Evaluation of performance of RPL and 6LOWPAN using Contiki and Cooja

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

Computer Science & Engineering

under the Supervision of

Dr.Shailendra Shukla

By

Priyam Shukla : 111256

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled **“Evaluation of performance RPL and 6LOWPAN using and Contiki and Cooja”**, submitted by Priyam Shukla in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:- **Shailendra Shukla**

Assistant Professor (Grade-II)

Acknowledgement

I would like to thank my supervisor **Dr. Shailendra Shukla** for his constant help and guidance and for supporting my ideas.

Date:22-12-2014 Priyam Shukla

Table of Contents

NOMENCLATURE

DAO	Destination Advertisement Object
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination Oriented Directed Acyclic Graph
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
LLN	Low Power and Lossy Network
MAC	Medium Access Control
OF	Objective Function
RDC	Radio Duty Cycling
RPL	Routing Protocol for LLN
TCP	Transmission Control Protocol
WSN	Wireless Sensor Network

Table Of Figures

List of Tables

ABSTRACT

The amount of research performed in the domain of Wireless Sensor Networks has increased to enormously in recent years. The reason of this is the advancement of fabrication technology of low-power devices containing various sensors along with a wireless interface, allowing costs to be reduced. However, the deployment of large-scale sensor networks, which could consist of several thousand nodes, still represents a challenge to researchers.

This project aims to evaluate the performance of routing protocols in lossy links for smart building networks. Special attention is directed towards the RPL protocol which is the IETF candidate for a standard routing protocol in wireless sensor networks.

The results reveal that RPL is suitable for use in large networks (over 200 nodes) because it does not present any scalability limitations as opposed to the other protocols tested in this thesis. However, for small to medium-sized networks (less than 100 nodes), RPL's performance does not outclass that of the other protocols that were tested. Each protocol has proven to have its strengths, thus when choosing a protocol for a wireless sensor network deployment, the decision has to be based on the specific requirements of the WSN application.

CHAPTER 1
INTRODUCTION

The main aim of this project is to investigate the performance of the RPL (IPv6 Routing Protocol for Low power and Lossy Networks) routing protocol in 6lowPAN (IPv6 low Power wireless Area Networks) networks, analysing its strengths and weaknesses.

RPL is a routing protocol defined by the IETF Routing Over Low power and Lossy (ROLL) networks Working Group, and is being promoted as the standard for use in wireless sensor networks. A number of situational requirements for RPL have been identified by the ROLL group for building services and industrial automation.

The goal of this project is to evaluate the performance of the protocol in meeting the complex requirements of such applications. The project evaluates the performance of the RPL protocol against existing wireless sensor network and MANET routing protocols in the SpeckSim network simulation environment.

The project was developed according to the following three phases: selection of protocols and criteria for evaluation, implementation of the selected protocols and network traffic models in the simulator, and performance evaluation and analysis.

1.1. Wireless Sensor Networks

Wireless Sensor Networks (WSN) consist of distributed sensors connected as a mesh, with the purpose of monitoring an area of interest and the surrounding environment. The sensors are able to capture various factors of the environment such as temperature, sound, pressure, movement and air composition, among others. Usually, the sensors are dispersed in large numbers in the area of interest. Information collection is often transmitted to a central collection point during, or after, an experiment.
Today, sensors hold a wide area of usage such as in industry (for monitoring and controlling the industrial process), environmental monitoring, healthcare applications and military

applications. More advanced networks support bidirectional communication so that the activity of the sensors can be controlled

The nodes in a WSN are multi-functional, low-power and low-cost devices. Each node can have one or more sensors, a radio transceiver, a microcontroller and runs on a battery. The range of the radio communication between the nodes is usually up to 30 meters.

An important challenge is represented by the scalability of network protocols when dealing with a large number of nodes. However, an equally important challenge is developing simple and efficient protocols to be used for various operations within a WSN. These design requirements are necessary in order for the protocols to be suitable to run on the network nodes, which are limited in terms of hardware (processing power, memory and battery size).

1.2. Routing in WSN

Routing within mesh-connected wireless sensor networks is important as most applications designed to run over these types of networks are based on the process of intra-node communication, as well as the process of transferring data from individual nodes to a central collection point (sink). Routing within wireless sensor networks differs from routing in IP networks in the following ways:

- The addressing scheme is often not important for data driven applications
- For most of the applications, data is collected from multiple sensors and sent to a central control point
- WSN nodes have serious resource constraints such as processing power, memory size, energy storage and communication bandwidth

Frequently, nodes in a WSN are deployed statically, requiring low mobility. Thus, compared to Mobile Ad hoc Networks (MANET) deployments (high rates of mobility that translates to frequent topological changes), there are very few changes in the topologies.

1.3. Motivation

The need for a standard for wireless sensor network routing led the IETF organization to promote the RPL protocol as its candidate. The results of this dissertation are a contribution to the consultation process. The main aims of the project are discussed next.

The goal of the project is to investigate the behaviour and evaluate the performance of RPL in WSNs. The project also evaluates the performance of RPL against existing WSN and MANET routing protocols in the SpeckSim network simulator.

In respect to the proposed goal, the RPL implementation provided in SpeckSim was validated against the ContikiRPL implementation from the Cooja network simulator. After the validation stage, RPL was tested and evaluated against the DSDV protocol and the four versions of the SimpleP protocol in the SpeckSim simulator.

1.4. Previous Work

This project has benefited from the previous work conducted in the same domain as it. The simulation environment used was the SpeckSim simulator which was developed by the Speckled Computing group , an early version of the RPL implementation in the SpeckSim simulator was provided by Ryan McNally, and the Cooja simulator along with the ContikiRPL implementation was developed by the Swedish Institute of Computer Science . The bulk of the previous and related work that was considered for this project is presented in *Chapter 2 - Background and Related Work.*

Chapter 2

Background and Related Work

This section sets the context of the project, provides a description for RPL and for other protocols that are considered relevant for comparison to RPL, describes the simulator used for evaluation and presents a survey of past research regarding routing protocols in Wireless Sensor Networks. It finally describes the concept of Smart Building networks and provides examples of several applications for such networks.

2.1. RPL

RPL has been proposed by the IETF ROLL (Routing Over Low power and Lossy networks) Working Group as a standard routing protocol for IPv6 routing in low-power wireless sensor networks. According to Tripathi et al [21], "existing routing protocols such as OSPF, IS-IS, AODV, and OLSR have been extensively evaluated by the working group and have been found to not satisfy, in their current form, all specific routing requirements for LLNs (Low power and Lossy Networks)"[21]. These networks (LLNs) are mainly characterized by high loss rates, low data rates and instability and can consist of up to several thousand nodes. The nodes in these networks are constrained in terms of processing power, memory and energy. The RPL routing protocol is tree-oriented, with RPL topologies generated starting from the Root nodes and organized as directed acyclic graphs. RPL forms non-transitive, non-broadcast, multiple access network topologies. RPL offers flexibility in building a topology, but is complex as described in detail in the specification of the IETF draft [1]. The predominant traffic patterns in LLNs are point-to-multipoint and multipoint-to-point. The RPL protocol supports these traffic flows, as well as point-to-point.
Point-to-point traffic is used between the devices within the LLN; point-to-multipoint pattern is used between a central control point and a set of sensor nodes; multipoint-to-point pattern is used between a set of sensor nodes and a central control point

2.2. Literature Survey

According to "A survey on routing protocols for wireless sensor networks" [2] almost all routing protocols for WSNs can be classified as data-centric, hierarchical or location-based. Even though the majority of routing protocols should fit in one of the above mentioned categories, some of them pursue slightly different approaches. Thus, a fourth category was created: “Network flow and QoS-aware protocols”. The paper provides a classification for the WSNs routing protocols as follows:

 Data-centric protocols – A data-centric protocol is query based and depends on the naming of the data. The process works like this: a node, which is a central control point, queries particular regions and then expects to receive data from these sensors. As most of the data collected will be redundant, it is up to the routing protocol to save energy by eliminating the redundant data. Data-centric routing differs from traditional routing (which is address-based) because routes are not created between addressable nodes. Due to the large number of nodes, for many applications for WSNs it is not practical to assign global identifiers (such as IP addresses).This makes it difficult to query a particular set of nodes. Since there is great redundancy of the data transmitted, routing protocols capable of data aggregation help by saving a significant amount of energy.

Example of data-centric protocols: SPIN (Sensor Protocols for Information via Negotiation), Directed Diffusion, Rumor routing, GBR (Gradient-based routing), CADR (Constrained anisotropic diffusion routing), COUGAR and ACQUIRE (ACtive QUery forwarding In senoR nEtworks).

 Hierarchical protocols – Hierarchical protocols mainly address the issue of scalability. A single gateway architecture is not scalable because increasing the density of sensors may lead to an overload of the gateway, thereby resulting a high latency for the communication within the network. Hierarchical routing decreases energy consumption by using multi-hop communication within a cluster and afterwards aggregating and performing fusion on the collected data in order to eliminate any redundant data.

Example of hierarchical protocols: LEACH (Low-Energy Adaptive Clustering Hierarchy), PEGASIS (Power-Efficient Gathering in Sensor Information Systems), TEEN (Threshold sensitive Energy Efficient Network protocol) and APTEEN (Adaptive Threshold sensitive Energy Efficient Network protocol).

 Location-based protocols – These protocols use location/position information to determine a particular region based on the collected data, so that queries are diffused only in that specific region.

Example of location-based protocols (some of the protocols mentioned are designed for ad-hoc networks, but are applicable to WSNs): MECN (Minimum Energy Consumption Network), SMECN (Small MECN), GAF (Geographic Adaptive Fidelity) and GEAR (Geographic and Energy-Aware Routing).

 Network flow and QoS-aware protocols – This category consists of protocols that pursue slightly different approaches such as network flow and QoS. (QoS-aware protocols are concerned with end-to-end delay requirements when selecting paths in WSNs).

This paper [2] surveys recent routing protocols and discusses each under their appropriate category. The aim is to provide a better understanding of the domain and to highlight outstanding open issues. The paper "Routing Techniques in Wireless Sensor Networks: A Survey" [4] focuses on routing techniques and provides classification and comparison of routing protocols in WSNs. It reveals design challenges for routing protocols in wireless sensor networks such as: node deployment, energy consumption without losing accuracy, data reporting model, node/link heterogeneity, fault tolerance, scalability, network dynamics, transmission media, connectivity, coverage, data aggregation, and quality of service. This paper structures routing protocols in a manner similar to [2], and it presents about the same protocols. It divides the protocols into two major categories: "Network Structure Based Protocols" and "Routing protocols based on Protocol Operation". The first category is divided into three subcategories which coincide with the classification provided by [2]. This paper provides well referenced future directions for routing in wireless sensor networks.

