POLITECNICO DI TORINO

Master degree course in Communication and Computer Networks Engineering

Master Degree Thesis

Open Source Solutions For Industrial Internet of Things

Supervisors

Prof. Cumani Sandro (DAUIN)

Prof. Cena Gianluca (CNR-IEIIT)

Prof. Scanzio Stefano (CNR-IEIIT)

Candidate

BIRU BIRUK ABDISSA

matricola 230079

December 2018

Summary

Industrial Internet of Things (IIoT) plays an important role in industries, fundamentally by enabling the accessibility of huge amount of data at high speed more efficiently and effectively, by incorporating the use of massive data, analytical tools, and wireless networks. Wireless Sensor Networks (WSNs) are important in a wide area of Internet of Things (IoT) applications. WSNs are a group of sensors responsible for collecting, analyzing and distributing data using a wireless link to a central storage or other networks types through a gateway. WSNs are rapidly emerging as a new type of distributed systems, with applications in different areas such as in industries, automotive areas, building environment, traffic management, etc. OpenWSN is an open source implementation of a complete protocol stack for IoT. To achieve reliable communication and to get an effective and efficient performance a network requires special routing mechanisms at the network layer. RPL protocol is a mechanism that is specified and developed in order to meet these requirements. RPL is an IPv6 routing protocol developed for low-power and lossy networks, where motes and routers are expected to be power-constrained. This thesis studies the RPL protocol and how to get reliable data transmission with no loss, low and deterministic latency in IIoT technology on industrial networks using open source hardware platform called OpenMote B by measuring the performance of the network with respect to the link. Simulation was performed using the OpenSim emulator, which allows simulating OpenWSN network without a real device. The Performance evaluation is based on traffic generation and measurement of relevant metrics, which are round-trip time (RTT) and Packet Delivery Ratio (PDR). The results obtained from the experiments show that the performance of a network will be effective and efficient when motes are near to the grounded root, or a mote has

a minimum hop with respect to the root mote. As future work, the performance of a network should be measured with respect to power consumption and in real devices in different scenarios.

Acknowledgements

First and foremost, I would like to take this opportunity to express my sincere appreciation and gratitude to my supervisors, *Prof. Cumani Sandro, prof. Cena Gianluca, prof. Scanzio Stefano* for their friendly assistance, advice, and my special thank to *prof. Scanzio Stefano* opening his door to me every time I could go to him with my technical question and dedicated feedback throughout the process of writing my thesis.

BIRU BIRUK ABDISSA

Contents

List of Tables

List of Figures

List of Acronym

IoT Internet of Things

IIoT Industrial Internet of Things

WSN Wireless Sensor Network

OpenWSN Open Source Wireless Sensor Network

TSCH Time Slotted Channel Hopping

IETF Internet Engineering Task Force

6TiSCH IPv6 over Time Slotted Channel Hopping

6LoWPAN IPv6 over Low-Power Wireless Personal Area Network

RPL Routing Protocol for Low-Power and Lossy Network

CoAP Constrained Application Protocol

DAG Directed Asyclic Graph

DODAG Destination Oriented Directed Asyclic Graph

ICMPv6 Internet Control Message Protocol version 6

Chapter 1

Introduction

1.1 General context

These days the Internet is going to change rapidly. Now there is a massive global network that allows people to communicate with each other. We use websites, send messages to communicate and to share data. The people who use the Internet and the data which come from client devices (laptop, personal computer, smartphone, tablet, etc.) go to some servers and those servers transmit that data or that information further. Due to this, we can say that the Internet is made up of three major actors, these are the people (client), the client devices they use and the servers. But now a new category (actors) has been added to the Internet, which are called 'Internet of Things (IoT)'.

IoT is the concept that any device is connected to the Internet and the devices connected to each other. IoT is any object or device that has a sensor attached to it that can transmit a data from that sensor into the Internet or into the cloud, where it can be analyzed or used to make decisions to fulfill specific needs. The sensors are connected to different platforms. The aim of these sensors is to target specifically what information is important and useful, and what can safely be ignored. This information can be used to notify the user with a suggestion or a recommendation, by the device to detect a specific pattern or to detect possible problems before they occur.

The impact of the IoT revolution is disruptive. Industries and homes are already registering major changes in efficiency and productivity. The transport sector is getting a new face of autonomous cars, connected street lights switch off to conserve energy when nobody is in the streets, automated traffic monitoring is controlling congestion, smart metering systems in cities and homes, waste management, entertainment, and an endless list of possibilities. These IoT devices applied in different aspects to simplify our lives, e.g., in a building and home automation, in smart cities, in smart manufacturing (industries), in automotive, in wearables object, in health care systems, in precision agriculture and many other aspects of our lives.

Industrial IoT (IIoT) is a branch of Internet of Things, it is the application of IoT to manufacturing industries. It changes fundamentally industries by enabling the acquisition and accessibility of a large amount of data at greater speed and more efficiently than before by integrating big data, analytical tools and wireless networks with physical and industrial equipment. There are different products in terms of hardware, software, and protocols, for IIoT implemented for commercial and open source markets. These products will open new possibilities, prototyping, and maintenance helps to exchange a large amount of data in a fast and more efficient way [2].

1.2 Objectives

The aim of this thesis is to study the Routing Protocol for Low-power and Lossy Networks (RPL) and to analyze the performance of the network generated by the RPL protocol. Experiments were performed by using OpenSim emulator in different scenarios and variable link qualities, and using as metrics the Round Trip Time (RTT) and Packet Delivery Ratio (PDR). We also propose a recommendation to improve the communication quality.

1.3 Organization of the manuscript

The remainder of this thesis is organized as follows: Chapter 2 starts with a presentation of the basics of wireless sensor networks design, characteristics, architecture, applications, and their challenges. Chapter 3 is dedicated to the open source WSN technologies and the OpenWSN protocol stack. In Chapter 4 a detailed overview of the Routing Protocol for Low-power and Lossy Networks (RPL) was presented. It will describe how routing is done, what are the routing requirements, the RPL control messages, and how the routing topology is constructed. Chapter 5 is dedicated to the simulation and execution platforms of OpenWSN. It focuses on the overview of simulation platforms and the configuration of the OpenSim emulator. Chapter 6 is dedicated to reporting experimental results obtained through simulation. In this chapter, the performance of the solution in terms of packet delivery ratio and latency is analyzed using the round trip time.

Finally, a concluding chapter highlights the main contribution of this thesis and outlines future works and perspectives.

Chapter 2

Wireless Sensor Networks

2.1 Overview of Wireless Sensor Network

A wireless sensor network (WSN) can be defined as a network spatially dispersed and composed of dedicated sensors (possibly low-size, low-complexity and low-cost devices) denoted as motes, which can sense the environment and communicate the information gathered from the monitored field through wireless links. Data is forwarded, possibly through multiple hops. The WSN is connected to other networks (e.g., the Internet) through a gateway [34].

