République Tunisienne
Ministère de l'Enseignement Supérieur,
De la Recherche Scientifique
et de la Technologie

Université de Sfax
École Nationale d'Ingénieurs de Sfax

Ecole Doctorale
Sciences et Technologies

Mémoire de *MASTERE*
NTSID

N° d'ordre: 2011– 152/11

MEMOIRE

Presented in

National School of Engineers of Sfax

Submitted in partial fulfillment of the requirements for the degree of

MASTER

NTSID

Prepared by

Rihab CHAARI
(Software Engineer)

Performance Evaluation of 6LoWPAN Protocol

Defended the 03 December 2011, in the presence of jury:

Mr.	**Mohamed ABID**	*President*
Mr.	**Wassef LOUATI**	*Referee*
Mr.	**Anis KOUBAA**	*Advisor*
Mr.	**Mohsen ROUACHED**	*Advisor*
Mrs.	**Olfa GADDOUR**	*Co-Advisor*

DEDICATION

I dedicate this work to my parents. My father, he did not only raise and nurture me but also taxed himself dearly over the years for my education and intellectual development. Incidentally he met his demise when I was preparing my master thesis on 13 August, 2011. I hope this would make him proud of me. My mother, she has given me so much, thanks for her faith in me, and for teaching me that I should never surrender. She has been a source of motivation and strength during moments of despair and discouragement. Her motherly care and support have been shown in incredible ways recently.

République Tunisienne
Ministère de l'Enseignement Supérieur,
De la Recherche Scientifique
et de la Technologie

Université de Sfax
École Nationale d'Ingénieurs de Sfax

Ecole Doctorale
Sciences et Technologies

Mémoire de *MASTERE*
NTSID

N° d'ordre: 2011– 152/11

PERFORMANCE EVALUATION OF 6LoWPAN PROTOCOL

Rihab CHAARI

الخلاصة: البروتوكول LoWPAN6 يمثل تكنولوجيا رئيسية للانترنت الجديدة. تصميم بروتوكول التوجيه لLoWPAN6 اصبح ذا اهمية قصوى. RPL يضهر باعتباره مشروع شبكة الانترنت التي اقترحتها مجموعة عمل IETF. RPL RPLلا يزال قيد التطوير، على الرغم من وجود اكتسبت النضج ، ومفتوحة لإدخال تحسينات. في الواقع ، هناك حاجة لفهم سلوكها جيدا، والتحقيق في أهميته. في هذه الأطروحة، وسوف نعرض أولا العمل الذي قمنا به لتصميم Monitor-Z اداة تدعم البروتوكولات RPL و LoWPAN6. وسوف نقدم بعد ذلك تقييم الأداء للبروتوكول التوجيه RPL لأنها تمثل المرشح الرئيسي للبوصفها بروتوكول التوجيه القياسية للشبكات LoWPAN6.

Résumé : le protocole 6LoWPAN représente une technologie clé pour le nouvel Internet. Concevoir un protocole de routage pour 6LoWPAN devient d'importance primordiale. RPL apparaisse comme une ébauche d'Internet proposée par le groupe de travail ROLL. RPL est en cours de développement, bien qu'après avoir gagné la maturité, et est ouvert d'améliorations. En effet, il y a un besoin de comprendre bien son comportement, et étudier sa pertinence. Dans cette thèse, nous présenterons d'abord le travail que nous avons fait pour concevoir l'outil Z-Monitor qui supporte les protocoles 6LoWPAN et RPL. Nous présenterons ensuite une évaluation des performances du protocole de routage RPL comme il représente le candidat principal pour agir en tant qu'un standard de routage pour les réseaux 6LoWPAN.

Abstract: 6LoWPAN protocol represents a key technology for the new Internet. Designing a routing protocol for 6LoWPAN becomes of a paramount importance. RPL shows up as an Internet draft proposed by the IETF ROLL working group. RPL is still under development, although having gained maturity, and is open to improvements. Indeed, there is a need to understand well its behavior, and investigate its relevance. In this thesis, we will first present the work we have done to design Z-Monitor monitoring tool that supports 6LoWPAN and RPL protocols. We will then present a performance evaluation of the RPL routing protocol as it represents the main candidate for acting as the standard routing protocol for 6LoWPAN networks.

المفاتيح: التوجيه, تقييم الاداء, RPL, LoWPAN6, Monitor-Z, Telosb.

Mots clés: 6LoWPAN, RPL, routage, Z-Monitor, évaluation de performance, Telosb.

Key-words: 6LoWPAN, RPL, routing, Z-Monitor, performance evaluation, TelosB.

ACKNOWLEDGMENTS

First of all, I thank Allah for giving me strength and ability to complete this master thesis.

I would like to express my gratitude to my supervisor Dr. Anis KOUBAA for his outstanding supervision, counsel, advice, support, inspiration, patience, exceptional seriousness and for always being available during the course of this work.

A special thanks to Mrs. Olfa GADDOUR and Dr. Mohsen ROUACHED for the interest they showed toward this work, their valuable suggestions and fruitful discussions.

My sincere gratitude goes to Professor Mohamed ABID, for giving me the honor of being the president of my Master jury. I also appreciate his kind acceptance in his research laboratory.

Furthermore, I deeply thank Dr. Wasef LOUATI, for accepting the reviewing and the judgment of my Master work.

I would like also to thank Nada and Hanan from Al-Imam Mohamed bin Saud University. I greatly appreciate the cooperation and the help they presented to accomplish the release of the project in the best condition.

Finally, I deeply thank all my friends and the people in the CES laboratory at the National Engineering School of Sfax for their support and enthusiasm that makes working in CES very stimulating and challenging.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Contents

1.1 Motivation

In recent years, the technology of Wireless Sensor Networks (WSNs) has grown thanks to the extraordinary rate of developments in communication and networking technologies. They are considered as the most important technologies in the 21^{st} century. Most of the researchers and technological analysts believe that WSNs have become an integral part of our daily life. They are deployed in factories for condition based maintenance, in emergent environments for search and rescue, in buildings for infrastructure health monitoring and even in bodies for patient monitoring.

With the emergence of wireless sensor networks, new themes have been opened and new challenges have emerged to meet the needs of individuals and the requirements of several application areas. One of the most important problems that must be overcome is integration with Internet. Nevertheless, the integration of IP in WSNs will offer several advantages for many applications. With more and more successful real environment WSN deployments on one hand, and the ubiquitous of IP networks on the other hand, it is natural to try to figure out some approaches to connect the WSN to the IP network infrastructure.

Most of the WSN implements IEEE 802.15.4 protocol for PHY and MAC layers, which specifies a wireless link for LoW-power Personal Area Networks (LoWPANs). Accordingly, the capabilities of the IEEE 802.15.4 protocol are very limited: low bandwidth, small frame sizes and low transmit power. Regarding these constraints, IPv6 protocol is too memory intensive to fit into IEEE 802.15.4 frames since it requires the support of packet size much larger than the largest IEEE 802.15.4 frame. Internet Engineering Task Force (IETF)[11] defined a specification to apply IP into WSN called 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks). 6LoWPAN protocol represents a key technology for the new Internet where trillion of tiny devices are expected to operate in the same network. 6LoWPAN technology has gained more and more interest by the research community.

Since the release of 6LoWPAN, routing has been considered as the main research and development challenge. In fact, several endeavors for specifying an efficient routing protocol for 6LoWPAN-compliant Low-power and Lossy Networks (LLNs) have been driven, such as Hydro [49], Hilow [50], and Dymolow [51]. Recently, the IETF ROLL working group [52] came up with the IPv6 Routing Protocol for Low power and Lossy Networks (*RPL*) routing protocol in an attempt to standardize the routing process for LLNs. The inherent LLNs of low data rates, high probability of node and link failures, and scare energy resources, have turned the design of RPL challenging and different from previous routing proposals. Moreover, *RPL* is still under development, although having gained maturity, and is open to improvements. Indeed, there is a need to understand well its behavior, and investigate its relevance. In the literature, several studies have evaluated the performance of the *RPL* protocol by using simulating tools such as in [41], [40], [42] and [43]. Despite simulation can give a good level of confidence about the protocol behavior, it cannot prove that the protocol operates correctly in all cases. In addition, it cannot describe the real protocol behavior against link or node failures. Therefore, it was necessary to evaluate the performance of *RPL* in real implementation.

1.2 Objectives

The main objective of this master project is to evaluate the behavior of 6LoWPAN networks in real implementation. We aim mainly to evaluate the performance of the *RPL* routing protocol as it represents the main candidate for acting as the routing protocol for 6LoWPAN networks. The evaluation will be performed for different network settings to understand the impact of the protocol attributes on the network formation performance, namely in terms of energy, storage overhead, communication overhead, network convergence time and maximum hop count. This work is motivated by the fact that simulation links doesn't reflect an important aspect of reality. They can't give insight of protocols operating characteristics which may confuse protocol's developers/analyzers.

To understand the behavior and find out the limitations of the protocols, it is necessary to have a powerful network monitoring and protocol analysis at hand. There exist a few solutions for LoWPAN monitoring, but they are either very expensive or of proprietary nature. Therefore, we aim in a first step to design a monitoring tool. Z-Monitor [30], is an open source software for monitoring IEEE 802.15.4-based networks, does not require special sniffer hardware and is easy to extend. However, it does not support many IPv6 based protocols such as 6LoWPAN and RPL.

1.3 Contributions

The contributions of this Master Thesis are both practical and theoretical:

- The first contribution is practical. It consists on implementing a monitoring tool that supports decoding of both 6LoWPAN and RPL protocols. We have extended Z-Monitor to support 6LoWPAN and RPL protocols for both operating systems Contiki and TinyOS.
- The second contribution is theoretical. It consists on conducting a set of experiments for evaluating the performance of protocols already implemented in the first contribution to understand their behavior.
- The third contribution is to experimentally evaluate the performance of the RPL routing protocol under different network settings. The choice of RPL is justified by the fact that this routing protocol was designed to meet the requirements of Low power and Lossy Networks compliant with the 6LoWPAN protocol.

1.4 Research Context

This Master project was developed within the CES laboratory at ENIS (University of Sfax), in collaboration with (1) COINS Research Unit at CCIS/ IMAMU (College of Computer and Information Sciences / Al-Imam Mohamed bin Saud University) (2) CISTER Research Unit at ISEP/IPP (Institution Superior de Engenharia do Porto/ Instituto Politécnico do Porto). Z-Monitor was firstly developed at COINS Research Unit. The first version supports only IEEE 802.15.4 and Zigbee protocols. Our mission is (i.)to reorganize Z-Monitor's code of the first version(ii.) to support the decoding of both 6LoWPAN and RPL protocols (iii.) to add some improvements to the design of Z-Monitor.

1.5 Outline

The remainder of this Thesis is structured as follow. Chapter 2 gives a brief overview on WSNs technology followed by a description of a set of protocols that form 6LoWPAN networks namely IEEE 802.15.4, 6LoWPAN and RPL. In Chapter 3, we describe the design and features of Z-Monitor. In Chapter 4, we describe the different tools and scenarios used for the performance evaluation of RPL and we detail the main results found from this experimental study. Conclusions and future works are finally presented in chapter 5.

CHAPTER 2 GENERAL BACKGROUND

Contents

2.1 Introduction

Recent advances in the technology of wireless communications and digital electronics have enabled the development of small inexpensive and low power devices that can communicate with each other. These devices integrate an acquisition unit for data collecting, a processing unit for aggregating the collected data , a storage unit and a radio transmission module. They cooperate with each other to form a communication infrastructure called wireless sensor mote.

Wireless Sensor Networks usually consist of many motes, that communicate via radio links for information sharing and cooperative processing. These devices are deployed randomly in an area of interest to monitor or supervise various phenomena. The importance of WSNs has been enforced by the delivery of many protocols that facilitate the communication between motes, Internet connection, routing... The IETF working group is concerned with the specification of protocol for WSNs. It forms a set of protocols in order to allow the sensor nodes to connect to internet.

In this chapter, we are going to present a brief overview on WSNs. We follow this presentation by a description of 6LoWPAN networks. We will detail namely the specification of the IEEE 802.15.4, 6LoWPAN and RPL protocols.

2.2 Wireless Sensor Networks

A Wireless Sensor Network is typically composed of a large set of wireless sensor nodes scattered in a controlled environment and interacting with the physical world. This set aims the collection of specified data needed for the monitoring/control of a predefined area/region [1]. Sensor nodes collaborate in order to merge individual sensor readings into a high-level sensing result. For example, to estimate the velocity, we need to integrate a time series of position measurements. Collected data is then processed and analyzed by a control station (or sink). Figure 2.1 [2] gives more explanation about information process in a WSN.

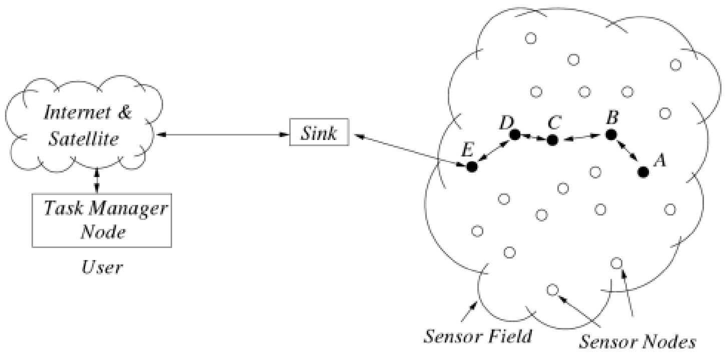


Figure 2.1: WSN Typical Architecture

2.2.1 Wireless Sensor Networks Applications

Wireless Sensor Networks are applied into a wide variety of applications and systems. There are several kinds of WSN applications. However, we can group them in two main classes. The first one aims to gather from the environment information of a relatively simple form like temperature. Such applications include entity monitoring and require limited signal processing. The other class of applications processes and transports large volumes of a complex data. A famous example of this application is industrial monitoring and video surveillance. Below some application of WSN are described:

- Healthcare: Integrating a WSN in healthcare applications is potentially beneficial for both doctors and patients. Attaching sensors to patients assures a long-term surveillance and disease administration. In hospitals, this technology saved the life of many patients by alerting doctors about patient's situation [3].

- Environmental monitoring: WSNs are used in such applications to monitor environmental parameters like humidity and temperature. An example of environmental monitoring applications is detection of forest fires.

- Industrial applications: Industries look for ameliorating products/services quality and process control. Wireless sensor technologies were involved in this domain to help increasing the industrial efficiency. They are used for factory automation, detection of liquid/gas leakage, real time inventory management, etc.

- Military applications: Such applications require a high level of security. WSNs are used in military area for that purpose. Being equipped with the appropriate sensors, these networks enable battlefield surveillance, sensing intruders on bases, detection of enemy units, military situation awareness, etc. [4]

2.2.2 Typical Architecture of a Sensor Mote

A wireless Sensor device is a battery-operated device, capable of sensing physical entities [5]. In addition to sensing, it is able to perform wireless communication, signal processing, data storage and a limited amount of computation. It is made up of four main components, as shown in figure 2.2 [6]:

- Sensing unit: is the actual interface that can observe and control physical parameters from target area. The analog signal is produced by the sensor based on the observed phenomenon. This signal is then converted to digital signal by an Analog-to-Digital Converter (ADC) and then delivered to the processing unit for analysis.

- Transceiver: is charged of connecting motes to the network.
- Power unit: is a major weakness for sensor networks. This unit is responsible of remaining energy control and measuring the time to live of the sensor mote. Some sensors are able to renew their energy from solar or vibration energy.
- Processing unit: has various functions. First, it receives data from sensors. Then, it processes this data. Finally, it decides when and where to send it.

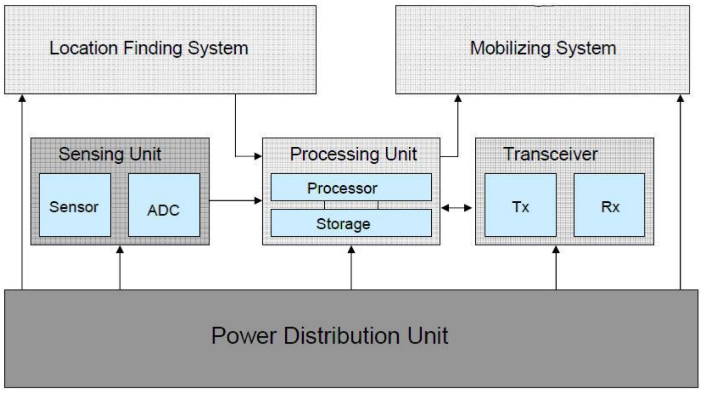


Figure 2.2: Main Component of Sensor Mote

In addition to the above units, a sensor mote may include some additional application dependent components such as mobilizing system, power generator and location finding system.

2.3 Overview on the IEEE 802.14.5 Protocol

IEEE 802.15.4 is a standard that defines a protocol and interconnection of devices via radio communication in a personal area network (PAN) [7]. It is maintained by the Institute of Electrical and Electronics Engineers (IEEE) 802.15 working group [8]. IEEE 802.15.4 defines both the Physical (PHY) and the Media Access Control (MAC) specification for Low-Rate Wireless Personal Area Networks (LRWPANs), leaving other higher-level layers undefined.

This standard defines two kinds of motes: Full Function Devices (FFD) and Reduced Function Devices (RFD). Full Function Devices implement the complete protocol set. They are capable to act as a device or as a coordinator when it is in charge of the whole network. Reduced Function Devices operate with a minimal implementation of the IEEE 802.15.4 protocol. They cannot act as a coordinator. They can only communicate with FFDs.

2.3.1 802.15.4 Addresses

In IEEE 802.15.4 protocol, a device is uniquely identified by a 64-bit address (or long address). Long addresses are globally unique and assigned when manufacturing. 00:12:75:00:11:6e:cd:fb is an example of a long address. Because of the limited packet size in IEEE 802.15.4 standard, nodes are allowed to use short addresses. These addresses are 16 bits length and they are assigned by the PAN coordinator. However, they can be used only within the PAN in which it was assigned. To communicate with devices outside its PAN, a device must include the 16-bit PAN identifier of its own PAN and the PAN of the device with which it communicates in each message.

