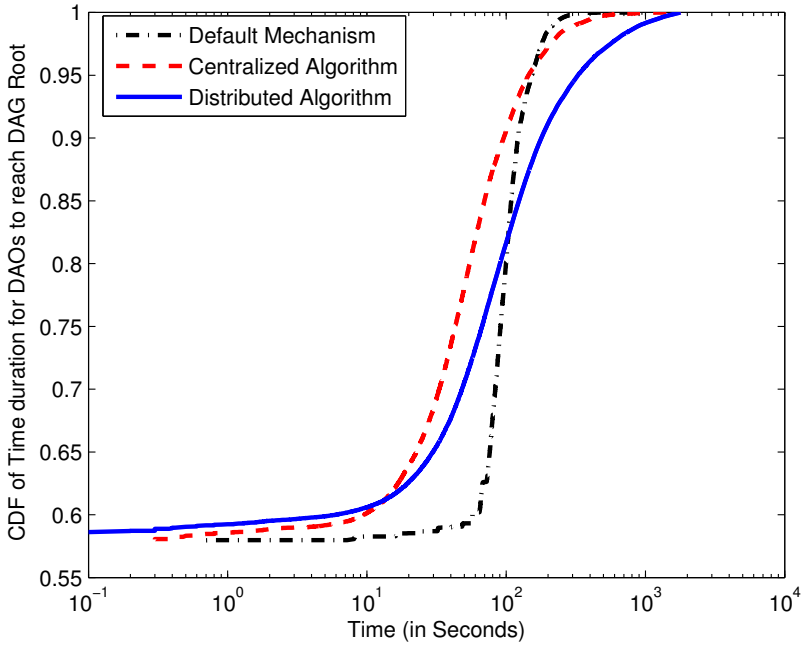

Figure 5.6: DAO reach time for all mechanisms.

re-issued due to congestion.

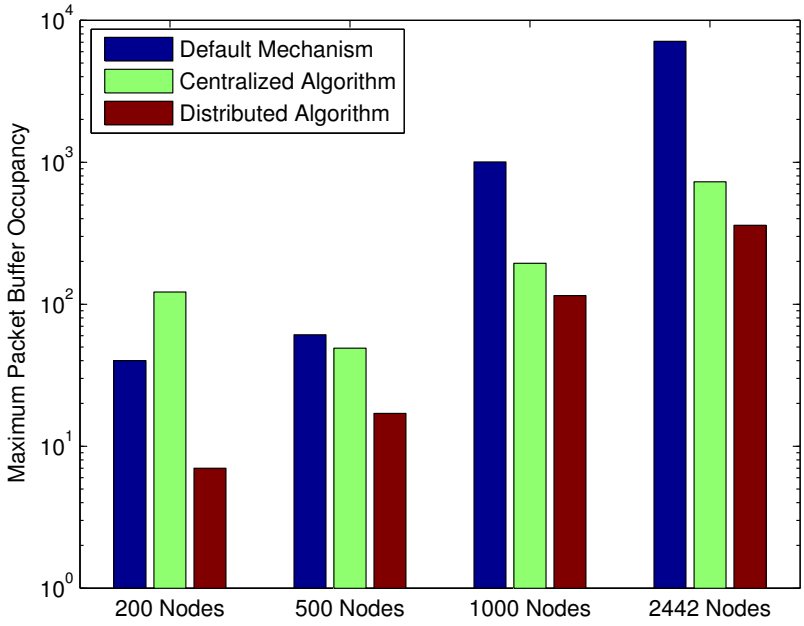

Figure 5.7: Maximum packet buffer size.

In Figure 5.10, we plot the CDF of the round-trip time of DAO packets, which is calculated as the time difference between issuing a DAO packet and receiving the corresponding ACK. Clearly, the two algorithms decrease the round-trip time by considerable amount. These results help establish that the proposed approaches do function as intended in a large network. We also considered the end-to-end delay experienced by data packets, since congestion highly effects this metric. In Figure 5.11, it can be observed that the delay experienced is much smaller for the proposed methods than for the default one.

Also, since we consider a distributed adaptive filtering based approach to obtain the parameters to determine DelayDAO Timer value, it is important to show that the approach leads to a stable output with respect to time. In Figure 5.12, we show how the 'K' parameter value varies with time for nodes of different ranks in the network. We plot the average 'K' parameter values for all nodes at rank 4, 8, 12 and 16, and observe that all

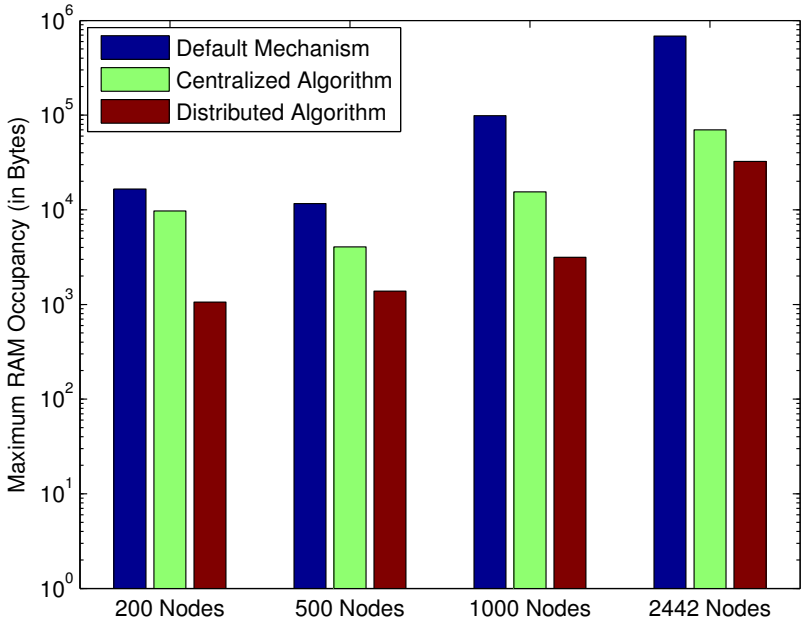

Figure 5.8: Maximum RAM consumption.

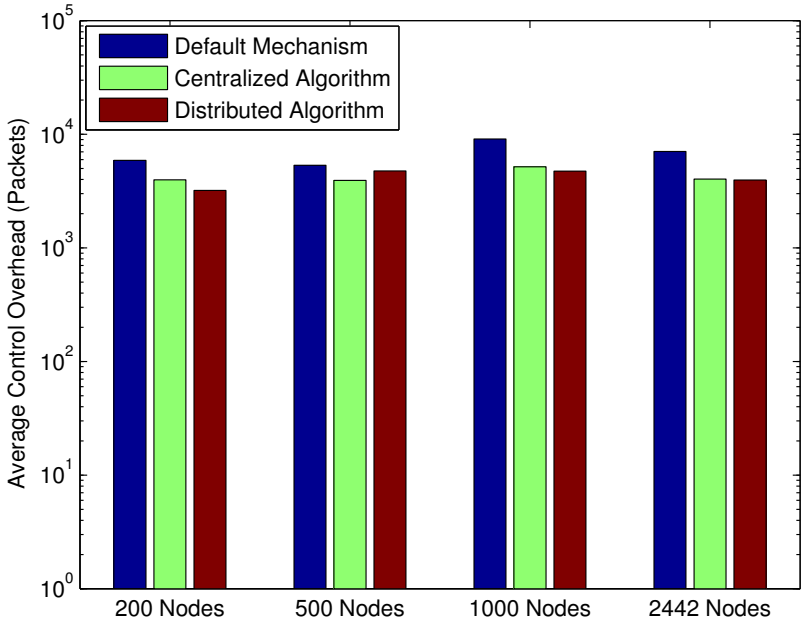

Figure 5.9: Average control packet overhead.

these values become stable after a few hours of operation.

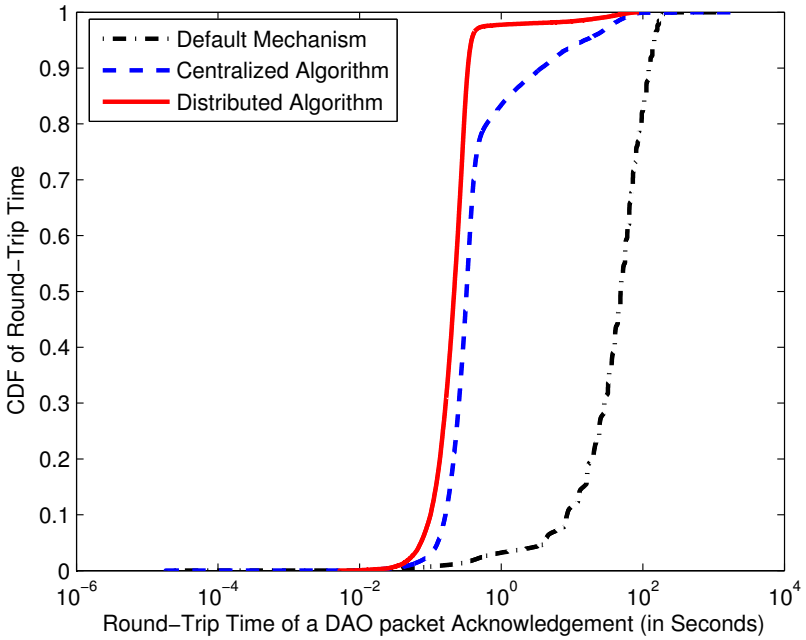

Figure 5.10: DAO round-trip time.

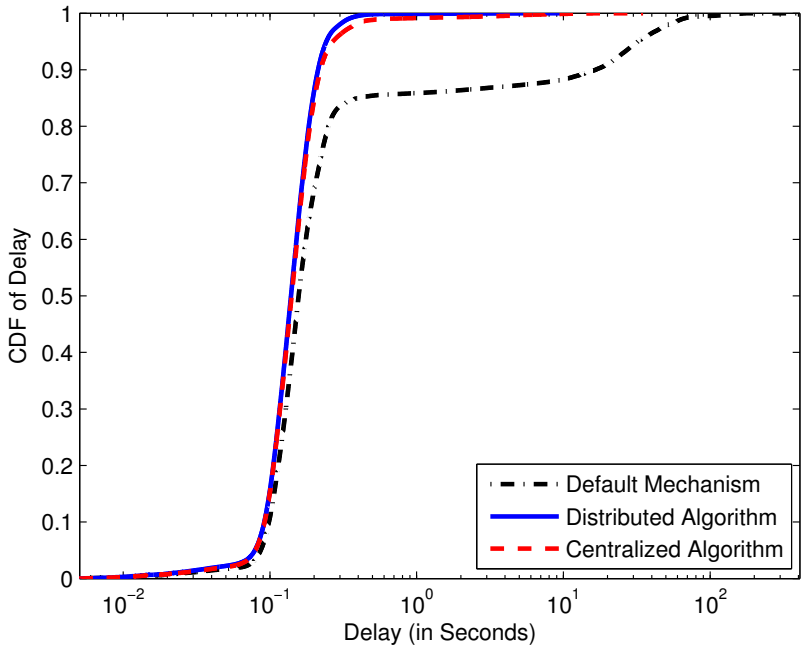

Figure 5.11: Data packet delivery latency.

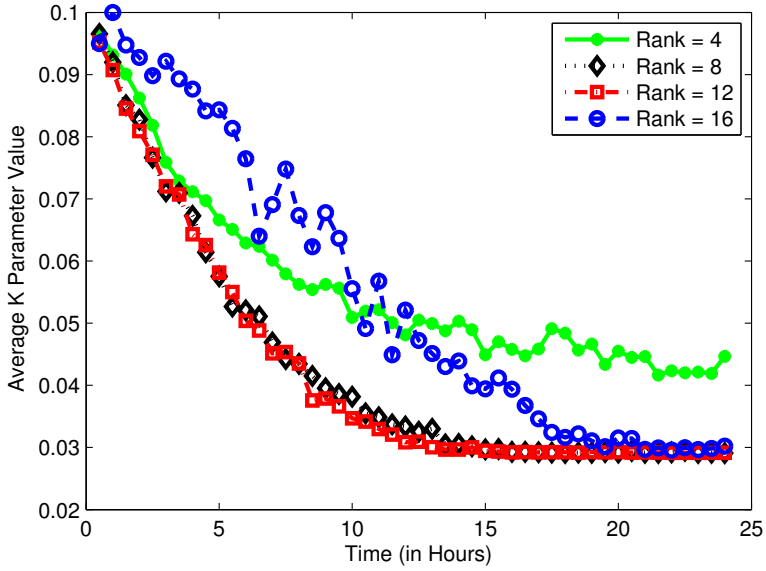

Figure 5.12: Stability of 'K' with time.

5.5 DelayDAO Controller - A Combined Algorithm Proposal to Improve RPL's Performance

As we have shown in Section 5.1 and 5.1.2, in large networks, when the LBR performs global repair to gather current routes, high congestion and excessive buffer occupancy is a by-product of the default RPL operation. This can be illustrated by a simple random topology of 1000 nodes with disc model for radio propagation. We assume the DAG constructed by RPL is the Breadth First Search (BFS) tree for the topology, and the value of DelayDAO Timer is T_{DD}. In general, the number of DAO packets received by a node at the ith T_{DD} interval will be the same as the number of nodes in the ith level of it's sub-DAG. In Figure 5.3, we plot the number of times a node's radio is busy transmitting or receiving DAO messages against rank and time interval. Note also that this analysis does not consider the DAO-ACK or acknowledgement packets that traverse down the DAG, while some DAOs are being forwarded up the DAG. Clearly, the results presented in this section are optimistic and provide at best a lower bound.

This high buffer requirement stems from the fact that each node needs to store the

DAO packets from its sub-DAG for possible aggregation of DAO routes. However, while implementing non-storing mode, route aggregation is not performed. Hence, a node should immediately forward a DAO from its sub-DAG to its parent, contrary to what is pointed out in RPL's RFC (15). Secondly, in a particular level in the DAG, the generation time for DAO packets should be distributed within a time duration to prevent propagation of a high number of DAO packets at the same time. The time duration should increase exponentially with rank as we have observed that the number of nodes tend to increase in this manner. This is illustrated in Algorithm 6, which determines the DelayDAO Timer duration. It assumes that each node generates a random number amongst an interval that increases exponentially with the rank of the node. The base of the exponent, or the parameter '$Base$,' is used for coarse tuning of the timer value, whereas the linear parameter 'K' is used for fine tuning. In the next sections, we will present mechanisms that will be used to estimate values of 'K' and '$Base$'. Each node initialize the value of these parameters from the value provided by the LBR in DIO packets, and updates them based on the algorithms described in this section.

Algorithm 6 Generation of DelayDAO timer duration

```
seed = <EUI_64>/MAC address or node ID
T_DD ← random(K * Base^(Rank-1), K * Base^Rank);
DAOStartTime := CURRENT_TIME;
Arm DelayDAO Timer with value = T_DD
Issue a DAO when DelayDAO Timer fires.
```

In Section 5.2 and 5.3, we proposed two methods to control the value of this timer. We observed that while a distributed algorithm provides less RAM consumption and lower DAO round trip time, a centralized algorithm helps the LBR gather the topology information faster. In this section, we propose a joint mechanism, so that we can retain the benefit from both approaches in terms of memory consumption and in faster building the whole DAG. We propose that border routers or collection points run a centralized

algorithm to estimate the parameters 'K' and '$Base$' from which DelayDAO timer should be computed at each node. The LBR stays updated on the current network topology by receiving DAO packets from nodes in the network. The parameters are broadcast before every global repair of the network, with the help of DIO messages with a new DAG sequence number from the LBR. Each node, on receiving the estimated parameters, and based on the round trip time of DAO packets and DAO acknowledgements, makes fine adjustment to these parameters, to further reduce the DAO congestion, thus limiting the buffer requirement.

5.5.1 Routine followed at LBR or collection point

Since all nodes send their DAO message to the DAG root, the root has an overall view of the whole network. Hence it would be advantageous to outsource the computations related to the network to the DAG root. In this section, we will present an algorithm that computes $Base$ and K upon receiving DAOs from the network. After computation, these values are distributed in the network during the next global repair or increased DTSN via a new DIO packet. The values of $Base$ and K can be added as objects in the DIO packet.