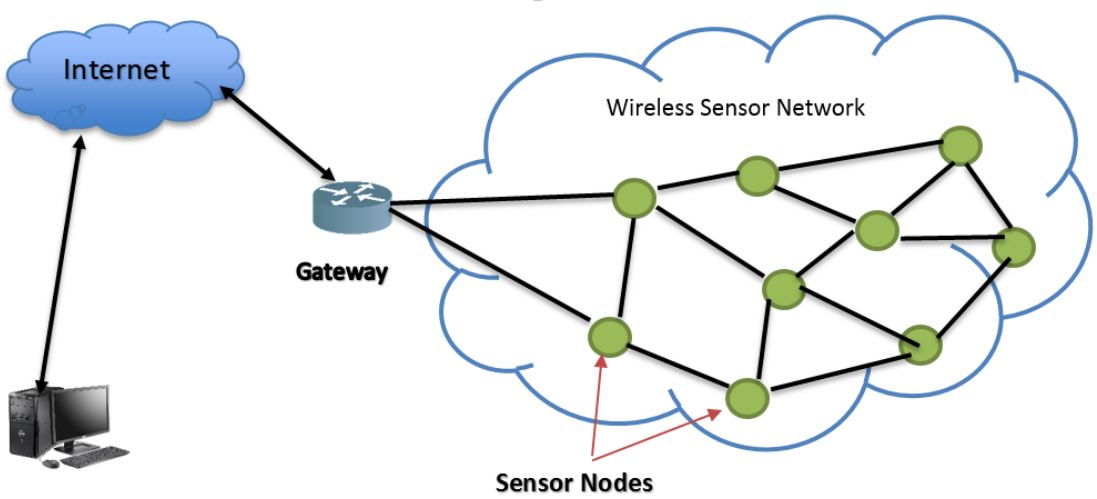


Figure 2.1: wireless sensor network model

WSNs are formed by hundreds or thousands of motes that communicate with each other and exchange data through paths that can involve more than one node. Motes are the building blocks of WSNs. They have very low cost and low power consumption. They can monitor one or more sensors, and they are connected to the outside world with a Radio Link. WSNs can be homogeneous or heterogeneous. Motes can be stationary or moving and they can be aware of their location or not. WSNs were initially designed to facilitate the battlefield surveillance of military operations, but after that its application was extended to new domains like health, traffic, and many other consumer and industrial areas. WSNs consist of protocols and algorithms with self-organizing capabilities [34].

2.2 Characteristics of WSNs

A WSN has several characteristics. The main ones include power efficiency, scalability, responsiveness, reliability, and mobility. A wireless sensor network with these characteristics can be profitably used in many applications. If a WSNs does not fulfill or ensure the characteristics mentioned above, its applicability in real application contexts is dramatically reduced.

- **Power efficiency:**

A typical WSN mote is small in physical size and in power. This implies that computation, communication, and memory resources in motes are a very limited resource.

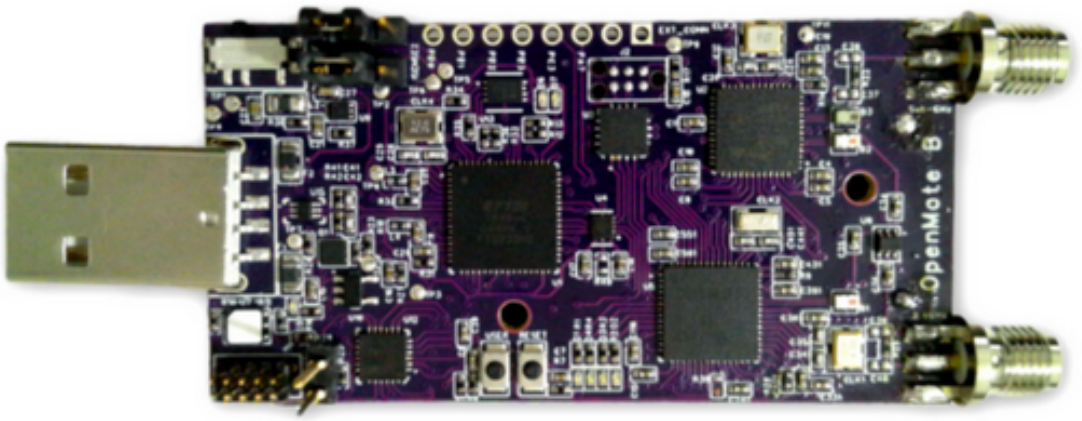


Figure 2.2: OpenMote B Hardware (Source: [36])

Due to these limitation, it becomes harder to design a large scale WSN. In fact, the most important problem (i.e. battery powered motes) can be counteracted only increasing the efficiency level of the device and of the communication protocol. The capability of the device to consume less energy by operating under extremely low power cosumption levels is an important prerequisite for such a kind of networks. This is a very important factor, since WSNs motes and their sensors are usually located far from power sources. Thus, these devices are usually built with the ability to work only with batteries, and eventually, they can make use of energy harvesting techniques to charge them. The optimal method of design is aimed at the maximum reduction of the duty cycle of each node. Additionally, wireless sensors are often put to sleep in order to save power, and thus are unresponsive to neighbor communication [20].

- **Scalability:**

The ability of a network to grow in terms of the number of motes connected to the wireless sensor network i.e. its scalability, is essential. WSN is built of "motes" from a few to several hundred or even thousand, and each mote is connected to a relatively high number of sensors. There is a higher chance that a communication link brokes as the network size increases. When the network size increases, the bandwidth available for applications decreases because a consistent percentage of

the available bandwidth is used by the control message aimed at managing the WSN. Several algorithms were proposed in the scientific literature to solve this performance degradation [20].

- **Responsiveness:**

The ability of a network to quickly adapt itself to change in the topology is named responsiveness. WSNs may change their topology due to their dynamic nature. In a typical WSNs node transmissions are susceptible to error because of environmental interference, and mobility of nodes. To obtain high responsive networks a compromise with other characteristics must be achieved. In particular, in a high dynamic environment both latency and scalability decrease [20].

- **Reliability:**

A high level of reliability is mandatory in any network. Consiquently, this is a basic requirement also for WSN. Reliable data transmission must be guaranteed also in the case of continuous changes in the network topology. This is a difficult requirement to be achived in real implemetations.

- **Mobility:**

Mobility is the networks ability to handle mobile motes and changeable data paths. A good design is necessary for wireless sensor networks to be highly responsive, in order to deal with mobility. As a result, it becomes harder to design a network with high scalability as well as a network capable to manage efficiently motes mobility.

2.2.1 Unique characteristics of WSN

There are some unique characteristics that differentiate WSNs from other communication networks:

- **Communication paradigm :** Compared to traditional communication networks, WSNs are data-centric. This means that communication should be targeted to motes dispersed in a given location and characterized by a specific type of traffic which depends on the type of sensors installed in the mote.

- **Application specific :** Wireless sensor network is implemented to perform a specific task.

- **Deployment :** In some large-scale WSNs, the deployment of motes is random and their maintenance and replacement is impractical. This implies that WSN motes should be reconfigured and reprogrammed remotely..

2.3 Architecture of Wireless Sensor Networks

Most common architectures for WSN follow the OSI Model. Basically, in sensor networks five layer are defined: application layer, transport layer, network layer, data link layer, and physical layer. In addition to the five layers there are three cross layers planes, as shown in Figure 2.3.

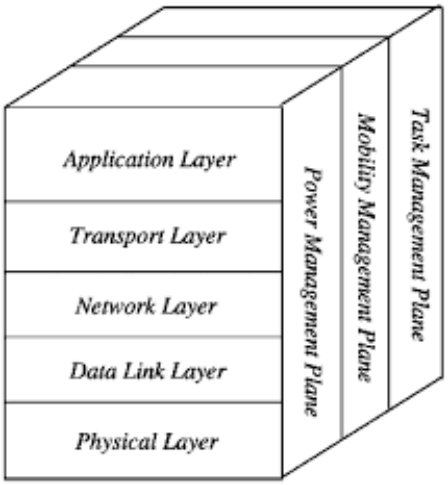


Figure 2.3: Wireless Sensor Network Architecture (Source: [37])

The three cross-layer plains are: power management plane, mobility management plane and task management plane. These layers are used to manage the network

and make the sensors work together, in order to increase the overall efficiency of the network.

- **Physical Layer:**

The main purpose of the physical layer in a WSN is modulation and demodulation of the signal used to transmit the data. The function of the physical layer in WSN is to perform the carrier frequency selection and generation, encryption and decryption, signal detection, modulation and demodulation, transmission and reception of data.

WSNs generally transmit in industrial, scientific and medical (ISM) band. Many other standards like IEEE 802.11 and Bluetooth use the same band. Therefore, all systems in this band have to be robust against interference from other communication technologies. The most commonly used protocol for WSN is IEEE 802.15.4. This protocol is designed to complement wireless technologies such as Bluetooth, Wi-Fi, and Ultra-wideband (UWB), and is targeted at commercial sensing applications, where cabled connections are not possible, and where ultralow power and low cost are required.