2.3.2 Physical Layer

The physical layer is responsible for data transmission, channel selection and energy and signal management functions. It operates on one of three possible frequency bands: a single communication channel between 868 MHz and 868.6 MHz, 10 communication channels between 902.0 MHz and 928.0 MHz and 16 communication channels between 2.4 GHz and 2.4835 GHz [7].

A packet is mainly composed of a synchronization header (SHR), a physical layer header (PHR) and PHY payload. The synchronization header contains the preamble and start of frame delimiter(SFD). This header allows the receiving device to synchronize and lock onto the bit stream. The physical layer header contains an information of the frame length (refer to Figure 2.3).

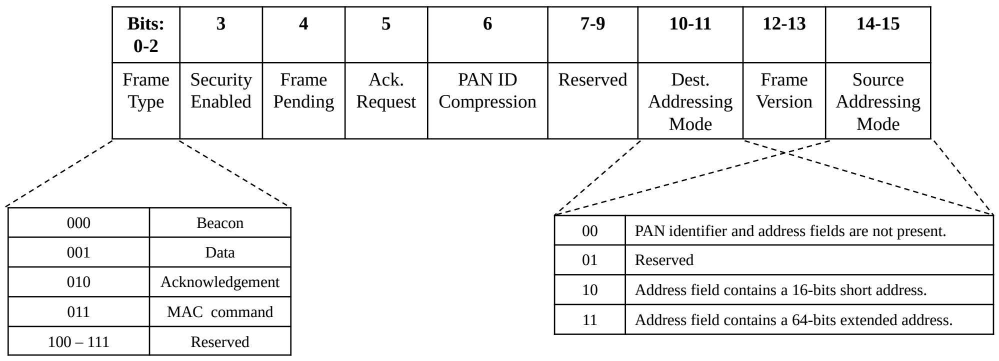
Octets:4	1	1		variable
Preamble	SFD	Frame length (7 bits)	Reserved (1 bit)	PSDU
SHR		PHR		PHY payload

Figure 2.3: General Packet Format

Moving to the next layer (MAC layer), another header will be added to the packet. This header will be inserted at the beginning of the PHY payload.

2.3.3 Medium Access Control (MAC) Layer

The Medium Access Control layer allows the transmission of MAC frames through the use of the physical channel. It aims to reduce or avoid packet collisions in the medium. MAC layer is capable of basic data frame-type as well as complex signalling and beaconing. This layer adds to the packet another header called MAC Header (MHR). The main fields of the MAC header are Frame Control, Sequence Number, Addressing fields, and Auxiliary Security Header. The Frame Control field is 2 octets in length. It contains information defining frame structure. It is composed of: (i.) Frame Type: 2 bits field that specifies the type of IEEE 802.15.4 frame, (ii.) Security Enabled: defines whether an Auxiliary Security Header should protect the MAC layer header or not, (iii.) Frame Pending: shall be used only in beacon frames and shall be set to one if the device sending the frame has more data for the recipient [7], (iv.) Ack. Request: specifies whether the recipient device requires an acknowledgement when receiving a data or a MAC command frame, (v.) PAN ID Compression: specifies if Source PAN Identifier or Destination PAN Identifier should be carried in-line, (vi.) Dest. Addressing Mode (refer to Figure ??) (vii.) Frame Version: indicates the version number corresponding to the frame, (viii.) Source Addressing Mode (refer to Figure ??). Figure 2.4 presents the Frame Control field.



Bits: 0-2	**3**	**4**	**5**	**6**	**7-9**	**10-11**	**12-13**	**14-15**
Frame Type	Security Enabled	Frame Pending	Ack. Request	PAN ID Compression	Reserved	Dest. Addressing Mode	Frame Version	Source Addressing Mode

000	Beacon
001	Data
010	Acknowledgement
011	MAC command
100 – 111	Reserved

00	PAN identifier and address fields are not present.
01	Reserved
10	Address field contains a 16-bits short address.
11	Address field contains a 64-bits extended address.

Figure 2.4: Frame Control Field

The Sequence Number is one octet length field. It specifies the sequence identifier for the frame used specifically to match an acknowledgement frame to a data or MAC command frame. The Addressing fields, is namely composed of Destination PAN Identifier, Destination Address, Source PAN Identifier and Source Address. The Destination PAN Identifier specifies the unique PAN identifier of the intended recipient of the frame [7]. The broadcast PAN identifier's value is 0xffff. The Destination Address and Source Address length depend respectively on the Destination Addressing Mode and Source Addressing Mode subfields of the Frame Control. The Source PAN Identifier specifies the PAN identifier of the originator of the frame. The Auxiliary Security Header is present only when the Security Enabled subfield in the Frame Control is set to 1. It specifies the information required for security processing like the security level. MAC payload field is appended with a 16-bits field: Frame Check Sequence (FCS). It is used to detect errors in each frame. Figure 2.5 [7] shows the frame format of the MAC layer header.

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	Variable	2
Frame Control	Sequence Number	Destination PAN Identifier	Destination Address	Source PAN Identifier	Source Address	Auxiliary Security Header	Frame Payload	FCS
		Addressing fields						
MHR							MAC Payload	MFR

Figure 2.5: MAC Frame Format

The MAC layer employs four frame types: beacon frame, data frame, acknowledgement frame and MAC command frame. In what follow, we present the structure of each frame according to IEEE 802.15.4-2006.

2.3.3.1 Acknowledgement Frame

Acknowledgement frames are used to confirm successful frame reception. The structure of the acknowledgement frame is depicted in Figure 2.6 [7]. MHR header in this case is very simple. It contains only the frame control and the sequence number fields.

Octets: 2	1	2
Frame Control	Sequence Number	FCS
MHR		MFR

Figure 2.6: Acknowledgement Frame Format

2.3.3.2 Beacon Frame

A beacon frame is sent by the PAN coordinator to organize the network. Figure 2.7 [7] gives a schematic view of beacon frame fields. The MAC payload in beacon frame contains superframe specification, Guaranteed Time Slot (GTS) fields, Pending address fields and beacon payload.

Octets: 2	1	4/10	0/5/6/10/14	2	variable	variable	variable	2
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Superframe Specification	GTS field	Pending Address field	Beacon Payload	FCS
MHR				MAC Payload				MFR

Figure 2.7: Beacon frame format

2.3.3.3 Data Frame

Data frames are used for carrying user data. In this case, the MAC payload contains the data that we want to transmit. As for beacon frames, data frames are prefixed with a MAC header and appended with a MAC footer (MFR). Figure 2.8 [7] gives an overview of the structure of a data frame.

Octets: 2	1	4/10	0/5/6/10/14	variable	2
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Data Payload	FCS
MHR				MAC Payload	MFR

Figure 2.8: Data Frame Format

2.3.3.4 MAC Command Frame

MAC command frames are used to handle all MAC peer entity control transfers. Figure 2.9 shows the structure of the MAC command frame. The MAC payload contains the command type field and the command payload which contains the MAC command itself.

Octets: 2	1	4/10	0/5/6/10/14	1	variable	2
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Command Frame Identifier	Command Payload	FCS
MHR				MAC Payload		MFR

Figure 2.9: MAC Command Frame Format

2.4 6LoWPAN Protocol

A 6LowPAN network is made up of low-power wireless area networks (LoWPANs). A LoWPAN is a subnet where 6LoWPAN motes share a common IPv6 address prefix. In fact, an IPv6 address is 128 bytes length. The first 64 bits are the IPv6 address prefix and the last 64 bits are interface identifier.

2.4.1 Adaptation Layer

The IEEE 802.15.4 link layer has a severe constraint on the size of the transmission unit. It allows up to 127 bytes for the whole IPv6 packets including header and payload information [10]. As mentioned in the previous section, the maximum length of IEEE 802.15.4 headers is 39 bytes which keeps 88 bytes for upper layers. However, IPv6 header length is 40 bytes. Moreover, the length of the TCP header is another 20 bytes. Thus, implementing IPv6 over IEEE 802.15.4 would result to only 28 bytes for application layer protocols. For this reason, IPv6 packets may need to be compressed and/or fragmented into multiple link layer frames.

The Internet Engineering Task Force (IETF) [11] formed the IETF 6LoWPAN working group [12] to standardize framing and header compression for the transmission of IPv6 packets over IEEE 802.15.4 links. It introduced an adaptation layer that handles several functionalities to enable IEEE 802.15.4 devices to connect to IP networks. From [13], the adaptation layer (called also the LoWPAN adaptation layer) was introduced between the IEEE 802.15.4 link layer and the network layer. The main functionalities of this layer are: packet fragmentation and reassembly, header compression and link layer forwarding.

2.4.1.1 Fragmentation

A packet payload is fragmented into several packets when it is too large to fit into a single IEEE 802.15.4 frame. Fragmentation offers the ability to encode a datagram into multiple link frames. However, it does not include a recovery mechanism for lost packets. The frame is broken into multiple fragments using fragment header shown in Figure 2.10 [14] below. The first fragment does not contain datagram-offset since it defines the offset of the fragment within the original payload. Datagram-size, the first field in fragment header, indicates the size of the entire IP packet before fragmentation. This field is present in both first and subsequent fragments because a subsequent fragment may arrive before the first fragment. Thus, the receiver would not know the memory size that should be allocated for the whole frame. Datagram-tag field is used to identify the fragmented packet.

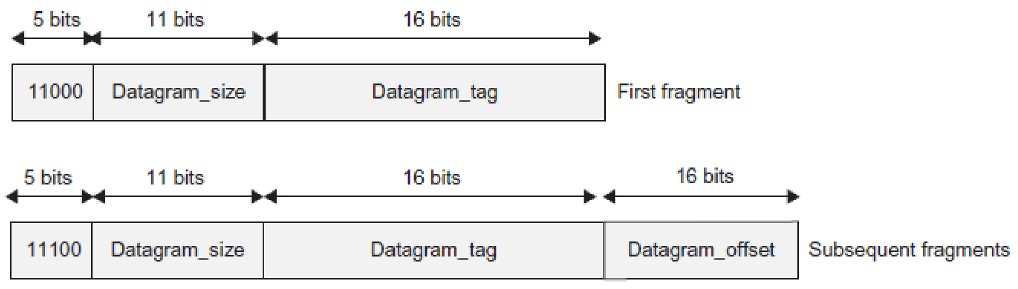


Figure 2.10: Fragmentation Header

2.4.1.2 Header Compression

The 6LoWPAN header compression was firstly defined in RFC 4944 [15]. This standard defines a stateless compression schema consisting of two types of header compression: HC1 (header compression 1) and HC2 (header compression 2). HC1 allows the compression of IPv6 header with the original size of 40 bytes to 3 bytes in the best case. Similarly, HC2 defines a compression format for the transport protocol layer. 6LoWPAN header compression according to RFC 4944 is presented in Figure 2.11.

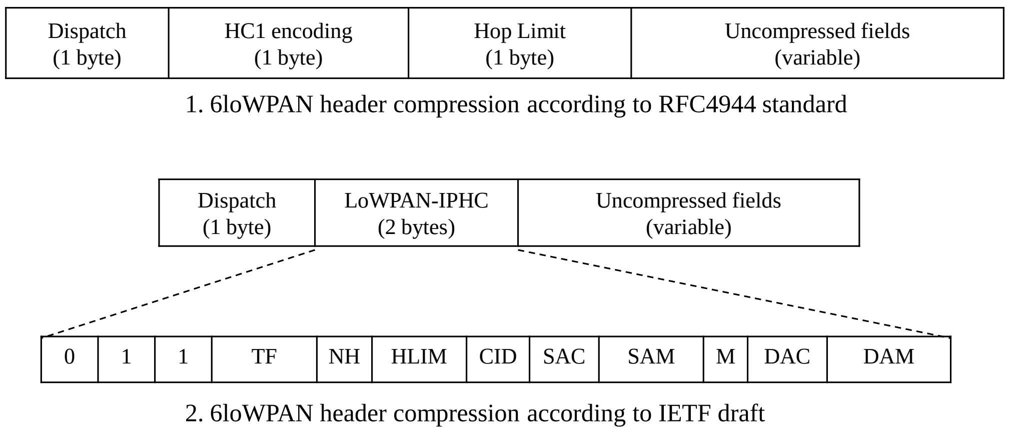

Dispatch (1 byte)
HC1 encoding (1 byte)
Hop Limit (1 byte)
Uncompressed fields (variable)
1. 6loWPAN header compression according to RFC4944 standard
Dispatch (1 byte)
LoWPAN-IPHC (2 bytes)
Uncompressed fields (variable)
0
1
1
TF
NH
HLIM
CID
SAC
SAM
M
DAC
DAM
2. 6loWPAN header compression according to IETF draft

Figure 2.11: 6LoWPAN Header Compression

The first byte in 6LoWPAN header is the dispatch. It specifies that the following header is a compressed IPv6 header. The next byte is called HC1 encoding. It defines whether a field is compressed or not. The hop limit is considered too hard to be compressed. It follows immediately the encoding field. The rest of 6LoWPAN header contains uncompressed IPv6 fields. The size of this field is variable. It depends on the encoding field value. Table 2.1

resumes how RFC 4944 compresses the IPv6 header.

Encoding bits	IPv6 Fields
Bits 0 and 1	Source Address 00: Prefix and interface identifier are carried in-line. 01: Prefix is carried in-line and interface identifier is elided. 10: Prefix is elided and interface identifier is carried in-line. 11: Prefix and interface identifier elided are elided.
Bits 2 and 3	Destination Address 00: Prefix and interface identifier are carried in-line. 01: Prefix is carried in-line and interface identifier is elided. 10: Prefix is elided and interface identifier is carried in-line. 11: Prefix and interface identifier elided are elided.
Bit 4	Traffic Class and Flow Label 0: 28 bits are carried in-line. 1: Traffic Class and Flow Label are zero.
Bits 5 and 6	Next Header 00: 8 bits are carried in-line. 01: next header is UDP. 10: next header is ICMP. 11: next header is TCP.
Bit 7	HC2 encoding 0: No more header compression bits. 1: transport layer header is compressed.

Table 2.1: 6LoWPAN Header Compression According to RFC 4944 Specification

In 6LoWPAN stateless compression, HC2 compresses only the UDP header. 1-octet field, called HC_UDP encoding, is used to compress the UDP header. The UDP header is compressed only if the last bit in HC1 encoding is set to 1. Table 2.2 depicts the HC2 compression according to the RFC 4944 standard.

HC_UDP encoding	UDP fields
Bit 0	UDP source port 0: carried in-line. 1: compressed to 4 bits. Source port in this case is 61616 (0xF0B0)+4 bits carried in-line.
Bit 1	UDP destination address 0: carried in-line. 1: compressed to 4 bits. Source port in this case is 61616 (0xF0B0)+4 bits carried in-line.
Bit 2	Length 0: carried in-line. 1: compressed. Length computed from IPv6 header length information. The value of the UDP length field is equal to the Payload Length from the IPv6 header, minus the length of any extension headers present between the IPv6 header and the UDP header.
Bits(3-7)	Reserved

Table 2.2: HC2 Header Compression According to RFC4944 standard

In some LoWPANs, a device may send or receive packets to some nodes external to the LoWPAN. In this case, only the interface identifier of the source address can be compressed. The IETF working group emerged a second generation of header compression specification called draft-ietf-6lowpan-hc [16]. Compared to RFC 4944 specification, it introduced a few changes to enable the compression of global addresses. This specification offers to nodes the opportunity to establish some additional context when joining the LoWPAN. This context is exchanged between the motes to be used then for the address field compression. To guarantee the context synchronization between the compressor and the decompressor, 6LoWPAN nodes should use the context when a higher layer protocol is used. This is due to the fact that higher layers protect IPv6 addresses by using a pseudo-header-based cheksum and/or authenticator. The structure of 6LoWPAN header compression in both specifications is depicted in Figure 2.11.

Simitar to the stateless compression with RFC 4944, the context-based header compression is composed of a dispatch, an encoding field (called also LOWPAN-IPHC) and the uncompressed fields. The LOWPAN-IPHC field is coded in 2 bytes . It controls which IPv6 fields are compressed. Tables 2.3 and 2.4 show how 6LoWPAN header is compressed/decompressed.

LOWPAN-IPHC	IPv6 fields
Bits 3 and 4 (TF)	Traffic Class and Flow Label 00: both Flow Label and Traffic Class are carried in-line. (4 bytes) 01: only Flow Label is carried in-line.(3 bytes) 10: only Traffic Class is carried in-line.(8 bits) 11: Traffic Class and Flow Label are compressed.
Bit 5 (NH)	Next Header 0: next header is carried in-line.(8 bits) 1: next header is compressed.
Bits 6 and 7 (HLIM)	Hop Limit 00: hop limit is carried in-line. 01: hop limit is compressed and equal to 1. 10: hop limit is compressed and equal to 64. 11: hop limit is compressed and equal to 255.
Bit 8 (CID)	Context Identifier Extension 0: No additional 8-bit Context Identifier Extension is used. 1: An additional 8-bit Context Identifier Extension field immediately follows the DAM field.
Bit 9 (SAC)	Source Address Compression 0: Source address compression uses stateless compression. 1: Source address compression uses stateful, context-based compression.
Bits 10 and 11 (SAM)	Source Address Mode If SAC=0: 00: source address is carried in-line.(128 bits) 01: the first 64-bits of the address are elided. The value of those bits is the link-local prefix padded with zeros. 10: the first 112 bits of the address are elided. The value of those bits is the link-local prefix padded with zeros. The remaining 16 bits are carried inline. 11: the address is fully elided. If SAC=1: 00: reserved 01: The address is derived using source context identifier (the first 4 bits of Context Identifier Extension) and the 64 bits carried inline. 10: The address is derived using source context identifier and the 64 bits carried inline. 11: The address is derived using context information and possibly link-layer addresses.
Bit 12 (M)	Multicast Compression 0: Destination address does not use multicast compression. 1: Destination address uses multicast compression.