A paper highly relevant to this project is "Low-Power Wireless IPv6 Routing with ContikiRPL" [3]. The paper aims to test the RPL protocol, and is based on an implementation of RPL for the Contiki operating system, called ContikiRPL. This implementation was tested both in simulation and in a low-power wireless network. The protocol was evaluated in terms of power efficiency and implementation complexity. The results show that the lifetime of the network with RPL routing on Tmote Sky motes can be several years. Additionally, the protocol has proven to be lightweight and power-efficient. This dissertation presents a different approach for evaluating RPL by taking into account other metrics than the ones mentioned in paper [3], and thereby providing a more comprehensive evaluation. It also provides a comparison between RPL and other relevant routing protocols in order to highlight RPL's strengths and weaknesses. The paper "A Performance Evaluation Study of RPL: Routing Protocol for Low Power and Lossy Networks"[21] provides an evaluation of the RPL protocol, simulated using the OMNET++ simulator with the Castalia2.2 framework that allows for wireless sensor networks simulation within the OMNET++ simulator. The nodes in the simulator used TelosB CC2420 radio and the 802.15.4 MAC protocol. The scenario simulated an outdoors network of medium size (with a total of 86 nodes). The topology was inspired from a real-life deployment and the metrics that were used for RPL's evaluation are the following:

 “Path quality metrics”
 “Control plane overhead”
 “Ability to cope with unstable situations (link churns, node dying)”
 “Required resource constraints on nodes (routing table size, etc.)”

The paper concludes by stretching out the usefulness of the Trickle timer used by RPL which keeps the control overhead to a lower level than the actual data packet count. Also, Global repair was used for managing broken links, which turned out to have “a significant effect on the number of control packets, which may be used to upper bound the overhead for trade off with time with no connectivity”[21]. In other words, Global repair provides a trade off between amount of control traffic generated and the time for loss of connectivity. Since this paper provided the evaluation of the RPL protocol based on an outdoor network, it is relevant to mention that the paper also concluded that the protocol operation has to be modified to better adapt to this type of networks.

2.3. Routing protocols relevant for comparing to RPL

2.3.1. DSDV

Destination-Sequenced Distance-Vector (**DSDV**) is a highly dynamic routing protocol for ad-hoc networks. It is classified as a distance-vector protocol and is based on multi-hop routing and the Bellman-Ford algorithm. By definition, an ad-hoc network comprises of mobile devices/nodes called hosts that communicate within the network without the intervention of any centralised Access Point. To form such a network, each device/host has to operate as a specialized router. When the DSDV protocol was created it has been taken into account that the topologies in ad-hoc networks may be highly dynamic and that most users will desire not to have to perform any administrative actions in order to set up the network. The protocol’s main aim is to solve the routing loop problem. The solution that it provides regarding this problem is the usage of a sequence number for each of the routes within the routing table. For differentiating between valid and non-valid links the protocol uses even (valid link) and odd (invalid link) numbers. These sequence numbers are generated by the destination. Since wireless sensor networks can be quite dynamic, it seemed logical to compare the RPL protocol to a MANET routing protocol such as DSDV. Other routing protocols (such as RIP, EIGRP, OSPF, IS-IS) that are used by regular computer networks could not be taken into account for comparison with RPL because they place too heavy a computational burden on the device running them, thus they are not suitable for use on wireless sensors (which have very limited resources). Even though RIP is somewhat similar to DSDV, since both are rather simplistic distance vector protocols and both use the number of hops as a metric, RIP will not be suitable for an ad-hoc environment because its design cannot handle highly frequent topology changes. And, according to Perkins and Bhagwat [17], the split horizon and poison reverse techniques used by RIP were appropriate for use in wired networks, and do not work in a wireless environment where the packets are sent via broadcast. DSDV avoids the looping problem by tagging each route with a sequence number. Based on the sequence number, nodes will be able to differentiate old and new routes. DSDV, being a distance vector protocol, “in order to keep the distance estimates up-to-date, each node monitors the cost of its outgoing links and periodically broadcasts, to each one of its neighbours, its current estimate of the shortest distance to every other node in the network”[17].

Before the work of Perkins and Bhagwat [17], there were numerous objections concerning the use of the Bellman-Ford algorithm as the basis for routing protocols. These objections targeted the poor looping properties of the algorithm caused by link failures. They provided the specification for the DSDV protocol which addresses some of these objections. Besides the thorough description of the protocol and useful and comprehensive use examples, the paper also provides a theoretical demonstration as to why DSDV guarantees loop-free paths to each destination and a comparison of DSDV to several other routing protocols. DSDV is described as an “innovative approach which models the mobile computers as routes which are cooperating to forward packets as needed to each other. It makes good use of the properties of the wireless broadcast medium and it can be utilized at either the network layer (layer 3), or below the network layer but still above the MAC layer software in layer 2”[17]. The paper’s findings is that mobile computers (modelled as routers) can effectively cooperate to build ad-hoc networks. This dissertation aims to discover how DSDV performs in wireless sensor networks and to compare its performances with that of RPL.

2.3.2. AODV

The **Ad hoc On-Demand Distance Vector** (AODV) routing protocol is designed for ad hoc mobile networks. The protocol manages to adapt to low processing and low available memory conditions and to the high dynamic nature of networks. It enables mobile devices that are part of the network to quickly acquire routes to new destinations. As it is a distance vector protocol, it uses multi-hop routing. However, it avoids the “counting to infinity” problem of the Bellman-Ford algorithm and provides loop-free routing by using destination sequence numbers for each route entry in the routing table. The destination sequence number is seen as a local route metric, thus in case of multiple routes for one destination, the route that holds the greatest destination sequence number is preferred. AODV offers quick convergence in case of network changes. AODV and RPL have several similarities, as they are both distance vector routing protocols, and tree based.

2.3.3. DSR

Dynamic Source Routing (DSR) is a beacon-less routing protocol for ad hoc mobile networks. It uses source routing, implying that the sender specifies the route that a packet takes through the network. The sender deposits the address of the intermediate nodes (which are part of the route that the packet must follow) in the packets. The protocol is based on two mechanisms: “Route Discovery” and “Route Maintenance”. These mechanisms have the purpose of discovering and maintaining routes to arbitrary nodes within the network.

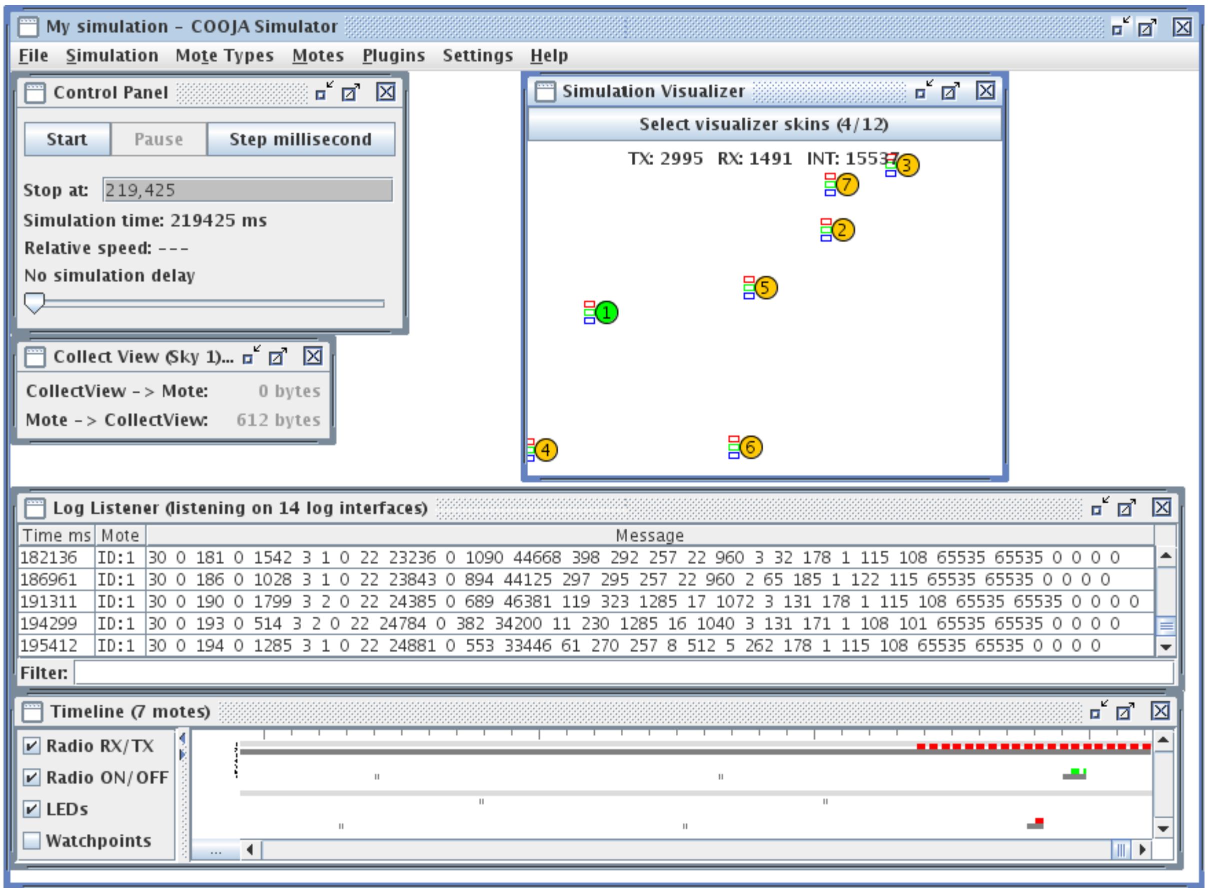
DSR is a simple and efficient protocol, which provides loop-free routing and can run on networks with unidirectional links. It is intended to have low overhead and be able to react quickly to network changes. The protocol determines and maintains all routing within its network, thereby enabling the network to be self-organizing and self-configuring. DSR is designed to scale to hundreds of nodes and is suitable for networks with a high degree of mobility.

Chapter 3

Design and Implementation

When I tried to run the ContkiRPL, the most convenient way was to run it in the Cooja simulator which was also written in Java. This led to the idea of finding out how the Cooja simulator (Java) called the ContikiRPL code (C) and try to do the same for the SpeckSim network simulator.

The easiest way to get Cooja along with the ContikiRPL implementation is to download a virtual machine that comes with all installations of Contiki software, development tools which can be downloaded from [24]. Different examples ("rpl-udp", "rpl-collect", "rpl-border-router") that use ContikiRPL can be found in the virtual machine in "~/contiki-2.x/examples/ipv6". When Cooja starts, it automatically loads some projects including these examples. Unfortunately, at this stage a problem appears due to the fact that Cooja is trying to load its project files from a different directory than the one they are stored in. This can be resolved by copying the needed files into the directory required by Cooja (this is more simple than changing the required directory in Cooja). When opening the ContikiRPL scenario examples, Cooja first compiles them. This also raises a problem. In order to not get errors at compilation, some small changes to the *Makefile* files for each of the examples are required. These changes consist of modifying the paths provided in the first line of the *Makefile* files to match the actual intended location/path. These problems were encountered after performing the update of the virtual machine in June 2011. It is possible that later/subsequent updates have solved these problems.