IEEE 802.15.4 defines wireless personal area networks for low rate devices. It specifies the standards for the Physical and the MAC layer. IEEE 802.15.4 works at a data rate of less than 250 kbps. It works up to a range of 75 m and it supports up to 254 nodes. This protocol is designed especially for networks with a small data rate, which have energy constraints, and which require a good QoS. IEEE 802.15.4 works in three frequency bands. These are 868 MHz (20 kbps data rate), 915MHz (40 kbps data rate) and 2.4GHz (250 kbps data rate). IEEE 802.15.4 use a Direct Sequence Spread Spectrum for modulation, BPSK for the first two bands, and a 16-bit array QPSK for the last band. The first band has 1 channel, the second band has 10 channels and the third band has 16 channels. This is outlined in Figure 2.4 [14], [15].

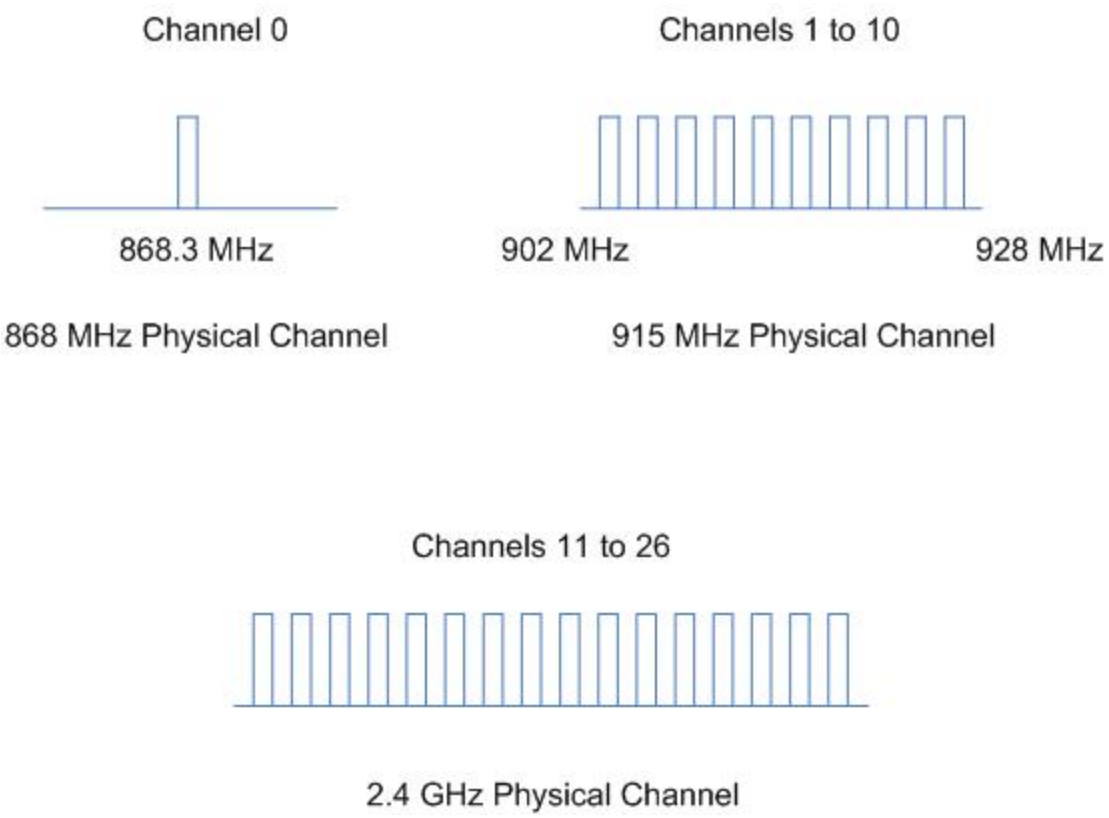


Figure 2.4: 802.15.4 frequency bands (Source: [38])

- **Data-Link Layer:**

The Data-Link Layer is responsible for multiplexing data streams, data frame detection, Media Access Control (MAC) and error control. This layer ensures a reliable implementation of the point-to-point or the point-to-multipoint communication paradigms. Errors or unreliability sometimes comes from Co-channel interference at the MAC layer. This problem can be solved by MAC protocols [16]. Multipath fading and shadowing at the physical layer is another problem that can be solved by *Forward Error Correction (FEC)* and *Automatic Repeat request (ARQ)* techniques.

ARQ, although typically used in WSNs, is not popular because of the additional re-transmission cost and overhead. ARQ is not efficient in addressing frame errors. In fact, all the frame has to retransmited also in the case of an error in a single bit.

FEC decreases the number of retransmissions by adding redundant data on each message. The receiver can use this additional information to detect and correct the errors with this method retransmissions can avoided, and consequently the time reserved to wait for the ACK confirmation frame can be used to schedule new transmissions.

The MAC layer is responsible to manage channel access policies, scheduling, buffer management, and error control. In WSN, we need a MAC protocol that considers energy efficiency, reliability, low access delay, and high throughput. The IEEE 802.15.4 standard, and its evolution IEEE 802.15.4e, is used in wireless sensor networks. The IEEE 802.15.4e standard defines a MAC amendment of the IEEE 802.15.4 standard protocol to support the industrial market in a better way. The main solution proposed in this standard is Time Slotted Channel Hopping (TSCH). Basically, TSCH was designed to allow WSN devices to support a wide range of applications including industrial applications IEEE 802.15.4e. TSCH is used by time synchronized motes to achieve low-power operation and channel hopping (to increase reliability). The IEEE 802.15.4e standard enable the production the latest generation of ultra-lower power and reliable networking solutions for low-power and Lossy networks [17], [18].

- **Network Layer:**

The main function of this network layer is routing the data. This service is needed by the transport layer. Routing protocols are used to define a reliable path (and eventually redundant paths) according to a certain metric, which differs from protocol to protocol. There are many routing protocols available for this layer, they can be divided into flat routing (for example, direct diffusion) and hierarchical routing (for example, LEACH), or can be divided into time driven, query-driven and event-driven. In continuous time driven protocols, the data is sent periodically, and time is the most important physical variable used by the application to periodically monitor the environment. In event-driven and query-driven protocols, sensor motes respond according to actions or user queries. For OpenWSN IPv6 address is used to identify the network interface of a mote instead, to manage routing OpenWSN and

the majority of WSNs use the RPL protocol. The RPL protocol is used for routing in a wide range of Low-power and Lossy network applications (see Chapter 4). The other protocol used at the network layer by OpenWSN and others is 6LoWPAN. 6LoWPAN is the acronym for IPv6 for Low-power wireless personal area networks [19].

- **Transport Layer:**

The function of the transport layer is to provide reliable data delivery (as required by the application layer) and congestion avoidance. There are many protocols designed to provide aimed at implementing this functionality. These protocols use different mechanisms for loss detection (ACK and sequence numbers) and loss recovery (End to End or Hop by Hop). This layer is specifically needed when a system needs to access and communicate with other networks. Providing a reliable hop by hop communication is more energy efficient than end to end. That is one of the reasons why TCP is not a suitable protocol for WSNs, while UDP is more suitable. Usually, the link from a gateway to a node is considered as a downstream link for multicast transmissions, and UDP traffic is suitable because of the limited memory consumption and overhead. In general, transport protocols can be divided into two categories: packet driven and event-driven. In Packet driven all packets sent by the source must reach the destination, while in Event-driven, it is enough that one notification message (an event) reaches the gateway [16].