We once again define the node rank set L, which contains for each rank $R, (1 \leq R \leq H+1)$, the nodes that are at rank R. Also, the DAG root maintains a parent list P that contains the parent for each node n in N. We define the function Find_Rank to intake a node ID, and returns the rank of the node, as described in Algorithm 7. As illustrated in the flowchart in Figure 5.13, upon receiving each DAO, the data structures are updated as in Algorithm 8. Before each global repair or DTSN increase, the DAG root estimates the parameters to determine the value of DelayDAO Timer for all nodes in the network, as shown in Algorithm 9.

5.5.2 Routine followed at nodes other than LBR

Each node, on their first global repair, sets the parameters with the values of the received DIO packets. As shown in the flowchart in Figure 5.14, nodes use adaptive filtering to

Algorithm 7 Rank Finder

```
int Find_Rank(node n)
{    int Rank = 0;
   P := Parent(node n);
if P == DAG_Root then
      return(1);
end if
   Rank := Rank + Find_Rank(node P);
   return(Rank); };
```

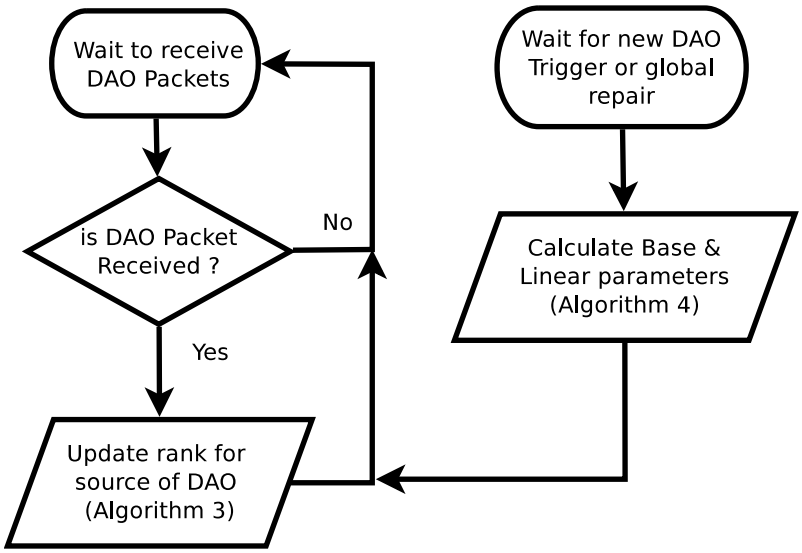

Figure 5.13: Routine at LBR - centralized parameter estimation.

Algorithm 8 Construction of Node Rank Set

```
n ← Source of DAO message;
R_n ← Find_Rank(n);
L[R_n] ← L[R_n] \ {n};
Update Route_Table with new parent for n;
R̂_n ← Find_Rank(n);
L[R̂_n] ← L[R̂_n] ∪ {n};
```

correctly estimate these parameters as they continue to receive acknowledgements from the LBR. On reception of DAO-ACKs, nodes use the round trip time to decrease congestion,

Algorithm 9 Centralized Parameter Calculator

$W \leftarrow \max(\{L[i]\}), 1 \leq i \leq H;$
$R_W \leftarrow R$ s.t. $\{L[R] = \max(\{L[i]\}), 1 \leq i \leq H+1;\}$
$Base \leftarrow \exp(\frac{\ln W}{R_W});$
$K \leftarrow \max(2 * T_{Tx} \times \frac{i * \ln i}{Base^i}), 1 \leq i \leq H+1;$

as shown in Algorithm 10. When global repair is performed, nodes do not give away their learned parameters, but adjust according to the received parameters in the DIO, as shown in Algorithm 11.

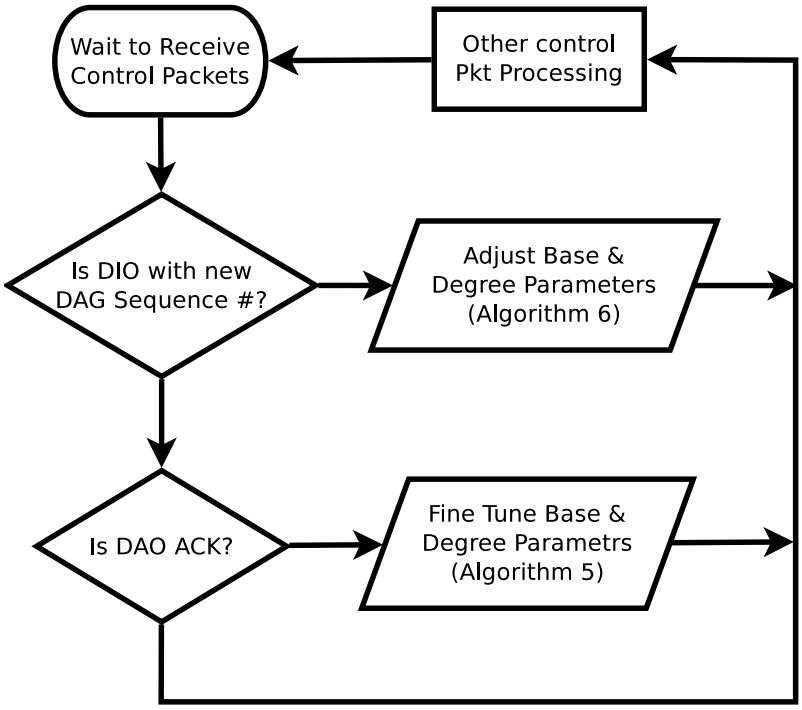

Figure 5.14: Routine followed at nodes - distributed parameter tuning.

The constants T_U, T_L and T_B depend on the transmission time T_{Tx} of the DAO packets. If the DAO packets have a length of L_{DAO}, and the data rate of the radio is given by B_R, we chose these constants as:

$T_{Tx} = \frac{L_{DAO}}{B_R}$, $T_U = 8 * T_{Tx}$, $T_L = 4 * T_{Tx}$, $T_B = 2 * T_{Tx}$.

Algorithm 10 Distributed Parameter Estimator - On DAO-ACKs

$RoundTripTime \leftarrow Current\_Time - DAOStartTime$;
if $RoundTripTime > T_U \times Rank$ **then**
  $K \leftarrow K \times (1 + \delta_K)$;
end if
if $RoundTripTime < T_L \times Rank$ **then**
  $DAO\_ACK\_TIME \leftarrow \alpha_D \times DAO\_ACK\_TIME + (1 - \alpha_D) \times RoundTripTime$
  $K \leftarrow K \times (1 - \delta_K)$;
end if
if $RoundTripTime > DAO\_ACK\_TIME$ **then**
  $Base \leftarrow Base \times (1 + \delta_B)$;
end if
if $RoundTripTime < T_B \times Rank$ **then**
  $DAO\_ACK\_TIME \leftarrow \alpha_D \times DAO\_ACK\_TIME + (1 - \alpha_D) \times RoundTripTime$
  $Base \leftarrow Base \times (1 - \delta_B)$;
end if

Algorithm 11 Distributed Parameter Estimator - On DIOs

$K_{New} \leftarrow K$ value received in DIO TLV.
$K \leftarrow \alpha_K.K + (1 - \alpha_K) \times K_{New}$
$Base \leftarrow \alpha_B.Base + (1 - \alpha_B) \times Base_{New}$

5.5.3 Evaluation of proposed approach

We simulate both the default specification of RPL, with a constant timer value, and our proposed method. We mainly concentrate on aspects of congestion, such as RAM occupancy, data packet delivery delay, etc. Figure 5.15 shows the CDF of data delivery delay and we see that the proposed mechanism drastically improves the latency with which the data packets are delivered. This is an obvious outcome of mitigated congestion, as congestion causes both data and control packet to be buffered.

In Figure 5.16, we plot the maximum number of packets buffered with network size and see that the propose mechanism achieves a significant reduction, with a gain in buffer size as high as 15× for the largest network. The gain increases as network size increases, a trend which is also observed for RAM consumption, as shown in Figure 5.17. Less use of buffered packet leads to less use of precious memory, and the proposed mechanism consumes only 40 KBytes of memory as opposed to the default mechanism, which consumes $\sim$ 700 KB

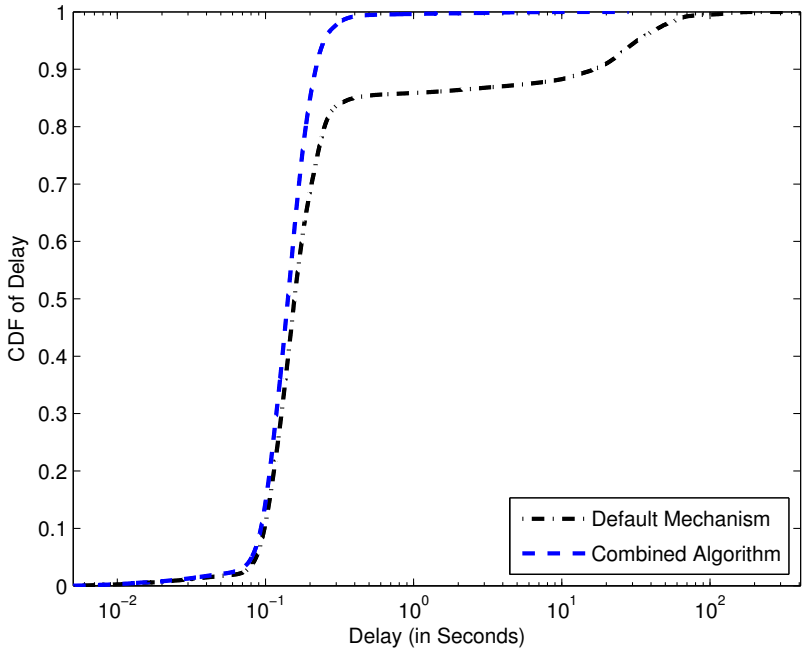

Figure 5.15: CDF of data delivery delay.

of RAM.

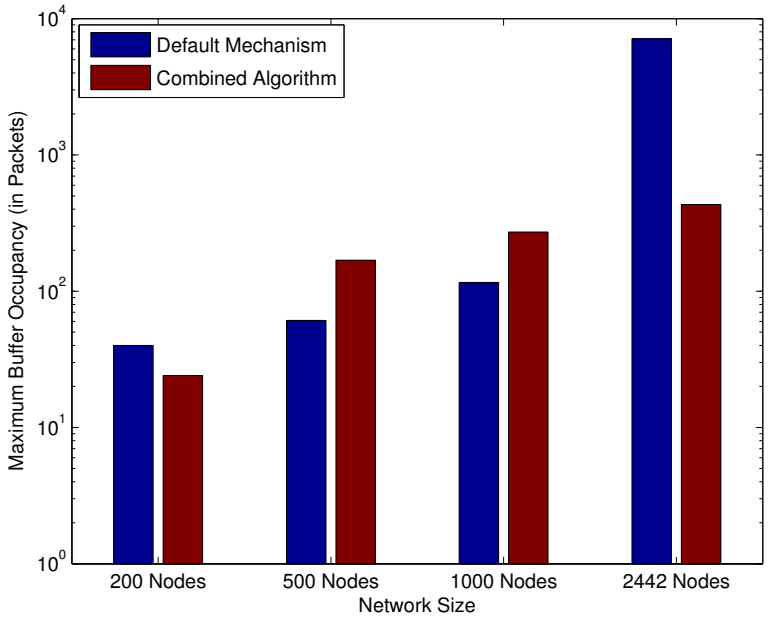

Figure 5.16: Maximum buffer occupancy against network size.

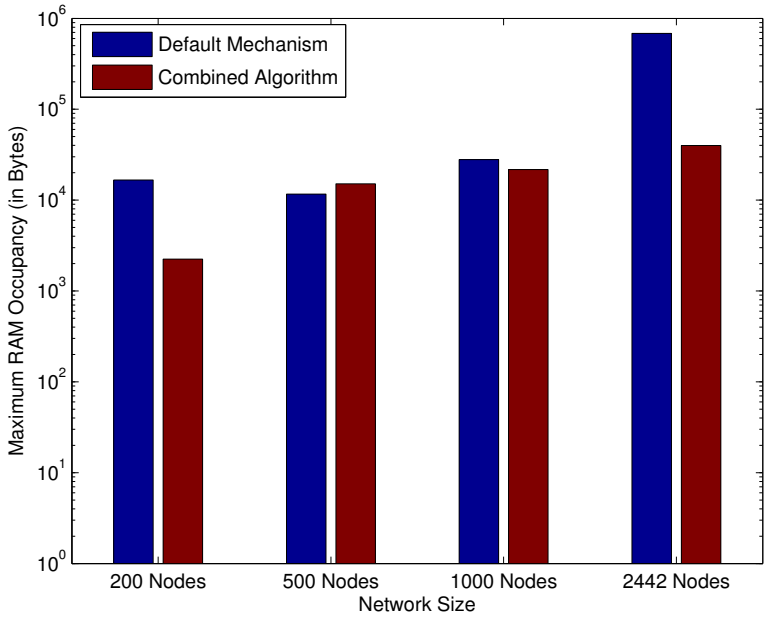

Figure 5.17: Maximum RAM occupancy against network size.

In Figure 5.18, we plot the CDF of the round trip time required by the DAO packets. Achieving a low round trip time, on one hand, shows signs of congestion free network. At the same time, a low round trip time also prevents unnecessary duplicate DAOs to be resent to the DAG root, thus saving on important control plane bandwidth. We again observe how the propose mechanism improves the round trip time when compared to the default RPL mechanism.

As it can easily be perceived, the Destination Advertisements (DAO) packets are scheduled in such a way that a congestion is avoided. By avoiding this congestion, we achieve less delay in order for data packets to reach the root, and less round trip time of DAO packets, which helps with the unnecessary issuance of destination advertisements. However, the price we pay by deploying this mechanism is high discovery time for some nodes, mostly the ones that are farthest from the root node. In Figure 5.19, we show the CDF of time taken by each DAO packet to reach the LBR.

As we observe, for around 85% nodes, DAO packets in the proposed method reach the LBR earlier than the proposed default method. However, the LBR learns about 15% of

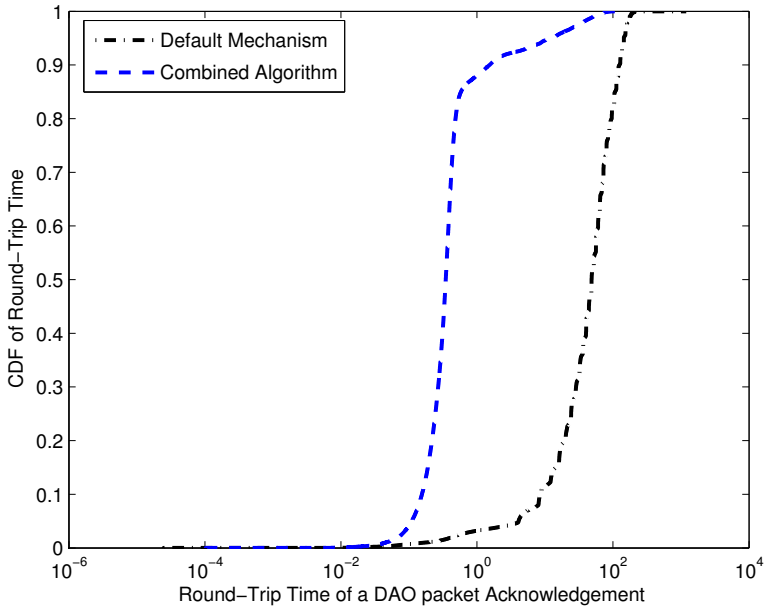

Figure 5.18: CDF of DAO round trip time.