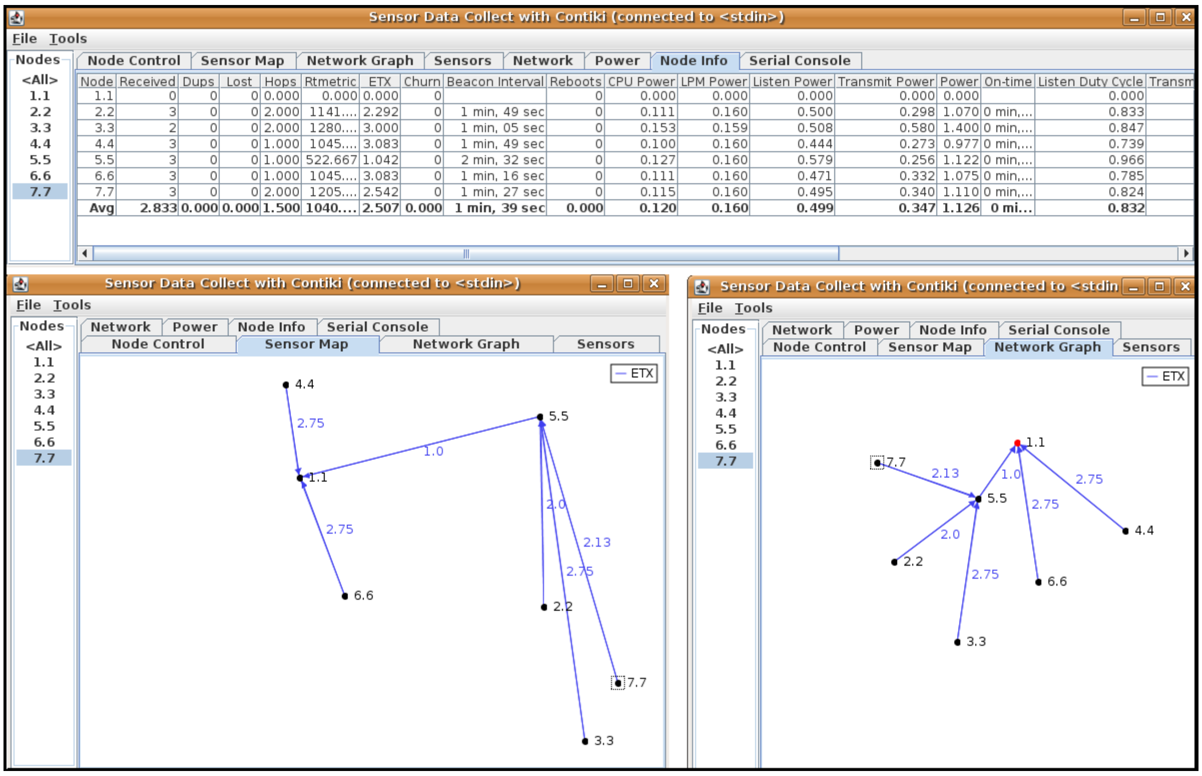

My simulation - COOJA Simulator
File Simulation Mote Types Motes Plugins Settings Help
Control Panel
Start Pause Step millisecond
Simulation Visualizer
Select visualizer skins (4/12)
TX: 2995 RX: 1491 INT: 15537
Collect View (Sky 1)...
Log Listener (listening on 14 log interfaces)
Timeline (7 motes)
Radio RX/TX
Radio ON/OFF
LEDs
Watchpoints

Stop at: 219,425
Simulation time: 219425 ms
Relative speed: ---
No simulation delay

CollectView -> Mote: 0 bytes
Mote -> CollectView: 612 bytes

```
Time ms Mote  Message
182136  ID:1  30 0 181 0 1542 3 1 0 22 23236 0 1090 44668 398 292 257 22 960 3 32 178 1 115 108 65535 65535 0 0 0 0
186961  ID:1  30 0 186 0 1028 3 1 0 22 23843 0 894 44125 297 295 257 22 960 2 65 185 1 122 115 65535 65535 0 0 0 0
191311  ID:1  30 0 190 0 1799 3 2 0 22 24385 0 689 46381 119 323 1285 17 1072 3 131 178 1 115 108 65535 65535 0 0 0 0
194299  ID:1  30 0 193 0 514 3 2 0 22 24784 0 382 34200 11 230 1285 16 1040 3 131 171 1 108 101 65535 65535 0 0 0 0
195412  ID:1  30 0 194 0 1285 3 1 0 22 24881 0 553 33446 61 270 257 8 512 5 262 178 1 115 108 65535 65535 0 0 0 0
Filter:
```

Figure 3.1: Cooja simulator - 1 sink, 6 senders running ContikiRPL


Sensor Data Collect with Contiki (connected to <stdin>)
File Tools
Nodes
Node Control Sensor Map Network Graph Sensors Network Power Node Info Serial Console
<All>
1.1
2.2
3.3
4.4
5.5
6.6
7.7
ETX
4.4
2.75
5.5
1.0
1.1
2.0
2.13
2.75
2.75
6.6
2.2
7.7
3.3
1.1
7.7
2.13
1.0
2.75
5.5
2.75
4.4
2.0
2.2
2.75
6.6
3.3

Node	Received	Dups	Lost	Hops	Rtmetric	ETX	Churn	Beacon Interval	Reboots	CPU Power	LPM Power	Listen Power	Transmit Power	Power	On-time	Listen Duty Cycle	Transm
1.1	0	0	0	0.000	0.000	0.000	0		0	0.000	0.000	0.000	0.000	0.000		0.000	
2.2	3	0	0	2.000	1141....	2.292	0	1 min, 49 sec	0	0.111	0.160	0.500	0.298	1.070	0 min,...	0.833	
3.3	2	0	0	2.000	1280....	3.000	0	1 min, 05 sec	0	0.153	0.159	0.508	0.580	1.400	0 min,...	0.847	
4.4	3	0	0	1.000	1045....	3.083	0	1 min, 49 sec	0	0.100	0.160	0.444	0.273	0.977	0 min,...	0.739	
5.5	3	0	0	1.000	522.667	1.042	0	2 min, 32 sec	0	0.127	0.160	0.579	0.256	1.122	0 min,...	0.966	
6.6	3	0	0	1.000	1045....	3.083	0	1 min, 16 sec	0	0.111	0.160	0.471	0.332	1.075	0 min,...	0.785	
7.7	3	0	0	2.000	1205....	2.542	0	1 min, 27 sec	0	0.115	0.160	0.495	0.340	1.110	0 min,...	0.824	
Avg	**2.833**	**0.000**	**0.000**	**1.500**	**1040....**	**2.507**	**0.000**	**1 min, 39 sec**	**0.000**	**0.120**	**0.160**	**0.499**	**0.347**	**1.126**	**0 mi...**	**0.832**	

Figure 3.2: Cooja simulator - Sensor Map and Network Graph

3.2. RPL

RPL was developed to be used as a standard routing protocol in WSNs deployed in different environments such as: ”urban networks, smart grid networks, industrial networks, building and home networks”[21]. It is optimised for collection networks (which are based on Multipoint-to-Point (MP2P) and Point-to-Multipoint (P2MP) traffic) with occasional Point-to-Point (P2P) traffic. The collection networks have multiple nodes which report periodically to few collection/sink nodes, and the sink nodes choose to communicate with one of the sender nodes in a P2P fashion. RPL uses MP2P traffic for the process of collecting data (measurements) and uses P2MP traffic for configuration purposes for the nodes that have to report to the sink nodes.

Starting from a node called **Root** (a sink node), RPL builds a **DODAG** (Destination Oriented Directed Acyclic Graph) with the help of the DIO (DODAG Information Object) messages generated by the Root. The DODAG has the role of minimising the cost of reaching the Root from any node in the network. Each node within the DODAG holds a **rank** value which is calculated based on the information received in DIO messages. The rank for a node increases as the node is further away from the Root node. The routing towards the DODAG Root is based on the rank values, as a node roughly picks the neighbour with the lowest rank when sending a packet up the DODAG. Nodes within a RPL network exchange information with the purpose of maintaining the DODAG. The messages that carry DODAG information are called **DIO** (DODAG Information object) messages. (“All the nodes in the network regularly issue a DIO which serves as a heartbeat to indicate their rank. The instant at which a node issues a DIO is governed by a **Trickle** timer which is reset only when an inconsistency arises. This way, when the topology is stable, very few DIOs are exchanged.”[22]) RPL uses another type of messages called **DAO** (Destination Advertisement Object) messages, which are sent from each node in the network towards the Root node. Based on the information contained in these messages (a list of node identifiers from all nodes that were traversed by this message), the Root node knows which nodes a packet has to go through in order to reach a certain destination.

3.2.1. Types of RPL messages

DIO – DODAG Information Object. DIO messages are sent periodically by the Root node and carry information that allows a node to “discover a RPL Instance, learn its configuration parameters, select a DODAG parent set, and maintain the DODAG.”[1] The most important information contained by this message is the DODAG-ID, the Objective Function and the rank of the broadcasting node (which is used by the receiving node to calculate its own rank). The DIO messages are sent periodically with an increasing sequence number in order to trigger the parent selection process (described in Section *3.2.2.1. Upward routing*) that leads to recalculation of the DODAG that further leads to repairing existing broken links.

DIS – DODAG Information Solicitation. It is used to trigger a DIO transmission from a RPL node. This type of message is usually used when nodes join a network. As an alternative to waiting to receive a DIO message, a node can choose to broadcast a DIS message so that other nodes will immediately trigger a DIO transmission upon receiving the DIS message.

DAO – Destination Advertisement Object. DAO is used to propagate information up the DODAG towards the Root node. Depending on the type of routing, DAO messages can be unicast messages sent from children to selected parents (*storing* mode) or unicast messages sent to the DODAG Root (*non-storing* mode). In either case, the DAO messages follow a common structure. The most significant information that is carried by a DAO message is the "Reverse Route Stack". As a DAO message travels up the DODAG towards the Root node, each node that it passes through appends its unique ID to this stack.