- **Application Layer:**

The application layer includes a variety of protocols that perform various sensor network applications, such as query dissemination, mote localization, time synchronization, and network security. For example, the Sensor Management Protocol (SMP) is an application-layer management protocol that provides software interfaces to perform a variety of tasks, for example, exchanging location-related data, synchronizing sensor nodes, moving sensor motes, scheduling sensor motes, and querying the status of sensor motes. WSNs at the application layer use Constrained of Application Protocol (CoAP), which is implemented in each OpenWSN devices. By using this protocol motes appear like a web servers on the internet.

2.4 Applications of Wireless Sensor Networks

Nowadays WSNs are popular due to their flexibility in solving different problems in different applications, and making at the end our life much easier. They also have the potential to change our lives in many ways. WSNs can be applied to different areas and to realized and for a variety of applications.

- **Military application:** WSN deployed in the military to control and detect enemy intrusion, for reliable communications, computing, intelligence, battlefield surveillance, and targeting systems [34], [43].

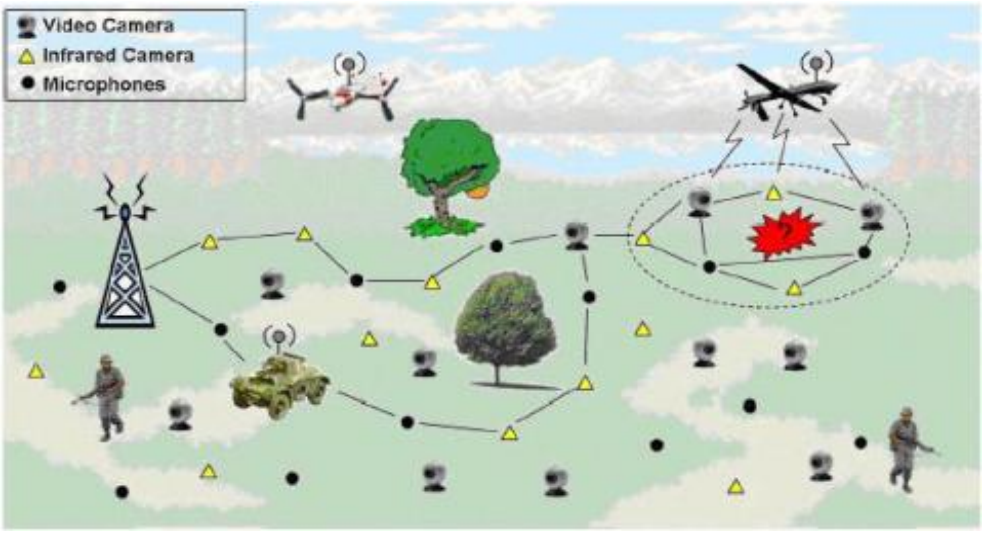


Figure 2.5: WSN for Military Application (Source: [40])

- **Area monitoring application:** The sensor nodes deployed to monitor some phenomena. When these nodes detect anomalies in situations to be monitored like heat, pressure, etc..., they have to report the problem to the base station as soon as possible. After detecting the problem they will take an action to solve the situation [34].

- **Transportation applications:** WSNs are deployed in the transportation system for real-time traffic information, and also to give to the deriver information related to congestion, parking area allocation systems, and traffic problems [43].

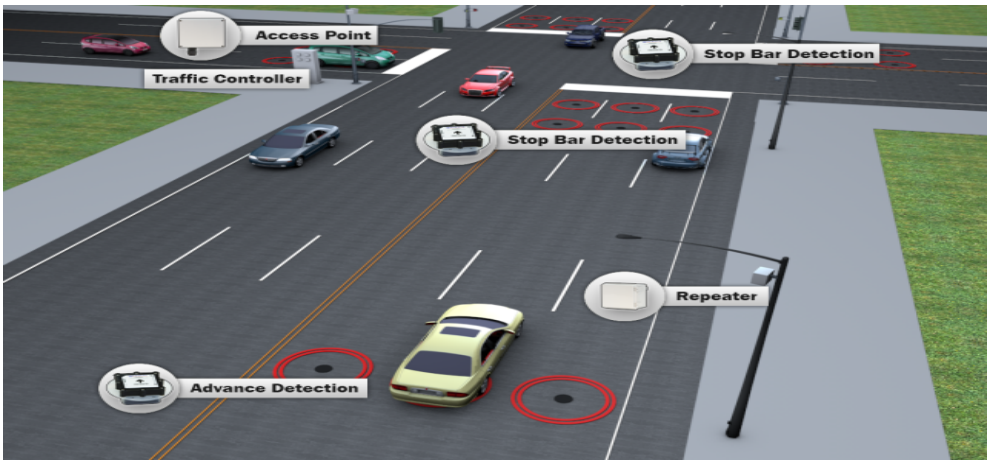


Figure 2.6: WSN for Transport Application (Source: [39])

- **Health applications:** Health applications for sensor networks are supporting interfaces for the disabled, integrated patient monitoring, diagnostics, and drug administration in hospitals, telemonitoring of human physiological data, and tracking and monitoring doctors or patients inside a hospital [34].

- **Environmental sensing applications:** The term environmental sensor networks have developed to cover many applications of WSNs to earth science research. This includes sensing volcanoes, oceans, glaciers, and forests. Some other major areas are Air pollution detection, animal monitoring, forest fire detection, greenhouse monitoring, and landslide detection [34].

- **Industrial monitoring application:** Wireless sensor networks have been developed for machinery Condition-Based Maintenance (CBM), as they offer significant cost savings and enable new functionalities. In wired systems, the installation of enough sensors is often limited by the cost of wiring. CBM is a type of predictive maintenance that uses sensors to measure the status of a prodion device over time while it is in operation. Wireless sensors can be placed in real-life locations to collect data in positions that are difficult to be connected with cables, such as rotating machines. WSNs are also used for data Center monitoring, to collect data for monitoring environmental information, the quality and the level of water, the condition of civil infrastructures

Figure 2.7: WSN for Environmental Applications (Source: [41])

and related geophysical processes close to real time, and for wine production monitoring in the field and also in the cellar [34].

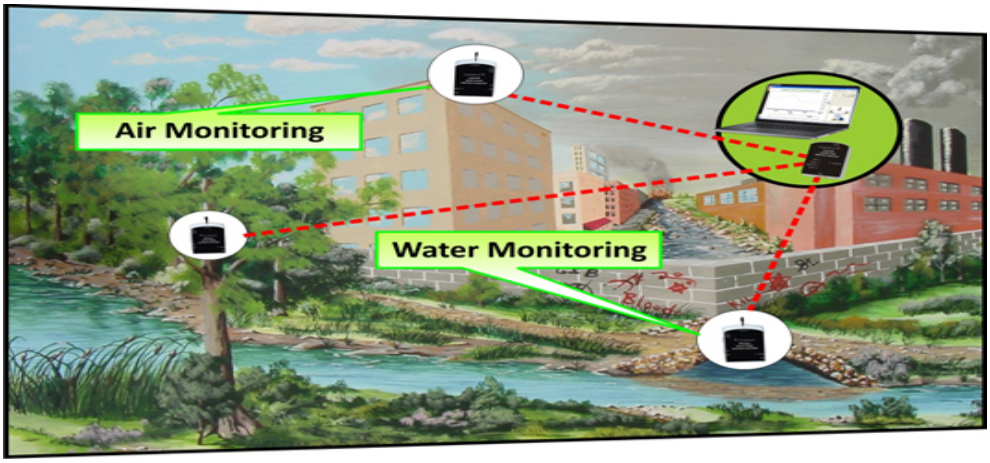


Figure 2.8: WSN for Industrial monitoring applications (Source: [42])

2.5 Wireless Sensor Network challenges

- **Energy:**

WSN applications require a power source, as any other electronic device, to do their activities. Energy is consumed in data collection, data processing, and data

communication. Additionally, continuous listening the medium for MAC operations demands a high amount of energy due to node components like CPU and communication chips. Batteries providing power need to be changed or recharged after they have been consumed. Sometimes, because of demographic conditions of sensors nodes, it will be difficult to recharge or change the batteries [21].