Table 2.3: LOWPAN-IPHC Header Compression

LOWPAN-IPHC	IPv6 fields
Bit 13 (DAC)	Destination Address Compression 0: Destination address compression uses stateless compression. 1: Destination address compression uses stateful, context-based compression.

Bits 14 and 15 (DAM)	Destination Address Mode
	If M=0 and DAC=0:
	00: Full address is carried in-line. (128 bits)
	01: The first 64-bits of the address are elided. The value of those bits is the link-local prefix padded with zeros.
	10: The first 112 bits of the address are elided. The value of those bits is the link-local prefix padded with zeros. The remaining 16 bits are carried inline.
	11: The address is fully elided.
	If M=0 and DAC=1:
	00: reserved.
	01: The address is derived using destination context identifier and the 64 bits carried inline.
	10: The address is derived using destination context identifier and the 16 bits carried inline.
	11: Full address is compressed.
	if M=1 and DAC=0:
	00: 48 bits are carried in-line. The address takes the form FFXX::00XX:XXXX:XXXX.
	01: 32 bits are carried in-line. The address takes the form FFXX::00XX:XXXX.
	10: 16 bits are carried in-line. The address takes the form FF0X::0XXX.
	11: 8 bits are carried in-line. The address takes the form FF02::00XX.
	if M=1 and DAC=1:
	00: Full address is carried in-line. (128 bits)
	01: 48 bits are carried in-line.
	10: Reserved.
	11: Reserved.

Table 2.4: LOWPAN-IPHC Header Compression (continuation)

2.4.1.3 Forwarding and Routing

In a LoWPAN, packets have to traverse multiple radio hops. This involves two processes: routing and forwarding. They can be implemented either in the link layer or in the network layer. 6LoWPAN supports both forwarding approaches. Forwarding when it is performed on link layer is called Mesh-Under [17]. Link layer addresses are used to forward packets. A device would therefore be seen as a single IP link. In this case, routing is performed on the adaptation layer. However, there is problem that must be solved: to forward a packet to its destination, a node must know the final destination address. Besides that, nodes need to know the originator address to achieve some services like reassembly. Thus, source and destination addresses must be stored somewhere since each forwarding step replaces the link-layer destination address by the address of the next hop and the link-layer source address by the address doing the forwarding. 6LoWPAN defines mesh header in order to store necessary information for forwarding. Mesh header is depicted in Figure 2.12.

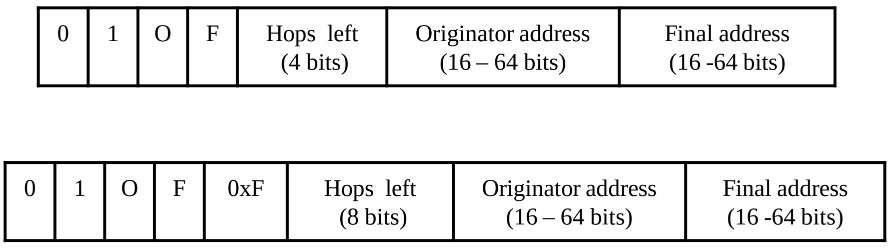

0
1
O
F
Hops left (4 bits)
Originator address (16 – 64 bits)
Final address (16 -64 bits)
0
1
O
F
0xF
Hops left (8 bits)
Originator address (16 – 64 bits)
Final address (16 -64 bits)

Figure 2.12: Mesh Header

Since IEEE 802.15.4 protocol supports 16-bit and 64-bit addresses, the value of the originator flag (O) and the final flag (F) in mesh header is set to one if address length is 16. Hops left field defines the number of intermediate hops between the source and the destination. When the number of hops is less than 15, the size of Hops left is optimized to 4 bits (first case in Figure 2.12). If a value of 15 or larger is needed, four bits are set to 1 (0xF) and another 8 bits are included for Hops left (second case in Figure 2.12). The value of Hops left is decremented by a forwarder before sending the message to the next hop. If it reaches zero, the packet is discarded. The sender set the originator address to its own link-layer address and the final address to the packet's ultimate address.

The second approach of forwarding is called Route-Over. It is performed on network layer. The Route-Over forwarding approach relies on IP routing. The routing is performed on network layer and each node can serve as an IP router.

2.4.2 Neighbor Discovery

Neighbor discovery mechanism was firstly introduced with the IPv6 protocol. The goal of this mechanism is to provide a set of key auto-configuration features for IPv6 like discovery of neighbor presence and discovery of routers on the link. This includes also prefix discovery, link layer addresses, information about paths to active neighbor, etc. Neighbor discovery specifies four ICMP message types: Router Solicitation (RS), Router Advertisement (RS), Neighbor Solicitation (NS) and Neighbor Advertisement (NA). In what follow, we describe the packet formats and the associated services.

Neighbor Solicitation (NS) message allows a device to check the existence and the reachability of a neighbor. It is sent by a mote to obtain or confirm the link layer address of a neighbor for which it knows the IP address. They are also used for address resolution. NS messages are multicast packets whose source address is the address of the requesting mote or the unspecified address. The format of NS is shown in figure 2.13 [14]. It is composed mainly of ICMP fields and the target address. The ICMP header is mainly composed by

- type: 8-bit type field that indicates the type of the message,
- code: 8-bit field used to provide additional granularity for a given ICMP message type,
- checksum: 16-bit field that detects accidental errors that may be introduced while transmitting or storing the message.

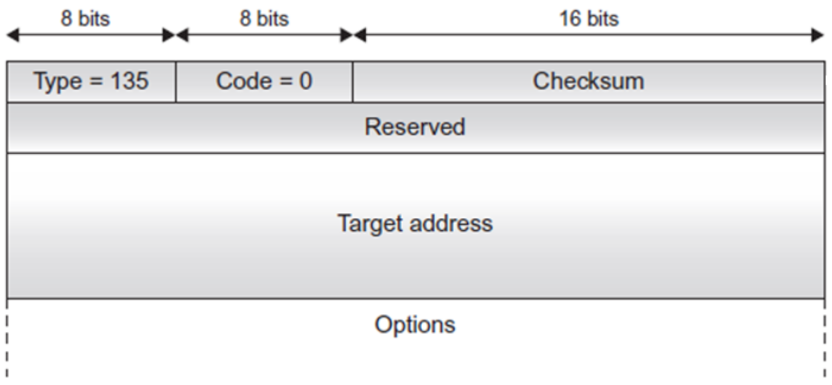


Figure 2.13: Neighbor Solicitation Message Format

Neighbor Advertisement (NA) messages are sent in response to NS messages. They provide the link layer address to the requesting mote. They are also used in unsolicited way to inform an address change or a mobility event.

The source address in these packets is the sender address. The destination address is the requested address presented in NS message. If the source address of NS messages is an unspecified address, then the destination address is the multicast address. When we compare NS message format with NA message format, presented in figure 2.14, we notice that NA contains three additional bits. The first bit is used to indicate that the sender is a router. The second bit indicates whether the message is sent in response to a NS request. The third bit, or O-bit, indicates that the advertisement should override an existing cache entry [14].

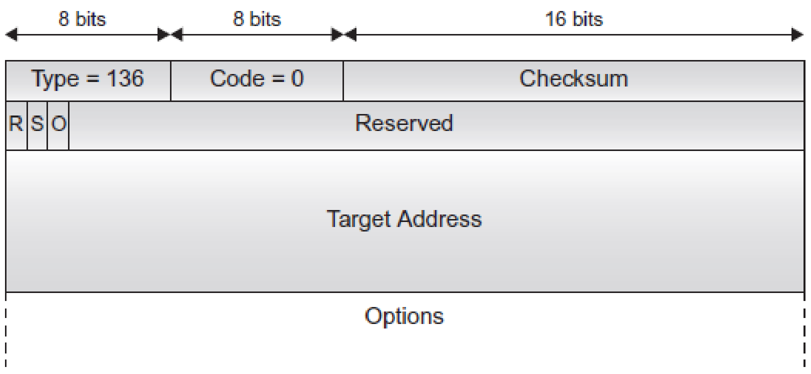


Figure 2.14: Neighbor Advertisement Message Format

Router Advertisement (RA) messages are used by routers to advertise their presence. They are used in unsolicited way as well as in a solicited way in response to router solicitation. Figure 2.15 shows the format of RA messages. In addition to ICMP fields, these messages contain:

- Current Hop Limit: is used in the hop count field of an IPv6 header.
- The M-bit: advertise hosts to use the administered stateful protocol for address auto-configuration.
- The 0-bit: indicates if other configuration information can be obtained.
- Router Lifetime: is the lifetime in seconds of the associated default route.
- Reachable Time: is the time between reachability confirmation and request.
- Retransmission timer: is the time in milliseconds between the retransmission of a neighbor solicitation messages.

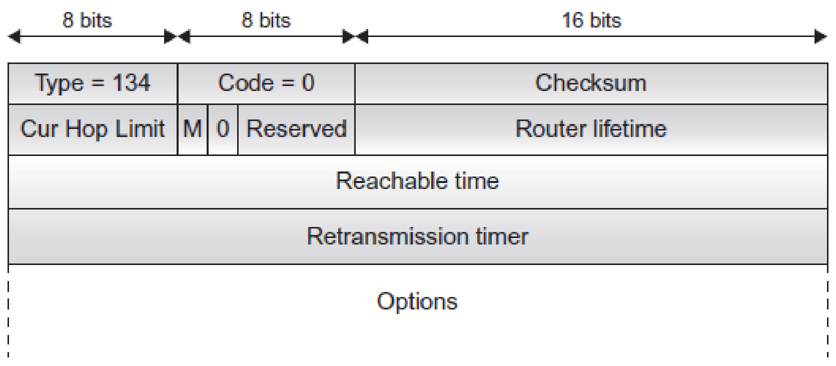


Figure 2.15: Router Advertisement Message Format

The Router Solicitation message is sent by a node to request the reachability of a router. The RS messages contain simply the ICMP fields. Figure 2.16 presents the format of a router solicitation message.

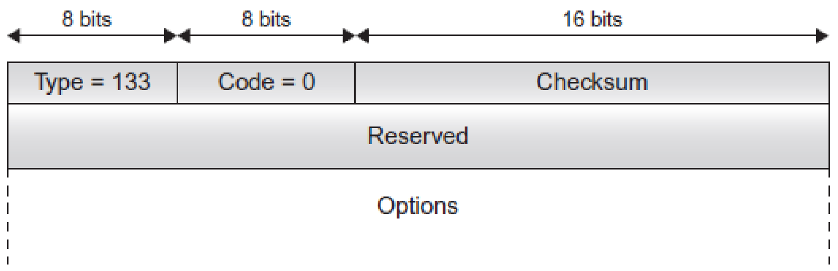


Figure 2.16: Router Solicitation Message Format

2.5 The RPL Protocol

RPL is a distance-vector (DV) and a source routing protocol that is designed to operate on top of several link layer mechanisms including IEEE 802.15.4 PHY and MAC layers. It targets collection-based networks, where nodes periodically send measurements to a collection point. A key feature in RPL is that it represents a specific routing solution for low power and lossy networks [14] [44], which stand for networks with very limited resources in terms of energy, computation and bandwidth turning them highly exposed to packet losses. In fact, it has been specifically designed to meet the requirements of resource-constrained nodes as mentioned in the routing requirement terminology document [45]. In particular, RPL enabled LLNs take into account two main features (i.) the prospective data rate is typically low (less than 250 kbps), and (ii.) communication is prone to high error rates, which results in low data throughput. A lossy link is not only characterized by a high

Bit Error Rate (BER) but also the long inaccessibility time, which strongly impacts the routing protocol design. In fact, the protocol was designed to be highly adaptive to the network conditions and to provide alternate routes, whenever default routes are inaccessible.

RPL is based on the topological concept of Directed Acyclic Graphs (DAGs). The DAG defines a tree structure that specifies the default routes between nodes in the LLN. However, a DAG structure is different from a typical tree in the sense that a node might associate to multiple parent nodes in the DAG, in contrast to classical trees where only one parent is allowed. More specifically, RPL organizes nodes as Destination-Oriented DAGs (DODAGs), where most popular destination nodes (i.e. sinks) or those providing a default route to the Internet (i.e. gateways) act as the roots of the DAGs. A network may consist of one or several DODAGs, which form together an RPL instance identified by a unique ID, called RPLInstanceID. A network may run multiple RPL instances concurrently; but these instances are logically independent. A node may join multiple RPL instances, but must only belong to one DODAG within each instance. Figure 2.17 shows an example of RPL instances with multiple DODAGs.

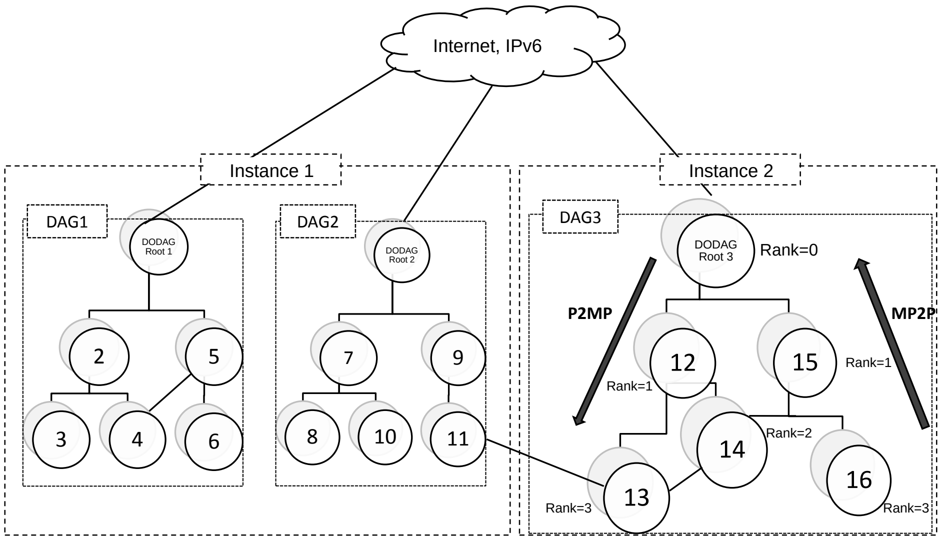


Figure 2.17: A RPL network with 3 DODAGs in 2 instances

2.5.1 Network Model

RPL defines three types of nodes:

- Low Power and Lossy Border Routers (LBR): it refers to the root of a DODAG that represents a collection point in the network and has the ability to construct a DAG. The LBR also acts as a gateway (or edge router) between the Internet and the LLN.

- Router: it refers to a device that can forward and generate traffic. Such a router does not have the ability to create a new DAG, but associates to an existing one.
- Host: it refers to an end-device that is capable of generating data traffic, but is not able to forward traffic.

The basic topological component in RPL is the DODAG, a Destination Oriented DAG, rooted in a special node called DODAG root. The DODAG root has the following properties: (i.) it typically acts as an LBR, (ii.) it represents the data sink within the directed acyclic graph, (iii.) it is typically the final destination node in the DODAG, since it acts as a common transit point that bridges the LLN with IPv6 networks, (iv.) it has the ability to generate a new DODAG that trickles downward to leaf nodes.

Each node in the DODAG is assigned a rank. The rank of a node is defined in [44] as the nodes individual position relative to other nodes with respect to a DODAG root. It is an integer that represents the location of a node within the DODAG. The rank strictly increases in the downstream direction of the DAG, and strictly decreases in the upstream direction. In other words, nodes on top of the hierarchy receive smaller ranks than those in the bottom and the smallest rank is assigned to the DODAG root.

The architecture of a DODAG is similar to a cluster-tree topology where all the traffic is collected in the root. However, the DODAG architecture differs from the cluster-tree in the sense that a node can be associated not only to its parent (with higher rank), but also to other sibling nodes (with equal ranks). The rank is used in RPL to avoid and detect routing loops, and allows nodes to distinguish between their parents and siblings in the DODAG. In fact, RPL enables nodes to store a list of candidate parents and siblings that can be used if the currently selected parent loses its routing ability.

In the construction process of network topology, each router identifies a stable set of parents on a path towards the DODAG root, and associates itself to a preferred parent, which is selected based on the *Objective Function* (OF). The Objective Function defines how RPL nodes translate one or more metrics into ranks, and how to select and optimize routes in a DODAG. It is responsible for rank computation based on specific routing metrics (e.g. delay, link quality, connectivity, etc. ...) and specifying routing constraints and optimization objectives. The design of efficient Objective Functions is still an open research issue. A couple of drafts have been proposed. In [46], the draft proposes to use the Expected Number of Transmission (ETX) required to successfully transmit a packet on the link as the path selection criteria in RPL routing. The route from a particular node to the DODAG root represents the path that minimizes the sum of ETX from source to the DODAG root. In [47], the draft proposes Objective Function 0 (OF0), which is only based on the abstract information carried in an RPL packet, such as Rank. OF0 is agnostic to link layer metrics, such as ETX, and its goal is to foster connectivity among nodes in the network.

2.5.2 RPL Control Messages

RPL messages are specified as a new type of ICMPv6 Control Messages, whose structure is depicted in figure 2.18.

According to [48], the RPL Control Message is composed of (i.) an ICMPv6 header, which consists of three fields: Type, Code and Checksum, (ii.) a message body comprising a message base and a number of options.