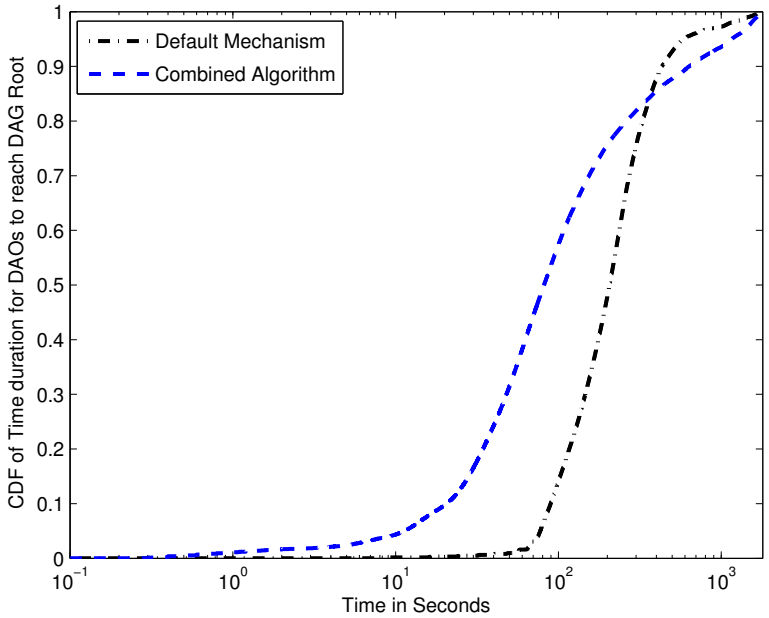

Figure 5.19: CDF of time taken by DAOs to reach the DAG root.

nodes in the network at a later point of time than with the default mechanism. These 15% of nodes are the ones that have large ranks. In other words, the proposed method

helps the LBR learn about nodes near itself faster, but for some nodes with large ranks it consumes more time, which is the price we pay for congestion avoidance. However, as DAO includes information on a node's parents only when it is generated, the information learned is not outdated.

5.6 Summary

In this Chapter, two mechanisms are proposed to resolve the congestion and high memory and buffer requirements in the standard RPL non-storing mode implementation, which makes use of a fixed, universal DelayDAO Timer. The proposed mechanisms are even more useful in topologies where the data collection point is at the center of the deployment, and where the number of nodes exponentially grows with distance from the root. It can be observed, that the centralized approach provides less time for the DAG root to gather routing information for the whole network than the distributed algorithm. Both proposed mechanisms provide fairly low RAM usage, especially in large topologies. These algorithms also incur less delay for data packet delivery by controlling the control plane congestion. From a high level overview, the proposed approaches in this study trades off the delay that Destination Advertisements (DAO) from higher rank nodes suffer with the buffer size in nodes with lower rank (near the DAG root) and end-to-end latency for data packets.

Different approaches can be considered which may use the information from the DAO-ACK packets to indirectly detect large buffer occupation and congestion near the DAG root or low rank nodes in RPL non-storing mode. This information may be used to delay transmission of DAO packets to reduce buffer requirement in LLNs. Different congestion control techniques for transport protocols exist in the literature, which might shade new light or hint at new directions to this problem. Nevertheless, it is of most importance to devise a suitable technique to schedule DAO packets in RPL for large networks.

6

RPL: DAO Propagation in Storing Mode

In this chapter, we will introduce a new packet format for DAO aggregation that does not violate RPL standard message format and adds to the original DAO Base Object defined in the RFC. Also, the message format should be able to handle compressed address format in case header compression is applied to decrease overall packet size. We will introduce an approach for DAO packet generation, storage and forwarding. At the same time, we also propose an aggregation method for DAO packets in storing mode.

6.1 DAO Aggregation and Delay Method in Storing Method

In RPL storing mode, DAO packets are not unicast to the DAG root, but are rather advertised to the node's parents. Thus, these DAO packets reach the DAG root in a hop-by-hop fashion, being acknowledged at every single hop. Nodes in storing mode may or may not aggregate DAOs from their subtree. If aggregation of DAO packets is not performed, we propose the storing mode to employ similar technique as non-storing mode for DAO generation. In absence of prefix aggregation, a node in storing mode has to forward the same number of DAO packets as in non-storing mode for the same topology. Hence, the same centralized approach as in non-storing mode is expected to provide similar

benefits in RPL storing mode. The proposed distributed approach, however, cannot be applied to storing mode, as in this mode the DAOs are acknowledged at every single hop, and no end-to-end acknowledgement is available.

As described in (15), a node should delay sending the DAO message in order to aggregate the DAO information from other nodes for which it is a DAO parent. RPL non-storing mode requires the parent node to send their DAO packets before the children, so the value of DelayDAO Timer should increase with the node's rank in this mode of operation. However, aggregation of destination addresses will surely get hurt by such a mechanism. The default mechanism of a constant universal timer will lead the nodes with higher rank to release their DAO later, thus harming the aggregation process. For aggregation to be successful, nodes of farthest distance (in rank) from the DAG root should release their DAO first. In an ideal situation, a node should wait till it receive DAOs from all nodes in its subgraph, aggregate them as much as possible, and finally send it to its parent. Intuitively, DAO aggregation needs a mechanism where the value of the DelayDAO Timer decreases with the rank. However, the RPL standard ((15)) does not specify rules on aggregated DAO packet generation and processing.

6.1.1 Aggregated DAO packet format

In (15), Type = 0x05 is reserved for a target option that adds to a DAO base object. This target option carries the destination address in a DAO packet. Type values within the range $0x01 - 0x09$ are reserved. We define a new DAO option, called *RPL Aggregated Target Option*, which is defined by the type value 0x10. The Aggregated Target Option possesses the same format as the target option originally defined. However, 4 bits amongst the 8 unused flag bits are used as a new field, denoted as *CmprT*. Our proposed RPL Aggregated Target Option is illustrated in Figure 6.1.

In our proposed RPL Aggregated Target Option, the CmprT field defines the level of compression applied to each target address, so that each target has an address length of $(16 - CmprT)$. As the possible values of CmprT range between 0 and 15, the length of

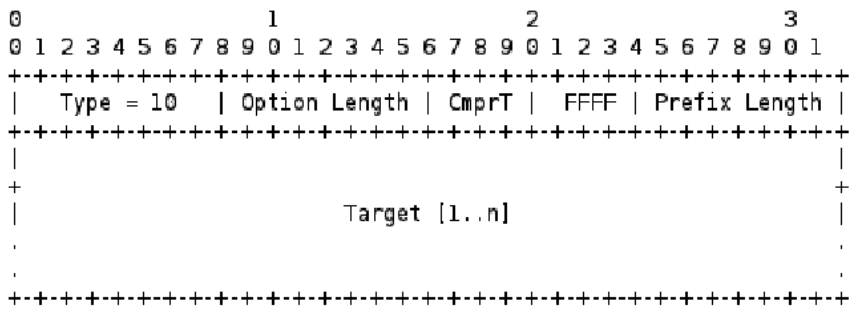

Figure 6.1: Proposed DAO target option for aggregation.

each address field in the Aggregated Target Option may vary between 1 and 16 octets. The number of targets present in the aggregated DAO can easily be calculated as:

$$n \leftarrow \frac{(Prefix\ Length - 4)}{(16 - CmprT)}$$

We define $MaxTarget$ as the maximum number of target destination addresses that may be accommodated in a single aggregated DAO. Clearly, if the Maximum Transmission Unit (MTU) of a network is known, and $DAOOverhead$ defines the overhead associated with a DAO packet without the 'Target' field, $MaxTarget$ can be calculated as:

$$MaxTarget \leftarrow \frac{(MTU - DAOOverhead)}{(16 - CmprT)}$$

We will explain this with a couple of examples. For 802.15.4 links, the MAC layer presents a maximum of 102 bytes of payload to the network layer. If we consider DAO packets with the proposed aggregation option, it can be shown that the total overhead incurred is 64 bytes without the 'Target' field. If the targets include a full IPv6 address of 16 bytes, the scope of aggregation is clearly, very limited. However, we can use header compression as in (63), or use a compressed address field of 2 bytes instead of the full 16 bytes. Using a 2 bytes address field and unchanged header format, a maximum of $(102 - 64)/2$ or 19 destinations can be aggregated in a single DAO packet. Also, 802.15.4g (67) has recently been amended to support Smart Utility Networks (SUN). 802.15.4g can handle a maximum frame-size of 2047 bytes, well over IPv6 defined maximum frame-size

of 1280 bytes. If we consider a full IPv6 packet with an MTU of 1280 bytes, even with uncompressed header and uncompressed address field of 16 bytes length, a single DAO may support up to 76 destinations together. However, link PDR may decrease with increased packet size, forcing an upper bound on $MaxTarget$ for certain environments.

6.1.2 DAO aggregation algorithm

The following notation and parameters are used in the algorithm:

- SubTree: The number of valid routes in a node's routing table;
- R: Current rank of the node;
- ReleaseQ: A queue of unreleased destinations present at each node, with a timestamp of when the destination is added;
- $T_L = 4 * T_{Tx}$, as defined in Section 5.2.

Upon receipt of a new DODAG sequence number or a new DTSN, Algorithm 12 is run to determine the generation of a DAO packet by the node. The DelayDAO timer is inversely proportional to rank, as discussed earlier in this section.

Algorithm 12 Generation of DelayDAO Timer Value

```
seed = <EUI_64>/MAC address or node ID
```
$T_{DD} \leftarrow \text{random}(\frac{T_L}{R-1} \times SubTree\ ,\ \frac{T_L}{R} \times SubTree)$;
```
Arm DelayDAO Timer with value = T_DD.
Enque (self, CURRENT_TIME) to ReleaseQ.
when DelayDAO Timer fires.
```

Clearly, even when a DAO is generated, it is not forwarded immediately. A node always keeps an eye on its ReleaseQ, to check if a destination is waiting too long to be transmitted. The routine is described in Algorithm 13.

Algorithm 13 ReleaseQ Checker

$T \leftarrow$ Timestamp of first entry in ReleaseQ.
if $T - CURRENT\_TIME \geq \frac{T_L}{R} \times SubTree$ **then**
  if Size of ReleaseQ $<$ MaxTarget **then**
    Aggregate all destinations in a DAO.
  else
    Aggregate $MaxTarget$ destinations in a new DAO.
  end if
end if

Upon receipt of a DAO packet from another node, Algorithm 14 defines the rules for DAO aggregation and forwarding.

Algorithm 14 Destination Aggregation and DAO Emission

if $n < MaxTarget - 1$ **then**
  Extreact each destination, D.
  update route table with D.
  Enqueue each (D, $CURRENT\_TIME$) into ReleaseQ.
  if Size of ReleaseQ $\geq$ MaxTarget **then**
    Aggregate $MaxTarget$ destinations in a new DAO.
  end if
else
  Update route_table with all destinations.
  Forward DAO Packet as is, without inserting it into ReleaseQ.
end if

6.2 RPL Storing Mode Evaluation

We used a similar setup and terminology as in Section 5.4. However, along with 802.15.4 links, we implemented 802.15.4g as well, to observe the effect of the maximum allowable aggregation. Simulations with 802.15.4 links consider a compressed address field of 2 bytes, whereas no compression is employed for 802.15.4g links. To be noted, even if 802.15.4g allows a maximum frame size of 2047 octets, a DAO packet in our simulation may have a maximum frame size of 1280 octets, as RPL is designed to run under IPv6.

In Figure 6.2, we plot the CDF of DAO reach time value for both, default mechanism and proposed modification for storing mode. This result shows how the proposed algorithm, with the help of aggregation, improves the DAO reach time helping them reach the DAG root faster than with the default mechanism. It should be noted that with increase in MTU, nodes wait longer to aggregate and transmit the DAO, and hence, the DAO reach time also increases.

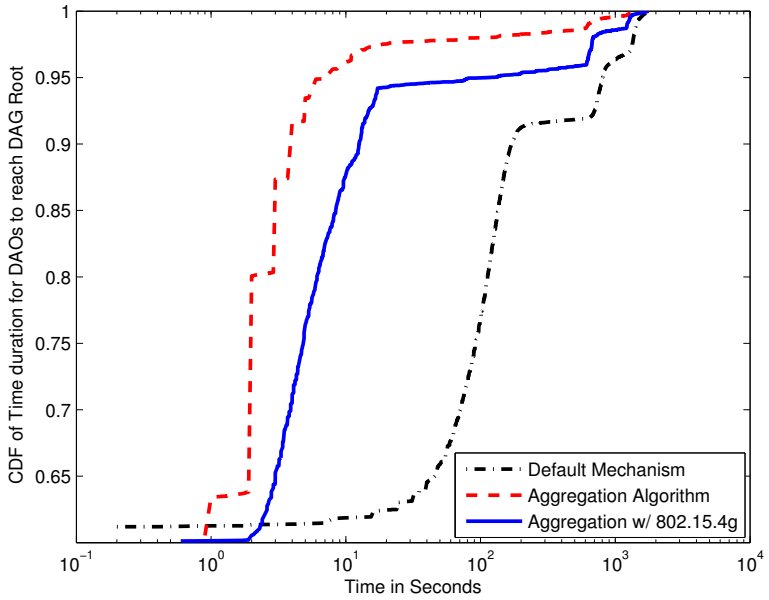

Figure 6.2: DAO reach time for both mechanisms.

We next consider 5 networks of size 86, 200, 500, 1000 and 2442 nodes to evaluate topology-dependency of the results. In Figure 6.3, we show how the proposed mechanism benefits RPL in terms of maximum memory consumption. As aggregation includes multiple addresses in a single DAO, the number of control packets is definitely smaller than with the default mechanism, but each packet has larger size. Hence, to be fair, in Figure 6.4 we plot the average control overhead (in bytes) for the two mechanisms for networks of different size.

We also considered the end-to-end delay experienced by data packets in the large

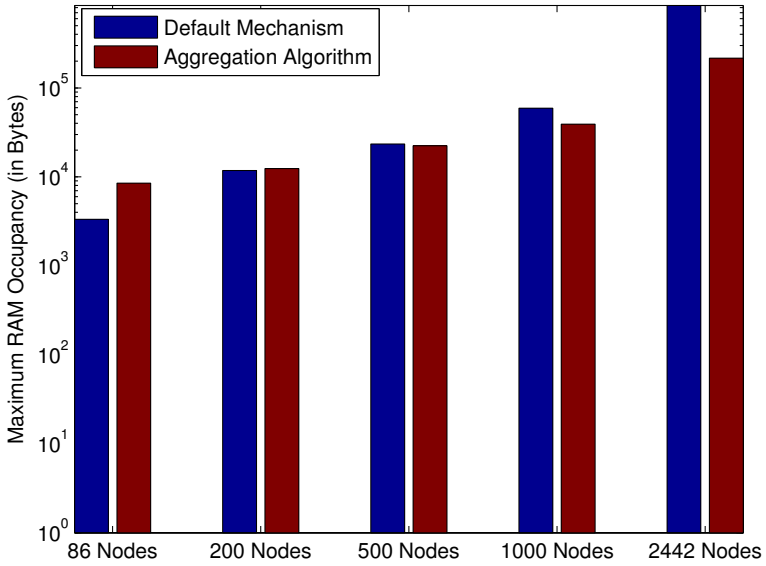

Figure 6.3: Maximum RAM consumption.

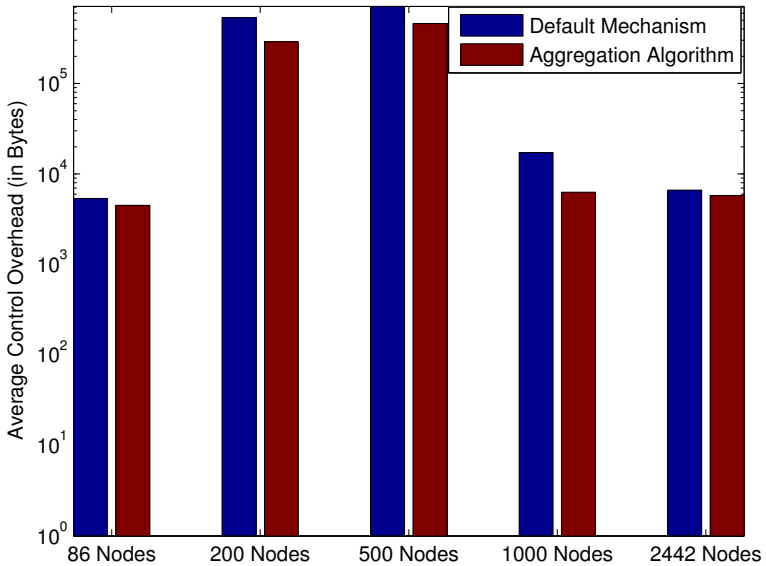

Figure 6.4: Average control packet overhead.