- **Self Management:**

Wireless sensor networks, after they are implemented in a real-world location, should be dynamic and highly adaptive to be able to work properly without any human intervention and they need to be self-organized. WSNs should be able to manage the network configuration, adaptation, maintenance, repair and communicate their current level of activity to the central router by themselves [22].

- **MAC Layer Issues:**

MAC solutions have a direct influence on energy consumption. Some of the primary causes of energy waste are found at the MAC layer. These are collisions due to sensor node trying to send packets in the same transmission channel at the same time, control packet overhead due to extra bytes added in the packet header, and listening of a channel (to see when it becomes idle) without sending any packet. It is not easy to implement power saving forward error control techniques, because of the high computing power they require and due to the big and variable packet size [23].

- **Limited Memory and Storage Space:**

Wireless sensors are small devices with very limited memory and storage space for the code. So, to build a good and effective software, it is necessary to limit the code size. The software designed for wireless sensors should be quite small due to their limited memory and computational resources [23].

- **Physical Attacks and Security:**

A wireless sensor deployed in a real-world location operates in an environment open to adversaries, environmental conditions like bad weather and other factors. Due to that, physical attacks are much more likely than the typical PC, which is located in a secure place and mainly faces attacks from the network. Physical security of wireless sensor motes cannot be assured. Attackers may modify or destroy mote hardware, for instance by replacing it with a malicious sensor or a dummy sensor.

Security is a quite challenging issue as WSN is not only being implemented in the real world like in battlefield applications but also for surveillance, building controlling, alarms and in critical systems such as airports and hospitals. Confidentiality is required in WSNs to protect information traveling between the sensor motes of the network or between the sensors and the base station. Otherwise, it may result in eavesdropping of the communication [23], [24].

Chapter 3

Open Source Wireless Sensor Network (OpenWSN)

3.1 OpenWSN

The OpenWSN project, on which this thesis is based, is an open-source implementation of a fully standardized protocol stack for WSNs, based on the new IEEE 802.15.4e Time Synchronized Channel Hopping standard and on the outcomes of the IPv6 over TSCH mode of IEEE 802.15.4e (6TiSCH) working group. IEEE 802.15.4e, coupled with others Internet of Things standards, such as 6LoWPAN, RPL, and CoAP, enables ultra-low-power and highly reliable mesh networks, which are fully integrated into the IPv6 protocol.

The goal of the OpenWSN project is to investigate the use of IEEE 802.15.4e in Internet-connected low-power lossy networks and to provide an open-source implementation of a complete protocol stack based on Internet of Things standards, on a variety of software and hardware platforms.

3.2 Technologies of open Source low-power wireless Operating Systems

In WSNs, the motes due to their size, low cost and low power consumption have limited resources. Currently, WSN mote technologies consist of processors with really low computational power (e.g. TIMSP43F, TICC2538, ...) and tens of kilobytes of memory. There is also a big variety of operating systems used by the motes. Some of the most known operating systems for WSN technologies are described.

3.2.1 TinyOS

It is an operating system which is flexible, component-based and used for low-power wireless platforms for example in WSN, PAN, Building Automation and smart meters. It supports the event-driven concurrency model. It defines events (which are asynchronous), tasks such as routing, and different computations. TinyOS is implemented in a programming language called NesC, a dialect of C language, which supports TinyOS components, concurrency, cross-component optimizations, compile-time race conditions. TinyOS is an embedded system framework and component-based application-specific operating system [44].

3.2.2 Contiki

Contiki is another operating system used in wireless sensor networks. It is an open source operating system for the Internet of Things. It allow connecting microcontrollers which are tiny, low-cost, and low-power. It also includes a powerful toolbox for building complex wireless systems.

It supports IPv4 and IPv6 along with low-power WSN standards such as RPL, 6LoWPAN, CoAP, and it can be installed in a wide range of low-power wireless devices available on the market. Contiki is highly memory efficient, provides full IP network stack with standard IP protocols and low-power standards, and it is designed to operate in low-power systems. Contiki is power efficient, as a consequence

it can run for long periods of time, supports low-power IPv6 standardized IETF protocols such as RPL multi-hop routing protocol, and CoAP application-layer protocol. Contiki supports dynamic loading and linking of modules at run-time [25].

3.2.3 RIOT

RIOT is a small and free open source operating system for networked, memory-constrained systems, with focus on low-power wireless Internet of Things (IoT) devices and microcontroller architecture (it runs on 32-bit CPUs such as ARM cortex, 16-bit such as TI MSP430, and 8-bit such as AVR Atmega). It implements all relevant open standards of the Internet of Things, including those related to security.

RIOT is IoT friendly, simplifying the implementation of users applications ready for small WSN nodes. Both IPv6 and 6LoWPAN are supported together with UDP, the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), CoAP, and Concise Binary Object Representation (CBOR) are available as application level communication protocols. RIOT allows application programming with the C and C++ programming language and provides full multithreading and real-time capabilities [26].

3.3 OpenWSN protocol stack

The OpenWSN protocol stack is used as the standard communication technology in this thesis. In the physical layer, OpenWSN uses the IEEE 802.15.4e standard to define the physical wireless frequency modulation or demodulation mechanisms. At the MAC layer, the IEEE 802.15.4e standard is used to define how motes communicate with each other. In the network layer, IETF 6LoWPAN is used to compress the header size of the network layer to fit in a 127 bytes packet. In this layer, the RPL routing protocol is used for routing. In addition, this layer defines how the motes communicate with each other by means of multiple hops. Finally, in the

transport layer, UDP is used to provide end-to-end communication over a network. The application layer makes use of CoAP and HTTP. CoAP is a protocol implemented on each OpenWSN device, which makes it appear like a web server on the internet. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. The diagram for the OpenWSN protocol stack shown in figure 3.1.

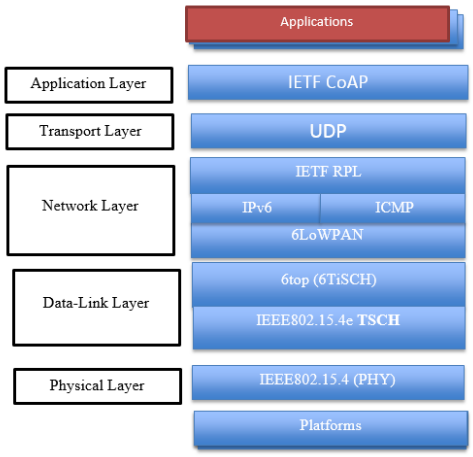


Figure 3.1: OpenWSN protocol stack (Source [35])

3.3.1 IEEE 802.15.4

IEEE 802.15.4 is a standard that defines low-cost and low-power communications and a large number of optional Physical Layers protocols operating in many frequency bands. In addition, IEEE 802.15.4 defines a simple and effective MAC layer as shown in the Figure 2.4, which is targeted for low-power wireless communications between battery-operated wireless devices [27]. The standard provides 127 bytes maximum payload size, 0-10 dBm transmission power and 250 kb/s data rate in 2.4GHz frequency band, which is subdivided 16 channels. IEEE 802.15.4 defines several addressing modes: it allows the use of either IEEE 64-bit extended addresses or 16-bit unique addresses within personal area network (PAN) [28].

In IEEE 802.15.4 standard, the transmissions follow a super-frame structure and nodes use CSMA/CA as channel access mechanism. Two types of CSMA/CA

weredefined: slotted and unslotted. If nodes use slotted CSMA/CA beacon mode is enabled, while if a node uses unslotted CSMA/CA,the non-beacon mode is enabled. The boundaries of the super-frame are defined using beacon frames sent from a personal area coordinator in order to inform nodes on the possibility to joining the network between two consequtive beacons, which are sent every beacon interval (BI) [4].