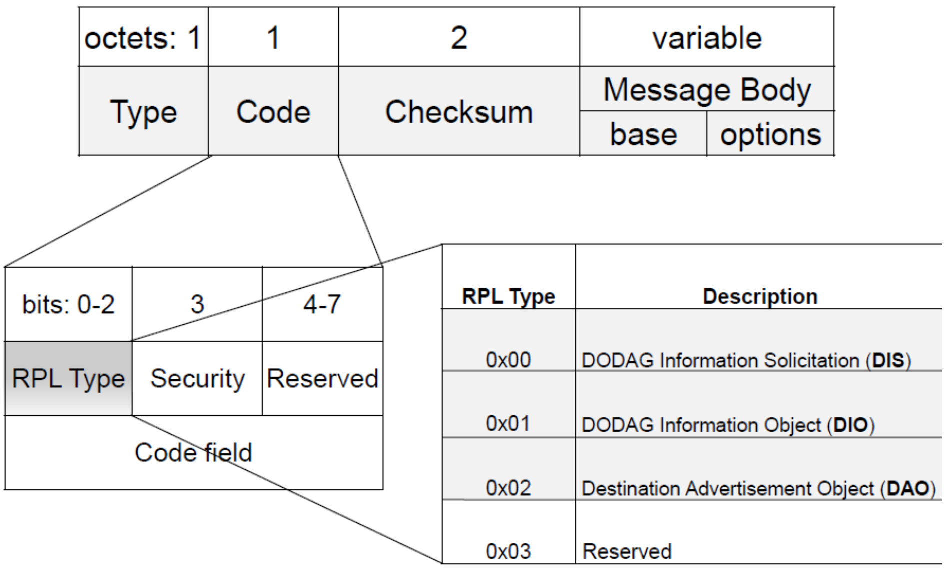


Figure 2.18: RPL Control Message

The Type field specifies the type of the ICMPv6 Control Message prospectively set to 155 in case of RPL (still awaiting confirmation from Internet Assigned Number Authority (IANA), as of version 17 of the draft). The Code field identifies the type of RPL Control Message. Four codes are currently defined:

- *DODAG Information Solicitation* (DIS): The DIS message is mapped to 0x00, and is used to solicit a DODAG Information Object (DIO) from an RPL node. The DIS may be used to probe neighbor nodes in adjacent DODAGs. The current DIS message format contains non specified flags and fields for future use.

- *DODAG Information Object (DIO)*: The DIO message is mapped to 0x01, and is issued by the DODAG root to construct a new DAG and then sent in multicast through the DODAG structure. The DIO message carries relevant network information that allows a node to discover a RPL instance, learn its configuration parameters, select a DODAG parent set, and maintain the DODAG. The format of the DIO Base Object is presented in Fig. 2.19.

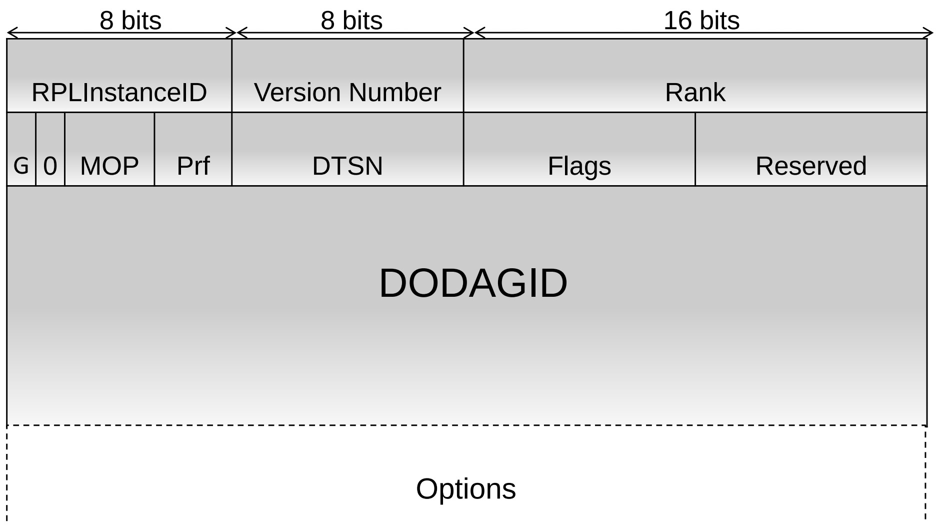


Figure 2.19: The DIO message format

The main DIO Base Object fields are: (i.) RPLInstanceID, is an 8-bit information initiated by the DODAG root that indicates the ID of the RPL instance that the DODAG is part of, (ii.) Version Number, indicates the version number of a DODAG that is typically incremented upon each network information update, and helps maintaining all nodes synchronized with new updates, (iii.) Rank, a 16-bit field that specifies the rank of the node sending the DIO message, (iv.) Destination Advertisement Trigger Sequence Number (DTSN) is an 8-bit flag that is used to maintain downward routes, (v.) Grounded (G) is a flag indicating whether the current DODAG satisfies the application-defined objective, (vi.) Mode of Operation (MOP) identifies the mode of operation of the RPL instance set by the DODAG root. Four operation modes have been defined and differ in terms of whether they support downward routes maintenance and multicast or not. Upward routes are supported by default. Any node joining the DODAG must be able to cope with the MOP to participate as a router, otherwise it will be admitted as a leaf node, (vii.) DODAGPreference (Prf) is a 3-bit field that specifies the preference degree of the current DODAG root as compared to other DODAG roots. It ranges from 0x00 (default value) for the least preferred degree, to 0x07 for the most preferred degree, (viii.) DODAGID is a 128-bit IPv6 address set by a DODAG root, which uniquely identifies a DODAG. Finally, DIO Base Object may also contain an Option field.

- *Destination Advertisement Object* (DAO): The DAO message is mapped to 0x02, and is used to propagate reverse route information to record the nodes visited along the upward path. DAO messages are sent by each node, other than the DODAG root, to populate the routing tables with prefixes of their children and to advertise their addresses and prefixes to their parents. After passing this DAO message through the path from a particular node to the DODAG root through the default DAG routes, a complete path between the DODAG root and the node is established. Figure 2.20 illustrates the format of the DAO Base Object.

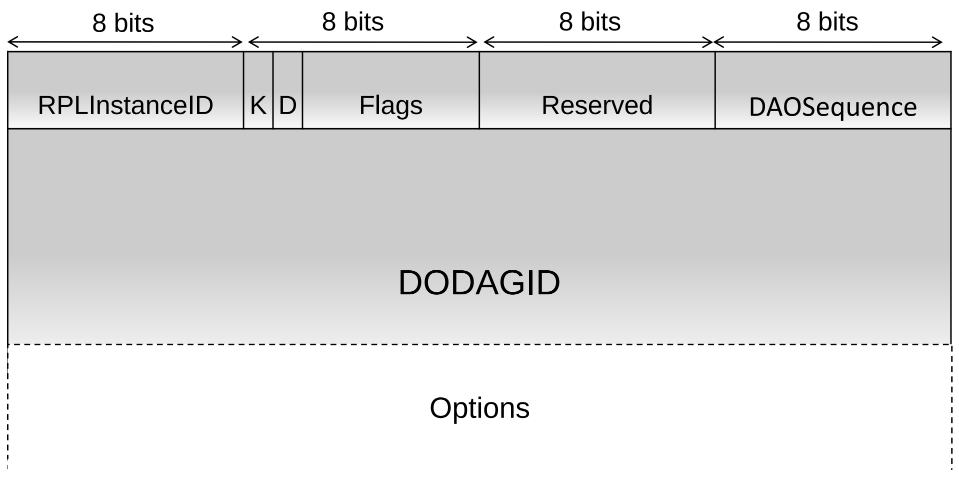


Figure 2.20: The DAO message format

As shown in the figure, the main DIO message fields are: (*i.*) *RPLInstanceID*, is an 8-bit information indicates the ID of the RPL instance as learned from the DIO, (*ii.*)*K flag* that indicates whether and acknowledgement is required or not in response to a DAO message, (*iii.*)*DAOSequence* is a sequence number incremented at each DAO message, (*iv.*)*DODAGID* is a 128-bit field set by a DODAG root which identifies a DODAG. This field is present only when *flag D* is set to 1.

- *Destination Advertisement Object Acknowledgment* (DAO-ACK): The DAO-ACK message is sent as a unicast packet by a DAO recipient (a

DAO parent or DODAG root) in response to a unicast DAO message. It carries information about RPLInstanceID, DAOSequence, and Status, which indicate the completion. Status code are still not clearly defined, but codes greater than 128 mean a rejection and that a node should select an alternate parent.

2.5.3 DODAG Construction

The DODAG construction is based on the Neighbor Discovery (ND) process, which consists in two main operation (1) Unicast transmission of DIO control messages issued by the DODAG root to build routes in the downward direction from the root down to client nodes, (2) Broadcast of DAO control messages issued by client nodes and sent up to the DODAG root to build routes in the upward direction.

In order to construct a new DODAG, the DODAG root broadcasts a DIO message to announce its DODAGID, its Rank information to allow nodes to determine their positions in the DODAG, and the Objective Function identified by the Objective Code Point (OCP) within the DIO Configuration option fields. This message will be received by a client node which can be either a node willing to join or an already joined node.

When a node willing to join the DODAG receives the DIO message, it (i.) adds the DIO sender address to its parent list, (ii.) computes its rank according to the Objective Function specified in the OCP filed, such that the nodes rank is greater than that of each of its parents, and (iii.) forward the DIO message with the updated rank information. The client node chooses the most preferred parent among the list of its parents as the default node through which inward traffic is forwarded. When a node already associated with DODAG receives another DIO message, it can proceed in three different ways (i.) discard the DIO message according to some criteria specified by RPL, (ii.) process the DIO message to either maintain its location in an existing DODAG or (iii.) improve its location by getting a lower rank in the DODAG based on computing the path cost specified by the Objective Function. Whenever a node changes its rank, it must discard all nodes in the parents list whose ranks are smaller than the new computed nodes rank to avoid routing loops.

2.6 Conclusion

In this chapter, we have presented an overview on wireless sensor networks. We have also introduced 6LoWPAN networks. In a first step, we described IEEE 802.15.4 protocol as it presents the lower layer protocol. Then, we highlighted the different features of the 6LoWPAN adaptation layer. Finally, we presented the newly defined routing protocol: RPL. In the next chapter, we describe in depth the original version of the Z-Monitor monitoring tool and our proposition to extend this tool to support IPv6 based protocols.

CHAPTER 3

Z-MONITOR MONITORING TOOL

Contents

3.1 Introduction

Monitoring Wireless Sensor Networks is a fundamental task to report the performance of the network behavior in real-world deployments. In order to ensure that a network is operating at the desired performance level, it is fundamental to have a powerful protocol analyzer. With respect to IP-based networking protocols, there are numerous tools to sniff and analyze packets. When it comes to developing low-level protocols for Wireless Sensor

Networks using IEEE 802.15.4-based radios for communication, the choice of such tools is limited and too expensive. Thus, it was necessary to design a monitoring tool to sniff radio packets: Z-Monitor. Towards this goal, the present chapter starts by presenting the new design of Z-Monitor to support 6LoWPAN networks. Then, we validate the tool by comparing its performance with other monitoring tools.

3.2 Related Works

3.2.1 Research Efforts

Monitoring and debugging of WSNs typically require instrumentation of sensor nodes and introduce monitoring protocols in-band with the actual sensor network traffic. For instance, in [18], the authors proposed *Sympathy*, a tool for automatically detecting and debugging failures in WSNs. The system is based on sending in-band metric data to a sink, where the information will be processed and potential failures are identified. No GUI nor protocol analysis components are provided in this tool. Also, it only supports a fixed set of problems (i.e. failure), while for LoWPAN, monitoring an extensible framework is needed.

Another recognizable effort for WSN monitoring is *Snif* [19], which is an extensible solution but does require multiple dedicated sniffers and does not focus on IEEE 802.15.4-based LoWPANs. *SpyGlass* [20] is another WSN visualization tool in which the data emitted by individual sensor nodes is collected by gateway software and is then passed on via TCP/IP to the visualization software on a potentially remote machine. The main focus of this work is on the visualization and not the data analysis and network performance. In [21] the authors have presented a multi-sniffer, multi-view application in which special hardware multiple sniffers are deployed and data emitted by individual sensor nodes is collected by these sniffers and is later used to view network topology, sensing data, network performance, hardware resource depletion, etc. This work also needs special hardware which is expensive and mostly proprietary. An effort to combine real-network testbed visualization and simulation results is made in [22]. This work is focused on providing both simulation and visualization functions to assist the investigation of algorithms in WSNs and cares less the testbeds monitoring.

There are some recent efforts focused on IEEE 802.15.4 monitoring, e.g. [23], [24], and [25]. In [23] a TinyOS-based sniffer is presented which focuses on capturing the packets and displaying the packet information. The work does not cover ZigBee or 6LoWPAN packet handling. A distributed monitoring and protocol analysis tool *SNDS* is presented in [24]. This work tries to combine TCP/IP networks with sensor networks but uses special sniffer hardware which is generally expensive and proprietary. The solution in [25] is focused on network monitoring and management of 6LoWPAN but it concentrates more on providing Simple Network Management Protocol (SNMP) inter-operability with 6LoWPAN rather than detailed packet analysis.

3.2.2 Related Products

There are some products that focused on IEEE 802.15.4 monitoring. In this section, we give a brief overview on the existing monitoring tools. Then, we discuss the features of each product and justify the choice of Z-Monitor.

Sensor Network Analyzer (*SNA*) [26] is a monitoring software for ZigBee-based WSN developed and released by Daintree Network, which is among the ZigBee Alliance members. The SNA has advanced visualization capabilities and has the ability to show the network as a whole. It also enables the end-user to view all devices and interactions simultaneously. From this system view, it is possible to collect critical performance metrics such as end to-end latency or packet loss. The Daintree *SNA* product is a comprehensive solution for ZigBee and 802.15.4 testing, analysis and commissioning. However, its high cost prevents its use at a wide level.

The *CC2420 packet sniffer* [27] captures, filters and decodes IEEE 802.15.4 MAC packets, and displays them in a convenient way. Frames may be filtered based on frame type and addressing information. Data may be stored to a binary file format. Data frames, beacon frames, command frames and acknowledgement frames are decoded separately, and each field for each frame is decoded and displayed separately on screen. This tool only considers the Physical, MAC and network layers of the IEEE 802.15.4/ZigBee protocol stack. It does not present any analysis of the data application layer and does not provide any means for monitoring and controlling large-scale multi-hop networks.

The *ZENA* tool [28] is a wireless network analyzer that graphically displays wireless network traffic following the IEEE 802.15.4 specification on the 2.4GHz band.The *ZENA* analyzer supports the ZigBee, MiWi and MiWi P2P protocols. In conjunction with the hardware packet sniffer, the software can analyze network traffic and graphically displays decoded packets. It can also display a graphical representation of the network topology and the messages as they flow through the network. This information can then be saved and/or exported for further analysis. However, it does not support the analysis of many LoWPAN protocols such as 6LoWPAN and RPL.

Wireshark [29] is a free and open-source packet analyzer tool. It is used largely for network troubleshooting, analysis, software and communications protocol development, and education. It provides a user-friendly interface with storing and filtering features. Wireshark supports capturing packets in both from live network and from a saved capture file. The capture file format is libpcap format like that in tcpdump. It supports various kinds of operating systems. However, Wireshark does not detect automatically the USB interface of some WSN platforms to capture data. In addition, it cannot display the graphical topology representation of the network.

Z-Monitor [30] is a free tool for monitoring, maintaining and testing ZigBee-based WSNs. Z-Monitor was designed and implemented to be interoperable with TKN and Open-ZB implementations, which are open source implementations of the standard distributed with TinyOS operating system. It is specially used by researchers and students for debugging and deploying WSN applications based on LoWPANs. It supports decoding IEEE 802.15.4 and Zigbee protocols. It has Graphical User Interface (GUI) that provides frame decoding, traffic timeline, packet statistics, etc. This solution overcomes most of the shortcoming of existing solutions. It is open source, does not require special sniffer hardware and is easy to extend. Thus, we assert to extend Z-Monitor to support more protocols.

3.3 Z-Monitor2.1 Design

Compared to the first version, the new version of Z-Monitor is more structured and provides a more sophisticated graphical user interface. The work that we have done to release this new version is as follow: first, we reorganized the code to make it more modular. Then, we designed a new decoding process. Finally, we extended it to support more protocols and features.

3.3.1 Code Reorganization

The code of the old version of Z-Monitor was not well structured: there are no separation between the graphical interfaces and the decoding task. Figure 3.1 presents the UML diagram of Z-Monitor classes as implemented in the old version. This version is mainly composed of seven classes:

- Zmonitor class: represents the main class of the tool. It is responsible of displaying all the GUI on the screen and user interaction. It doesnt contain any decoding task.
- Zsniffer class: is responsible of decoding task and drawing packet sniffer interface.
- Zmote class: is a class that contains all the attribute of a sensor mote in the network like its MAC address.
- PointMote class: is used to localise a sensor mote in the network. This class is mainlly used to draw the topology.
- ThemeChange class: is used to change Z-Monitor theme.

- Zmap class: used to draw the network topology. It looks like a simulation of the network parameters: motes coordinates, links between motes, motes type...
- MoteTimeLine class: is a class that displays the received packet over the time.