2442-node network, as we did for the non-storing mode algorithms. In Figure 6.5, it can be observed that the delay experienced is much smaller for the proposed method with aggregation than for the default one. Note that the proposed aggregation method does not aggregate any data packet. Still, by reducing congestion and thus providing more resource

to the network, the proposed approach provide less delay for 65% of the packets. As observed, the maximum delay experienced by any data packet for the proposed approach is also much smaller. Clearly, the link MTU does not have a big effect on data delivery once the congestion is mitigated, as we can observe the two MAC/PHY models of 802.15.4 provide very close data delivery delay.

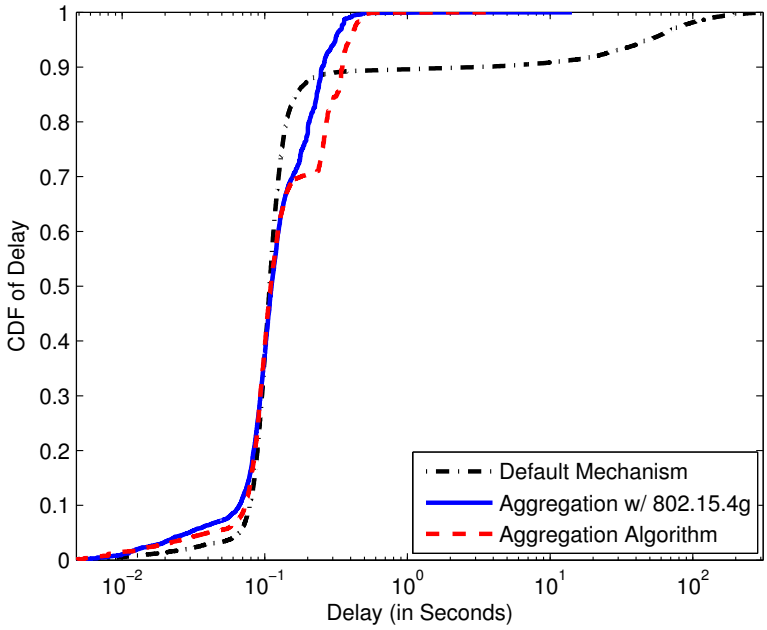

Figure 6.5: Data packet delivery latency.

6.3 Summary

In this Chapter, a DAO generation and aggregation mechanism is proposed for RPL storing mode, which is observed to provide the DAG root with faster congregation of addresses in the network, less control overhead and faster delivery of data packets, by lowering DAO congestion. Here it can be observed, that using higher link MTU (such as in 802.15.4g) does not necessarily provide high performance benefits. Nodes waiting to aggregate destination address information causes further delay and improves the overall

control overhead by a small fraction. However, the DAO generation algorithm for storing mode of RPL implementation is still able to provide a fraction of improvement in packet buffer consumption.

7

Evolution of RPL: Towards Load Balanced LLNs

The vision of the Internet of Things (IoT) brings together networks and thousands of battery operated wireless devices to share common resources. Due to their large scale, these networks employ hierarchical routing along with IPv6, and a border router to connect to the Internet. Distribution of data traffic is essential in order to avoid congestion and hot spots in the vicinity of the gateway or border routers, as well as to decrease/balance energy expense of the devices. Therefore, load balancing is crucial in such deployments, and accordingly there have been numerous attempts to achieve balance in traffic forwarding in order to decrease congestion and packet drops.

RPL, however, was not designed to achieve load balancing. In RPL, some nodes may be used more often than others for forwarding, increasing the load in some branches. Urban or forest deployments, however, typically have high node density, with many detours available. Similarly, in smart grid networks, many AMI meters in the same building and locality interconnect to form a network of nodes with high degree of connectivity. Hence, a node may chose amongst many alternative parents to send data to the base station, and vice-versa. By distributing the traffic load across available links, load balancing may achieve less hot spots and greater network lifetime by distributing energy expense.

The contributions of this chpater are as follows:

- Firstly, this work defines a quantitative metric to measure load imbalance for any instance of data aggregation to a single sink. The metric assumes periodic data generation by all the nodes in the network and can be generalized for uneven traffic generated by the nodes;
- Secondly, this work proposes techniques to minimize the imbalance metric over time, using only a very limited subset of neighbors or parents, consistent with the RPL protocol. The proposed approach should not consume any extra control overhead other than the control message exchange already needed for RPL to operate.
- Finally, to show in Chapter 8 how this can be achieved via proposed HIPC architecture.

7.1 Related Work

There have been numerous attempts to achieve balance in traffic forwarding in order to decrease congestion and packet drops. In wireless sensor networks (WSN) in particular, these approaches can be broadly classified into two categories. One on category, load balancing is achieved through clustering algorithms, varying the cluster size and/or cluster-head over time to balance energy expense of the nodes, such as in (68, 69, 70, 71, 72). There are also heuristics that try to alternate among possible next-hop nodes to balance energy expense (73, 74). The second category includes approaches where the whole topology of the network needs to be known in advance, such as in (75, 76, 77, 78). There is a large body of literature concentrating on energy efficient clustering techniques in general (79, 80, 81).

However, there exists a large gap between the load balancing techniques that currently exist for WSNs and a practical approach that can be deployed for large scale LLNs and the IoT. Traffic is sensing or event driven in WSNs, whereas for urban LLNs, such as smart grids, they are mostly periodic. Many routing algorithms exist for WSNs which

provide best routing path between two peers. In large scale LLNs, due to the incapability of nodes storing routing tables, P2P routing is possible through some capable nodes only, and generally by source routing. At the same time, clustering approaches need variable transmission power, where cluster heads transmit with higher energy. This leads to higher energy consumption for cluster heads, hence leading to quick depletion of energy and therefore the need for cluster head rotation. Also, for a base station to either know the full topology or neighbor set of its LLN deployment or all link details, every node needs to advertise all of its available link states. Given the temporal characteristics of LLNs, this approach leads to prohibitive overhead, and therefore is impractical. Normally, only a few neighbors may be used for data forwarding with confidence. Approaches that try to achieve load balancing in node via several optimization methods, linear programming etc., are not practically applicable in LLNs and most of the WSNs either. In LLNs or Internet of things, nodes have very low memory (A few KBs of flash and RAM) and computational capacity (e.g. 8 MHz processor in TI MSP430 processor used in TelosB motes). This sets up a platform for a load balancing technique which does not consume extra control overhead, does not require the sink or central node to know the full topology, and/or requires minimal processing for in-network nodes.

Moreover, existing networking load balancing literature almost always lacks any quantitative measure for balance or imbalance of traffic load in the network. Currently, given two or more aggregation tree instances for the same network, there is no way to quantitatively define which instance can achieve more balance in traffic load than others. Measure of load balance, or lack thereof, has mostly been determined by energy expense or network lifetime. Clearly, load balancing has been pursued without quantitatively being defined.

The contributions of this work are mainly two-fold.

- Firstly, this chapter defines a quantitative metric to measure load imbalance for any instance of data aggregation to a single sink. The metric assumes periodic data generation by all the nodes in the network and can be generalized for uneven traffic generated by the nodes;

- Secondly, this work proposes techniques to minimize the imbalance metric over time for an LLN deployment, using only a very limited subset of neighbors or parents, consistent with the RPL protocol. The proposed approach does not consume any extra control overhead other than the control message exchange already needed for RPL to operate. When used in conjunction with RPL, this technique can achieve load balance in a LLN deployment, aiding in network lifetime. Performance results are shown for real LLN links.

Section 7.2 introduces the new load imbalance metric, and shows how it really reflects the imbalance in different branches of data aggregation. In Section 7.4, we show how a greedy method to minimize the imbalance factor can be integrated with RPL for operations in LLNs. Section 7.5 shows the effectiveness and improvement of proposed method with basic RPL implementation and comparison with Brute Force optimum in smaller scale

7.2 A Metric to Define Load Imbalance

7.2.1 Example of balanced collection tree

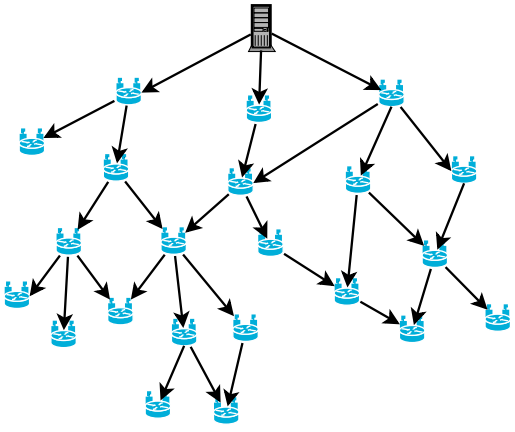

Figure 7.1: A DAG created by RPL.

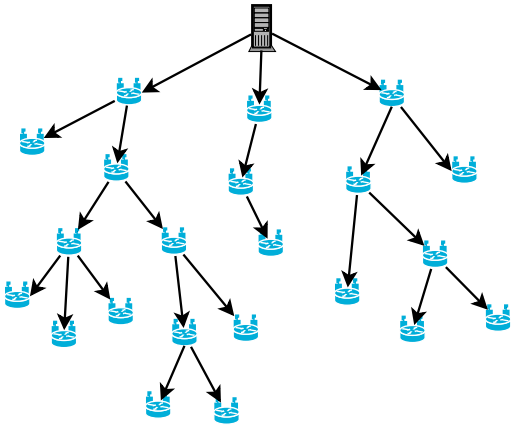

Figure 7.2: Example of unbalanced parent selection.

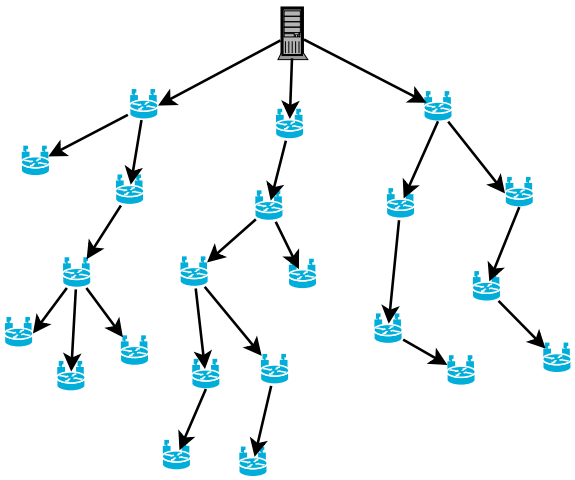

Figure 7.3: Example of balanced parent selection.

To approach the problem of load balancing in LLNs, one needs to create a data collection tree as balanced as possible. Clearly, not all nodes will carry same data traffic load in the network. However, the intuitive idea behind load balancing is that nodes at the same level in the DAG or collection tree, or same hop distance from the root node, should carry as equal volume of traffic as possible. As an example, consider a DAG constructed by RPL in Figure 7.1. RPL often lets nodes select a preferred parent that is advertised first, and hence results in a tree as depicted in Figure 7.2. If we consider the three nodes from left to right at rank 2, we observe that they provide route for 12, 3, and 7 destinations in the network, respectively. Clearly, the leftmost node in the three will deplete its energy much sooner than the middle one. On the other hand, the same DAG created by RPL may provide a preferred parent selection that yields the tree shown in Figure 7.3. Such an approach is most desirable for any data collection network. We propose to minimize the difference between the traffic load handled by the nodes of the same rank for every level in the DAG.

7.2.2 Imbalance factor

We define the estimator of balanced load, for a given level or rank in the collection tree, to be the average traffic each node in that rank forwards. To minimize load imbalance, one

therefore needs to minimize the variance of traffic load amongst nodes of a given rank. We assume N_R refers to the number of nodes in the network at rank R, S_i denotes the number of nodes accessible through node i from the root or collection point, and S_R denotes the average number of routes available for all the nodes at rank R. Basically, S_R is the number of nodes at ranks R or greater, divided by N_R. For the node i, the deviation from balanced load can be given as $|(S_i - S_R)|$. For the rank R, we thus have N_R samples of such load, and our objective is to minimize the variance among them. Hence, we define the mean square error of traffic load at any level of the DAG as an estimator of load imbalance at that given level. Thus, the load imbalance factor at rank R in the network is,

$$I_R = \frac{1}{|N_R|} \sum_{i \in N_R} (S_i - S_R)^2 \tag{7.1}$$

Since we are using the mean square error of load to estimate load imbalance, we see that the higher the number of nodes in a particular rank or level of the tree, the lesser will be the effect of imbalance, as traffic is distributed amongst more forwarders, following our intuition of load balancing. If the network has a height of H, the total imbalance of data traffic of the network can be defined as

$$I = \sum_{2 \leq R \leq H+1} I_R = \sum_{2 \leq R \leq H+1} \frac{1}{|N_R|} \sum_{i \in N_R} (S_i - S_R)^2 \tag{7.2}$$

Looking back at our previous example, the tree of Figure 7.2 produces an imbalance of $I_2 = 13.56$ at rank 2, and a total imbalance of $I = 28.74$, whereas the tree at Figure 7.3, produces $I_2 = 0.22$ and $I = 7.12$, respectively. Also, a complete balanced tree (binary, tertiary, ... etc.), would produce an imbalance of 0 at every level and for the overall tree. Clearly, a routing/data collection tree that minimizes the metric I would be the most balanced tree for the specific network. In this definition, we have assumed all destinations generate equal traffic, and hence the number of nodes accessible through a node is considered equivalent to the traffic load. However, the definition can be easily generalized to include different traffic rates, where the total generated traffic in the subtree

replaces subtree size in the two equations. The leaf nodes will have subtree size equal to the generated traffic by them instead of 1.

7.3 How Hard Is It to Minimize Imbalance?

Let us define a 'Parent Set' of a node, to be the set of forwarders, through which the node can access the root or collection point and vice-versa. This Parent Set may be size bound by the implementation (e.g. advertising up to 3 parent nodes to the root node) or may be bound by the highest node degree Δ. Let's assume, the upper bound on Parent Set is K. For a network of N nodes, we may have up to K^N combinations of how parents may be assigned and thus a naive polynomial time assignment is not possible.

Let there be an imbalance function $I(\mathbb{S})$, where $\mathbb{S}$ denotes the subtree set for all nodes. The input matrix, $\mathbb{P}$ is the $N \times N$ parent option set where each element p_{ij} of $\mathbb{P}$ is 1 if node i is a candidate parent for node j, or 0 otherwise. Also, Let X be the $N \times N$ output parent assignment matrix, so each element of X, X_{ij} will be equal to 1 if node i is the chosen parent for node j, or 0 otherwise. Hence, the problem to minimize load imbalance can be formulated as follows:

$$\begin{aligned} &\text{minimize } I(\mathbb{S}), \text{ where } \mathbb{S} = \{S_1, S_2, \cdots, S_N\}, \\ &S_i = \sum_{j \in N} S_j \cdot X_{ij} + X_{ii} \quad \forall i \in N, i \neq j \\ &\text{such that, } \sum_{i \in N} X_{ij} \cdot p_{ij} = 1 \quad \forall j \in N, i \neq j \\ &\text{and } X_{ii} = 1 \quad \forall i \in N \end{aligned} \tag{7.3}$$

Since all elements in matrices X and P are integer, more specifically 0 or 1, this is clearly an example of $0-1$ Integer Programming, a well known NP-hard problem. Note that this analysis may also be generalized for unequal traffic generated by each node. In that case, X_{ii} will be equal to the traffic generated by node i. In our case, the imbalance function, $I(\mathbb{S})$ is given by Equation 7.2, which is a non-linear function. Hence, balancing the load turns out to be a NP-hard problem.