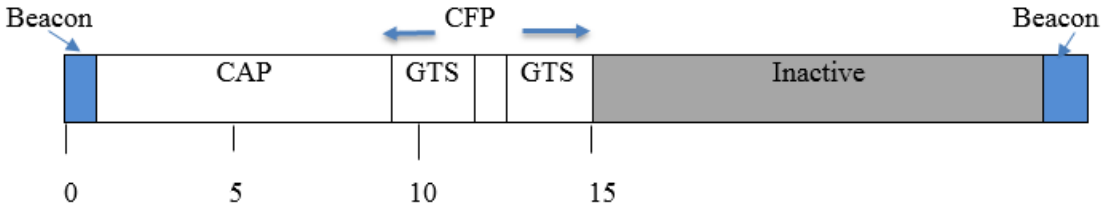


Figure 3.2: Super frame structure in IEEE 802.15.4

As shown in the Figure 3.2, the active period is a combination of the contention access period (CAP) and the contention free period (CFP). During CAP nodes try to join the network using slotted CSMA/CA. CFP is the sum of 16-time slots to offer an exclusive access to the communication medium for nodes with high needs in terms of quality of services. IEEE 802.15.4 has some limitations. The main one is performance under dense implementations in real-world scenarios like smart cities, smart buildings, smart homes, and industrial applications. Due to this, the use of CSMA/CA mechanisms decreases the performance of the MAC standard and increases co-existence problems. If many networks share the same frequency band, each time a node needs to transmit on a specific network, it could collied with other nodes belonging to other networks that might be communicating with a different technologies (such as IEEE 802.11 ot Blutooth) [27].

3.3.2 IEEE 802.15.4e

The IEEE 802.15.4e standard approved in 2012 addresses the problems of interference and multipath fading. It is a revision of the IEEE 802.15.4-2006 standard. This standard increases robustness against external interference and gives better

support in terms of better latencies and reliability for the industrial markets. It defines three types of access mode.

The first one is Deterministic and Synchronous Multichannel Extension (DSME) which defines an extended vision of the superframe structure used in the beacon-enabled mode of IEEE802.15.4, and increases the number of frequency channels. The other access mode is a Low Latency Deterministic Network (LLDN), which is designed for star topologies without the channel-hopping mechanism [29]. The third access mode is called Time Slotted Channel Hopping (TSCH) which was designed to allow IEEE 802.15.4 devices to support a wide range of applications including the latest generation of Ultra-low power and reliable networking solutions for LLNs. In particular, as described in the next subsection, TSCH allows obtaining better WSN with lower latencies, higher reliability, and lower power consuption.

3.3.3 Time Slotted Channel Hopping (TSCH)

Time Slotted Channel Hopping (TSCH) is used to reduce interference and it is particularly suitable for multi-hop mesh networks. IEEE 802.15.4 TSCH networks communicate by following a Time Division Multiple Access (TDMA) schedule.

The time slotted access schedule provides predictable and bounded latencies, guarantees a predefined bandwidth and increases network capacity using multi-channel communication and using different channels, which are identified by different channel offsets. It is also an emerging standard for industrial automation and process control for LLNs. Many ideas were borrowed from previous industrial standards WirelessHART, ISA100.11a, and others. To avoid collisions, the absolute sequence number (ASN) is used to measure time. This time indicatation is sent by the router or the root node and shared by all the nodes [30].

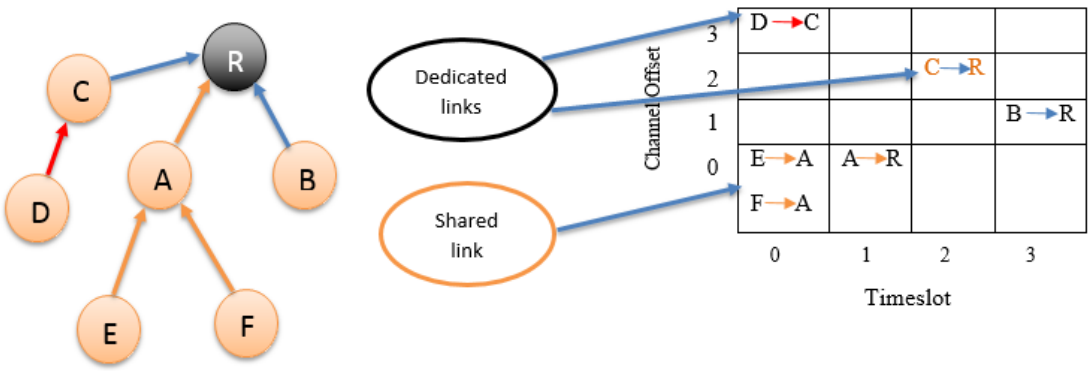


Figure 3.3: slotframe structure in IEEE 802.15.4e TSCH

TSCH is supporting both Dedicated links and Shared links. A "Link" is a Pairwise assignment of a directed communication between devices in a specific slot, with a given channel offset, as shown in the Figure 3.3. Dedicated links are allocated to a single device to device communication. These slots are assigned to one transmitter and one receiver, and the links are accessed without any delay (i.e. there is no need to execute a channel access algorithm because only the two scheduled nodes can use the dedicated slot). Shared links are special slots assigned to multiple transmitters/receivers and can be accessed concurrently by different nodes through the CSMA/CA algorithm, to reduce the probability of collisions. There are 16 channel which are used to offer channel hopping functionalities.

3.3.4 6top

The IPv6 over the TSCH mode of IEEE802.15.4e (6TiSCH) defines the 6top sublayer which is a set of protocols for setting up a TSCH schedule. TSCH is one of the key element of the 6TiSCH stack [30]. The 6top protocol dynamically assigns bandwidth resources to the nodes in the network according to the application requirements. All communications in a 6TiSCH network (i.e. a network thatimplements TSCH) are coordinated by a schedule. The schedule is composed of cells identified by a slot offset in the time-axis and by a channel offset in the frequency-axis. 6top is a 6TiSCH operation sublayer [13]. The 6top protocol which is also named 6p in the newer versions of the standard. 6P allows nodes to communicate

with a neighbor node, to add or delete TSCH cells and to run one or multiple 6top Scheduling Functions (SFs), which define the rules that decide when to add or delete cells. There are hard and soft cells. A soft cell can be read, added, deleted or updated by 6top and a hard cell is read-only for 6top [13], [30].

3.3.5 IETF 6LoWPAN

IPv6 over low-power wireless personal area networks (6LoWPANs) are formed by devices which are compatible with the IEEE802.15.4 standard. It is a wireless network with low-power, where all the nodes have its own IPv6 address, which allows it to connect directly with the internet using open standards [27]. In 6LoWPAN a link is characterized as lossy, supports 16-bit short or 64-bit extended MAC addresses, low-power, low bit rate, short range; with many nodes saving energy with long sleep periods [54]. 6LoWPAN offers end-to-end IP addressable nodes and a router can connect the 6LoWPAN network to IP. It offers self-healing, robust and scalable mesh routing with P2M and M2P routing. The mesh routers of 6LoWPAN can route data destined to others, whereas hosts can sleep for the long duration of time [31], [27].

3.3.6 RPL

RPL is the IPv6 routing protocol for low-power and lossy networks (LLNs). RPL was designed to be suitable for resource-constrained devices in home automation, urban and industrial WSNs and IoT domain, It is considered a critical component that links the low-power network connectivity to the application layers in IETF protocol suite for LLNs. It operates on top of 6LoWPAN to maintain a routing topology. Both collection and source routing mechanisms are implemented in RPL. The routing layer is responsible for relaying packets across multiple hops, separating the source and destination nodes. LLNs are a class of networks in which both the router and their interconnection are constrained. RPL uses an Objective Function (OF) which is used to specify how a specific route has to be weighted. The main goal of RPL is to provide IPv6 connectivity to a large number of battery-operated

embedded wireless devices that use low-power radios to communicate, and that deliver their data over multiple hops. RPL is also a distance vector type routing protocol that builds directed acyclic graphs (DAGs) based on the selected routing metrics and constraints [4], [35].