DAO ACK – Destination Advertisement Object Acknowledgement. This type of message is optional and is sent as an unicast message to a DAO recipient which can either be a DAO parent or the DODAG Root. **CC – Consistency Check.** "The CC message is used to check secure message counters and issue challenge/responses. A CC message MUST be sent as a secured RPL message."[1]

3.2.2. RPL routing

3.2.2.1. Upward routing

RPL provides routes up the DODAG towards the DODAG Root, constructing a DODAG optimized according to an Objective Function. This is accomplished by sending **DIO** messages from the DODAG Roots **down** the DODAG, towards the leaf nodes. With the help of these messages the DODAGs are formed and maintained. A node is able to join a DODAG by discovering neighbours that are already part of the DODAG of interest and by having built a parent set. The **parent set** represents a subset of the **candidate neighbour set** (which contains the nodes reachable via link-local multicast). Any neighbour with a lower rank than the node itself is considered as a candidate parent. From the parent set, a **preferred parent** is elected which will be used as the next hop for all upward routes. This allows for Multipoint-to-Point communication within a WSN that runs RPL. Regarding the Upward routing process, the construction of the DODAG is of great importance. The process of the DODAG construction is based on the Distributed Algorithm Operation and it involves assigning / configuring some nodes as DODAG Roots. These nodes send DIO messages (link-local multicast) to all the RPL nodes with the intention of advertising their "presence, affiliation with a DODAG, routing cost and related metrics" [1]. The nodes always listen for DIO messages and they use the information provided by these messages to join a new DODAG or maintain the existing one. ("Nodes provision routing table entries, for the destinations specified by the DIO message, via their DODAG parents in the DODAG Version. Nodes that decide to join a DODAG can provision one or more DODAG parents as the next-hop for the default route and a number of other external routes for the associated instance."

3.2.2.2. Downward routing

RPL can also provide routes down the DODAG in order to reach certain nodes (that are not DODAG Roots). This is accomplished by sending **DAO** messages **up** the DODAG towards the DODAG Root. Usually downward routes are Point-to-Multipoint routes (from a DODAG Root towards leaf nodes), but with the help of the DAO messages, **Point-to-Point** routing can also be supported. ("Point-to-Point messages can flow toward a DODAG Root (or a common ancestor) through an upward route, then away from the DODAG Root to a destination through a downward route.") The downward routes can be maintained in two modes, called **"storing"** and **"non-storing"**. In **"storing"** mode, all non-root and non-leaf nodes store a routing table for the downward routes for their sub-DODAG. The routing tables are formed based on the information provided by the DAO messages. When a message is travelling on a downward route in **"storing"** mode, routing decisions (determine the packet's next hop) are taken at every hop (based on each hops' stored routing table). In **"non-storing"** mode, the non-root nodes do not have to store a routing table. Only the DODAG Root has to store and maintain a routing table, thus making it responsible for all downward routing decisions. The messages are routed down the DODAG based on source routes provided by the DODAG Root. This mode is highly dependent on the DODAG Root node (if the Root node fails to store some information, then some of the destinations may not be reached). Point-to-Point routing can be supported by both **"storing"** and **"non-storing"** modes.

3.2.3. Objective Function

The Objective Function (**OF**) indicates the method that must be used to construct the DODAG. Nodes become aware of the OF when receiving a DIO message. ("An OF defines how nodes translate one or more metrics and constraints into a value called Rank, which approximates the node's distance from a DODAG Root. An OF also defines how nodes select parents."[1]) RPL supports a variety of Objective Functions, due to its large number of applications. The OF allows customising the way RPL builds the routing topology based on various metrics and constraints. RPL was developed offering this flexibility in order to be able to better accommodate networks used by a wide range of applications. The Objective Function used by the RPL implementation in the SpeckSim simulator mainly consists in calculating a node’s rank based on simple hop counting.

3.2.4. Trickle timer

(“The Trickle algorithm allows wireless nodes to exchange information in a highly robust, energy efficient, simple, and scalable manner. Dynamically adjusting transmission windows allows Trickle to spread new information on the scale of link-layer transmission times while sending only a few messages per hour when information does not change. A simple suppression mechanism and transmission point selection allows Trickle's communication rate to scale logarithmically with density.

Trickle is an efficient algorithm for propagating data changes in a network. RPL uses Trickle to ensure that nodes in a given neighbourhood have consistent, loop-free routes. The RPL implementation (in SpeckSim) uses the **Trickle** timer for the **DIO** transmissions. The transmission of this type of message is periodic and it is triggered by the Trickle timer. The timer requires setting two intervals between two DIO transmissions: a **minimum interval** and a **maximum interval**. It starts from the minimum interval and with every transmission it doubles its duration until it reaches the maximum interval. When the maximum interval is reached, the transmissions will be maintained at a constant rate (which is dictated by this maximum interval). However, the timer is forced to reset to the minimum interval with any change that occurs in the DODAG structure.

Chapter 4

Results and Critical Analysis

This chapter presents the results gathered from running the tests described in *Chapter 3*, and is structured on the metrics that are evaluated for each of the protocols. After presenting the results and their interpretation for each metric, the chapter concludes with an evaluation of all the tested protocols.

In this section we first evaluate OF0 and ETX in terms of performance metrics of interest. Later we configure a number of RPL parameters (DIO Minimum Interval and Doublings, Duty cycling interval, Frequency of application messages) to observe their cause and effect on performance metrics of interest.

A number of protocol parameters influence the efficiency of RPL; the four significant among them are DIO minimum interval, DIO Doublings, Radio Duty Cycling interval and Frequency of application messages. To observe the effect of these parameters on ContikiRPL performance we use five performance metrics of interest: Convergence Time, Control Traffic overhead, Energy consumption, Network Latency, and Packet delivery ratio (PDR).

In the first phase of the simulations we evaluate the performance of OF0 and ETX in terms of three metrics: Energy consumption, Network Latency and Packet delivery ratio to propose an efficient Objective function. In the second phase we evaluate RPL in terms of five metrics of interest stated above for the preferred OF suggested at first phase.

In the rest of the chapter we describe the performance metrics of interest in section 6.1, that we will use to observe the performance of RPL after changing the parameters. The parameters that impact RPL performance are described in section 6.2. In section 6.3 we describe the Link failure model which is used to simulate the lossy radio medium in Cooja. We perform the simulations in phase 1 and phase 2 in sections 6.4 and 6.5 respectively.

4.1 Performance metrics

We use five standard performance metrics namely Network convergence, Control Traffic overhead, Energy consumption, Packet Latency and PDR to evaluate the performance of RPL.

The first performance metric of interest is *Network Convergence Time*. The nodes in a sensor network need to form a topology in order to communicate. Therefore the network setup time is a crucial metric that need to be evaluated for any routing protocol. The Convergence Time of the RPL DAG is defined as the amount of time needed by all the reachable (in terms of radio) nodes in the network to join the DAG. This convergence should be considered as the initial Convergence Time in a RPL network with static nodes. As the nodes can be mobile and the links are lossy in LLN, the absolute Convergence Time cannot be defined for a mobile network.

The second performance metric is *Control Traffic Overhead* for the network. This includes DIO, DIS and DAO messages generated by each node and it is imperative to confine the Control Traffic keeping in mind the scarce resources in LLN. RPL control the redundant control messages by making use of trickle timers. The aim of this metric is to analyze the effect of the stated parameters on the Control Traffic overhead.

The third performance metric is *Energy Consumption*. To make good energy estimation we use percent radio on time of the radio which dominates the power usage in sensor nodes. Furthermore we take the average percent radio on time for all the nodes in the whole network setup.

The fourth performance metric of interest in the study is *Packet Latency*. The Latency is defined as the amount of time taken by a packet from node to reach the sink and is the average of the latencies of all the packets in the network from all the nodes.

The fifth metric is *Packet Delivery Ratio* (PDR) and is defined as the number of received packets at the sink to the number of sent packets to sink. We take the average PDR of all the packets received successfully at sink.

In the following sections we observe one-by-one the effect of RPL parameters namely: DIO minimum interval, and Doublings, a Duty cycling parameter: ContikiMAC interval and an Application parameter: Frequency of application messages generated; on the performance metrics of interest in a RPL network with lossy environment.

4.2 RPL, Duty-Cycling, Application: Parameters

1. DIO Interval Minimum

This parameter controls the rate of DIO transmission and therefore crucial for Control Traffic overhead, Energy Consumption and Convergence Time etc. The more quickly the DIOs are transmitted the more quickly the network gets converged but at the expense of Control Traffic overhead and more Energy Consumption etc. This parameter is influential on the performance of the whole protocol performance. A careful tweaking of this parameter is necessary for improved performance keeping in view the different application areas of WSN and environmental conditions. In Contiki this parameter is set by RPL_DIO_INTERVAL_MIN.

2. DIO Interval Doublings

This parameter defines the number of times the DIO minimum interval can be doubled and is useful to keep the traffic low for steady network conditions. Its low value can cause the Control Traffic to flow even if not needed. Consequently it's essential to configure this parameter precisely. In Contiki this parameter is set by RPL_DIO_INTERVAL_DOUBLINGS.

3. Duty-Cycling Interval

This parameter is used by the duty-cycling mechanism to set the number of times the medium is sensed in one second for any traffic. In Contiki this parameter is set by NETSTACK_RDC_CHANNEL_CHECK_RATE and its default value is 16 which means the medium is sensed 16 times or after 62.5 ms in each one second interval.

4. Frequency of Application messages

This is the rate at which a node sends application level messages to the sink. The more often the application sends messages the more likely for it to drain the network resources because application packet transmissions takes considerable amount of energy, bandwidth etc for LLN. We tune this parameter by setting SEND_TIME in our sample contiki application.

4.3 Simulation and Network Setup

The experimental process can be divided into four steps, definition, planning, operation and analysis & interpretation [42]. We conduct all the simulations in the respective sections and present the overall analysis from all the phases and simulations in a separate section 7. This makes it easy to understand the overall analysis from these extensive simulations at one place and to provide greater readability at the same time.

In order to conduct a WSN simulation the simulation of losses in wireless medium is very important because it simulates the actual environment where the sensor nodes will work. The more accurate the simulations of the radio medium the more closer are the results to the actual radio medium. In this section we first describe how we simulate the radio medium in Cooja followed by the network setup used for the simulations in this study. After this we explain how we compute the performance metrics from the data that we obtain from these simulations and finally running the simulations in an organized manner.

4.3.1 Link Failure Model

The link failure model is emulated by using Unit Disk Graph Model (UDGM) in Cooja [9]. It uses two different range parameters one for transmission and one for interference with other radios as shown in Figure 6.1

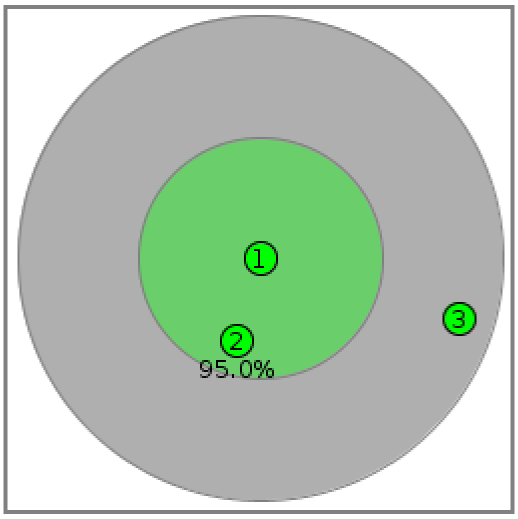


Fig. 4.1: The radio medium is simulated by UDGM in the Simulations using Cooja. The bigger green circle denotes the transmission range (R) of node 1 while the gray circle denotes its collision with other radios. The figure as percentage shows the reception ratio of the transmission between node 1 and 2. The node 3 is inside the collision range of node 1.