7.4 Implementation of Load Balancing in LLNs

We propose a few modifications to RPL in order to accomplish load balancing for LLNs: firstly, each node, while implementing RPL non-storing mode, should advertise a Parent Set instead of a single parent to the DAG root or LBR. This is achieved by DPM - Parent Selection module in the proposed HIPC architecture. In our implementation, nodes advertise a maximum of three parents to the DAG root. Of course, if a node is reachable only through a single parent, it advertises the only parent. Secondly, after gathering information about the whole DAG, the DAG root should run a heuristic that assigns a parent to each node from the Parent Set provided by that node. This process is performed in the LBR by CPM - Traffic Engineering (CPM-TE) Module. The advertised parent set might be subject to policy enforcement module before running any load-balancing heuristics, though. Our proposed heuristic to minimize the load imbalance, is given in Algorithm 2 and 16. The CPM-TE module interacts with the CPM - DAO Collector module to gather assigned parent set ia DAO packets, and send DAO-ACKs to inform every node about their assigned parent, that should be used for periodic or bulk data reporting.

The following data structures are needed to run the algorithm:

1. $SubSize(j)$ = Subtree size of a node j.
2. $AvgSubSize(R)$ = Average subtree size for nodes at rank R.
3. $Child(j)$ = Set of all nodes that have node j as a parent, preferred or backup.
4. $PrefChild(j)$ = Set of all nodes that have node j as preferred parent.
5. $Par(n)$ = Preferred parent of node n.

Periodically, as the LBR collects the parent information of all nodes in the network, it runs the optimization methods to choose the best parent for each node for balance data collection. In the next period, if the obtained Parent Set (P) covers previously detected

Algorithm 15 High Level Balancing - I1

for $R = H + 1$ downto 2 **do**
  for Node $j \in N(R)$ **do**
    $Diff(j) = SubSize(j) - AvgSubSize(R)$
  end for
  Create $N\_Sorted(R)$ as sorted list of $N(R)$, increasing order of $Diff(j)$, j$\in N(R)$
  $Curr(N\_Sorted(R)) = top(N\_Sorted(R))$
  while dec **do**
    dec = Compensate($N\_Sorted(R)$);
  end while
end for

Algorithm 16 Compensate(N_Sorted(R)) - I2

1: **if** N_Sorted(R+1) is empty **then**
2: `return(false)`
3: **end if**
4: **for** node $j = Curr(N\_Sorted(R))$ up to $end(N\_Sorted(R))$ **do**
5: $Curr(N\_Sorted(R))$ = Position of j in $N\_Sorted(R)$
6: Create Candidate Set $\mathbb{V} = \{Child(j) \setminus PrefChild(j)\}$, sorted w.r.t subtree size
7: Find node $n \in \mathbb{V}$ with maximum subtree size, s.t. n is not marked AND $SubSize(n) \leq (Diff(Par(n)) - Diff(j))/2$
8: `Assign node n from Par(n) to j as below:`
9: $PrefChild(Par(n)) = PrefChild(Par(n)) \setminus \{n\}$
10: $Diff(Par(n)) = Diff(Par(n)) - SubSize(n)$
11: $SubSize(Par(n)) = SubSize(Par(n)) - SubSize(n)$
12: INSERT $Par(n)$ to new position in $N\_Sorted(R)$
13: $Par(n) = j$
14: $PrefChild(j) = PrefChild(j) \cup \{n\}$
15: $Diff(j) = Diff(j) + SubSize(n)$
16: $SubSize(Par(n)) = SubSize(Par(n)) + SubSize(n)$
17: INSERT j to new position in $N\_Sorted(R)$
18: Mark n
19: $N\_Sorted(R+1) = N\_Sorted(R+1) \setminus \{n\}$
20: `return(true)`
21: **end for**
22: **if** $Curr(N\_Sorted(R)) == end(N\_Sorted(R))$ **then**
23: `return(false)`
24: **end if**

best set (B), the LBR advertises the same. However, if $B \not\subset P$, the LBR should run the heuristic again.

7.4.1 Runtime analysis of proposed method

Since we are providing a greedy heuristic for an NP-hard problem, it is important to provide the runtime complexity. We propose the following lemmas in order to establish the worst-case runtime complexity of the whole heuristics.

Lemma 1: For any rank R in the DAG / tree, 'compensation' (Lines $8-20$ in Algorithm 16) is executed at most N_{R+1} times, where N_{R+1} is number of nodes at rank $R+1$.

Proof: It is to be noted, that a node is unmarked and marked in 'Compensate' function a maximum of 1 time. A node, if used to compensate between two parents in algorithm 16, is not used again for compensation purpose. Also, in each compensate function call, exactly one node is marked; when no node is available for compensation for a given rank R, the high level algorithm moves on to a lower rank. Thus, in worst case, for every Rank R, N_{R+1} nodes are marked.

Lemma 2: For each node j chosen in Line 4 of algorithm 16, compensation procedure has a worst case runtime of $O(N)$.

Proof: Clearly, for each node j, the constructed children set is bounded by the node degree Δ. Finding a suitable node to satisfy the condition of line 7 in algorithm 16, can be done in $O(\Delta \log \Delta)$ time. To do this, one may just construct $\mathbb{V}$ in $O(\Delta)$ time, and sort this set in order of subtree size. To find a suitable candidate from a sorted set can also be done in $O(\Delta)$ time. Hence, finding n has a worst case runtime of $O(\Delta \log \Delta)$. In case a suitable child is found, inserting the current and new parent at new positions can take upto $O(N_R)$ time. Hence, for a given node j, this compensation process may take up to $O(N_R + \Delta \log \Delta) = O(N)$ time, since $N > N_R$ and it is safe to assume $N \geq \Delta \log \Delta$.

Consider that, if a node j does not have a candidate child to be compensated with, that node is not visited again for the given rank R, as we advance the current position

in line 5 in algorithm 16. Combining this fact with the above lemmas, we find the total runtime complexity of the proposed method to be,

$$T = \sum_{1 \leq R \leq H} N_{R+1} \cdot O(N) = O(N) \cdot \sum_{1 \leq R \leq H} N_{R+1} = O(N^2)$$

7.5 Evaluation

This work simulates RPL in OMNeT++ discrete event simulator engine, using Castalia-2.2 WSN framework. However, LLNs posses highly time dependent attributes, a fixed probabilistic packet delivery or link condition does not represent typical challenges in an urban network. Hence, to make the results more realistic, the authors gather topology and link quality data with respect to time from real outdoor deployments to build a database, and use these information instead of simple probabilistic Packet Delivery Ratio (PDR). Each link in the simulated work behaves in the exact same way as a real link fluctuates in the real life deployment. All the nodes simulate TelosB motes with CC2420 radio with a data rate of 250 Kbps, and a Tx current of 17.4 mA operated by 2 AA batteries, along with 802.15.4 MAC/PHY specifications.

7.5.1 Simulation results

In Figure 7.4, we plot the calculated imbalance as per equation 7.2, for networks of different size. We generate routing tables by implementing RPL. With proposed load balancing algorithms, total imbalance is observed to be much less, and the difference is more prominent for larger networks. Clearly, larger the size of the network is, more option the algorithm has to balance the load in the the network. Since load balancing is most required at the top level of aggregation where the nodes carry most amount of traffic. Hence in Figure 7.5, we show the variance in energy expense for the nodes at the top level of the network, directly connected to the aggregation point. This outcome shows similar property as Figure 7.4, providing the insight that proposed imbalance metric in equation 7.2 gives more importance to the top level.

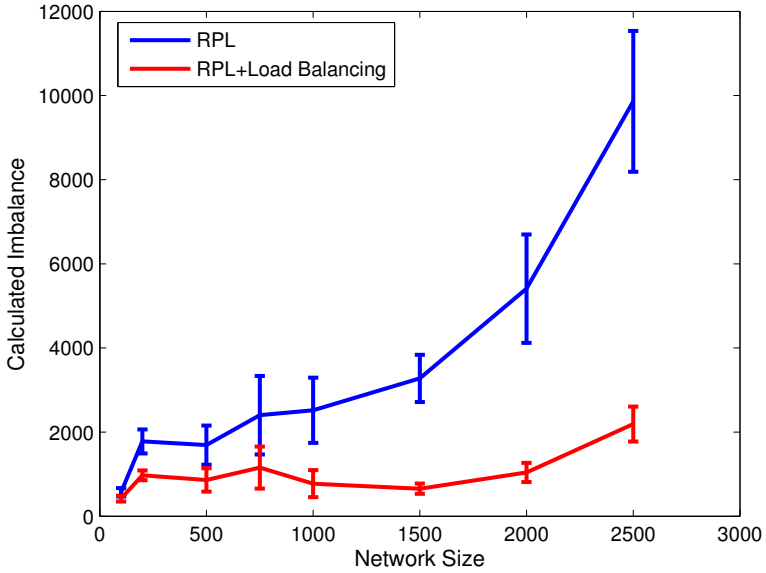

Figure 7.4: Imbalance metric (I) vs Network size.

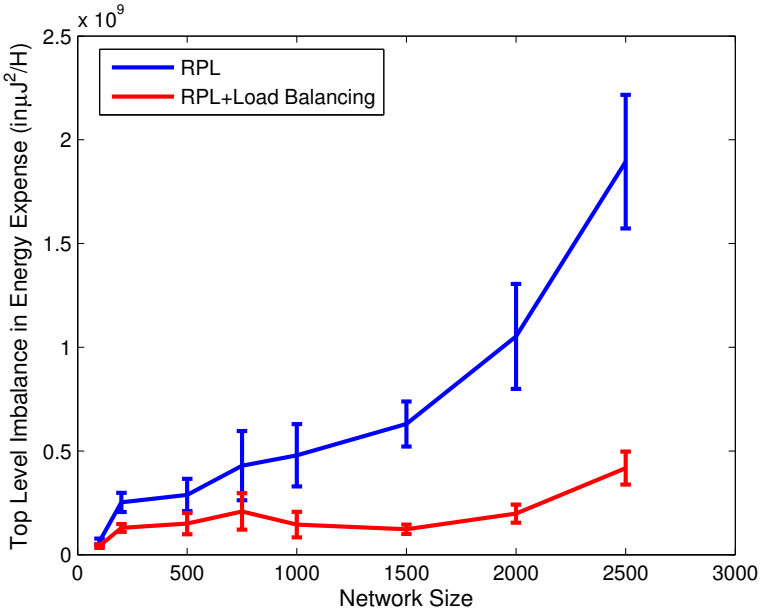

Figure 7.5: Top level variance in energy expense.

In accordance with Equation 7.2, the top level energy expense variance is calculated

as follows:

$$I_{E_R=2} = \frac{1}{|N_{R=2}|} \sum_{i \in N_{R=2}} (E_i - E_{R=2})^2 \tag{7.4}$$

, where similar to S_i and S_R, E_i and $E_{R=1}$ refer to energy spent by node i and average energy expense of all nodes at rank $R = 1$ respectively. Figure 7.6 shows how the energy expense is distributed among nodes for a network of size 2000. By providing proposed load balancing approach, this heuristic increases number of nodes that consume less energy, and decrease the number of nodes that consume maximum energy. As it can be observed, the maximum energy consumption by any node drops from $3.5 \times 10^5 \mu J$/hour to $9.5 \times 10^4 \mu J$/hour.

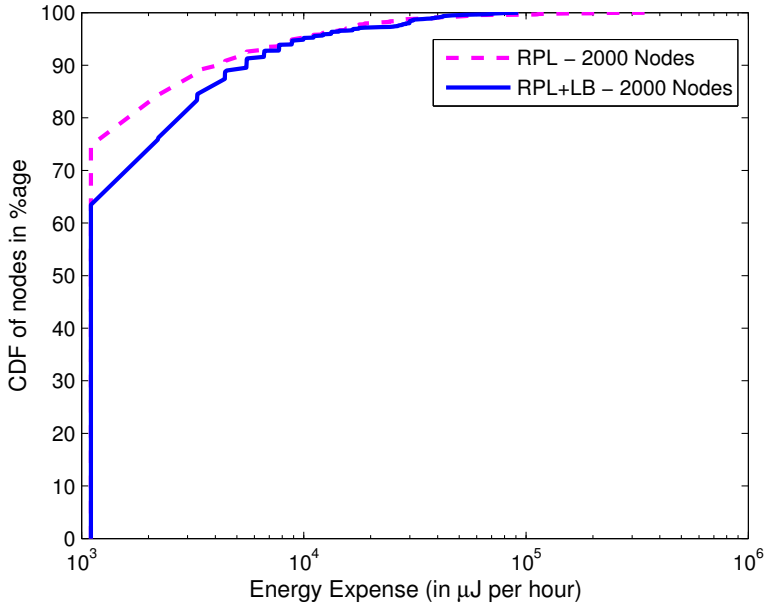

Figure 7.6: CDF of energy expense vs % of nodes.

It is unarguable, that the most desired and prominent effect of load balancing should be to increase the lifetime of a network. By decreasing the maximum energy expense, one can hope to keep the network alive for more time. For our evaluation, lifetime is defined

to be the time when the first node depletes its battery. Since this node is most expected to be a node at the top level, this incident may lead to unreachability of a number of nodes in the network. In Figure 7.7, the average lifetime of different size networks for RPL with and without the proposed load balancing approach is plotted. Clearly, the improvement is significant. The lifetime improvement factor, defined as the ratio of network lifetime with load balancing approach and without it. As one can observe in Figure 7.8, the improvement factor increases with the network size. Proposed algorithm provides only an average of 10% of network lifetime improvement in a network of 100 nodes, as the opportunity of load balancing in such a small network is limited and unnecessary. However, for a network of size 2500, we can achieve a mean improvement of over 160%.

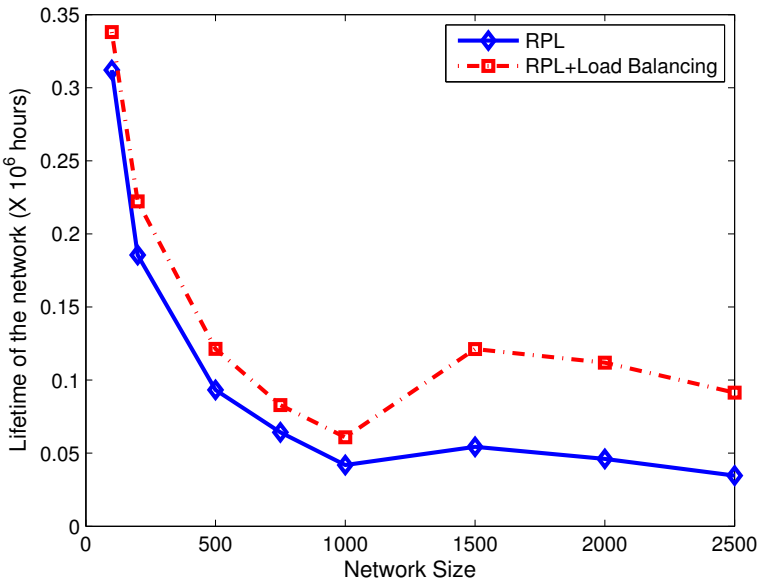

Figure 7.7: Network Lifetime vs Network size.