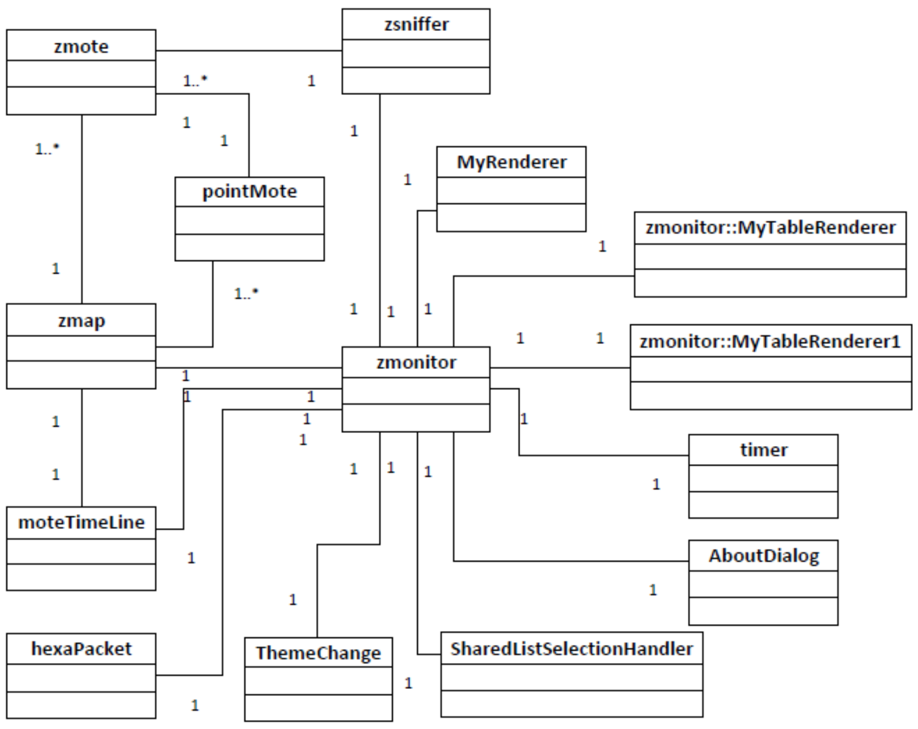


Figure 3.1: Original Code Structure

It was really difficult to understand the original code of Z-Monitor. So, in a first step, we separated the decoding task from the display task. In addition, we added some improvements to ameliorate the display. Then, we structured it into packages. Finally, we integrated the necessary methods or classes to support 6LoWPAN and RPL protocols. The code of Z-Monitor after performing all these modifications is depicted in figure 3.2.

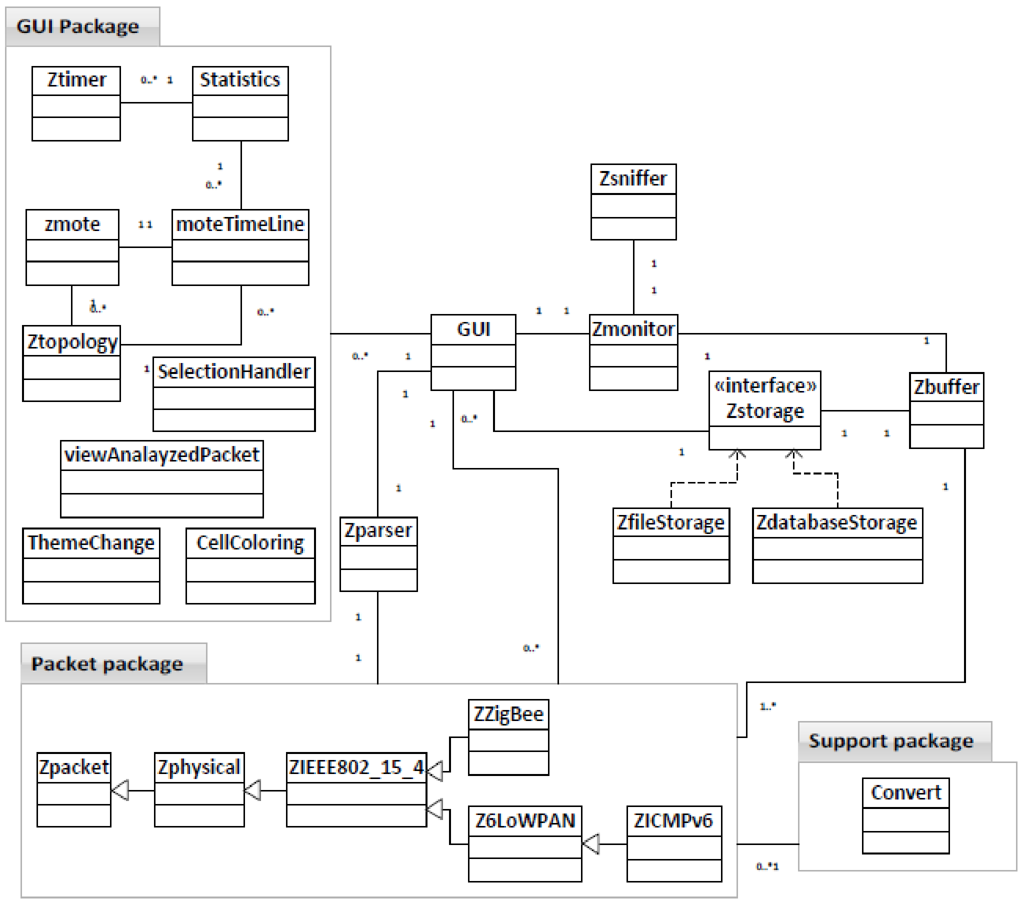


Figure 3.2: Z-Monitor New Code Structure

3.3.2 Design

The software design objective of Z-Monitor is to provide an open source, extensible, modular and user-friendly solution for LoWPAN monitoring [56]. Z-Monitor allows for passive monitoring of IEEE 802.15.4-based networks and for analyzing the network behavior through statistical data analysis. Z-Monitor relies on a particular sensor node acting as a passive sniffer that captures network traffic and redirects it to a user-friendly Graphical User Interface (GUI). The fundamental advantage of Z-Monitor as compared to commercially available products such as CC2420 Sniffer, Daintree Network Analyzer and Zena is that it is independent of any special hardware and simply relies on a simple mote to capture traffic.

To meet the aforementioned objectives, a component-based approach has been used to design Z-Monitor. The block diagram of the main components is shown in figure 3.3 [31].

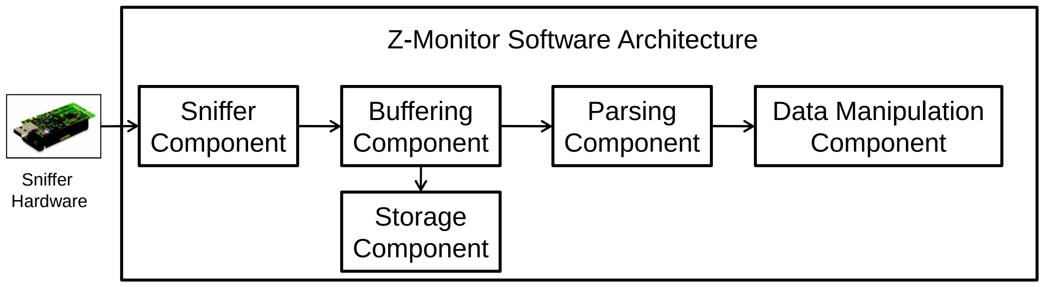


Figure 3.3: Z-Monitor Design Architecture

According to figure 3.3, Z-Monitor has two types of component: hardware and software. The hardware component is used to capture the packets; while software components are used to store collected packets in a buffer, to perform parsing and packet decoding and finally to display parsed frames and output network statistics.

3.3.2.1 Hardware Component

The sniffer hardware is simply an IEEE 802.15.4-compliant sensor mote, which passively captures the network traffic. Each received packet is redirected to the serial interface through which the sniffer is attached to forward that packet to the software sniffing threads. The sniffer hardware that we have used is a simple TelosB mote that implements tknsniffer application available under TinyOS. The tknsniffer application switches the USB port into promiscuous mode and subsequently sniffs all packets that come along.

Z-Monitor takes the serial port of the router as an input to analyze packets exchanged through the network. When we wanted to test it with BLIP implementation, Z-Monitor was not able to capture packets because the router's USB port is used by the routing protocol driver. The solution that we have proposed is to use a second router that plays the role of the sniffer for Z-Monitor. To be sure that we grab all packets on CC2420 hardware, we forced the address recognition and acknowledgements to be off by modifying the *makefile* of the second router. However, this solution can only capture 6LoWPAN packets, and thus acknowledgement frames, for example, were not captured. According to 6LoWPAN specification, IP and 6LoWPAN packets are transmitted only through data frames. This means that only IEEE 802.15.4 data frames are captured by the sniffer application in this case. Thus, we looked for another solution to capture all packet types including acknowledgments.

To this end, we used TknSniffer application as a second solution to capture the 6LoWPAN packets. TknSniffer is a tinyOS application developed by Jan Hauer. It can be found in Z-Monitor package download. This application works as follow: it switches the USB port into promiscuous mode and subsequently sniffs all packets that comes along.

3.3.2.2 Software Component

After being captured by the hardware component, a packet is sent to the software component starting by the sniffer component. The sniffer component reads the data from the mote through a serial forwarder and processes it to be interpreted into a more logical format. We have implemented a Java class Zsniffer which is used to sniff the traffic, capture packets coming from the attached mote running tknsniffer application and redirect it to the buffer component.

The main role of the buffering component is to store the incoming bit-stream of data in the volatile memory to avoid any packet loss. We implemented the class Zbuffer to handle this operation. There are two possible options for buffering. The first option is to store packets into a data structure after parsing. The second option is to store packets as a string bit-stream before parsing. The first option has the advantage of quicker fetching of packets fields as compared to the second option, which needs to apply parsing each time a packet is needed. However, the second buffering option has the advantage of being independent of packet types and requires less memory size than the first option. For that purpose, we have opted for buffering the raw bit-stream before parsing, in particular fetching fields of previously received packets is not a frequent or time-constrained operation.

The parser component has a key role as it is responsible of decoding the packet correctly and extracting all its fields. The Java class Zparser was implemented to parse packets of different supported protocols namely IEEE 802.15.4 MAC Layer, 6LoWPAN and ZigBee Network Layers, and ICMP Headers (i.e. RPL routing protocol). This parsing has been verified so that it supports the different open source implementations of these protocols. Decoded packets are assigned to objects of the same type of packets. To this end, we have considered a hierarchical design of packets where:

- Zpacket class represents Physical Layer attributes,
- ZIEEE802154 class inherits from Zpacket and defines IEEE 802.15.4 MAC Layer attributes,
- Z6LOWPAN class represents 6LoWPAN packets,
- ZZigBee class for ZigBee packets,
- ZICMPv6 class for ICMPv6 packets.

These classes implement: (i.)the decoding of all IEEE 802.15.4 frames namely data, beacon, MAC command and acknowledgement frames. (ii.) both 6LoW-PAN stateless compression and context-based compression. (iii.) the decoding of ICMP header including RPL packets.

The role of the data manipulation component is to display packet fields and perform useful statistics. The display is made through a comprehensive GUI that provides the end-user with two viewing modes for packet analysis: plain view, in which the fields of each packet are displayed in a row with a time line panel that shows sequence of exchanged packets, and layered view, similar to WireShark display, in which a packet is displayed layer by layer (i.e. Physical Layer, then MAC Layer, then Network Layer, ...). In addition to display, Z-Monitor provides users with statistical analysis of the traffic including total number of packets, average number of packets, average packet size, etc.

One of the problems of the Z-Monitor first version is that it does not have a storage component. The received packets are automatically decoded and displayed on the screen. This presents a big problem specially when performing statistics. Thus, we have implemented a component storage in the new version of Z-Monitor to make possible the off-line analysis of the incoming traffic. Two types of storage means are proposed: (i.) file storage: packets are dumped into a file formatted as XML or text format. Our storage also supports WireShark format (.pcap) to ensure compatibility with it. (ii.) database storage: packets are inserted as bit-stream records in a remote database. This enables to efficiently share data among users over the Internet and would help to extend Z-Monitor for supporting multiple sniffers in the future. We have implemented the abstract class ZStorage from which two classes ZfileStorage and ZdatabaseStorage inherits for handling file and database storage, respectively.

3.3.3 Features

Several implementations have been released for 6LoWPAN protocol. Z-Monitor supports three implementations:

- BLIP [33]: is an implementation of 6LoWPAN under TinyOS [34] operating system.
- uIPv6: is an implementation of 6LoWPAN under Contiki[35] operating system.
- ContikiRPL: an implementation of the RPL protocol under Contiki operating system.

In addition to these implementations, Z-Monitor provides an interactive GUI and a statistical analysis of the traffic. The configuration of the network was performed using three panels in the old version: the first panel is used to choose the operating system type, the second panel to select the port number and the third panel to set the mote type. To make the configuration easier and faster, we implemented a new configuration panel that gathers all steps together. The new resulting panel is shown in figure 3.4.

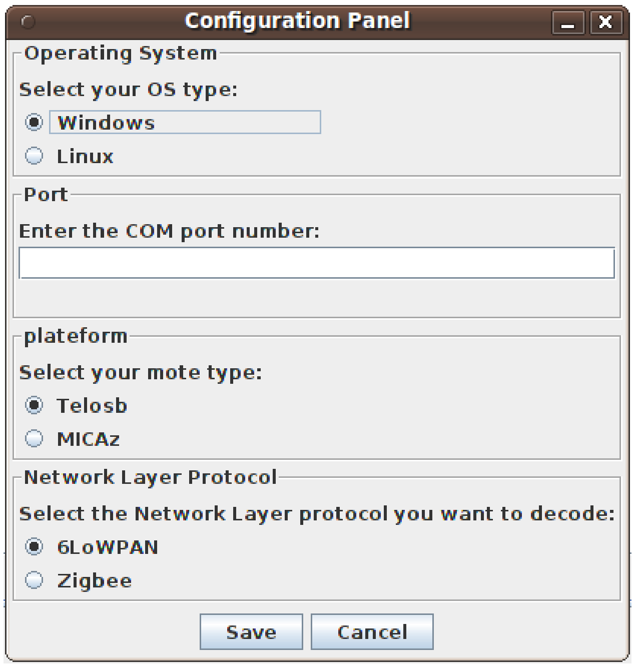
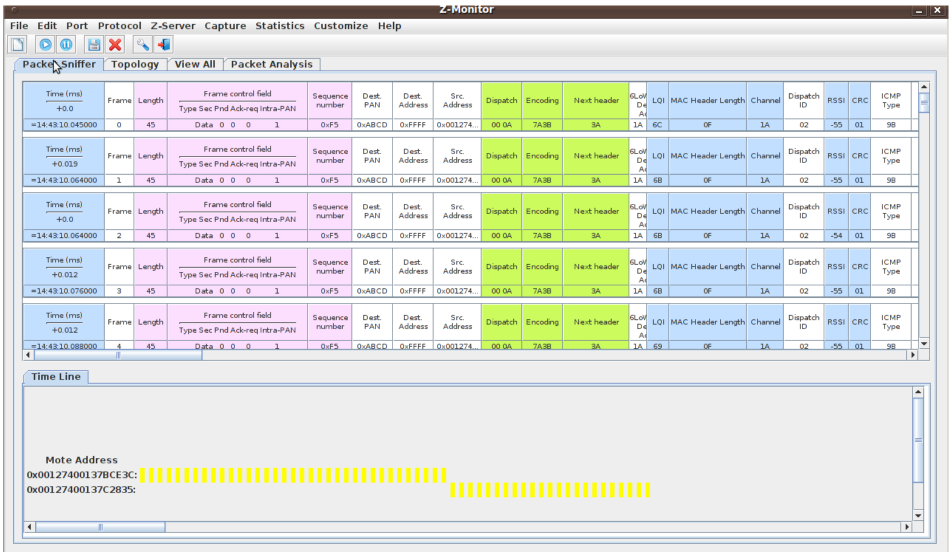


Figure 3.4: Z-Monitor Configuration Panel

As already mentioned, Z-Monitor offers two panels for displaying packets decoding. The first one, called packet sniffer, shows the meaning of each field in the packet. This panel shows also the sequence of the packets received over time. The packet sniffer panel is depicted in figure 3.5.

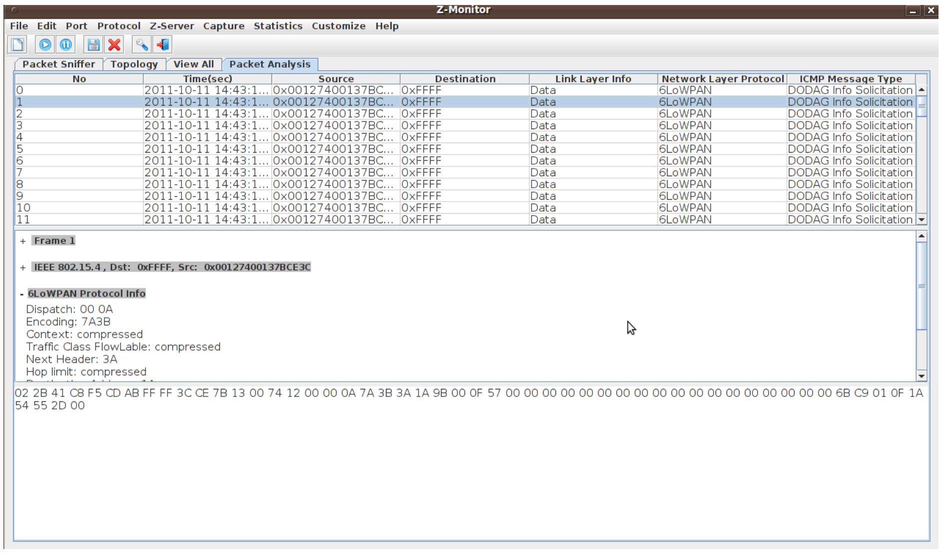

Z-Monitor
File Edit Port Protocol Z-Server Capture Statistics Customize Help
Packet Sniffer
Topology
View All
Packet Analysis
Time Line

Time (ms)	Frame	Length	Frame control field Type Sec Pnd Ack-req Intra-PAN	Sequence number	Dest. PAN	Dest. Address	Src. Address	Dispatch	Encoding	Next header	6LoW De Ac	LQI	MAC Header Length	Channel	Dispatch ID	RSSI	CRC	ICMP Type
+0.0 =14:43:10.045000	0	45	Data 0 0 0 1	0xF5	0xABCD	0xFFFF	0x001274...	00 0A	7A3B	3A	1A	6C	0F	1A	02	-55	01	9B
+0.019 =14:43:10.064000	1	45	Data 0 0 0 1	0xF5	0xABCD	0xFFFF	0x001274...	00 0A	7A3B	3A	1A	6B	0F	1A	02	-55	01	9B
+0.0 =14:43:10.064000	2	45	Data 0 0 0 1	0xF5	0xABCD	0xFFFF	0x001274...	00 0A	7A3B	3A	1A	6B	0F	1A	02	-54	01	9B
+0.012 =14:43:10.076000	3	45	Data 0 0 0 1	0xF5	0xABCD	0xFFFF	0x001274...	00 0A	7A3B	3A	1A	6B	0F	1A	02	-55	01	9B
+0.012 =14:43:10.088000	4	45	Data 0 0 0 1	0xF5	0xABCD	0xFFFF	0x001274...	00 0A	7A3B	3A	1A	69	0F	1A	02	-55	01	9B

Mote Address
0x00127400137BCE3C:
0x00127400137C2835:

Figure 3.5: Packet Sniffer Panel

The second displaying panel is called packet analysis. This panel shows the details of each protocol namely IEEE 802.15.4, 6LoWPAN and ICMP in case of 6LoWPAN implementation. This panel is presented in figure 3.6.