Both radio ranges increase with the increase in radio power. Both the transmission and reception ratios can be configured using UDGM. However we keep the transmission ratio to 1 (100%) in these simulations as we are interested in loss ratios at the receiver end only.

The probability of success of packet reception at a node increases as node's distance (D) decreases towards the other node in its transmitting range (R). Thus the minimum probability of success would be at the edge of transmitting range R and equal to RX ratio.

Whereas the probability of success of packet reception at a node at a distance D from another node can be computed as:

Probability of success $= 1 - (D^2 / R^2) * (1 - RX)$ …………………………………… *(Eq.1)*

Where D is the distance between the two nodes and D is less than or equal to R.
R is the reception range and greater than 0.
RX defines the success ratio.

4.3.2 Network Setup

We design a sample network in the Cooja simulator containing 80 client nodes and 1 server node acting as root of the DODAG. The network scenario is shown in Figure 6.2. The server is using a sample application *udp-server.c* while all other nodes are using *udp-client.c*. We use a Cooja plugin called Contiki Test Editor to measure the simulation time and stop the simulation after the specified time. This plugin also creates a log file (COOJA.testlog) for all the outputs from the simulation which we will analyze at the end of the simulation using a Perl script (Appendix C).

In order to introduce lossyness in wireless medium we use the Cooja Unit Disk Graph Medium which introduces lossyness with respect to relative distances of nodes in the Radio Medium as discussed in section 4.3.1. The parameters for the Simulation and its environment are shown in Table 4.1.

As shown in Table 6.1 the start delay is the initial start delay for the application to start transmitting it messages to the sink node. This initial start time is the approximate time sufficient for the initial network convergence. This also ensures that the packet sent to the server will not get lost because of the lack of network connectivity. Therefore a correct evaluation can be performed on the number of packets sent.

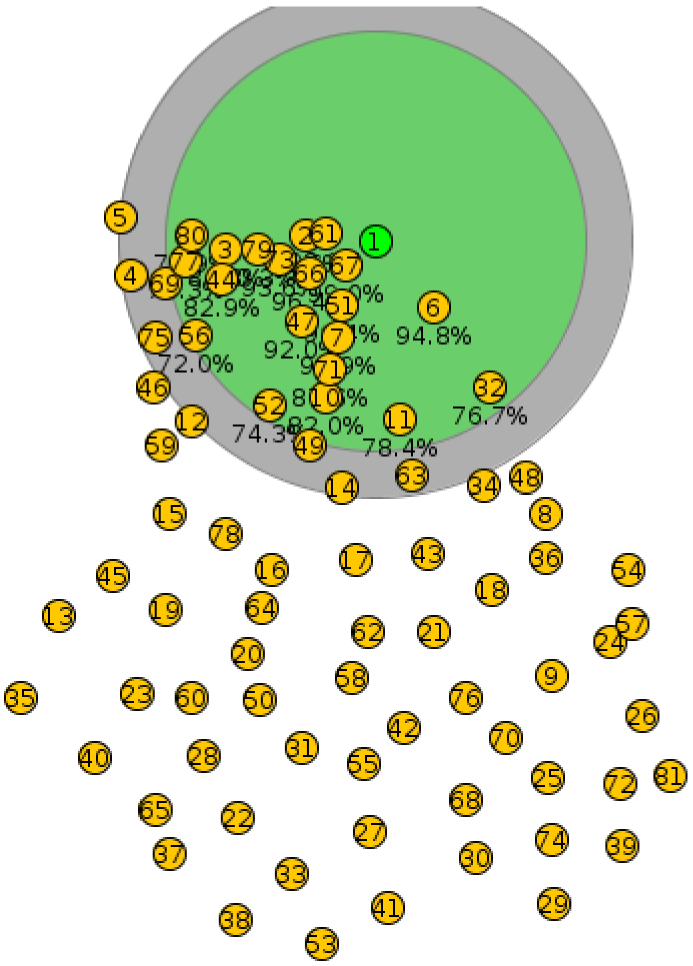


Fig. 4.2: RPL Network Setup of 80 client nodes and 1 sink node in COOJA Simulator, The green node labeled as 1 is the sink, all other nodes with orange circles are the client nodes and send packets to sink. The bigger green circle denotes the area of transmission range where as the gray circle shows the area of radio collision. The figures as percentages show the reception ratio.

Parameters	**Value**
Start Delay	65 s
Randomness	2 s
Total Packets	32115 approx.
RPL MOP	NO_DOWNWARD_RO UTES
OF	OF0, ETX
DIO Min	12
DIO Doublings	8
RDC Chanel Check Rate	16
Send Interval	4 s
RX Ratio	30-100%
TX Ratio	100%
TX Range	50m
Interference Range	55m
Simulation Time	1 Hr
Client Nodes	80

Table 4.1: Simulation Parameters

The send interval represents the interval between two successive application level messages. Both the start delay and send interval have been added time interval randomness. The number of total packets from a node transmits at a rate of 1 packet/(Send Interval± Randomness). So the minimum number of packets sent is 1 packet/(Send Interval) * Simulation Time; whereas the maximum number of packets sent is calculated as 1 packet/(Send Interval+Randamness) * Simulation Time. Since the packet transmission starts after the Start Delay (65s) so the actual Simulation Time will be less by Start Delay (Simulation Time – Start Delay). Since each sensor node will get different randomness therefore the correct number of packets sent cannot be pre-computed but our metric measuring mechanism which is explained next, measures the number of packets sent precisely in real time. This enables the fair computation of packet delivery ratios, Control Traffic overhead.

We set the RPL mode of operation to No Downward routes because we are interested in using the multipoint to point traffic for this evaluation and to limit the scope of this study. DIO Min and DIO Doublings are set to ContikiRPL default values. The reception ratio (RX) represents how lossy is the radio medium and is set in percentages during the successive iterations of the simulation. In the first phase we set it to different levels and thus we observe the performance of OF0 and ETX for different values of lossyness. The transmission ratio (TX) is set to 100% (loss free) because we do not aim to introduce losses at the transmitter end but only at the receiving end. The TX range is set to 50m and interference range to 55m. We run the simulation for 1 hour (3600s) for 80 nodes in the simulation.

4.3.3 Measuring the performance metrics

The network shown in Figure 6.1 has 80 nodes and each node sends a sample UDP packet ("Hello N", where N is the sequence number) to the server (sink) after each Send Interval (10 seconds) and after an initial Start Delay (65 s). The client prints a message 'Hello N sent to Server' as soon as it sends a packet and this enable us to note the sending time. Similarly Server prints a message 'Hello N received from Client M' and this enables us to note the receiving time of the packet at server. The print messages are stored in log file from the simulation. From the sending time and receiving time we calculate the Latency for all the packets.

We read the log file line by line and keep on noting the node id, packet number and sending time in a Send_Table. When we read a line of 'Hello N received from Client M' we look up the Send_Table for node id and packet number to find the send time of that packet. If there is match we compute the Latency and remove the corresponding entry from the Send_Table. All the remaining entries in the Send_Table indicate lost packets. The Network Latency can be computed using the following equation.

$$Total\ Latency\ =\ \sum_{k=1}^{n}(Recv\ Time(k) -\ Sent\ Time(k))$$

Where n is the total number of packets received successfully. The timing information is provided by Cooja Simulator.

To compute the average Latency we divide the Total Latency from Eq.2 by the number of total received packets. The Total number of received packets is counted at the sink node.

Average Latency = Total Latency / Total Packets Received ……………………….… (Eq.3)

To compute the average Packet Delivery Ratio we measure the number of sent packets from all the nodes to the sink and divide it by the number of successfully received packets at the sink.

Average PDR= (Total Packets Received / Total Packets Sent) * 100 ………..…… (Eq.4)

To compute the power consumption we use the mechanism of Powertrace system available in Contiki [37] [17]. Powertrace is a system for network-level power profiling for low-power wireless networks which estimates the energy consumption for CPU processing, packet transmission and listening. This mechanism maintains a table for the time duration a component like CPU, radio transmitter was on. Based on this computation we calculate the percentage of radio on time. We compute the power consumption for radio transmission and listening as these are the most energy consuming components [1].

In order to get the Convergence Time in a RPL network we determine the time for first DIO sent from the client nodes and the last DIO joined the DAG. The Convergence Time is obtained by subtracting first DIO sent time from the time of the last DIO joined DAG.

Convergence Time = Last DIO joined DAG - First DIO sent …………………. (Eq.5)

RPL uses ICMPv6 based control messages called DIS (DODAG Information Solicitation) and DIO (DODAG Information Object) for building and maintaining DODAG. The ICMPv6 is a network layer protocol and therefore we are capturing the control messages from the network level as we are not interested in the application level messages in this experiment.

RPL is composed of several source files in Contiki including rpl-icmp6.c and rpl-dag.c for ICMPv6 control messages and rpl dag creation process respectively.

When a node sends a control message, RPL will print the status message 'DIO sent' and 'DIO joined DAG' to the console. These messages are collected in Cooja log file (COOJA.testlog) and we process it to compute setup time using a Perl script (Appendix C). The interval for sending these messages can be configured via the Contiki parameter RPL_DIO_INTERVAL_MIN.

$$Control\ Traffic\ Overhead\ = \sum_{k=1}^{n} DIO(k) + \sum_{k=1}^{m} DIS(k) + \sum_{k=1}^{o} DAO(k)$$

To compute the Control Traffic overhead each node will print the message DIO/DIS/DAO sent, depending on the type of control message. We collect these control messages per node basis and sum them up for the total RPL network control overhead.

4.3.4 Measuring the Control Traffic

Trickle algorithm is used to limit the number of control packets sent. The algorithm uses two important parameters namely DIO Interval Minimum and DIO Interval Doublings. The DIO Interval Minimum is used to for the initial interval of the control packet transmission where as DIO Interval Doublings is used to place an upper limit on the rate of this transmission. When the RPL network starts the DIOs are transmitted at a rate equal to Imin and this rate is doubled each time DIO is fired until it reaches Imax. When Imax is reached the DIOs are transmitted at the rate equal to Imax. The Imin is determined from DIO Interval Minimum while Imax is determined from DIO Interval Doublings (see section 2.6.6).

The Imax is computed as follows (section 2.6.6).

Imax = Imin * 2 ^ n …………………………………………………...…… (*Eq.7*)

Where n represents DIO Interval Doublings and is the number of times Imin can be doubled

Suppose DIO Interval Minimum is 12 and DIO Interval Doublings is 4 during a simulation of duration SimulationTime (s), then Imin and Imax can be calculated as:

Imax = Imin * 2 ^ DIO Interval Doublings

Imax = 4096 * 2 ^ 4

Imax = 65636 ms

Imax = 65.536 s

Imin = 2 ^ DIO Interval Minimum

Imin = 2 ^ 12

Imin = 4096 ms

Imin = 4.096 s

Suppose DIO Interval Minimum is 12 and DIO Interval Doublings is 4 during a simulation of duration SimulationTime (s), then Imin and Imax can be calculated as:

These two values mean that transmission will start at the rate of 4.096s (Imin) and then it can be doubled 4 times (DIO Interval Doublings or n) before reaching 65.536s (Imax) after which the rest of the transmission will take place at the rate of 65.536s. This process can be portrayed in Figure 6.3.