3.3.7 Constrained Application Protocol (CoAP)

CoAP is a specialized web transfer protocol for constrained node and networks. Constrained networks that make use of 6LoWPAN support the fragmentation of IPv6 packets into small link-layer frames, even if this causes a significant reduction in packet delivery probability. CoAP is a protocol supported by OpenWSN that enables RESTful interaction with individual motes, without the overhead of the Transmission Control Protocol (TCP) and the verbose nature of HTTP. A CoAP-enabled mote acts both like a web client and a web server [32].

A CoAP request is sent by a web client to request an action on a resource located in a web server. The server sends back to the client a response. These exchanges are performed asynchronously over a datagram embeded on a UDP packet. CoAP main goal is to design a generic web protocol for the special requirements of constrained environments, especially considering energy, building automation, and other machine-to-machine (M2M) applications. One of the targets of CoAP is to keep message overhead small, limiting the need for fragmentation, and not to blindly compress HTTP, but rather to realize a subset of Representational State Transfer (REST) managed with HTTP, but optimized for M2M application. CoAP rewrites simple HTTP interfaces into a more compact protocol. More importantly, it also offers features for M2M like built-in discovery, asynchronous message exchange, and multicast support. The main feature of CoAP is a web protocol fulfilling M2M communication pasradigm. It uses UDP, and it implements asynchronous message exchanges. The messages have low header overhead and parsing complexity. It is also a stateless HTTP protocol that allows proxies to be easily built, providing access to CoAP resources via HTTP in a uniform way, and CoAP uses security binding to Datagram Transport Layer Security (DTLS) [33].

Chapter 4

Routing Protocol for Low-power and Lossy Network

4.1 Overview

Low-power and Lossy networks (LLNs) are a class of networks in which both the routers (typically WSN motes) and their interconnections are constrained. LLNs consist largely of constrained motes operating with limited processing power, memory, and energy. Limited energy is a constraint when the motes are battery operated and they can not make use of energy scavenging [1]. The routers are usually interconnected by using the IEEE 802.15.4 low-power wireless standard, which is characterized by low data rate, high loss rate, and instability [4].

LLNs support the following traffic patterns: Point-to-Point (P2P), Point-to-Multipoint (P2MP, from a central point to a subset of devices inside the LLN) and Multipoint-to-Point (MP2P, from devices inside the network towards a central control mote) as shown in Figure 4.1.

Most of the time LLNs use the Point-to-Multipoint or Multipoint-to-Point traffic patterns. As mentioned in the previous chapters, such networks can contain up to thousands of motes. For these reasons, such kind of network traffic requires a routing solution that considers the above characteristics. A commonly adopted

solution is RFC 6550 [4], also known as IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL).

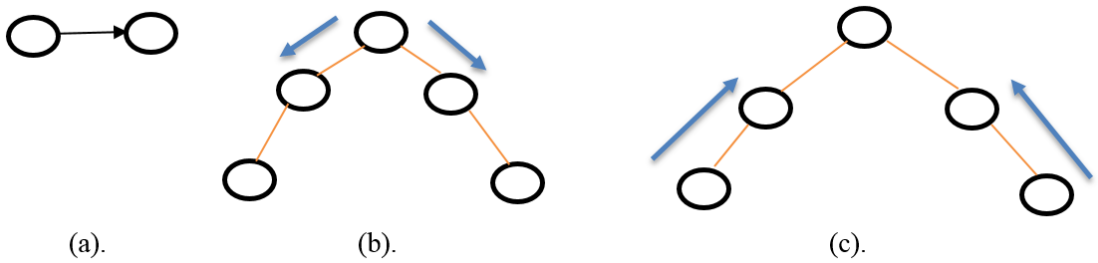


Figure 4.1: Traffic supported by LLNs (a) P2P, (b) P2MP and (c) MP2P

RPL is a distance vector protocol which is based on calculating a direction (the next hop address and exit interface) and a distance (a measure of the cost to reach certain mote calculated using different routing metrics) [2]. RPL is a routing protocol which operates on top of IEEE 802.15.4. The routing layer is responsible for relaying packets across multiple hops separating the source and destination motes. It is divided into a *forwarding engine*, which uses a routing table to decide which neighbor node is the next hop for that packet, and a *routing protocol*, which populates that routing table.

RPL is designed for Low Power and Lossy Wireless Networks such as WSNs. Hence, it is optimized for collection networks (where motes periodically report measurements to a small number of collection points) with infrequent communication from the collection point to individual motes. RPL manages separately collection traffic MP2P and configuration traffic P2MP. RPL does not support P2P traffic well, although workarounds can be used.

RPL routes are optimized for traffic to or from one or more roots motes that act as gateways for the topology. For MP2P traffic, RPL builds a graph into the network, grounded at the collection points called Low power and Lossy network Border Router (LBR) or Direct Acyclic Graph root (DAG root). To each node, a rank is assigned, such that the rank increases as the mote gets farther from the border router. In RPL, the topology is called Destination Oriented Directed Acyclic

Graph (DODAG). Forwarding a packet to the LBR roughly consists in picking the neighbor mote with the lowest rank [3], [4], [35].

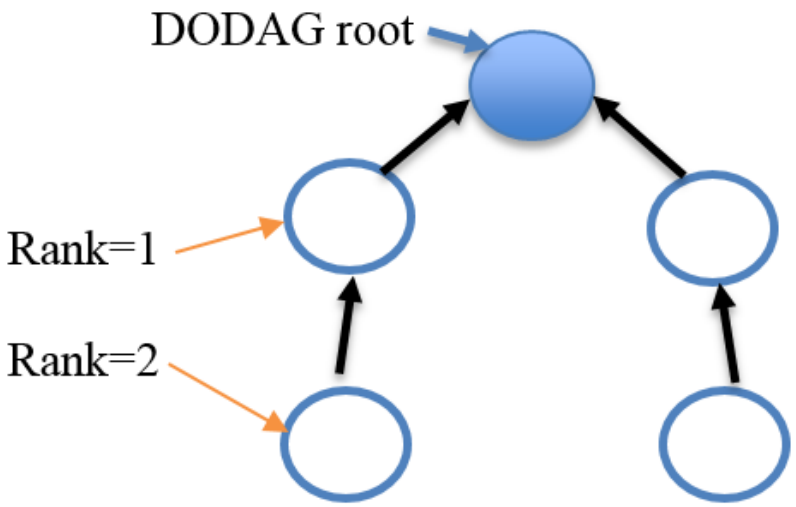


Figure 4.2: DODAG Graph

As shown in Figure 4.3, RPL organizes a topology as a DAG that is portioned into one or more DODAG. Both DAG and DODAG are directed graphs with no cycles. All paths are oriented toward and terminate at one or more root motes. DODAG is a DAG rooted at a single destination, i.e., at a single node (the DODAG root) with no outgoing edges, as shown in the two figures 4.3.

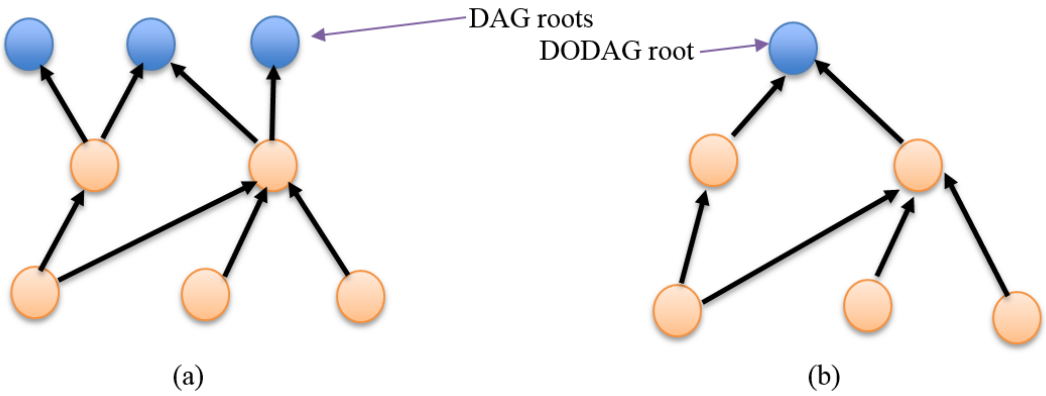


Figure 4.3: (a) DAG (b) DODAG

To maintain and identify the topologies, RPL uses four values. These are RPLInstanceID, DODAGID, DODAGVersionNumber, and Rank.