Z-Monitor
File Edit Port Protocol Z-Server Capture Statistics Customize Help
Packet Sniffer
Topology
View All
Packet Analysis

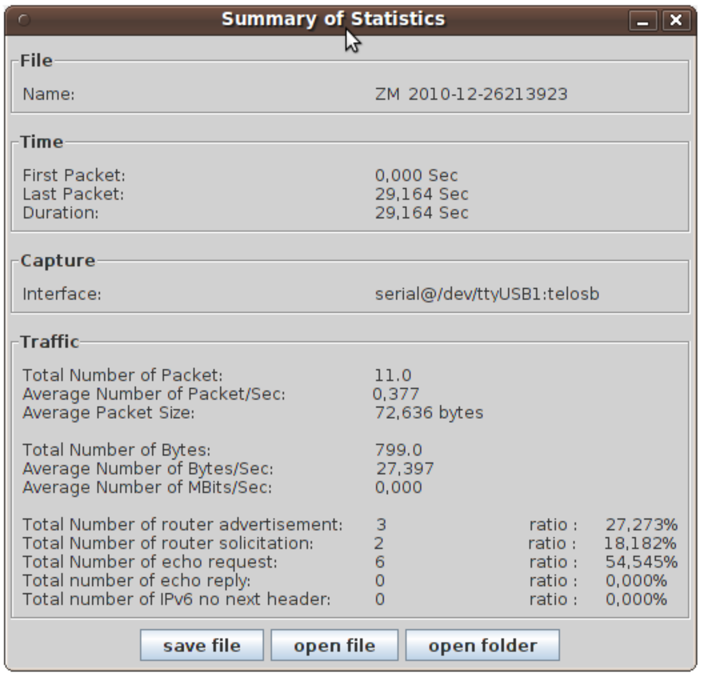
No	Time(sec)	Source	Destination	Link Layer Info	Network Layer Protocol	ICMP Message Type
0	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation
1	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation
2	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation
3	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation
4	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation
5	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation
6	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation
7	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation
8	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation
9	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation
10	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation
11	2011-10-11 14:43:1...	0x00127400137BC...	0xFFFF	Data	6LoWPAN	DODAG Info Solicitation

```
+ Frame 1

+ IEEE 802.15.4 , Dst:  0xFFFF, Src:  0x00127400137BCE3C

- 6LoWPAN Protocol Info
  Dispatch: 00 0A
  Encoding: 7A3B
  Context: compressed
  Traffic Class FlowLable: compressed
  Next Header: 3A
  Hop limit: compressed
```

```
02 2B 41 C8 F5 CD AB FF FF 3C CE 7B 13 00 74 12 00 00 0A 7A 3B 3A 1A 9B 00 0F 57 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 6B C9 01 0F 1A
54 55 2D 00
```

Figure 3.6: Packet Analysis Panel

Another useful panel in Z-Monitor is packet statistics. This panel is used to display useful statistics about the number and the type of packets received by the sniffer. This part of GUI is depicted in figure 3.7.



```
File
Name:                                   ZM 2010-12-26213923

Time
First Packet:                           0,000 Sec
Last Packet:                            29,164 Sec
Duration:                               29,164 Sec

Capture
Interface:                              serial@/dev/ttyUSB1:telosb

Traffic
Total Number of Packet:                 11.0
Average Number of Packet/Sec:           0,377
Average Packet Size:                    72,636 bytes

Total Number of Bytes:                  799.0
Average Number of Bytes/Sec:            27,397
Average Number of MBits/Sec:            0,000

Total Number of router advertisement:   3          ratio :   27,273%
Total Number of router solicitation:    2          ratio :   18,182%
Total Number of echo request:           6          ratio :   54,545%
Total number of echo reply:             0          ratio :   0,000%
Total number of IPv6 no next header:    0          ratio :   0,000%
```

Figure 3.7: Statistics Panel

3.4 Validation

In this section, we present an experimental study that shows how to perform monitoring and performance evaluation of ZigBee, 6LoWPAN and RPL protocols using Z-Monitor. The objectives of the experimental study are manifold:

- To demonstrate the capabilities of Z-Monitor for network monitoring.
- To validate Z-Monitor tools support for various IEEE 802.15.4-based networks.
- To show how Z-Monitor is useful in evaluating the performance of IEEE 802.15.4-based WSNs.
- To present the collection of network statistics using Z-Monitor.

3.4.1 Network Test-Bed

The constructed network consists of 12 nodes: one sniffer mote running tknsniffer TinyOS application, one Base Station running IPBaseStation TinyOS application for BLIP or uip6-bridge for Contiki that does the bridging to the nodes running uIPv6, and 10 identical router nodes distributed around the

Base Station. The network topology scenario used in these experiments is presented in figure 3.8.

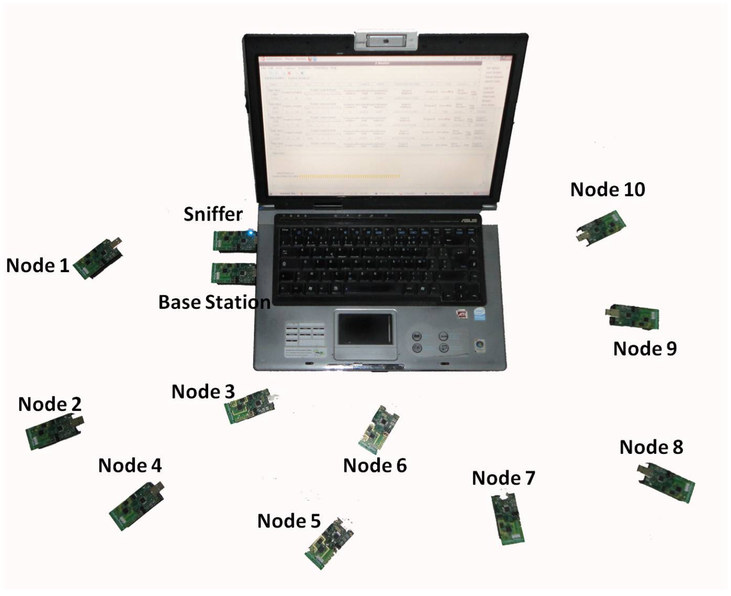


Figure 3.8: Experimental Testbed

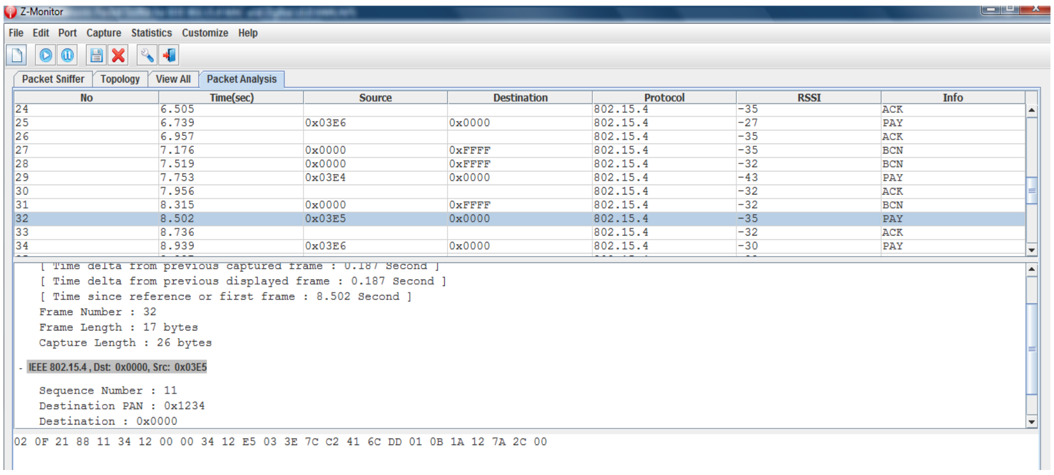
As shown in figure 3.8, TelosB motes were deployed within a single broadcast domain, i.e. a single-hop network. The transmission power of nodes was set to 0 dBm which is the maximum available transmission power. This transmission power makes the communication range of about 50 m. The frequency channel was set to 26. We have considered the available open-source implementations of ZigBee and 6LoWPAN protocols namely, the TinyOS IEEE 802.15.4/ZigBee implementation, the TinyOS 6LoWPAN implementation (BLIP) and the Contiki 6LoWPAN implementation (uIPv6).

In the following sections, we (i.) demonstrate how Z-Monitor is used to monitor and analyze BLIP and uIPv6, (ii.) evaluate and compare the performance of two well-known 6LoWPAN implementations, i.e., uIPv6 on Contiki and BLIP on TinyOS, and, (iii.) evaluate the performance of the recently drafted RPL routing protocol under Contiki operating system (this part will be detailed in the next chapter), using Z-Monitor.

3.4.2 WPAN Monitoring and Analysis of Standard Protocols

Zigbee [9] is a specification ratified by ZigBee Alliance on 2004. It defines a protocol for upper layers ranging from the network layer to the application layer. It uses the IEEE 802.15.4-2003 standard for lower layers. This technology is intended to be less expensive and simpler than other technologies such as Bluetooth. Zigbee is targeted to applications that require long battery life, a low data rate, and secure networking.

Figure 3.9 shows a screenshot of Z-Monitor tool analyzing a ZigBee protocol based on the official TinyOS implementation of the IEEE 802.15.4/ZigBee protocol stack [37]. It is clear from the screenshot that the ZigBee packets are correctly parsed and all the packet types (Beacon, Acknowledgement, Data frames, MAC command) are supported.

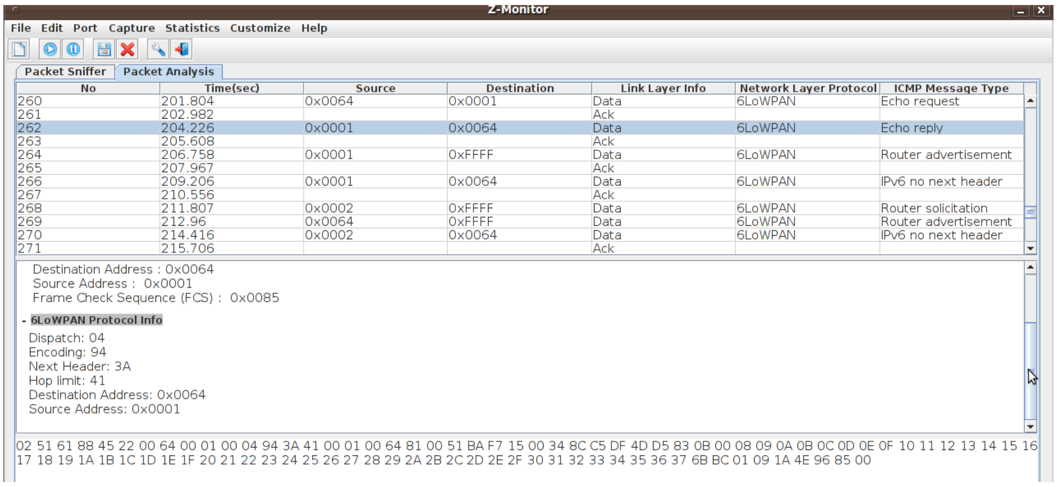


No	Time(sec)	Source	Destination	Protocol	RSSI	Info
24	6.505			802.15.4	-35	ACK
25	6.739	0x03E6	0x0000	802.15.4	-27	PAY
26	6.957			802.15.4	-35	ACK
27	7.176	0x0000	0xFFFF	802.15.4	-35	BCN
28	7.519	0x0000	0xFFFF	802.15.4	-32	BCN
29	7.753	0x03E4	0x0000	802.15.4	-43	PAY
30	7.956			802.15.4	-32	ACK
31	8.315	0x0000	0xFFFF	802.15.4	-32	BCN
32	8.502	0x03E5	0x0000	802.15.4	-35	PAY
33	8.736			802.15.4	-32	ACK
34	8.939	0x03E6	0x0000	802.15.4	-30	PAY

```
    [ Time delta from previous captured frame : 0.187 Second ]
    [ Time delta from previous displayed frame : 0.187 Second ]
    [ Time since reference or first frame : 8.502 Second ]
    Frame Number : 32
    Frame Length : 17 bytes
    Capture Length : 26 bytes

 - IEEE 802.15.4 , Dst: 0x0000, Src: 0x03E5

    Sequence Number : 11
    Destination PAN : 0x1234
    Destination : 0x0000
02 0F 21 88 11 34 12 00 00 34 12 E5 03 3E 7C C2 41 6C DD 01 0B 1A 12 7A 2C 00
```

Figure 3.9: ZigBee Protocol Analysis using Z-Monitor

The experiment consists in monitoring the 6LoWPAN network during the set-up phase (i.e. node joining process with the Base Station). The joining process between a router and the Base Station under BLIP can be observed in Figure 3.10. In this example, we observe the node join process for the node with ID 0x0002 that sends a router solicitation message to the Base Station with ID 0x0064, which responds to its request by sending a router advertisement message. The node 0x0002 joins the network after receiving this network advertisement.



No	Time(sec)	Source	Destination	Link Layer Info	Network Layer Protocol	ICMP Message Type
260	201.804	0x0064	0x0001	Data	6LoWPAN	Echo request
261	202.982			Ack		
262	204.226	0x0001	0x0064	Data	6LoWPAN	Echo reply
263	205.608			Ack		
264	206.758	0x0001	0xFFFF	Data	6LoWPAN	Router advertisement
265	207.967			Ack		
266	209.206	0x0001	0x0064	Data	6LoWPAN	IPv6 no next header
267	210.556			Ack		
268	211.807	0x0002	0xFFFF	Data	6LoWPAN	Router solicitation
269	212.96	0x0064	0xFFFF	Data	6LoWPAN	Router advertisement
270	214.416	0x0002	0x0064	Data	6LoWPAN	IPv6 no next header
271	215.706			Ack		

```
    Destination Address : 0x0064
    Source Address :  0x0001
    Frame Check Sequence (FCS) :  0x0085

 - 6LoWPAN Protocol Info
   Dispatch: 04
   Encoding: 94
   Next Header: 3A
   Hop limit: 41
   Destination Address: 0x0064
   Source Address: 0x0001

02 51 61 88 45 22 00 64 00 01 00 04 94 3A 41 00 01 00 64 81 00 51 BA F7 15 00 34 8C C5 DF 4D D5 83 0B 00 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16
17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 6B BC 01 09 1A 4E 96 85 00
```

Figure 3.10: 6LoWPAN Protocol Analysis using Z-Monitor

It is clear from the above examples that Z-Monitor can effectively handle IEEE 802.15.4-based protocols, and helps in collecting statistical data based on the observed traffic. In what follows, we demonstrate how Z-Monitor can be used to evaluate the performance of the 6LoWPAN and Zigbee protocols, providing another dimension of Z-Monitor capabilities.

3.4.3 Performance Evaluation of Implementations using Z-Monitor

In our study, we measure the network convergence time metric of each router node, which is the duration a node spends to join the 6LoWPAN network for both implementations under study. We also measure the convergence time for RPL-based networks, which is the necessary time to construct the DODAG structure for RPL.

Z-Monitor goes beyond the passive monitoring (i.e. sniffing and displaying) of network traffic as it can be used for analyzing the performance of the IEEE 802.15.4-based protocols. To illustrate this fact, we run experiments where Z-Monitor was used to analyze the network joining time of each node, and consequently the network convergence time. In what follows, we present the experimental results of the evaluation of ZigBee and 6LoWPAN protocols.

3.4.3.1 Performance Evaluation of Zigbee Protocol

To evaluate the performance of Zigbee, We considered a cluster-tree topology composed of one ZigBee Coordinator, three ZigBee routers and seven End Devices. The Beacon Order was set to 8 leading to a Beacon Interval of 3.97 seconds and the Superframe order was set to 4. In this experiment, each node uses the LocalTime interface to compute its joining time to the network and sends the computed values to the ZigBee Coordinator. We observed with Z-Monitor the time when the ZigBee Coordinator receives this message and this time is considered as the joining time of the node. All the 10 motes (routers and end-devices) wake up at the same time. Figure 3.11 shows the joining time of each node captured by Z-Monitor.

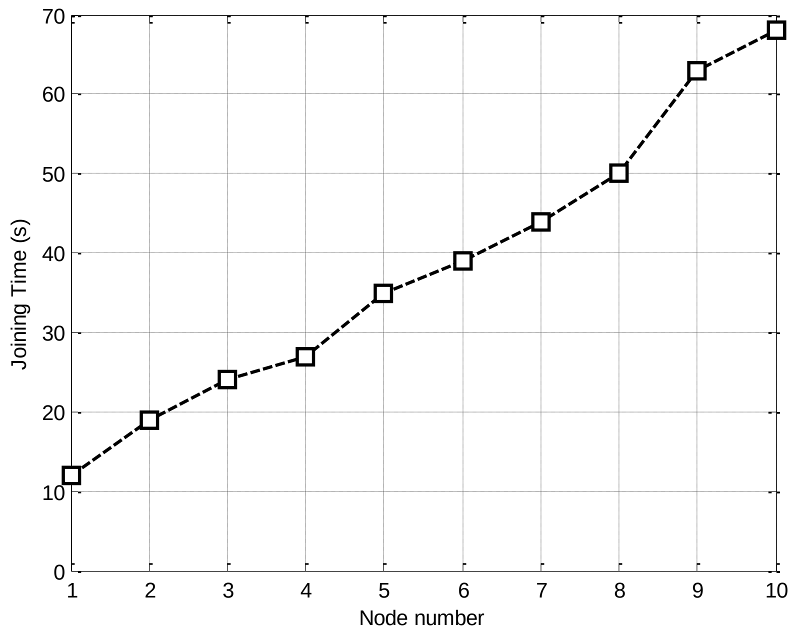


Figure 3.11: Convergence time of ZigBee WPAN (10 nodes)

It is clear from the figure that the first ZigBee router joins the network after three beacons (12 seconds) which is expected. In addition, the joining time grows linearly with the ZigBee network size. This is because ZigBee end-devices cannot join the network before the joining of their parents (the routers).