Since what has been developed in this work is essentially a heuristic working in a greedy fashion, we also turn our attention towards finding out how effective the proposed method is against the best solution. Being an NP-hard problem, The best solution is no way achievable for a network of sizes considered in this study other than solving the

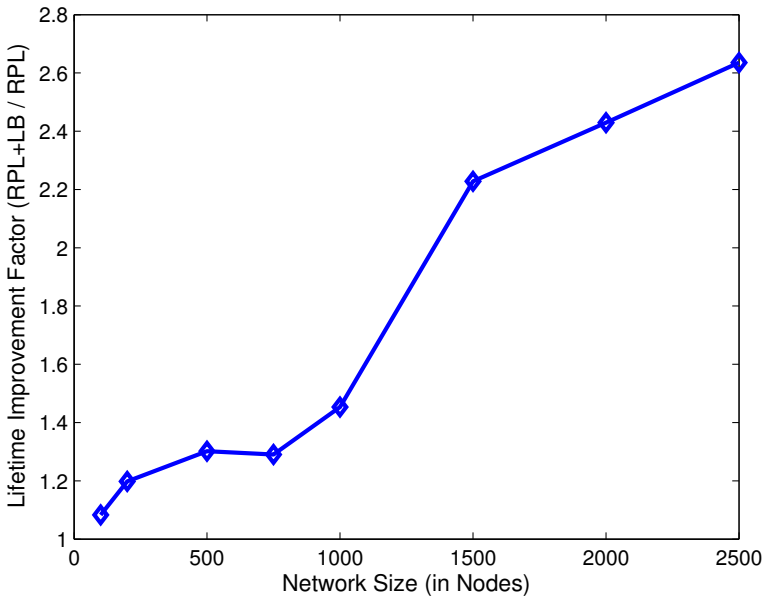

Figure 7.8: Improvement fator by load balancing vs Network size.

maximization problem given in constraint refered by Equation 7.3 seperately. Instead, we generate 25 instances of random topologies of 20, 25, 30 and 35 nodes, and run the proposed heuristic along with Brute Force mechanism that finds the optimum solution that minimizes the imbalance objective function listed in Equation 7.3. The Brute force is needed to find the aggregation tree structure with absolute minimum imbalance, . In Figure 7.9, we compare average imbalance factor for default RPL, proposed Load balancing heuristic and the maximally balanced tree for each instance of topology generated. Also, In Figure 7.10, we plot the imbalance factor at the top level (Rank = 2), based on number of packets handled. The Top level imbalance on packet forwarding is calculated based on Equations 7.2 and 7.4, and is shown in the following equation 7.5.

$$I_{S_{R=2}} = \frac{1}{|N_{R=2}|} \sum_{i \in N_{R=2}} (S_i - S_{R=2})^2 \tag{7.5}$$

However, even for small networks, implementing a brute force mechanism will be exponentially time consuming. For an example, In a network with 35 nodes, if each node has only 3 candidate parents to chose from, the brute force search mechanism needs to

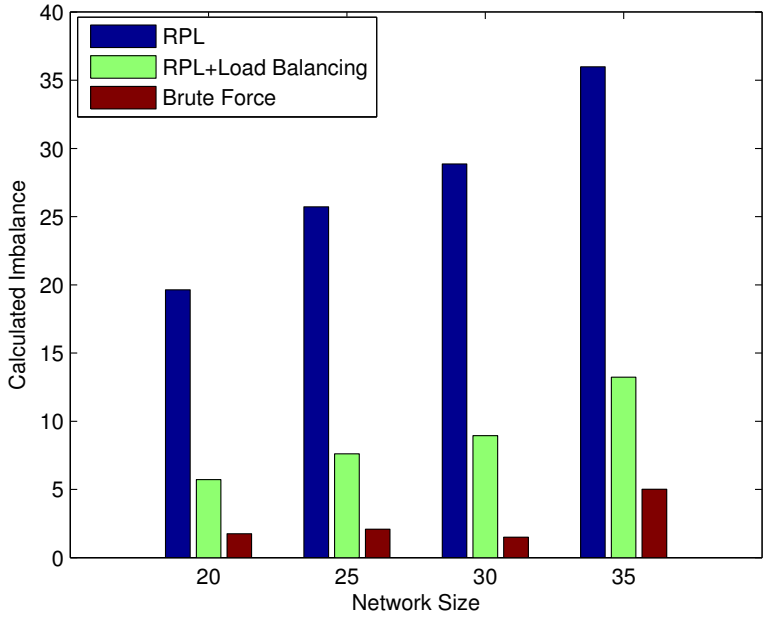

Figure 7.9: Imbalance Factor for Default RPL, Proposed Heuristic, and Maximally Balanced DAG.

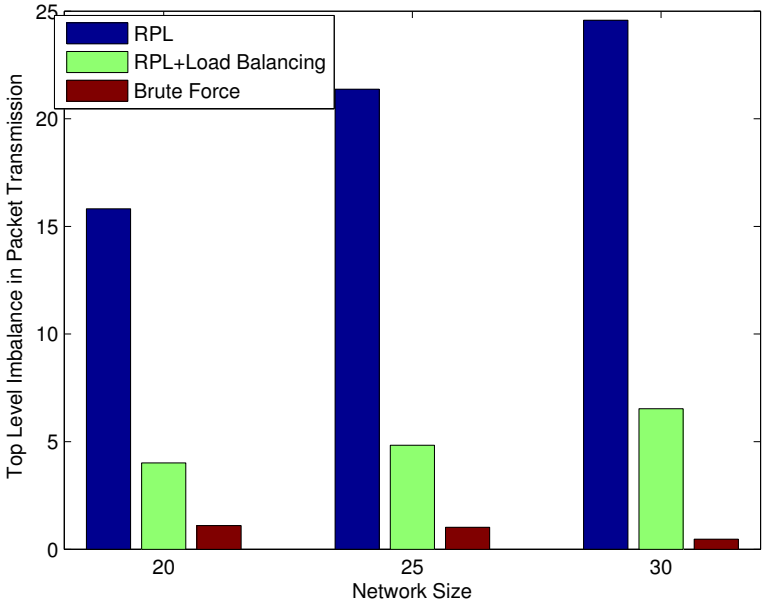

Figure 7.10: Top level Variance in Packet Transmission for Default RPL, Proposed Heuristic, and Maximally Balanced DAG.

generate and evaluate 3^{34} ($\sim O(10^{16})$) possible instances of tree. On the other hand, the proposed heuristic should be extremely efficient with $O(N^2)$ operations, N being the number of nodes in the network. For the computational purpose, the average time taken to complete the 'Compensate' algorithm is calculated for topologies of 200 – 2500 nodes, and is plotted in Figure 7.11. We can observe, even in a large topology of 2500 nodes, the heuristic takes less than 4 seconds to complete. All these tests have been performed on a computer with 6 GB RAM and 2.4 GHz Intel processor.

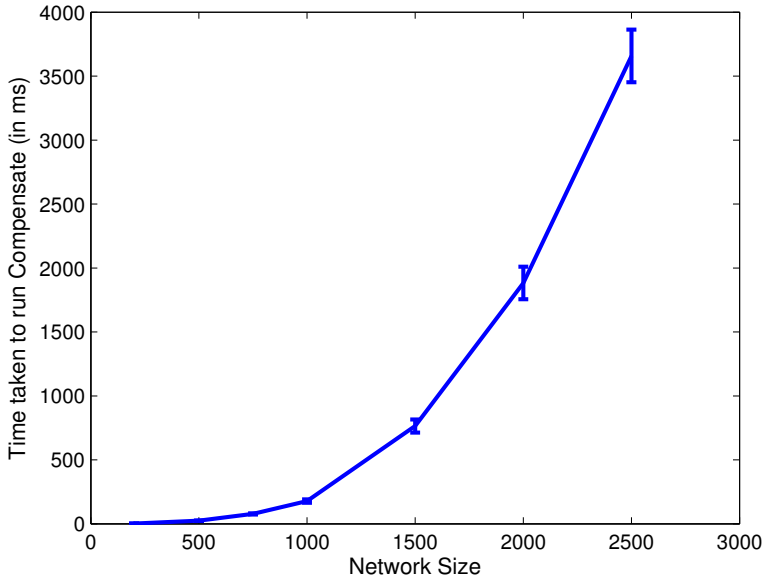

Figure 7.11: Time to run compensate algorithm.

7.6 Summary

This Chapter shows how existing load balancing techniques cannot satisfy the needs of large scale LLNs. A metric to calculate load imbalance in any data aggregation tree is hereby proposed, minimizing which would yield least variation in data traffic handled as well as in energy consumption by the same level nodes. This work shows that minimizing any such imbalance function of a network, given a partial topology knowledge, is a NP-

hard problem. Thus a greedy heuristic is proposed, which runs in $O(N^2)$ time to minimize imbalance at every level and thus the overall imbalance in the network. The simulation results show that the proposed approach indeed achieves less variance in handled data traffic, low imbalance and help improve the network lifetime by considerable amount. This work shows that with a very partial knowledge of the topology, such as at most 3 parents, notable improvement in energy expense and load balancing can be achieved. Other optimization methods, such as genetic algorithms can be applied to achieve the best parent assignment over time.

8

Future Work - RPL Adaptation in IoT via HIPC

So far, we have only developed an appropriate routing protocol for LLNs. However, adaptation to the Internet of Things is a much bigger task than simple routing, control overhead scaling and congestion control. IoT is not going to be limited within a single deployment scenario, or rather neither will be controlled by single vendor or implementation policy. Objective function (OF) to build a DAG or to route data is envisioned to depend on several policies. Different users may have different Service Level Agreements (SLA) or Quality of Service (QoS) requirements. It is indeed required that while routing the data, each policy and requirement should be enforced. However, the individual routing elements within the LLN, or the smart objects does not often have the processing power for policy details, performing traffic engineering or even simple tasks such as load balancing. Hence clearly, some centralized decision making is needed. Clearly different from Wireless Sensor Networks (WSN), an LLN can not avoid centralized routing techniques, as the objects need to be globally accessed.

It should be noted RPL non-storing mode forces the border router (LBR) to insert a source routing heading into the packets to route a data. However, the source routing header is constructed based on the parent node provided by each routing element in the

LLN. With this approach, RPL alone can not support different traffic requirements. Also, fully centralized approach is not possible due to several reasons. Firstly, gathering all link states are not possible since control overhead has to be kept at a minimum. Secondly, due to varying link quality, the updates will be frequent, and consume scarce bandwidth and energy resources.

8.1 Function of Proposed Architecture

In this chapter, we propose Hybrid Intelligent Path Computation (HIPC), which is a routing architecture based on RPL (preferably non-storing mode), but is capable to extend the services of this routing protocol so that policy enforcement, traffic engineering, etc., are possible. By decoupling policy control from objective function, we require the smart objects in IoT to use their processing power less often. We also aim to provide an example based on a functionality offered (load balancing) how this architecture can be used.

As we can observe in Figure 8.1, an Autonomous System (AS) can have multiple LLN border routers, along with other border routers and interior gateways. An autonomous system that encompasses both kinds of borders is most likely to belong to a single vendor or a single entity, such as a University or apartment complex. For an apartment complex, the individual LBRs may connect to different building automation systems. For a utility provider, an LBR can be installed in a local substation, that connects to individual meters (AMI or gas meters) in the locality, forming a border gateway for an urban LLN. The autonomous system in this case may be the proprietary intra-network of the utility provider company. However, the same meters and individual components also needs to be accessed through standard IP network, for which the traffic arrives from external autonomous systems, which can belong to same vendor, or a different vendor agreeing to common set of terms and policy. Clearly, priority of traffic and provided bandwidth to access individual devices or sensors inside the LLN would differ from traffic to traffic, and thus also from application to application. For example, a security application within the same autonomous would have more priority than a polling application traffic originating outside

the AS, while the later can only be routed through mains power operated nodes within the LLN. Thus, within the LLN, a node may need to chose different parent in the DAG based on the application.

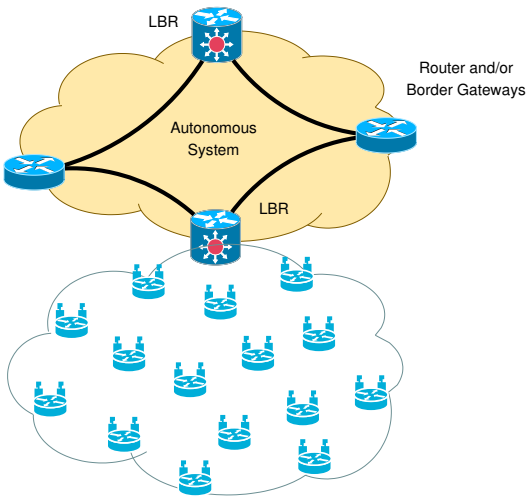

Figure 8.1: LLNs Connected to an AS.

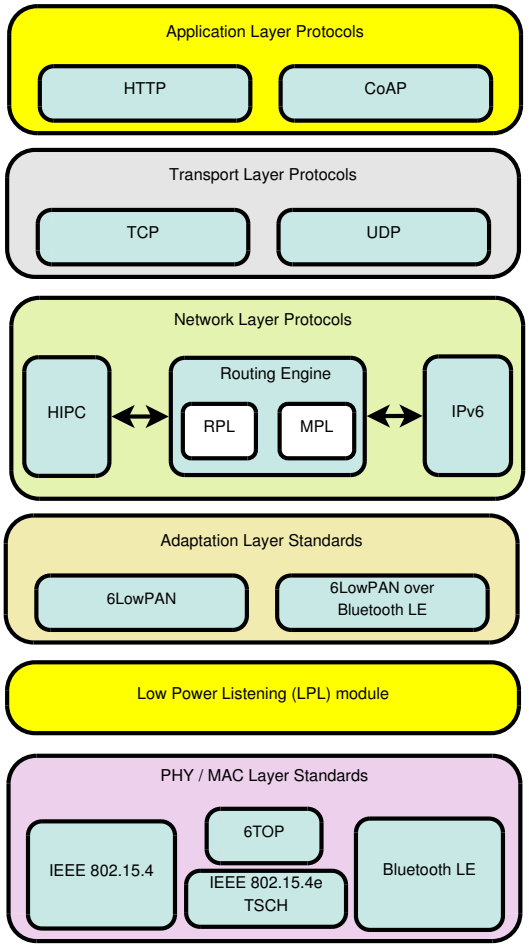

Figure 8.2: HIPC in a complete Protocol stack for LLN Devices

However, delegating the responsibility of determining different routes for different applications to the individual smart objects or sensors within the LLN can be quite burdensome to the already low resourceful nodes. At the same time, as we pointed out in Section 7.1, distributed optimization approaches for load balancing are not viable either.

Hybrid Intelligent Path Computation (HIPC) is designed to be included in layer 3, along with the core RPL mechanism, as shown in Figure 8.2. While the core RPL will be responsible for generating and processing control packets such as DIO, DAO etc., handling different timers including trickle timer for DIO generation, and forwarding the data packet to the next hop, HIPC will determine the suitable parent for corresponding traffic, estimate average traffic, provide policy control and service level agreement, implement load balancing, etc. The RPL core with provide the HIPC block with the set of parents which are either obtained by receiving DIO messages (for nodes other than LBR) or via receiving DAO advertisements by the LBR from in-LLN nodes. The HIPC will provide the RPL core with processed forwarder list, based on either flow label or application ID.

8.2 Detailed HIPC Architecture

HIPC architecture will consists different types of processing modules, which will be interacting among themselves. These processing modules can be broadly classified to two different classes based on their location in the network and functionality. Modules that form HIPC component in the routing elements of smart objects inside the LLN are termed as Distributed Processor Modules (DPM), as they process information related to a single node. The second type of modules are termed Centralized Processor Modules (CPM), which are active in the border routers, and normally acquire a high level view of the network. Distributed Processor Modules (DPM) are responsible to interact with RPL core mechanisms and lower levels to gather information on best possible parents for the node. On the other hand, CPMs are responsible for analyzing information forwarded by individuals nodes from within the LLN along with Policy rules, service level agreements, QoS requirements etc. CPMs thus provide the RPL core of a border router with an appropriate parent selection for each node, which is forwarded to the nodes via piggy-backing some acknowledgement, preferably DAO-ACKs.

8.2.1 Distributed Processor Modules (DPM)

Each node in the LLN, instead of assigning a single parent in the advertised destination advertisements (DAO), includes a confident set of parents / forwarders. This is to ensure that the CPMs in LBR may have more than one choices to to forward the data to any node without gathering the full network topology. Specifically, gathering a confident forwarder set from individual nodes are more effective for a dynamic environment such as an LLN than gathering the whole neighbor set. The LBR may and should not have knowledge about the link quality of each node with each of its neighbor, as gathering such huge and temporally unstable information would assume large control traffic volume enough to halt normal network operations. If each node provides the LBR with options that can be confidently used, the routing topology would tend to be much more stable.

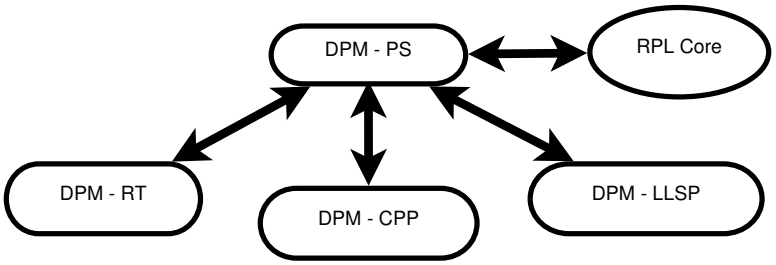

Figure 8.3: Distributed Processor Modules in HIPC.