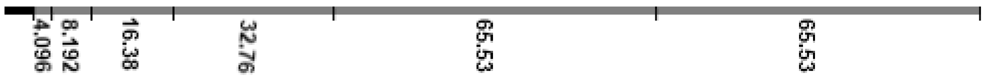


Fig. 4.3: Timeline for DIO transmission using Trickle algorithm.

From Figure 4.3 we can observe that the DIO Interval Minimum can be doubled 4 times (=DIO Doublings) before reaching Imax (65.53s) and this is also equal to the number of packets (n) sent before Imax is reached

From the Eq.7 we can compute the number of DIOs transmitted (= DIO Doublings) before Imax is reached as follows:

Imax = Imin * 2 ^ n

Imax / Imin = 2 ^ n

2 ^ n = Imax / Imin

Taking log of both sides

Log (2 ^ n) = Log (Imax / Imin)

n = Log (Imax / Imin) ……………………………………………...…………(*Eq.8*)

Since n is the number of times the Imin can be doubled therefore it is equivalent to the number of packets sent during the Imin intervals.

After the Imin and Imax are reached the remaining time in Simulation is SimulationTime – Imax - Imin and the remaining number of packets sent during this time can be calculated as:

(SimulationTime – Imax - Imin) / Imax …………..…..…………………..….. (*Eq.9*)

From Eq.8 and Eq.9 we calculate the total number (N) of DIOs sent as follows.

N = Log (Imax / Imin) + (SimulationTime – Imax - Imin) / Imax …….. (*Eq.10*)

Where N represents the number of DIOs sent and Log is Logarithmic function to the base 2.

4.3.5 Running the Simulations

The simulations conducted in this research study not only involve deep analysis of the protocol behavior but repetition of simulations for different values of the parameters and configurations. Running a series of these simulations using Cooja simulator is a time consuming and a mundane activity. The analysis of the effect of different parameters on RPL performance need a large set of data and take significant amount of time to get such data from the simulations to perform reliable and statistically correct results. Due to this reason, the process of executing simulations as well as the processing of log data has been automated.

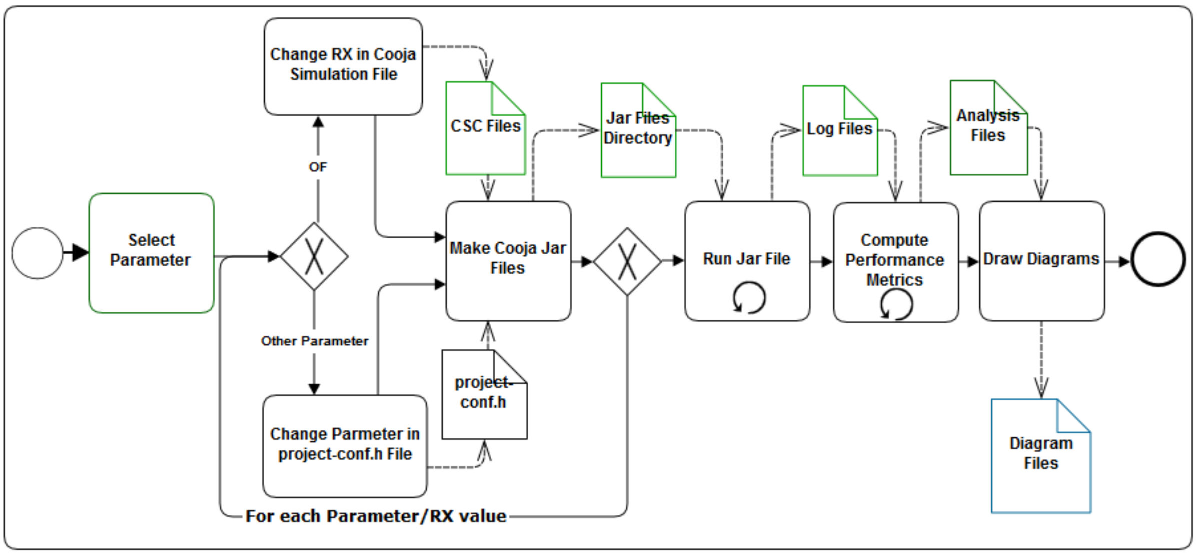


Fig. 4.4: Outline for running the automated simulation scripts

This process is regulated by using Perl scripts by first changing the parameters of the simulation, then building Cooja Simulation Jar Files, then an analysis script computes different performance metrics and finally when all analysis files are complete another Perl script creates diagrams based on outcomes using RRD (round robin database).

Figure 6.3 depicts the simulation and analysis process of the Perl automated tool using Business Process Model and Notation (BPMN) graphical modeling tool. The process starts with the selection of the parameter, if it is OF then the script changes the RX values in the Cooja simulation file and a Cooja Jar file is created iteratively but if the parameter selected is

DIO Interval Minimum or others the script changes the values of the parameter in the project-conf.h file of the Contiki application. The output of both cases yields in Cooja jar files in a separate directory. All the jar files in the directory are executed by the script and COOJA.testlog files are maintained in the respective directories for computing the performance metrics iteratively. The performance metrics are written to an analysis files in text format. The script uses the information in the analysis files and draws the diagrams using open source RRD tool (Documentation is available at

4.4 Phase 1: Comparison of Objective Functions

An Objective Function defines how a RPL node selects the optimized path within a RPL instance based on the routing metrics and constraints. It provides specific optimization criteria like minimize hop count, path ETX, Latency etc. RPL forms Directed Acyclic Graph (DAGs) based on the objective function. The OF guides RPL in selection of the preferred parents and candidate parents. It is also used by RPL to compute the ranks of a node. All upward traffic is forwarded via the preferred parent.

Contiki implements two objective functions OF0 and ETX. Objective Function zero (OF0) select the path to the root with minimum hops. This can be achieved by comparing the ranks of parents. Contiki uses 16 bit rank in units of 256 (min_hoprankinc) which allows a maximum of 255 hops.

The ETX metric of a wireless link is the expected number of transmissions required to successfully transmit a packet on the link. Objective Function ETX uses ETX metric while computing the shortest path.

4.4.1 OF0 vs. ETX

ContikiRPL implements two objective functions and it is essential to evaluate these OFs as the path selected by these OFs may differ in Energy consumption, Network Latency and PDR etc.

The objective of this experiment is to evaluate the two objective functions OF0 and ETX in terms of Energy consumption, Latency and Packet Delivery Ratio of the network for the upward traffic with respect to different levels of lossyness. The upward traffic in a RPL network means traffic from client nodes towards the sink node and is a kind of multipoint to point traffic scenario. We use Cooja Network Simulator for this evaluation which is the most commonly used open source simulator for WSN using Contiki.

The network setup is shown in Figure 6.2 and different parameters are shown in Table 6.1. We repeat the simulation for different RX values ranging (30,40,50- 100%). We set Send Interval to 8s, Start Delay to 65s. We extract the data from the simulation log file using a Perl script (appendix C).

The average values of Packet Latency and PDR are computed using equations Eq.3 and Eq.4 respectively, while Energy consumption is computed using Powertrace mechanism. The power trace mechanism provides CPU and low power mode (lpm) values which gives the total time consumed in each power trace interval. The sum of transmission and reception times gives the time spent by the transceiver. Furthermore we focus on computing the % radio on time which is reproducible for Energy Consumption.

A. Network Latency

The Latency of OF0 is plotted against ETX as shown in Figure 6.5. We can see from the figure that ETX performs better than OF0 because it considers link level details i.e. ETX in computing the best paths. The difference is more obvious in more lossy links but as the lossyness decrease (RX ratio increase) the Latency becomes equal as we can see for RX values 80% and greater.

B. Energy Consumption

The Energy Consumption of the network is more at the start with more lossy network but as we decrease the lossyness in the network the Energy consumption also decreases as shown in the Figure 6.6. This is because the lossy environment has more retransmissions than a lossless environment.

The objective function OF0 selects the shortest path, which is not the best in terms of ETX metric than objective function ETX. As a result, the nodes will experience more retransmissions, which consequently consume more energy.

Since LLN has a lossy nature the rank quality changes frequently and therefore the ETX for the path also changes frequently. ETX takes into account the path ETX for computation and can therefore reflect the link status more precisely at any time in a RPL network.

49

C. Packet Delivery Ratio

The average packet delivery ratio is computed from equation Eq.4. The packet delivery ratio of ETX is slightly better than OF0 in this small sized simulation, which proves the efficiency of ETX.

D. Summary of the outcomes

The average Network Latency of ETX is 0.8s compared to OF 1.0s. This is a considerable difference because the network size is small and the longest route possible can be only 15 nodes while in real network scenarios it can be larger. The PDR of ETX is 92% which is better than OF0 88%. ETX is using energy even more efficiently and the % Radio on time during an hour simulation is only 3.7%
while that of OF0 is 4.7%. We also note that difference in PDR and Latency for both OFs becomes less as the lossyness in the radio medium decreases.
Therefore we conclude that ETX provides better routes than OF0 by taking into account the link characteristics which consequently provides better Network Latency because the ETX selects the best routes which has better path ETX than minimum hop path selected by OF0. The best paths ensure less re-transmissions and radio collisions across the network and this provides better Latency, PDR and Energy consumption.
We conclude this section by proposing ETX as a preferred objective function for most of the scenarios. In the second phase we proceed with the RPL evaluation using objective function ETX.

4.5 Phase 2: RPL Evaluation

As observed in the previous simulations that ETX performs better than OF0. Also ETX is considered the most preferred OF by the research community. Therefore in the rest of the simulations we use objective function ETX for simplicity and to limit the scope of this study. The same evaluation can be done for OF0 as well in future studies.
6.5.1 DIO Interval Minimum
The Objective of this experiment is to analyze the effect of DIO Minimum Interval on RPL performance metrics of interest. The order of these metrics is important as Convergence Time and Control Traffic overhead directly affects Energy consumption whereas Network Latency and Packet Delivery ratio are considered next.
We use the same experimental setup as explained in section 6.3 and shown in Figure 6.2 which consists of 80 client nodes and one server node in the Cooja Simulator. The simulation parameters used in this simulation are shown in Table 6.1. We change the DIO values to (3, 4-16) in the subsequent iterations of the simulation and set RX ratio to 70%. We change Start Delay to 3s and Send Interval to 4s with Randomness of 2 s.
A. Network Convergence

As we increase the Interval, the setup time also increases and for DIO Interval Min between 3 and 9 it is approximately equal. For n = 15 and greater it becomes large and thus unacceptable in some applications and in case of radio duty cycling.