3.4.3.2 Performance Evaluation of 6LoWPAN Protocol

Figure 3.12 shows the required time by each node to join the 6LoWPAN network for the uIPv6 and BLIP implementations. We powered all the nodes on at the same time and measured the time when the node receives the router advertisement message from the Base Station for both implementations.

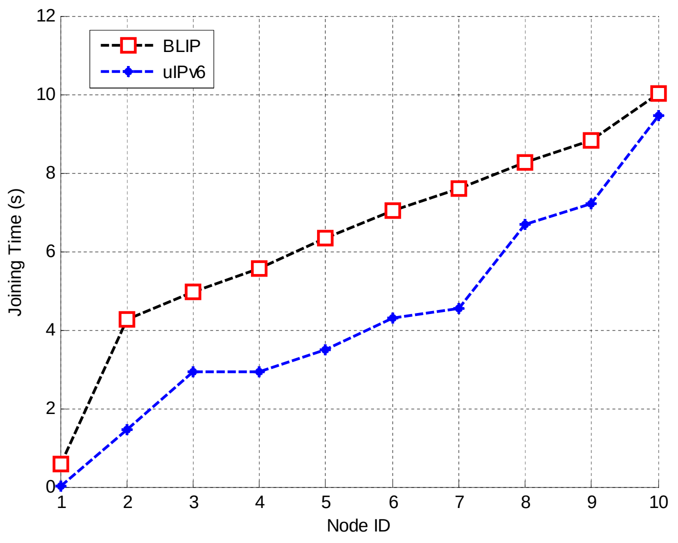


Figure 3.12: Convergence Time of 6LoWPAN with BLIP and uIPv6

It can be observed from Figure 3.12 that the nodes join the network successively although they are deployed at the same time. This is due to the fact that the Base Station is unable to respond simultaneously to the requests of neighbor solicitation when the network becomes bigger. In addition, we observe how the convergence time increases with the network size almost linearly. Also, when comparing the two implementations, it can be seen that the uIPv6 allows a shorter time convergence than BLIP. This is due to the fact that the nodes installed with BLIP send synchronization packets periodically which causes the latency in the joining process.

3.5 Reliability of Z-Monitor

We have run several experiments to test the reliability of Z-Monitor software. The reliability is measured by the packet loss ratio, i.e. the percentage of packets transmitted but not captured by Z-Monitor. Several scenarios were considered in which we varied the number of active nodes (i.e. transmitting nodes) and the duration of the experiment. The scenarios are summarized in Table 3.1. The traffic consists of UDP Echo messages sent from the gateway (IPBaseStation) to the active nodes using the UDPEcho TinyOS application, with a period of 1 packet/second.

Scenario ID	Number of Active Nodes	Duration	Packet Loss Ratio
Scenario 1	1 node	10 minutes	0.00
Scenario 2	2 nodes	10 minutes	0.001
Scenario 3	2 nodes	60 minutes	0.001
Scenario 4	5 nodes	10 minutes	0.00
Scenario 5	5 nodes	30 minutes	0.00
Scenario 6	7 nodes	10 minutes	0.002
Scenario 7	7 nodes	30 minutes	0.00
Scenario 8	7 nodes	60 minutes	0.006

Table 3.1: Experimental Scenarios for Reliability Test

One can observe that the packet loss ratio does not exceed 0.6% in all scenarios, where most of the scenarios show around 100% of successful packet capture. We have also compared the packet loss ratio of Z-Monitor against that observed with the well-known WireShark sniffer, and we observed that both results are very close to each other. This clearly demonstrates the reliability of Z-Monitor. Though, we are currently working towards analyzing the cause of temporary packet losses to reduce them at the maximum aiming at achieving zero loss ratio in future releases.

We argue that the high reliability achieved with Z-Monitor is a result of prioritizing the capture and buffering threads over the parsing and display threads in the Java application. In fact, we granted more priority to the sniffer and buffering threads to avoid losing packets due to the processing of others by the parsing and the display processes.

3.6 Conclusion

In this chapter, we have presented our extension of the monitoring tool and protocol analyzer Z-Monitor. Firstly, we have explained our contribution in the new design of Z-Monitor. Secondly, we presented main features of Z-Monitor. Finally, We demonstrated the capabilities of Z-Monitor to monitor the behavior and evaluate the performance of IEEE 802.15.4, ZigBee and 6LoWPANs. We have used Z-Monitor in conjunction with other tools to evaluate the performance of the RPL protocol and this performance evaluation is detailed in the next chapter.

CHAPTER 4 RPL PERFORMANCE EVALUATION

Contents

4.1 Introduction

In this chapter, we pursue the performance evaluation of the RPL routing protocol. This performance study is conducted with real implementation under Contiki operating system called ContikiRPL [38]. In the literature, several research works have evaluated the performance of the RPL protocol from different perspectives. Their goals were roughly to understand and analyze the behavior of certain mechanisms specified in the IETF draft and to propose some enhancements to potential shortcomings and open issues. Firstly, we describe these works and present their results. Secondly, we give an overview on the RPL protocol. Then, a description of the experimental methodology is introduced. In the final part, we present and discuss the experimental results.

4.2 Related Works

Several works have proposed simulation and experimental models for the sake of experimenting with RPL and evaluating its performance. In [39], the authors presented a performance evaluation study of RPL through an OMNET++ simulation model. They used ETX as a default link metric to build the DAG. Simulation results showed that the performance of P2P routing in RPL for the considered topology is close to the shortest path between source and destination and it is better when the root is in the middle. In addition, they found that 90% of the nodes need to store less than 20 entries in their routing tables. They found also that for a node closer to the sink, the data packet amount is much higher than control packet. For leaf nodes, the amount of control packets are more than data packets. They demonstrated the efficiency of the trickle timer in controlling the packet overhead and stabilizing the network. In addition, the paper demonstrated the significant effect of the global repair duration on the number of control packet overhead. However, the simulations were performed for a small-scale network, preventing the generalization of results for large-scale networks.

In [40], the authors compared between the P2P routing based on RPL and the simple shortest path routing algorithm. They demonstrated via NS-2 simulations of a large-scale network that the RPL P2P routes are significantly sub-optimal as compared to the minimum cost (shortest) routes, mainly when the source and destination are close to each other. However, the authors did not propose a solution to this problem.

In [41], the authors proposed a performance evaluation of the RPL protocol in the context of smart grid applications. Using OMNET++ simulation model of RPL, they demonstrated the capability of the trickle timer in RPL to bound the control overhead and reduce communication latency. In addition, they demonstrated that RPL quickly performs local repair of link outage and provides a path quality close to an optimized shortest path for an outdoor environment. Also, they showed that in RPL Point-to-Point (P2P) routing, the path quality is not drastically worse than the shortest path. This result even improves when the DAG root is located in the middle of the network. The results showed that, in most cases, the total end-to-end delay is in the order of milli-seconds.

In [42], the authors presented a performance evaluation study of the RPL routing protocol using the Contiki COOJA simulator [43]. They mainly evaluated the network overhead, the throughput and the end-to-end delays for two network sizes of 20 and 100 motes. They concluded that RPL leads to a fast network set-up and limited communication delays. The authors reported a network set-up time, i.e. the time required to let the control overhead drop from 100% to about 25%, of about 10 minutes for 20 nodes, which is relatively high for such a medium-scale networks. They also observed that DAO messages represent the main factor that increases the control overhead of RPL. It can be concluded that RPL is open to further amendments to optimize the network formation process in terms of convergence time and control overhead.

All the works mentioned above evaluated the RPL protocol using simulators. Simulation models are appropriate for providing insights on the RPL protocol behavior. However, in general, these models are not able to report the exact performance of the protocol as they rely on emulated channel models, which may differ from real channels, and make abstraction on hardware resources and their usage. As such, experimental models are of paramount importance to assess the real protocol behavior and performance. Thus, we provide the main efforts that contributed to the implementation of experimental prototypes of the RPL protocol.

4.3 Experiment Settings

To evaluate the performance of the RPL protocol, we deployed a real network composed of 30 motes comprising one sink node acting as the DAG root, and 29 RPL routers. The motes are deployed in an indoor office environment composed of three rooms. all wireless devices are battery powered except the DAG root mote which is connected to a PC through its USB port. The PC runs Z-Monitor to control and analyze the experiments. The transmission power is set to -15 dBm. We considered two scenarios: (i.) a single broadcast domain scenario, where all nodes hear each other, and (ii.) a multiple broadcast domain scenario, where nodes are placed in a multi-hop topology such that the routers are distant from each other with a range varying from 1 to 4 hops. Figure 4.1 shows our experimental testbed for the evaluation of RPL in the multi-hop topology. We have deployed the motes in 3 rooms in the laboratory. We show in this figure one observed instantiation of the parent to child relationship within the formed DAG in the experiment. The node with ID 23 is the DAG root.

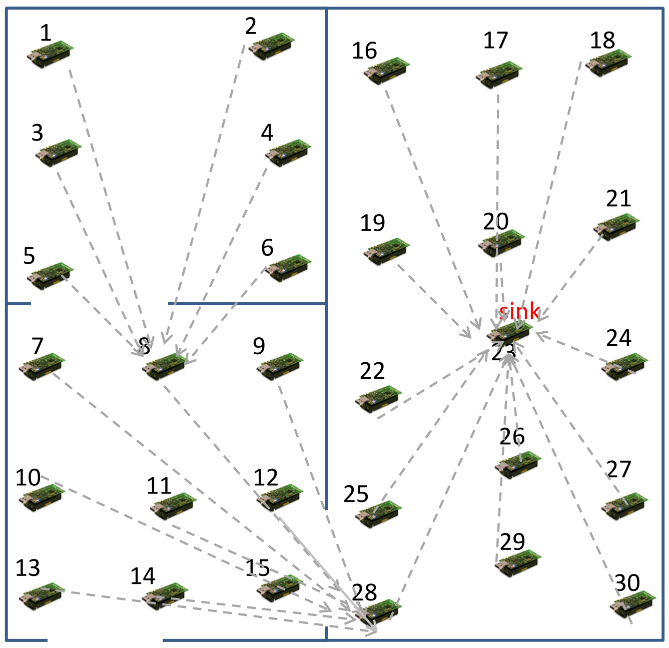


Figure 4.1: Experimental testbed for multi-hop topology

4.3.1 Hardware platform

The hardware platform used in these experiments is *TelosB* [36]. This platform was originally developed at UC Berkeley. Now, it is produced by the Crossbow Technology company. It is a battery powered wireless module with USB programming capabilities. TelosB are equipped with 16-bit RISC MCU clocked at 8 MHz, 16 registers, 10 kB of RAM, 48 kB of flash memory and 16 kB of EEPROM. It integrates an 8MHz Texas Instruments MSP430 micro-controller and a Temperature, Humidity and Light sensors. This structure of the TelosB mote is presented in Figure 4.2.

Figure 4.2: Telosb Module

4.3.2 Software implementation

The RPL implementation used in these experiments is ContikiRPL [1]. ContikiRPL is the first real-world implementation of RPL developed under Contiki operating system [6]. It implements the version 18 of the specification. One of the main features of ContikiRPL is that it provides a simple programming interface for designing and evaluating Objective Functions.

ContikiRPL is structured into layers. microIPv6 layer is responsible of packet forwarding. IPv6 packets are then passed to 6LoWPAN layer for header compression and fragmentation. 6LoWPAN layer in tern sends packets to contiki MAC layer. The default MAC layer in Contiki OS is CSMA/CA. The CSMA/CA mechanism places outgoing packets on a queue. These packets are transmitted in order to the radio duty cycling (RDC) layer. Packets are finally transmitted through the radio link by the RDC link. Figure 4.3 [7] shows the layers that a packet must go through in Contiki OS.

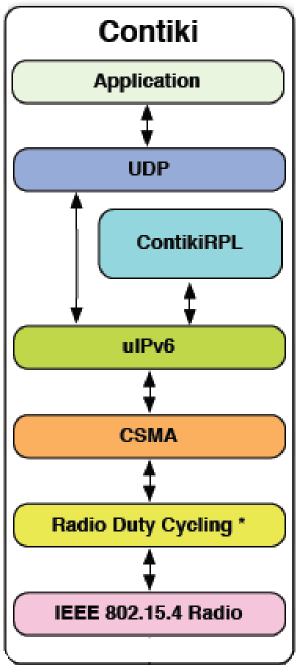


Figure 4.3: ContikiRPL Layers

4.4 Experimental Results

4.4.1 DAG Convergence Time

The DAG convergence time represents the time at which the DAG is completely constructed and all the motes have joined the network. Figure 4.4 shows the average measured convergence time for different network sizes, and for single and multiple broadcast domains. Each experiment is repeated five times and results are presented with 90% confidence interval.

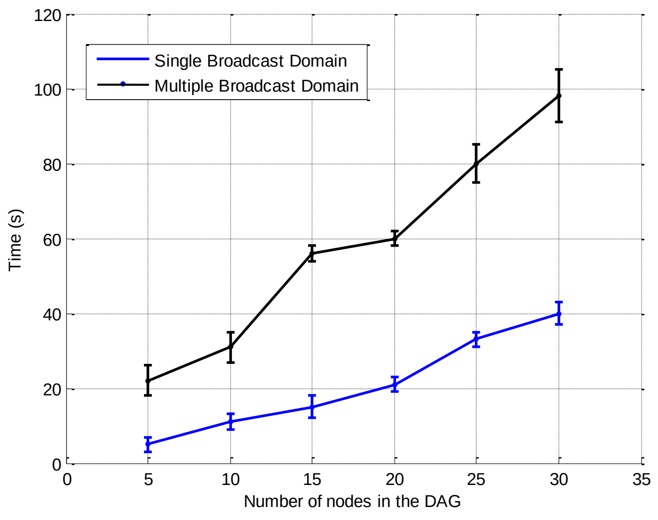


Figure 4.4: Convergence Time for RPL networks

We observe that the convergence time linearly increases with the number of nodes in the DAG for both single and multiple broadcast domains. However, the convergence time in the multiple broadcast domain is at least 4 times larger than that of the single broadcast domain. This illustrates the impact of the number of hops on the time needed to join the network. Furthermore, we notice that the convergence time becomes remarkably large when the number of nodes increases. This is mainly due to three reasons: (i.) the lossy nature of the channel, since control packets need to be retransmitted when they are lost; This confirms the results drawn in [53], where the authors demonstrated using COOJA simulator that the convergence time with a lossy channel is much larger than that with a perfect channel (ii.) The duty cycle radio protocol used in ContikiMAC, which induces additional delays in particular when the topology grows, (iii.) the impact of the trickle timer parameter (Imin is equal to 4s), which makes the DIO retransmissions occur after an important delay.

4.4.2 Power Consumption

The power consumption represents the average energy consumed of each node in the DAG during the DAG construction process. Figure 4.5 shows the average power consumption for different network sizes. The power consumption was measured during the joining process.

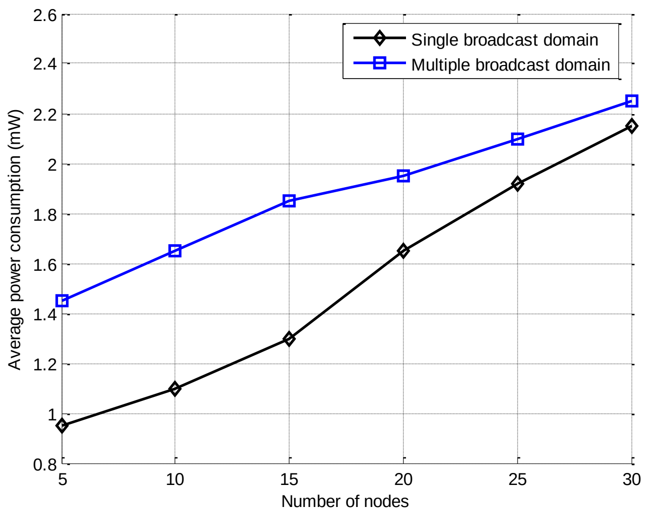


Figure 4.5: Average power consumption

A straightforward observation is that the power consumption increases with the number of routers in the DAG, which is expected. We also observe that the multiple broadcast domain consumes more power in the network set-up process than the single broadcast domain does. However, the gap becomes smaller as the number of nodes increases and the average consumed energy converges towards 2 mW for both scenarios. This demonstrates that the effect of multi-hop becomes less important for large networks and that the number of nodes represents the major factor of energy dissipation.

4.4.3 Packet Loss

For the quest to assess the reliability of RPL, we measured the packet loss ratio, which is defined as the ratio of the number of lost packets to the total number of packets. Packet losses occur when one or more data packets traveling across the DAG fail to reach their destinations. Each router randomly sends Hello data packets to the root at an average rate of 1 packet/minute. We have chosen a large period to avoid overloading the network and prevent collisions. We have used the data collection tool of ContikiOS to collect the Hello data packets at the root. Figure 4.6 depicts the packet loss ratio for different hop counts between the DAG root and a router mote.