DPM-PS : As shown in Figure 8.3, these modules interact with RPL core via the DPM - Parent Selection (DPM-PS) module, as for routing in non-storing module the only routing database a node needs to have is its preferred parent. Downward routing, when performed by insertion of source routing header can bypass the RPL core and can be achieved through IPv6 (or μIPv6) module itself. In this case, the DPM-PS module perform the functions listed below:

- It interacts with the RPL control packets transmitted and received to create a confident parent set. In order to achieve this, DPM-PS module need to communicate

with Control Packet Processing Module (DPM-CPP). It receives DIO packets to build a potential parent and sibling set, and returns a set with parents with more confidence to include in the destination advertisement (DAO) packets.

- The module interacts with the Lower Level Signal Processing (DPM -LLSP) module to gather lower level information to determine confidence on its neighbors. DPM-PS sends a set of potential parents to the DPM-LLSP modules, and based on the returned information, it uses a filtering function to sort the set or eliminate weak candidates.

- The module, also receives acknowledgements of destination advertisement (DAO-ACK) packets sent from the LBR via the RPL-core, and forwards back to the Routing Table module (DPM-RT).

DPM-LLSP : The Lower Level Signal Processing (DPM -LLSP) module is responsible to interact with the MAC layer, where it gathers statistics on each transmission and retransmission attempt made. It also receives a candidate forwarder set from the parent selection (DPM-PS) module. For each node in the candidate set, this module maintains a moving average of data transmission success rate, thus creating a database of Expected Transmission Count (ETX) metric for the candidate set. Once a node falls under a minimum configurable ETX threshold from the communicating node, DPM-LLSP blacklists the node. This module may also gather information from the IPv6 (or μIPv6) module, and can also be benefited from it's Neighbor Discovery (ND) or Neighbor Unreachability Detection (NUD) techniques, described in RFC 4861 (82). It returns the DPM-PS module with a filtered or sorted set of parent / neighbor nodes for routing decisions.

DPM-CPP : The Control Packet Processing Module (DPM-CPP) is responsible for receiving control packets (DIO, DAO, DAO-ACK) from the RPL core via the parent selection (DPM-PS) module, parsing the information. It also sends destination advertisements (DAO) to the most preferred parent node via the RPL core. At the same time, it may also receive parameters to emit different control packets. For example, in Section 5.5, a combined approach of determining the value of DelayDAO parameters was proposed.

It can be easily implemented using the HIPC architecture. The DPM-CPP module can calculate the round trip time, since it sends out the DAO packets, and receives the DAO-ACK packets. The DAO-ACKs also contain 'Base' and 'K' parameters as pointed out in Algorithm 10, Section 5.5, which in conjunction with the round trip time, fine tune the parameters needed for DAO emission.

8.2.2 Centralized Processor Modules (CPM)

Centralized Processor Modules (CPM) in the LBR would be responsible for tasks such as handling DAO packets, maintaining a topology of the whole DAG as viewed from the LBR, removing parents from the assigned parent set by individual nodes based on policy etc. Similar to the DPM, these centralized modules communicate with the RPL core implementation via a topology Management (CPM-TM) module. The RPL core is responsible for receiving Data and control packets, Managing trickle timers, sequence numbers, emitting DIO packets etc. The RPL core also receives a preferred topology or calculated parent set, and creates source routing header for routing data packets. The HIPC modules are provided control packets from the RPL core, and policies, application profiles and traffic requirements from the provider. An instance of such an HIPC architecture implementation is shown in Figure 8.4.

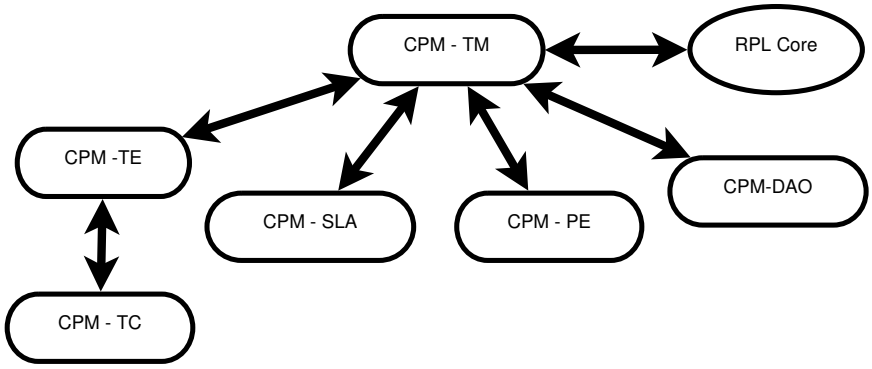

Figure 8.4: Centralized Processor Modules in HIPC.

CPM-TM : The Topology Management (CPM-TM) module is responsible for interacting with the RPL core, receiving control packets such as DAO, and providing with control packets such as DAO-ACK. It is also responsible for creating a preferred topology, given policy information, service level agreement, and preferred traffic engineering rule, such as load balancing or lifetime maximization. It also interacts with other centralized processor modules to perform the following actions.

- It receives a preliminary parent set from Destination Advertisements Module (CPM-DAO) as formed by DAO packets received from the in-LLN nodes and provides the module with a selected parent, so that the CPM-DAO module can issue a DAO-ACK to the DAO issuing node.
- It provides the Policy Enforcement Module (CPM-PE) with the received parent set, and receives a filtered set of parents.
- It provides the Service Level Agreement Module (CPM-SLA) with the filtered parent set received from the CPM-PE module (if any), or with the preliminary parent set received from the CPM-DAO module, and receives another filtered parent set from CPM-SLA.
- It provides Traffic Engineering Module (CPM-TE) with the filtered parent set received from the CPM-SLA module (if any), or from the CPM-PE module if CPM-SLA is not implemented, or with the preliminary parent set received from the CPM-DAO module. From CPM-TE, this module receives a calculated parent set (one for each node), which is forwarded to the CPM-DAO module.

CPM-DAO : It receives Destination Advertisements (DAO) packets via the CPM-TM module as received by the RPL core, creates a preliminary parent set for each in-LLN nodes, which consists of all the candidate parents as chosen by the nodes. This module may or may not use filtering on the received parent set before passing it on to the CPM-TM module. Also, it receives a calculated parent set, which contains one parent per node, or one parent per flow label, per node, and creates DAO-ACK packets for the

node accordingly. At the same time, this module can determine parameters needed for DelayDAO timer triggering, as pointed out in Section 5.5. The CPM-DAO module, in this case, can receive a topology view from the CPM-TM module, and return the values of 'K' and 'Base' parameters to include in the DAO-ACK packets or for the RPL core to use. Precisely, Algorithms 8 and 9 will be run in this module.

CPM-PE : The centralized Policy Enforcement Module (CPM-PE) is responsible to filter a received parent set from the CPM-TM module according to policy rules enforced for a particular LLN on basis of application and / or traffic origin. Normally, operator policies, such as avoiding nodes with certain conditions or preferring nodes certain features while routing can create a filtering criteria. At the same time, Avoiding certain nodes while routing data from a particular origin (vendor) or belonging to a particular flow, can create a filtered parent set with two key index, node and flow label. This would further facilitate label based routing within future LLNs. This module returns the filtered parent set to the CPM-TM module.

CPM-SLA : Centralized Service Level Agreement module (CPM-SLA) will further filter a parent set received from the CPM-TM module, based on traffic differentiation either on the base of origin or based on flow label. Hence, it also create a filtered parent set with two key index, node and flow label and returns it to the CPM-TM module.

CPM-TC : For Traffic engineering, it may be required to know the traffic present in the network. In case of normal traffic pattern being unknown or no pre-determination of traffic passed through the border router, the Traffic Calculator Module (CPM-TC) creates a traffic estimate, and returns the same to the Traffic Engineering Module (CPM-TE). As an example, the following Algorithm 17 can be applied each time the border router is presented with a traffic either originated from its LLN, or destined to an object within the LLN. In the algorithm below, $T_{j,k}^{Sample}$ denotes the traffic volume originated or destined to node j within the current time frame for Application ID k or Flow Label k, whichever applicable. The matrix T^{Avg}, is exponential moving estimate of traffic, where $T_{j,k}^{Avg}$ stands for the average traffic originated or destined to node j for Application ID k or Flow Label

k, over time. The T^{Avg} is returned to the CPM-TE module. This algorithm creates nothing but a moving average estimate of the traffic, and it is on the implementor to design an algorithm for traffic estimation suitable to the LLN in context.

Algorithm 17 Example of Traffic Calculation in CPM-TC Module

1: N = Number of LLN nodes, K = Number of distinguished Application ID / Flow Label
2: Initiate Traffic Matrix T^{Avg}, of dimension $N \times K$
3: Capture each traffic, extracts application ID / Flow Label '$k \in K$', source and destination of the traffic
4: **for** Each node $j \in N$ **do**
5: **if** (Source == j) OR (Destination == j) **then**
6: $T^{Avg}_{j,k} = \beta \times T^{Avg}_{j,k} + (1 - \beta) \times T^{Sample}_{j,k}$
7: **end if**
8: **end for**

CPM-TE : CPM - Traffic Engineering Moderator would run heuristics to chose the most preferred forwarder from the resultant parent set received from the CPM-TM module so that certain requirements can be fulfilled. This module also may also interact with the Traffic Calculator Module (CPM-TC), as pointed out in the previous bullet. Input to this module is mainly two-fold, a filtered candidate parent set received from CPM-TM module, and a traffic matrix from CPM-TC module, which states traffic volume originated from or destined to from each node, for each flow label, if label based routing is present. Once it receives such a traffic detail, it can run further heuristics to determine the final parent set based on traffic engineering needs. For example, Algorithms 15 and 16, can be run in CPM-TE to provide a parent set to maximize the network lifetime. On each occasion of global repair (by releasing new DAG Sequence Number), or significant change in average traffic (T^{Avg}) matrix, or a parent failure, these algorithms can be rerun to generate new parent set.

8.3 Summary

This Chapter provides a brief introduction to the proposed HIPC architecture, and highlights the key procedures. This architecture mainly aims at reducing the processing burden from smart objects with low resources, yet is capable of delivering current Autonomous Systems features of policy control, traffic engineering, etc. to thge context of LLNs. This architecture, thus extends the capability of emerging Internet of Things to be seamlessly assimilated into current world IP architecture. The future prospect of this small idea can be immense, As there is currently no standard on how a border router (LBR) can be implemented as a gateway to the LLNs, and most certainly there is no such architecture yet to provide Autonomous Systems (AS) capabilities into an LLN.

9

Conclusion and Future Directions

This chapter summarizes the goals achieved in this thesis, and how future systems can be built upon what we have today.

9.1 Summary of This Thesis

We have achieved the following in previous years while working on RPL and routing in LLN in general:

- We have designed a WSN simulator, which can take network link traces as input, generate topologies identical to real deployment or random ones, and vary the link quality in similar ways to a real deployment.
- We have implemented RPL for different networks and validated that local repair mechanism with poisoning sub-DAG indeed keep connectionless time to a low value, so nodes always have a path to reach the LBR. The performance evaluation studies of RPL has been published in (17, 18).
- We have performed exhaustive comparative study between RPL and LOADng, and researched whether proactive protocol or reactive protocols are better suited for LLN needs. We have also shown that for large networks, buffer space consumed and memory requirement for RPL tends to increase. Hence with default configuration in

proposed standard, RPL will not behave optimally. This work has been submitted for consideration in Elsevier's Ad Hoc Networks Journal. A short version of the paper appeared in (83).

- Two mechanisms are proposed to resolve the congestion and high memory and buffer requirements in the standard RPL non-storing mode implementation, which makes use of a fixed, universal DelayDAO Timer. We have observed that both proposed mechanisms provide fairly low RAM usage, especially in large topologies. These algorithms also incur less delay for data packet delivery by controlling the control plane congestion. These findings have been published in (64).

- A Combination of Distributed and Centralized mechanisms to schedule Destination Advertisements (DAO) is proposed to retain the benefits of both approaches in Chapter 5, and has been evaluated.

- A DAO generation and aggregation mechanism is proposed for RPL storing mode, which is observed to provide the DAG root with faster congregation of addresses in the network, less control overhead and faster delivery of data packets, by lowering DAO congestion.

- We show how existing load balancing techniques cannot satisfy the needs of large scale LLNs, and have demonstrated how load balancing can be implemented through proposed HIPC architecture. We have implemented a greedy heuristic for creating a balanced tree from a Directed Acyclic Graph (DAG). Thus, forming the base of load balancing in LLNs. Some preliminary results of this work have appeared in (84).

- We have drafted the HIPC architecture for routing over LLNs, so that future needs such as policy enforcement, traffic engineering, quality of service, etc., can be handled. We have shown how traffic estimation, load balancing, policy enforcement, etc. can be achieved through this architecture, without burdening the smart objects having low resources, thus paving the way for LLNs to be compatible with the broader Internet of Things (IoT).

9.2 Future Work Items

The future prospects of the work in this thesis are immense. Routing in Low-Power Lossy Networks (LLNs), smart grid etc., is a nascent area, where we have many stones yet to be turned. The proposed HIPC architecture provides a stepping stone towards practical implementation of LLNs and augmentation with existing IP world, so that sensor networks can implement service differentiation for end users, which was hard to foresee even a decade back. This thesis can be extended to achieve the following in near future:

- Developing the HIPC architecture, and detailing how QoS, Traffic Engineering, etc., can be achieved via this architecture.
- It should be interesting to investigate how the number of reported parents effect the achievable load balancing. It is our intuition, that as we increase maximum number of parents reported by a node, better load balancing and network lifetime would be achieved. It is also possible to extend this work with the help of online genetic algorithm, so that a change in network topology or parent set does not trigger recomputing the whole routing matrix, but the network gradually approaches towards its optimum performance.
- Implementation of HIPC architecture along with core RPL functionalities in TinyOS or Contiki OS with MicaZ motes, and verification of proper and intended operation in real deployments along with performance results, thus probably creating a prototype of HIPC enabled network.
- Finally, it is possible to extend Software Defined Networking (SDN) concept to the Low Power Lossy networks as well, by adopting and fine tuning the proposed HIPC structure. The DPM-RT module proposed in Section 8.2.1, can be used where flow tables can be installed. Also, source routing as performed by the non-storing mode, fits right into the software defined networking concept of decoupling the data plane and control plane from each other. The CPM-TM module as described in Section 8.2.2 can be used as the controller, which has the (partially) full view of the

topology, and performs all routing decisions based on the parent sets and policies enforced. This would, in future open the gateway for easy SDN extension from where it is today.

9.3 Summary of Publications

1. **J. Tripathi**, J. C. de Oliveira, and JP Vasseur, "Proactive versus Reactive Routing in Low Power and Lossy Networks: Performance Analysis and Scalability Improvements", Accepted for publication in Elseviers Journal Ad Hoc Networks.

2. **J. Tripathi**, J. C. de Oliveira, C. Chauvenet, and JP Vasseur, "Why Reactive Protocols are Ill-Suited for LLNs, draft-tripathi-roll-reactive-applicability-02", IETF draft, Jan. 2014. http://tools.ietf.org/html/draft-tripathi-roll-reactive-applicability-02

3. **J. Tripathi** and J. C. de Oliveira, "Quantifying Load Imbalance: A Practical Implementation for Data Collection in Low Power Lossy Networks," in the Proceedings of the 47th Annual Conference on Information Sciences and Systems (CISS 2010), Baltimore, Maryland, March 2013.

4. **J. Tripathi** and J. C. de Oliveira, "Proactive versus Reactive Revisited: IPv6 Routing for Low Power Lossy Networks," in the Proceedings of the 47th Annual Conference on Information Sciences and Systems (CISS 2010), Baltimore, Maryland, March 2013.

5. **J. Tripathi** and J. C. de Oliveira, "On Adaptive Timers for Improved RPL Operation in Low-Power and Lossy Sensor Networks," in the Proceedings of the 5th International Conference on COMmunication Systems and NETworkS (COMSNETS 2013), Bangalore, India, Jan. 2013.