The 'First DIO sent' time (for DIO Interval Min=14, 15, 16) is also worth noticeably large because the DIS is sent after about 5s and it means that the nodes should start joining the DAG soon after 5s. But the first DIO is sent after about 16s, 32s and 62s for DIO Interval Min=14, 15, 16 respectively. Even if the nodes request explicitly for a DIO by generating DIS messages the neighbor nodes may not have joined the DAG yet and so cannot provide DIO to their neighbor. This behavior clearly depicts the importance of DIO minimum interval parameter on the network convergence. A suitable calibration of this parameter is important for any network otherwise the network setup time may rise to about 150 seconds as shown in the Figure 6.7.

Another good observation is that the Convergence Time is very fast (less than 10s) for all values of DIO minimum interval in the range (3-13). We suggest a recommended value of DIO minimum interval among these values (3-13) provided the other performance metrics meet that we study and analyze in the following sections (*OBS1: DIO Interval Min.Convergence Time*).

B. RPL Control Overhead

The Control Traffic overhead is very high for lower DIO Interval Minimum (lower than 9) as shown in Figure 6.8. These values yield intervals (Imin) of less than 1 second which enables the trickle timer to fire the DIO very quickly (to compute Imin see section 2.6.7).

We also observe that the control overhead is lower for DIO Interval Minimum of 9 (4700 Packets) which decrease further to 1200 Packets for Interval of 16. We deduct a subset of DIO interval Min in the range (8-16) is the most optimum set for Control Traffic overhead (*OBS2: DIO Interval Min.Control Traffic Overhead*).

C. Energy Consumption

To compute the Energy Consumption by the network we use the Powertrace tool [17], [37]. This tool uses power state tracking to estimate the power consumption of the system for activities such as packet transmission and receptions.

In this experiment we only compute the power consumption by the transceiver part which is the most power consuming component [1]; the CPU power consumption is very low and can be ignored for simplicity reasons in this simulation. We use the ContikiMAC duty cycling mechanism for these simulations, which is the default radio duty cycling mechanism in Contiki. It is a type of sampled listening technique discussed in section 3.5.

The RPL network consumes a lot of energy about 7% Radio ON time when DIO interval is very small (i.e. 3) as shown in the Figure 6.9. DIO Interval Minimum of 3 generates about 146000 control packets during this simulation of one hour as shown in Figure 6.8. But when we increase the DIO Min interval to 4 the Energy Consumption becomes less as the control packets generated now is about 77000 and Radio % ON time reduces to about 3%. The quick generation of control packets not only increases control packets but it also increases collisions and therefore more retransmissions are required to successfully send a packet and the Energy Consumption increases consequently. For DIO interval minimum, between

5 and 7 the Energy Consumption keeps on decreasing in a linear fashion. However the best Energy Consumption can be observed for DIO Interval Minimum of 8 and higher where the Energy Consumption is about 1.5% and remains constant for all values of DIO Interval Minimum of 8 and higher. This means that optimum value for DIO Interval Minimum regarding Energy Consumption is between (8-16) and a single value among it is subjected to fulfilling other performance metrics. (OBS3: DIO Interval Min. Energy Consumption).

D. Latency

The average Network Latency decreases from 3.5s to 1.7s for DIO Interval Minimum between 3 and 6 respectively. This sudden decrease is due to the reason that lower values DIO Interval Minimum generated heavy Control Traffic as we observed in the previous section (B).

The heavy Control Traffic causes more radio collisions and the packets need to be buffered before the radio collision resolves which increase the overall Latency of the packet along the path. As soon as the Control Traffic decreases for DIO Interval Minimum of higher than 8 (OBS2) the packet buffering decreases and radio collision also decrease and as a result the packet reaches the destination relatively quickly than before. The average Network Latency is lower than 0.5s for DIO Interval Minimum from (8-16) as shown in Figure 6.10 (OBS4: DIO Interval Min.Latency).

E. Packet Delivery Ratio

Packet delivery ratio is used by wireless sensor network to compute the best route, optimum transmission rate and power consumption [18], [44], [45].

In Figure 6.10 the PDR is below 85% at the beginning, for DIO Minimum interval (3-5) which means due to high control overhead the RPL network suffer collisions and therefore the PDR is poor. However as we increase the DIO interval (6-14) RPL provides a good Packet Delivery ratio of more than 96%. We can also observe that PDR falls for DIO Interval Minimum of 15 and greater. The reason is that the value of DIO minimum interval higher than 14 does not provide a quick network convergence as we can see from the result obtained from previous section (OBS2:); Consequently the network is not converged fully for DIO Minimum interval of 15 and higher and as a result incurring packet loss to some of the destinations in the network.

We conclude that to achieve a high PDR the recommended DIO minimum interval is between [6...14] inclusively. (OBS5: DIO Interval Min.PDR)

F. Summary of the Outcomes

The trickle algorithm was designed to limit the number of control packets transmitted in LLN and to use the limited resources in LLN more optimistically. This process is controlled by several parameters and DIO Interval Minimum is one of them and is the most influential since it controls the generation of Control Traffic (DIO) directly. The tweaking of this parameter causes a tradeoff between the performance metrics and we observed carefully the effect of every possible change of this parameter on the performance metrics of interest. We summarize the observations (OBS1-OBS5) made in this section in Table 6.2. As we observed the tradeoff among the performance metrics; for instance to achieve low Convergence Time the Control Traffic increases therefore a subset of these values is required which optimize all the five performance metrics equally.

Performance Metric	**DIO Interval Min**
Convergence Time	3-13
Control Traffic	8-16
Energy Consumption	8-16
Latency	8-16
PDR	6-14

Table 4.2: Recommended values for DIO Interval Min for different Performance metrics

As shown in Table 4.2 the lower value (3) of DIO Interval Minimum provides fast Convergence Time but this value is not recommended in case of Control Traffic overhead which seeks a value at the higher end. So the intersection of these two sets (3-13) and (8-16) gives a set of (8-13) that optimizes both the performance metrics Similarly its intersection with set of Energy Consumption (8-16), Latency (8-16) and PDR (6-14) gives a final set of (8-13). We conclude this section by recommending a DIO interval Minimum in the range between (8–13) as this range satisfies all the metrics of interest (OBS: DIO Interval Minimum).

4.5.2 DIO Interval Doublings

The objective of this simulation is to evaluate the performance of RPL for different values of DIO Doublings. The network scenario consists of 80 udp client nodes which send Hello packets to sink node after a random time period between 8 and 10 seconds as shown in Figure 6.2. The simulation parameters are shown in the Table 6.1 however we set DIO Doublings in the range (3-16) for each iteration of the simulation. We choose DIO Interval Minimum of 12 from the recommended values in the previous section (OBS: DIO Interval Minimum) and it gives Imin of 4.096s.

A. Network Convergence Time

After changing the DIO Doublings interval in the range 3 to 16, we observe that the network setup time is not affected and the network setup time is about 19 seconds in all cases as shown in Figure 6.12.

The maximum interval between two successive DIOs is computed (see section 2.6.7) as:

Imax = Imin * 2 ^ DIO Doublings

This implies that the minimum Imax in this simulation will be for DIO Doublings of 3 and computed as:

Imax = 4.096 * 2 ^3 = 32.7 s

This Imax time is larger than the network Convergence Time of 19s as shown in Figure 6.13 and the network will get converged before reaching even the minimum

Imax value and hence has no effect on network Convergence Time (*OBS1: DIO Interval Doublings.Convergence Time*).

B. Control Traffic Overhead

The Control Traffic overhead of RPL decreases as we increase DIO Doublings and becomes nearly equal for values higher than 8 as shown in Figure 6.13. The maximum overhead is about 9955 control packets for DIO Doublings of 3 during a simulation of 1 hour. The Imax for DIO Doublings of 3 equals 32.7 s (section 2.6.7) which is the minimum interval between two successive DIOs in steady state network conditions.

If the network is steady DIO Doublings of 3 will generate about 111 (*Eq.10*) control packets (DIOs) per node during a simulation of one hour which yield a total of 111 * 80 nodes = 8880 packets (Computed) for the whole network. However from Figure 6.14 we get Control Traffic of 9955 which contains 367 DAO and 61 DIS as well. So the number of DIOs from simulations is 9955-367-61 = 9527 (Simulation). The difference between the computed and the simulated value is 647. These 647 DIOs are either lost (or retransmitted) or due to topology changes the DIO trickle timer has been reset and DIOs are transmitted more quickly with the rate of Imin.

Since the DIO Doublings of 8 and higher generate very low extra Control Traffic overhead and can be observed in Figure 6.14, so the recommended value for this parameter is between 7 and 16 (*OBS2: DIO Interval Doublings.Control Traffic Overhead*).

C. Energy Consumption

The energy estimation is affected by the number of packets (both control packets and application messages) because the transceiver needs to remain ON for the transmission of these packets. The more transmission (and re-transmission) of packets across the network cause more Energy Consumption per unit time. The frequency of application messages can be controlled by application programmer depending on the application type and requirements however RPL is responsible for controlling and limiting the frequency of control packets transmission.

The DIO Doublings limit the number of times the DIO Minimum interval can be doubled. For a small value of DIO Doublings the DIO need to be transmitted more often even in steady network situations, but a very large value of DIO Doublings can also cause the network to remain inconsistent for long durations, therefore a minimum value of DIO Doublings causing less Energy Consumption is required. The energy estimation (or % Radio ON time) decreases as we increase DIO Doublings but at value of 10 and higher it remains almost the same as shown in Figure 6.15 (*OBS3: DIO Interval Doublings.Energy Consumption*).

D. Latency

There is not a considerable change in Network Latency for changing DIO Doublings in the range (4-16). The parameter DIO Doublings impact the number of control packets transmitted. The Latency for DIO Doublings 3 is very high (1.8 s) but decreases quickly for 4 and higher. The DIO Doublings 3 gives Imax of 32s which makes DIO Interval Minimum to stop doubling and the DIOs are transmitted after this time which increases Control Traffic and hence the overall network delay increase but DIO Doublings 4 the Imax is about 65s the DIOs transmitted is relative less and does not create congestion that increase packet delay across the network. Therefore DIO Doublings does not make considerable change in the Latency in a simulation of one hour and consisting of 80 nodes as shown in Figure 6.16 (*OBS4: DIO Interval Doublings.Latency*).

60

network delay increase but DIO Doublings 4 the Imax is about 65s the DIOs transmitted is relative less and does not create congestion that increase packet delay across the network. Therefore DIO Doublings does not make considerable change in the Latency in a simulation of one hour and consisting of 80 nodes as shown in Figure 6.16 (*OBS4: DIO Interval Doublings.Latency*).