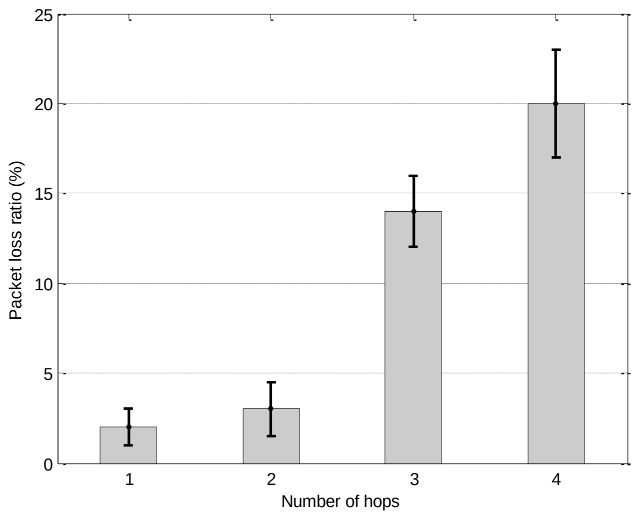


Figure 4.6: Packet Loss Ratio for different hops

Figure 4.6 shows that the packet loss ratio for 1 and 2 hops is relatively low (between 1% to 4%). However, this ratio significantly increases with greater hop counts reaching 20% for 4 hops. This is due to the following reasons: (i.) packet losses have a cumulative effect when the number of hops increases, due to increasing chances for a packet to get lost from one hop to another, (ii.) link quality fluctuations of low power and lossy links, which results in temporary losses of connectivity in particular for those disturbed by obstacles. This was the case of large hop counts in our experiment. These results raise questions about the effectiveness of the default objective function relying on the ETX metric to select routes. Considering more efficient link quality estimators such as 4-Bits [54] and F-LQE [55] is a promising approach to promote the reliability of RPL through the selection of more stable and higher-quality routes. This represents an open research issue in RPL design.

4.4.4 Packet Delay

We evaluated the end-to-end delays in a RPL network to understand its timeliness behavior. We used the Ping application to measure the round-trip time between two nodes placed at a certain number of hops. The packet delay is defined as the duration between the transmission time of the Ping Request message and the reception time of the Ping Reply message. We have used the analyzer tools Wireshark [29] and Z-Monitor [30] for delay measurements. Figure 4.7 shows the measured packet delays for different hop counts between the source and destination with a confidence interval of 90%.

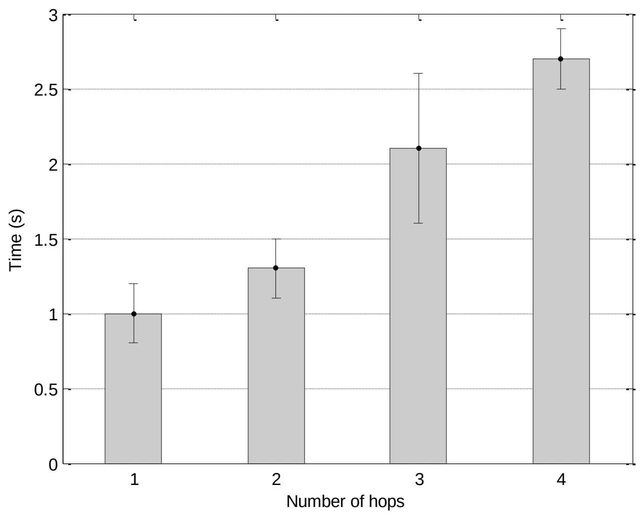


Figure 4.7: Packet delay for different hop count

Figure 4.7 shows that the packet delay increases almost linearly with the number of hops between the source and destination. It varies from 1 second for single hop distant motes to 2.7s for motes that are 4 hops away. This represents an acceptable real-time performance considering the low-power and resource limitation nature of sensor nodes. However, there is a room and a need to improve the timeliness performance of RPL through the adoption of QoS mechanisms to optimize the routing process and reduce end-to-end delays.

4.5 Conclusion

In this chapter, we have detailed our experimental performance evaluation of the RPL routing protocol. The main conclusion from this experimental work is that RPL has several benefits for LLN networks as it minimizes the power consumption of the nodes mainly in sparse networks and does not introduce much latency in receiving packets.

However, the RPL protocol requires much convergence time mainly in large scale networks, and causes packet losses when the motes are far from the DAG root. These issues should be considered by researchers in order to improve the quality of service of the RPL protocol.

CHAPTER 5 CONCLUSIONS AND FUTURE WORK

Low power and Lossy Networks (LLNs) are composed of many embedded devices with limited power, memory, and processing resources. They are interconnected by a variety of links, such as IEEE 802.15.4, low power WiFi, etc. There is a wide scope of application areas for LLNs, including industrial monitoring, building automation, healthcare, environmental monitoring, etc. The lack of an IP-based network architecture prohibited sensor networks from interoperating with the Internet, limiting their real-world impact. To overcome this disconnect, the IETF has specified standards at various layers of the protocol stack with the goal of connecting low-power and lossy networks to the Internet.

One very promising technology is 6LoWPAN, which consists in integrating the IPv6 network layer on top of IEEE 802.15.4 MAC layer, as a complete protocol stack for tiny sensor nodes. This integration enables the interoperability with the Internet, which enable to deploy, monitor wireless sensor networks as large-scale. While 6loWPAN is considered as the most promising technology for enabling such a large scale deployment and interoperability, there are several issues that need to be addressed to ensure its adequacy with WSNs requirements, mainly in terms of energy and limited resources of sensor nodes. In addition, there is a need to discover the behavior of this protocol in a real-world deployment.

This thesis addressed the problem of evaluating the performance of 6LoWPAN based networks and the RPL routing protocol. The reason behind the choice of RPL is that it represents the main candidate for acting as the standard routing protocol for 6LoWPAN networks that allows the integration of IPv6 into wireless sensor networks. To facilitate this performance evaluation and to be sure of the accuracy of the results, it was necessary to use a monitoring tool. RPL specification was recently released. Therefore, it is not supported by the most of the monitoring tools. After analyzing all the solutions, we assert that there is a need of a LoWPAN network monitoring and analyzing tool that is not only compatible with latest technologies like 6LoWPAN and RPL, but also provides extensibility. In this master project, we extended the Z-Monitor monitoring tool to overcome most of the shortcoming of the existing solutions and to support 6LoWPAN and RPL protocols.

In this Thesis, we started by reorganizing the original Z-Monitor code and identified its limits. In a second step, we have designed a new sniffing approach for packet decoding. Then, we have adapted Z-Monitor to support three open source implementations: two for 6LoWPAN (BLIP and uIPv6) and one for the RPL protocol (ContikiRPL). Finally, we added some features that would help later on in the performance evaluation (Chapter 3).

Our developed tool was used to evaluate the performance of 6LoWPAN and the RPL routing protocol in a real implementation (chapter 4). We have measured the packet loss, the packet delay, the convergence time and the average power consumption for different network settings and for different network sizes.

From this experimental study, we concluded that the number of nodes has a great impact on the convergence time. We noticed also that the number of nodes represents the major factor of energy dissipation. In addition, we concluded that the packet loss is high mainly for more distant routers, which raises the question of promoting the reliability of RPL through the selection of more stable and higher quality routes. Packet delays in RPL are acceptable for applications considering the low-power and resource limitation nature of sensor nodes.

In summary, we identified a set of problems related to the behavior of the RPL protocol. Thus, there is a great need to add some improvements to this protocol in order to fulfill the requirements that some LLN applications may impose.

We are currently working towards extending Z-Monitor to support more advanced features including (1) support of multi-hop topologies through the use of multiple sniffers so that it will be easier and practical to analyze the behavior of large scale networks, (2) extending parsing component to support new COTS protocols implementations such as TinyRPL, which has recently been released, (3) integrating advanced filtering and statistical analysis features.

We aim also to propose some improvements to the RPL protocol in order to overcome the drawbacks that were found in our experimental study. Future works include the optimization of the design of RPL Objective Functions in order to fit the application requirements and to optimize the paths between routers in the network.

BIBLIOGRAPHY

[1] Anis Kouba, Mrio Alves, and Eduardo Tovar. Communication Protocols for Wireless Sensor Networks: On the Lower Protocol Layers. Technical report, 9 november 2005.

[2] lan F.Akyildiz and Mehmet Can Vuran. Wireless Sensor Networks: lan F.Akyildiz Series in Communications and Networking. pp9-12, United Kingdom, 2010.

[3] Holger Karl and Andreas Willig. Protocols and Architectures for Wireless Sensor Networks. pp. 5, Chennai, India, july 2007.

[4] Carlos F. García-Hernández, Pablo H. Ibargüengoytia-González, Joaquín García-Hernández, and Jesús A. Pérez-Díaz. Wireless Sensor Networks and Applications: a Survey. IJCSNS International Journal of Computer Science and Network Security, March 2007

[5] Yingshu Li, My T. Thai, and Weili Wu. Wireless Sensor Networks and Applications. chapter1, page 3. USA, 2008.

[6] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. IEEE Communication Magazine, August 2002.

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

Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). New York, USA. September 2006.

[8] IEEE 802.15 Working Group for WPAN. [Online]. Available: http://www.ieee802.org

[9] Zigbee 2006 specification. [Online]. Available: http://www.zigbee.org/

[10] Pedro Jos Marron, Stamatis Karnouskos, and Daniel Minder and Anibal Ollero Editors. The Emerging Domain of Cooperating Objects. pp. 114, chapter 3, London, New York, 2011.

[11] The Internet Engineering Task Force (IETF)working group. [Online]. Available: http://www.ietf.org

[12] The IPv6 over Low power WPAN (6lowpan) working group. [Online]. Available: www.ietf.org/html.charters/6lowpan-charter.html

[13] Zach Shelby and Carsten Bormann. 6LoWPAN: the wireless embedded Internet: Wiley series in communications networking and distributed systems. pp. 16-34, chapter 1, A John Wiley and sons publication.

[14] Jean-Philippe Vasseur and Adam Dunkels. Interconnecting smart objects with IP: the next Internet. pp. 215-236, Burlington, USA, 2010.

[15] 6LoWPAN header compression: RFC 4944 standard. [Online]. Available: http://tools.ietf.org/html/rfc4944

[16] 6LoWPAN header compression: draft-ietf-6lowpan-hc. [Online]. Available: http://tools.ietf.org/html/draft-ietf-6lowpan-hc

[17] Jonathan Hui, David Culler, and Samita Chakrabarti. 6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture: Internet Protocol for Smart Objects (IPSO) Alliance. pp. 13, January 2009.

[18] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger, in: Proceedings of the 3rd international conference on Embedded networked sensor systems. SenSys 05, ACM, pp. 255267, New York, NY, USA, 2005.

[19] M. Ringwald, K. Romer, and A. Vitaletti. Passive inspection of sensor networks, in: Proceedings of the 3rd IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2007). pp. 205222, Santa Fe, New Mexico, USA, 2007.

[20] C. Buschmann, D. Pfisterer, S. Fischer, S. P. Fekete, and A. Kroller. SpyGlass: a Wireless Sensor Network Visualizer. SIGBED Rev. 2 (2005) 16.

[21] Y. Yang, P. Xia, L. Huang, L. Zhou, Y. Xu, and X. Li. SNAMP: A Multi-sniffer and Multi-view Visualization Platform for Wireless Sensor Networks, in: Proceedings of the 2006 1ST IEEE Conference on Industrial Electronics and Applications. pp. 14, IEEE Computer Society, Washington, DC, USA, 2006.

[22] L. Shu, C. Wu, Y. Zhang, J. Chen, L. Wang, and M. Hauswirth. NetTopo: Beyond Simulator and Visualizer for Wireless Sensor Networks. SIGBED Rev. 5, 2008.

[23] J. C. McEachen, T. H. Siang, and G. Kirykos. Design and Implementation of a Modular Wireless Sensor Network Sniffer. DigitalCommons@University of Nebraska - Lincoln.

[24] X. Kuang, and J. Shen. SNDS: A Distributed Monitoring and Protocol Analysis System for Wireless Sensor Network, in: Proceedings of the 2010 Second International Conference on Networks Security. Wireless Communications and Trusted Computing - Volume 02, pp. 422425, NSWCTC 10, IEEE Computer Society, Washington, DC, USA, 2010.

[25] S. A. Chaudhry, G. Boyle, W. Song, and C. J. Sreenan. EMP: A Network Management Protocol for IP-based Wireless Sensor Networks, in: ICWUS10, pp. 16, 2010.

[26] Daintree Sensor Network Analyzer. [Online]. Available: http://www.daintree.net

[27] CC2420 Packet Sniffer, Texas Instruments. [Online]. Available: http://www.ti.com

[28] Zena Network Analyzer. [Online]. Available: http://www.microchip.com/

[29] Wireshark tool. [Online]. Available: http://www.wireshark.org/

[30] Z-Monitor: A Monitoring Tool for IEEE 802.15.4 WPANs (2011). [Online]. Available: http://www.z-monitor.org

[31] Anis Koubâa, Shafique Chaudhry, Olfa Gaddour, Rihab Chaari, Nada Al-Elaiwi, Hanan Al-Soli, and Hichem Boujelben. Z-Monitor: Monitoring and Analyzing IEEE 802.15.4-based Wireless Sensor Networks: The 6th IEEE LCN Workshop on Network Measurements. October 2011.

[32] Tinyos operation system home page. [Online]. Available: http://www.tinyos.net/

[33] BLIP tutorial. [Online]. Available: http://docs.tinyos.net/tinywiki/index.php/BLIP_Tutorial

[34] Tinyos operation system. [Online]. Available: http://www.tinyos.net/

[35] Contiki operating system. [Online]. Available: http://www.contiki-os.org/

[36] TelosB datasheet. [Online]. Available: URLhttp://www.memsic.com

[37] Open-ZB open-source toolset for the IEEE 802.15.4/ZigBee protocols. [Online]. Available: URLhttp://www.open-zb.net

[38] N. Tsiftes, J. Eriksson, N. Finne, O. Fredrik, J. Hglund, and A. Dunkels. A Framework for Low-Power IPv6 Routing Simulation, Experimentation, and Evaluation. SIGCOMM10, New Delhi, India, pp. 479480, November 2010.

[39] J. Tripathi, J. de Oliveira, and J. Vasseur. A Performance Evaluation Study of RPL: Routing Protocol for Low Power and Lossy Networks. Information Sciences and Systems (CISS), 2010 44th Annual Conference on, pp. 1-6, May 2010.

[40] W. Xie, M. Goyal, H. Hosseini, J. Martocci, Y. Bashir, E. Baccelli, and A. Durresi. A Performance Analysis of Point-to-Point Routing along a Directed Acyclic Graph in Low Power and Lossy Networks. 2010 13th International Conference on Network-Based Information Systems, pp. 111-116, 2010.

[41] J. Tripathi, J. de Oliveira, and J. Vasseur. Applicability Study of RPL with Local Repair in Smart Grid Substation Networks. Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on, pp. 1-6, November 2010.

[42] N. Accettura, L. Grieco, G. Boggia, and P. Camarda. Performance analysis of the rpl routing protocol. in Mechatronics (ICM), 2011 IEEE International Conference on, april 2011, pp. 767 772.

[43] Contiki Operating System. [Online]. Available: `http://www.sics.se/contiki/`

[44] T. Winter and P. Thubert. RPL: IPv6 Routing Protocol for Low Power and Lossy Networks. IETF Internet-Draft draft-dt-roll-rpl.txt, vol. 3, 2010.

[45] J. Vasseur. Terminology in Low power and Lossy Networks. IETF Internet Draft: draft-ietf-roll-terminology-04.txt, September 2010.

[46] O. Gnawali and P. Levis, The ETX Objective Function for RPL, IETF Internet Draft: draftgnawali-roll-etxof-00, 2010. [Online]. Available: http://tools.ietf.org/html/draft-gnawali-roll-etxof-00

[47] E. P. Thubert, RPL Objective Function 0, IETF Internet Draft: draft-ietf-roll-of0-03, 2010. [Online]. Available: http://tools.ietf.org/html/draft-ietf-roll-of0-12

[48] A. Conta, S. Deering, and E. M. Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC: 4443, March 2006.

[49] A. Tavakoli. draft-tavakoli-hydro-01. HYDRO: A Hybrid Routing Protocol for Lossy and Low Power Networks, September 2009.

[50] K. Kim, S. Yoo, J. Park, S. Daniel Park and J Lee. draft-deniel-6lowpan-hilow-hierarchical-routing-00.txt, Hierarchical Routing over 6LoWPAN Hilow, December 2005.

[51] K. Kim, S. Park, I. Chakeres and C. Perkins. draft-montenegro-6lowpan-dymo-low-routing-03, Dynamic MANET On-demand for 6LoWPAN (DYMO-low) Routing, June 2007.

[52] Routing Over Low power and Lossy networks (ROLL). [Online]. Available: https://datatracker.ietf.org/wg/roll/charter/

[53] JeongGil Ko, Stephen Dawson-Haggerty, Omprakash Gnawali, David Culler and Andreas Terzis. Evaluating the Performance of RPL and 6LoWPAN in TinyOS. Workshop on Extending the Internet to Low power and Lossy Networks (IP+SN), April 2011.

[54] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson and Philip Levis. Four Bit Wireless Link Estimation, Proceedings of the Sixth Workshop on Hot Topics in Networks (HotNets VI), 2007.

[55] Nouha Baccour, Anis Koubâa, Habib Youssef, Maissa Ben Jamâa, Denis do Rosário, Mário Alves and Leandro Buss Becker. F-LQE: A Fuzzy Link Quality Estimator for Wireless Sensor Networks, EWSN, 2010.

[56] Olfa Gaddour, Anis Koubâa, Rihab Chaâri, Fernando Royo, Nada Al-Elaiwi, Hanan Al-Soli, Stefano Tennina, and Hichem Boujelben. Demo

Abstract: Z-Monitor: A Monitoring Software for IEEE 802.15.4 Wireless Sensor Networks. 37th IEEE Conference on Local Computer Networks (LCN), October 2011, Germany.