6. **J. Tripathi**, J. C. de Oliveira, and JP Vasseur, "IETF RFC 6687, Performance Evaluation of the Routing Protocol for Low-Power and Lossy Networks (RPL)", October 2012. http://tools.ietf.org/html/rfc6687

7. **J. Tripathi**, J. C. de Oliveira, and JP Vasseur, "Applicability Study of RPL with Local Repair in Smart Grid Substation Networks," in the Proceedings of the 1st IEEE International Conference on Smart Grid Communication (IEEE SmartGridComm 2010), Gaithersburg, Maryland, Oct. 2010.

8. **J. Tripathi**, J. C. de Oliveira, and JP Vasseur, "A Performance Evaluation Study of RPL: Routing Protocol for Low Power and Lossy Networks," in the Proceedings of the 44th Annual Conference on Information Sciences and Systems (CISS 2010), Princeton, New Jersey, March 2010.

References

[1] Dave Evans. **The Internet of Things - How the Next Evolution of the Internet Is Changing Everything**, April 2011. 1

[2] **The EPCglobal Architecture Framework**. 2

[3] **Internet of Things in 2020 - Roadmap for the Future**, May 2008. 3

[4] Luigi Atzori, Antonio Iera, and Giacomo Morabito. **The Internet of Things: A Survey**. *Computer Networks*, **54**(15):2787–2805, Oct 2010. 3

[5] A. Brandt, J. Buron, and G. Porcu. **Home Automation Routing Requirements in Low Power and Lossy Networks**. RFC 5826, April 2010. 3, 4, 8, 33, 70, 99, 100

[6] K. Pister, P. Thubert, S. Dwars, and T. Phinney. **Industrial Routing Requirements in Low-Power and Lossy Networks**. RFC 5673, October 2009. 3, 4, 8, 11, 33, 34, 70, 90, 99, 100

[7] M. Dohler, T. Watteyne, T. Winter, and D. Barthel. **Routing Requirements for Urban Low-Power and Lossy Networks**. RFC 5548, May 2009. 3, 4, 8, 34, 70, 82, 83, 99, 100

[8] J. Moy. **OSPF Version 2**. RFC 2328, April 1998. 3

[9] T. Clausen and P. Jacquet. **The Optimized Link State Routing Protocol (OLSR)**. RFC 3626, October 2003. 3

[10] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg. **The Optimized Link State Routing Protocol version 2**. DRAFT, October 2012. 3

[11] P. Levis, A. Tavakoli, and S. Dawson-Haggerty. **Overview of Existing Routing Protocols for Low Power and Lossy Networks**. DRAFT, April 2009. 3

[12] R Albrightson, JJ Garcia-Luna-Aceves, and Joanne Boyle. **EIGRP-a fast routing protocol based on distance vectors**. In *Proc. Networld/Interop*, **94**, pages 136–147, May 1994. 3

[13] C. Perkins, E. Belding-Royer, and S. Das. **Ad hoc On-Demand Distance Vector (AODV) Routing**. RFC 3561, July 2003. 3

[14] **IETF Routing Over Low-Power Lossy Networks Working Group**. 3, 68

[15] T. Winter et al. **RPL: Routing Protocol for Low Power and Lossy Networks**. RFC 6550, March 2012. 4, 17, 22, 25, 26, 27, 28, 113, 116, 122, 129, 139

[16] J. Martocci, P. De Mil, N. Riou, and W. Vermeylen. **Building Automation Routing Requirements in Low-Power and Lossy Networks**. RFC 5867, June 2010. 4, 8, 14, 33, 70

[17] J. Tripathi, J.C. de Oliveira, and J.P. Vasseur. **Applicability Study of RPL with Local Repair in Smart Grid Substation Networks**. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, October 2010. 4, 120, 176

[18] J. Tripathi, J. de Oliveira, and J.P. Vasseur. **Performance Evaluation of the Routing Protocol for Low-Power and Lossy Networks (RPL), RFC 6687**. RFC 6687, October 2012. 4, 23, 67, 79, 80, 120, 176

[19] N. Accettura, L.A. Grieco, G. Boggia, and P. Camarda. **Performance analysis of the RPL Routing Protocol**. In *Mechatronics (ICM), 2011 IEEE International Conference on*, pages 767–772, April 2011. 4, 63, 80

[20] **Tiny OS**. 4

[21] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. **System Architecture Directions for Networked Sensors**. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS IX, pages 93–104, 2000. 4

[22] Jonathan Hui and David E. Culler. **IP is Dead, Long Live IP for Wireless Sensor Networks**. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 15–28. ACM, 2008. 4, 6, 7, 71

[23] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. **Next Century Challenges: Scalable Coordination in Sensor Networks**. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '99, pages 263–270. ACM, 1999. 4, 5

[24] Sungmin Hong, Daeyoung Kim, Minkeun Ha, Sungho Bae, Sang Jun Park, Wooyoung Jung, and Jae-Eon Kim. **SNAIL: an IP-based wireless sensor network approach to the internet of things**. *Wireless Communications, IEEE*, **17**(6):34–42, 2010. 5

[25] Minkeun Ha, Daeyoung Kim, Seong-Hoon Kim, and Sungmin Hong. **Inter-MARIO: A Fast and Seamless Mobility Protocol to Support Inter-Pan Handover in 6LoWPAN**. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, Dec 2010. 5

[26] Wooyoung Jung, Sungmin Hong, Minkeun Ha, Young-Joo Kim, and Daeyoung Kim. **SSL-Based Lightweight Security of IP-Based Wireless Sensor Networks**. In *Advanced Information Networking and Applications Workshops, 2009. WAINA '09. International Conference on*, pages 1112–1117, May 2009. 5

[27] J. Hui and J.P. Vasseur. **Routing Architecture in Low-Power and Lossy Networks (LLNs), draft-routing-architecture-iot-00**. DRAFT, March 2011. 6, 71

[28] **IETF Constrained RESTful Environments Working Group**. 6

[29] Z. Shelby, K. Hartke, and C. Bormann. **Constrained Application Protocol (CoAP), draft-ietf-core-coap-18**. DRAFT, June 2013. 6

[30] Adam Dunkels. **Full TCP/IP for 8-bit Architectures**. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, MobiSys '03, pages 85–98, 2003. 7

[31] A. Dunkels, J. Alonso, and T. Voigt. **Making TCP/IP Viable for Wireless Sensor Networks**. In *Proceedings of the 1st European Workshop on Wireless Sensor Networks (EWSN), 2004*, January 2004. 7

[32] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. **Transmission of IPv6 Packets over IEEE 802.15.4 Networks**. RFC 4944, September 2007. 7

[33] Joseph Polastre, Jason Hill, and David Culler. **Versatile Low Power Media Access for Wireless Sensor Networks**. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 95–107, 2004. 7

[34] A. El-Hoiydi and J-D Decotignie. **WiseMAC: an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks**. In *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, **1**, pages 244–251, June 2004. 7

[35] J.P. Vasseur, M. Kim, K. Pister, M. Dejean, and D. Barthel. **Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks**. RFC 6551, March 2012. 17, 28

[36] A. Varga. **The OMNeT++ Discrete Event Simulation Systems**. In *European Simulation Multiconference (ESM'2001)*, June 2001. 24, 120

[37] A. Boulis. **Castalia: Revealing pitfalls in designing distributed algorithms in WSN**. In *5th international conference on Embedded networked sensor systems (SenSys'07)*, 2007. 24, 80

[38] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko. **The Trickle Algorithm**. RFC 6206, March 2011. 26, 99

[39] T. Clausen et al. **The LLN On-demand Ad-hoc Distance-Vector Routing Protocol-Next Generation (LOADng)**. DRAFT, July 2012. 66, 67, 79

[40] 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, pages 73–80, New York, NY, USA, 2011. ACM. 66, 67, 69

[41] D. Wang, Z. Tao, J. Zhang, and A. Abouzeid. **RPL Based Routing for Advanced Metering Infrastructure in Smart Grid**. In *Communications Workshops (ICC), 2010 IEEE International Conference on*, May 2010. 66, 69, 70

[42] C. Petrioli, M. Nati, P. Casari, M. Zorzi, and S. Basagni. **ALBA-R: Load-Balancing Geographic Routing Around Connectivity Holes in Wireless Sensor Networks**. *Parallel and Distributed Systems, IEEE Transactions on*, **25**(3):529–539, March 2014. 67

[43] B. Lichtensteiger, B. Bjelajac, C. Muller, and C. Wietfeld. **RF Mesh Systems for Smart Metering: System Architecture and Performance**. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 379–384, Oct 2010. 67

[44] G. Iyer, P. Agrawal, E. Monnerie, and R.S. Cardozo. **Performance analysis of wireless mesh routing protocols for smart utility networks**. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 114–119, 2011. 68

[45] A. Koliousis and J. Sventek. **Proactive vs Reactive Routing for Wireless Sensor Networks**. Technical report, University of Glasgow, Department of Computing Science, 2007. 68, 70

[46] S. Mohseni, R. Hassan, A. Patel, and R. Razali. **Comparative Review Study of Reactive and Proactive Routing Protocols in MANETs**. In *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*, pages 304–309, 2010. 68

[47] C. Mbarushimana and W. Vanderbauwhede. **Comparative Study of Reactive and Proactive Routing Protocols Performance in Mobile Ad Hoc Networks**. In *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*, **2**, pages 679–684, 2007. 68, 70

[48] Kavita Pandey and Abhishek Swaroop. **A Comprehensive Performance Analysis of Proactive, Reactive and Hybrid MANETs Routing Protocols**. *International Journal of Computer Science Issues*, **8**, November 2011. 68, 70

[49] S.R. Das, R. Castaneda, Jiangtao Yan, and R. Sengupta. **Comparative Performance Evaluation of Routing Protocols for Mobile, Ad hoc Networks**. In *Computer Communications and Networks, 1998. Proceedings. 7th International Conference on*, pages 153–161, 1998. 68, 69, 70

[50] Qing Zhao and Lang Tong. **Energy Efficiency of Large-scale Wireless Networks: Proactive versus Reactive Networking**. *Selected Areas in Communications, IEEE Journal on*, **23**(5):1100–1112, May 2005. 68

[51] D. Johnson, Y. Hu, and D. Maltz. **The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4**. RFC 4728, February 2007. 68

[52] Charles E. Perkins and Pravin Bhagwat. **Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for mobile computers**. In *Proceedings of the conference on Communications architectures, protocols and applications*, SIGCOMM '94, pages 234–244, New York, NY, USA, 1994. ACM. 68

[53] Z. Haas, M. Pearlman, and P. Samar. **The Zone Routing Protocol (ZRP) for Ad Hoc Networks**. DRAFT, July 2002. 68

[54] V. Park and S. Corson. **Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification**. DRAFT, July 2001. 69

[55] 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)*, pages 1974–1979, April 2013. 69, 70

[56] Alessandro Camillò, Michele Nati, Chiara Petrioli, Michele Rossi, and Michele Zorzi. **IRIS: Integrated Data Gathering and Interest Dissemination System for Wireless Sensor Networks**. *Ad Hoc Networks*, **11**(2):654–671, March 2013. 69

[57] A. Camillo and C. Petrioli. **Hands on IRIS: Lessons learned from implementing a cross layer protocol stack for WSNs**. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 157–163, Dec 2012. 69

[58] O. Gaddour, A. Koubaa, S. Chaudhry, M. Tezeghdanti, R. Chaari, and M. Abid. **Simulation and performance evaluation of DAG construction with RPL**. In *Communications and Networking (ComNet), 2012 Third International Conference on*, 2012. 80

[59] Yanjun Sun, Omer Gurewitz, Shu Du, Lei Tang, and David B. Johnson. **ADB: An Efficient Multihop Broadcast Protocol Based on Asynchronous Duty-cycling in Wireless Sensor Networks**. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 43–56, 2009. 98

[60] Yanjun Sun, Omer Gurewitz, and David B. Johnson. **RI-MAC: A Receiver-initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks**. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, 2008. 98

[61] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. **X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks**. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 307–320, 2006. 98

[62] C. Perkins. **Dynamic MANET On-demand (AODVv2) Routing, draft-ietf-manet-aodvv2-03**. DRAFT, February 2014. 99

[63] J. Hui and P. Thubert. **Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks**. RFC 6282, September 2011. 105, 140

[64] J. Tripathi and J.C. de Oliveira. **On Adaptive Timers for Improved RPL Operation in Low-Power and Lossy Sensor Networks**. In *Fifth International Conference on Communication Systems and Networks (COMSNETS)*, January 2013. 108, 177

[65] 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 *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, March 2010. 120

[66] **IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)**, September 2006. 121

[67] **IEEE 802.15 WPAN Task Group 4g (TG4g)Smart Utility Networks.**, 2012. 140

[68] He Huang, Yun Xu, Yu e Sun, and Liusheng Huang. **Cluster-based load balancing multi-path routing protocol in wireless sensor networks**. In *7th World Congress on Intelligent Control and Automation. WCICA 2008.*, pages 6692 –6696, june 2008. 148

[69] Nauman Israr and Irfan Awan. **Multihop clustering algorithm for load balancing in wireless sensor networks**. 2007. 148

[70] Namhoon Kim, Jongman Heo, Hyung Seok Kim, and Wook Hyun Kwon. **Reconfiguration of clusterheads for load balancing in wireless sensor networks**. *Computer Communications*, **31**(1):153 – 159, 2008. 148

[71] Devendar Mandala, Xiaojiang Du, Fei Dai, and Chao You. **Load balance and energy efficient data gathering in wireless sensor networks**. *Wireless Communications and Mobile Computing*, **8**(5):645–659, 2008. 148

[72] Suat Ozdemir. **Secure Load Balancing via Hierarchical Data Aggregation in Heterogeneous Sensor Networks**. *J. Inf. Sci. Eng.*, pages 1691–1705, 2009. 148

[73] A.D. Amis and R. Prakash. **Load-balancing clusters in wireless ad hoc networks**. In *Proceedings. 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology.*, pages 25 –32, 2000. 148

[74] Sem Borst, Iraj Saniee, and Phil Whiting. **Distributed dynamic load balancing in wireless networks**. In *Proceedings of the 20th international teletraffic conference on Managing traffic performance in converged networks*, ITC2007, pages 1024–1037. Springer-Verlag, 2007. 148

[75] Hui Dai and R. Han. **A node-centric load balancing algorithm for wireless sensor networks**. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, **1**, pages 548 – 552 Vol.1, dec. 2003. 148

[76] Xin Guan, L. Guan, X. Wang, and Tomoaki Ohtsuki. **A New Load Balancing and Data Collection Algorithm for Energy Saving in Wireless Sensor Networks**. *Telecommunication Systems*, **45**:313–322, 2010. 148

[77] S. Toumpis and S. Gitzenis. **Load Balancing in Wireless Sensor Networks using Kirchhoff's Voltage Law**. In *IEEE INFOCOM 2009*, pages 1656 –1664, april 2009. 148

[78] Yaling Yang, Jun Wang, and Robin Kravets. **Interference-aware Load Balancing for Multihop Wireless Networks**. Technical report, 2005. 148

[79] Seema Bandyopadhyay and E.J. Coyle. **An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks**. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, **3**, pages 1713 – 1723. 148

[80] O. Younis and S. Fahmy. **Distributed Clustering in Ad-hoc Sensor Networks: a Hybrid, Energy-efficient Approach**. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, march 2004. 148

[81] Ping Ding, JoAnne Holliday, and Aslihan Celik. **Distributed Energy-Efficient Hierarchical Clustering for Wireless Sensor Networks**. In *Distributed Computing in Sensor Systems*, **3560** of *Lecture Notes in Computer Science*, pages 322–339. Springer Berlin Heidelberg, 2005. 148

[82] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. **Neighbor Discovery for IP version 6 (IPv6)**. RFC 4861, September 2007. 170

[83] J. Tripathi and J.C. De Oliveira. **Proactive versus reactive revisited: IPv6 routing for Low Power Lossy Networks**. In *Information Sciences and Systems (CISS), 2013 47th Annual Conference on*, 2013. 177

[84] J. Tripathi and J.C. de Oliveira. **Quantifying load imbalance: A practical implementation for data collection in low power lossy networks**. In *Information Sciences and Systems (CISS), 2013 47th Annual Conference on*, March 2013. 177