E. Packet Delivery Ratio

The packet deliver ratio fluctuates between 99% and 99.4% for DIO Doublings in the range (3-16) which is not a considerable change. The ContikiRPL has good support to handle control packet transmission of about 9955 packets in one hour simulation of 80 nodes without affecting the PDR value (*OBS5: DIO Interval Doublings.PDR*).

61

F. Summary of the Outcomes

The outcomes (OBS1-OBS5) from this section are summarized in Table 6.3

Performance Metric	**DIO Interval Doublings**
Convergence Time	Any
Control Traffic	7-16
Energy Consumption	10-16
Latency	Any
PDR	Any

Table 4.3: Recommended values for DIO Interval Doublings for different Performance metrics

We conclude this section by recommending an intersection of the values in Table 6.3 which comes out to be (10-16) for DIO Interval Doublings and it optimizes all the performance metrics of interest (OBS: DIO Interval Doublings).

4.5.3 Duty-Cycling Interval

The duty cycling mechanisms controls the number of times the medium is sensed for a potential packet in a second. In this simulation we use the same RPL network scenario as shown in Figure 6.2. The parameters of the simulation are shown in Table 6.1. The RDC channel check rate can be set in powers of two only and hence we set it in the range 2,4,8,16,32,64,128,256.

A. Network Convergence Time

The network Convergence Time decrease as we increase the Radio Duty Cycling channel check rate.

There is a difference of about 17s between channel check rate of 2 and 8 as shown in Figure 6.18. For channel check rate higher than 8 the network setup time is almost the same which equals about 15s for the network of 80 nodes with 70% RX ratio. So the set of values of RDC channel check rate which optimizes Network setup time is (8-256) provided the other performance metrics also meet (OBS1: RDC.Convergence Time).

B. Control Traffic Overhead

The Control Traffic overhead remains linear for radio duty cycling channel check rate for 2,4,8,32,64 and 128 but suddenly increases for channel check rate of 256 and higher which is a high value for most of the LLN devices and should be avoided as long as possible because it will drain out the scarce resource quickly. By keeping the radio duty cycling high the radio transmitter remains ON for longer time which cause heavy radio collisions and packet loss and hence the overall Control Traffic increases. A RDC check rate of 256 and higher also means that the radio duty cycling becomes ineffective because the large number of channel sense rates and is nearly equivalent of no-duty cycling. The optimum set of RDC values is (2-64) which optimizes Control Traffic overhead (OBS2: RDC.Control Traffic overhead).

Chapter 5

DISCUSSION

The results of the five experiments are obtained, analyzed, and organized in a logical manner to provide results in support of answering the research questions. The performance of the protocol is influenced by several factors including both external environmental factors and internal protocol parameters. In this study we studied both the external factors and internal.

At first phase the effect of lossyness was investigated, which is caused by several environmental factors discussed in section 6.4. We evaluated the performance of OF0 and ETX and compared them for several levels of lossyness.

The LLN is lossy and vulnerable to different interferences which mean an efficient OF is required to handle all the interferences efficiently and utilize the scarce resources more optimally at the same time.

From the set of experiments in the first phase (section 6.4) we observed that ETX provides best paths which improve Network Latency, Energy consumption and Packet delivery ratio. The average Network Latency of ETX is 0.8s which is better than OF0 (1.0s) in the sample network of 80 nodes with depth of about 15 nodes only but we expect that this difference will be more prominent in larger networks of hundreds and thousands nodes. ETX provides an average PDR of 92% whereas OF0 provided 88%.

The Energy Consumption of ETX is 3.7 (% Radio ON time) which is much better than OF 4.7 (% Radio ON time) and so ETX provides longer life for the sensor nodes than OF0. Therefore we infer that ETX can handle the adverse environmental conditions more intelligently than OF0 and this answer our RQ1.

Many applications can benefit from using ETX without utilizing any superfluous LLN resources. However OF0 can also be used in less lossy environments as we observed that the performance of both OFs is nearly the same as the lossyness of the medium decreases.

In the second phase, we evaluated ContikiRPL in terms of several performance metrics of interest for objective function ETX. We illustrated step by step and in a systematic manner the effect of several RPL parameters on RPL performance. We fed the output of one simulation to another in an attempt of suggesting the best combination of parameters. This extensive simulation exercise is strongly appealed by the vast application areas of Sensor networks.

Routing is an important component in sensor networks because it performs the packet forwarding and routing decisions and therefore accounts for the utilization of the network resources. The worst routes cause more retransmissions and wastage of resources.

The performance of RPL is controlled by several parameters both directly related to RPL like DIO intervals, and parameters that effect RPL performance like RDC channel check rate and frequency of application messages. We observed that performance of RPL can be efficiently optimized by tweaking the parameters. The DIO Interval Minimum combined with RDC channel check rate and Frequency of application messages can cause a very quick network convergence, and we observed in our simulations that it can be as efficient as 14s. This is a very useful outcome and many applications will observe no delay as soon as the sensor nodes turn on.

The Control Traffic overhead is heavily affected by DIO interval minimum, RDC channel check rate and a very high frequency of application messages. The more frequent recurring values of these parameters cause packet collisions and packet loss due to more radio on times by the individual nodes. So any extra radio on time by the improper configuration of these parameters is harmful for the Control Traffic overhead. A careful configuration of these parameters keep the Control Traffic overhead less than 1000 control packets / hour in our simulations.

Similarly Energy consumption is also affected by the entire four parameters DIO interval minimum, DIO Interval Doublings, RDC channel check rate and Frequency of application messages. Our simulation shows that it can be as less as 1% for packet interval of more than 8s.

The Network Latency is directly affected by both DIO Interval Minimum and RDC channel check rate and suitable values of these parameters can make it as efficient as less than 0.5 s for the entire network in our simulations.

The PDR is affected considerably by DIO interval minimum, RDC channel check rate and Frequency of application messages and a value of more than 90% in a congested sensor network can be obtained by limiting the application messages.

RPL is the new standardized IPv6 routing protocol by IETF for LLN. It supports both upward routing and downward routing. It has been made very flexible to adopt the vast application areas of WSN. The core responsibility of RPL as a routing protocol is to provide the best paths and to utilize the scarce resources of WSN efficiently. The routing logic has been separated from the routing in the form of Objective Function to provide the flexibility of adding it according to the routing requirements and metrics. OF0 is not the recommended OF because it takes into account hop count for making routing decisions but ETX takes into account link status and can choose better paths.

Chapter6

CONCLUSION AND FUTURE WORK

Wireless network has several constraints like energy, bandwidth, computing and memory, which make routing in these devices more challenging. RPL being a new proposed protocol for low power and lossy network is under experimentation and important aspects are needed to be evaluated. It is a major component of consuming the energy of sensor nodes.
RPL selects routes based on the routing metrics and the objective functions.

The two objective functions in Contiki are OF0 and ETX uses hops count and ETX metric to compute the best paths. Simulation results show that ETX performs better than OF0 as it considers the link statistics. This performance is more visible in more lossy environments.
We also observed that a set of RPL parameters are crucial for its better performance in vast areas of sensors applications. Trickle timer is an important component of RPL in order to utilize the scarce resources of WSN more efficiently. The trickle timer controls the emission of Control Traffic by using two important parameters DIO Minimum Interval and DIO Doublings. A scaled set of these parameters provides a very quick Network convergence, Energy efficiency, decreased Control Traffic overhead, lower Latency and high PDR.
Although Contiki provides very good support for high traffic density, which can be observed in the form of lower packet loss but unnecessary application messages consumes resources like energy, Control Traffic overhead. Extra care is required to limit the application messages and save the limited resources of WSN.

Radio Duty Cycling has direct impact of RPL performance and high duty cycling is unnecessary in a WSN. The ContikiMAC radio duty cycling configured with a very lower rate enables RPL to provide good energy efficiency, lower Control Traffic overhead and better PDR.

BIBLIOGRAPHY

[1] T. Winter (Ed.), P. Thubert (Ed.), and the ROLL Team. RPL, Internet-Draft, “RPL: IPv6 Routing Protocol for Low power and Lossy Networks”, draft-ietf-roll-rpl-19, March 13, 2011

[2] AKKAYA, K., AND YOUNIS, M. A Survey on Routing Protocols in Wireless Sensor Networks. In Ad-hoc Networks (2005)

[3] Nicolas Tsiftes , Joakim Eriksson , Adam Dunkels, Low-power wireless IPv6 routing with ContikiRPL, Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, April 12-16, 2010, Stockholm, Sweden

[4] AL-KARAKI, J. N., AND KAMAL, A. E. Routing Techniques in Wireless Sensor Networks: A Survey. IEEE Wireless Communications (December 2004), 6–28

[5] Siddhu Warrier, Characteristics and Applications of MANET Routing Algorithms in Wireless Sensor Networks, 2007

[6] Networking Working Group, Internet-Draft, “The Trickle Algorithm”, draft-ietf-roll-trickle-01, January 10, 2011

[7] AKYILDIZ, I. F., SU, W., SANKARASUBRMANIAN, Y., AND CAYIRCI, E. A Survey on Sensor Networks. IEEE Communications Magazine (August 2002),

[8] C.S. Raghavendra, Krishna M. Sivalingam and Taieb Znati, Wireless Sensor Networks, 2004

[9] SpeckNET. SpeckSm. http://www.specknet.org/dev/specksim, 2011 [Online; accessed 13-August-2011]

[10] Charles Waltner, Smart Buildings Offering Clever Ways to Reduce Energy Consumption, July 21, 2008, http://newsroom.cisco.com/dlls/2008/ts_072108.html [Online; accessed 13-August-2011]

[11] FIRE/STREP ICT – 257466, HOBNET, Holistic Platform Design for Smart Buildings of the Future Internet, Deliverable D1.1, Scenario Analysis Report, http://www.hobnet-project.eu/files/D1.1.PDF [Online; accessed 13-August-2011]

[12] http://www.smart-buildings.com [Online; accessed 13-August-2011]

Appendix

How to use the Cooja Simulator

How to start the Cooja Simulator

1. Open an Ubuntu console/terminal
2. Go to the following path: " *~/contiki-2.x/tools/cooja* "
3. Execute the "*ant run*" command

Once the simulator has started, it is possible to create a new simulation by accessing the *File* menu.To add nodes into the simulation: Mote – Add mote of type (and choose a mote type). Create and add RPL nodes in the simulation Access the upper-left menu bar: Mote Types – Create mote type – Sky Mote Type. This will trigger the appearance of a pop-up menu. In the "*Contiki process / Firmware*" field of the pop-up menu, it has to be chosen either the "*udp-sender.c*" file (for creating a RPL router or leaf) or "*udp-sink.c*" (for creating a RPL Root) file from the "*rpl-collect*" project that can be found at the following path "*~/contiki-2.x/examples/ipv6/rpl-collect*". After selecting one of these files, it needs to be compiled by using the "*Compile*" button, and if the compilation is successful, a "*Create*" button will appear that can be used to import the desired type of speck into the simulator. The next step is to choose the number of the selected type of sensors and the type of positioning and the position interval of the sensors.