RPLInstanceID identifies a set of one or more DODAGs. An RPL Instance is defined as a set of one or more DODAGs that share the RPLInstanceID identifier. An RPL node can belong to at most one DODAG associated to an RPL Instance. Each RPL Instance is independent from other RPL Instances. An RPL Instance may comprise a single DODAG with a single root like in Home Automation Networks or multiple uncoordinated DODAGs with independent roots (different DODAGIDs) like urban data collection application. This thesis is focused on a single RPL Instance [4].

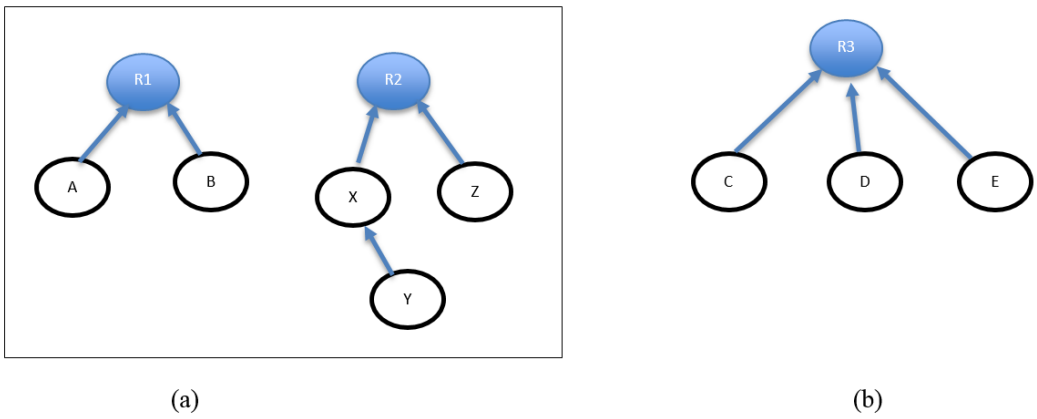


Figure 4.4: RPL Instance

As an example, Figure 4.4(a) shows an RPL Instance comprising two DODAGs with DODAG roots R1 and R2. Each of these DODAG roots advertises the same RPLInstanceID, but DODAGID is different. In Figure 4.4(b), DODAG root R3 have different RPLInstanceID from DODAG roots R1 and R2 in the Figure 4.4(a), therefore, the DODAG in the Figure 4.4(b) have different RPL Instance from Figure 4.4(a).

The scope of the DODAGID is RPL instance. A single DODAG in the network is identified by a combination of RPLInstanceID and DODAGID. DODAGVersionNumber is a sequential counter that is increased by the root to form a new version

of DODAG. Its scope is the DODAG. DODAG Version is a specific iteration "Version" of a DODAG with a given DODAGID. DODAG Version is uniquely identified by a combination of RPLInstanceID, DODAGID, and DODAGVersionNumber.

Rank defines individual positions of nodes with respect to DODAG root and its scope is DODAG Version. The computation of the value Rank is obtained by means of an Objective Function (OF). A DODAGVersionNumber increment leads to a new DODAG version. The DODAGVersionNumber increment results in a different DODAG topology. A DODAG root can thus institute a global repair operation by incrementing the DODAGVersionNumber, thus allowing for DODAG repairing in RPL. The repair operation will lead to a new DODAG Version that contains the new organization of the motes. Due to this, motes in the DODAG will have a new position, and they can choose the new position using the RPL protocol. In particular, motes in the new DODAG Version nodes can choose a new position whose Rank does not depend on the Rank they had in the old DODAG Version. RPL also allows for local DODAG repairing within the DODAG Version [4]. The following Figure 4.5 illustrates how a DODAGVersionNumber increment leads to a new DODAG Version.

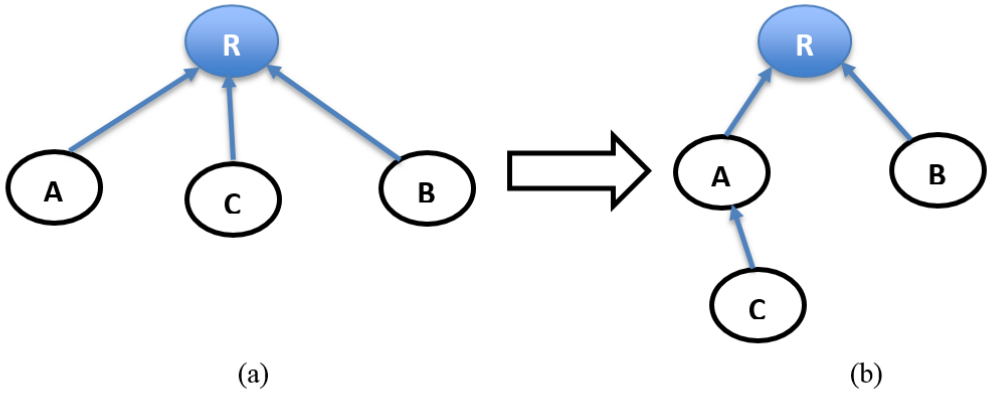


Figure 4.5: (a) Version N (b) Version N+1

For any application running in the mote, security is an important requirement, and RPL is not an ecception. RPL supports message integrity and confidentiality. RPL

has three basic security modes. These are unsecured mode, preinstalled mode and authenticated mode [4].

In unsecured Mode the control messages are sent without any additional security mechanisms. But this does not mean that a RPL network is not secured. RPL network uses other security mechanisms like Link-layer security to meet application security requirements.

In preinstalled Mode, hosts and routers have preinstalled keys that enable only authorized motes to join a given RPL Instance. The keys also provide message authenticity, integrity, and confidentiality.

In authentication Mode, motes have preinstalled keys, but the preinstalled key is only used to join the RPL Instance when a node is a leaf and it has consequently no children. A router joining RPL Instance must be authenticated. It requires an authentication key from an authentication authority.

4.1.1 Upward Route and DODAG Construction

RPL provisions routes *Upward* towards DODAG roots, forming a DODAG optimized according to an Objective Function (OF). Upward routing is a standard procedure which enables network devices to send data to a root mote. RPL nodes routes are constructed, and the topology of DODAG maintained using a RPL control message called DODAG Information Object (DIO).

Objective Function

An Objective Function (OF) [5] defines how RPL nodes select and optimize routes within an RPL instance and related functions used to compute the rank. In addition, OF dictates how parents are selected in the DODAG and it is used for DODAG formation. OF allows RPL to be adapted to meet different optimization criteria that are required by different applications and network designs. A mote decision of which DODAG and DODAG Version it should join is based on the value of the OF. Each node also uses the OF to select its preferred and backup parent motes in the current DODAG Version. The preferred parent will be used in Upward routes,

while the backup parent will be used whenever the connectivity with the preferred parent is lost.

In a typical WSN scenario, nodes periodically generate data packets that have to find their way through the network to reach a specific destination encoded in the packet. There are two types of OF used for RPL routing. These are Object Function Zero (OF0) and Minimum Rank with Hysteresis OF (MRHOF).

OF0 is designed to find the nearest Grounded root and it selects a preferred parent and a backup feasible successor if it is available. This can be achieved if the Rank of a node is close to its distance from the root. The preferred parent of a node belongs to the set of the node parents, that have been selected as preferred motes for the next hop of the Upward route. Usually, the preferred parent set contains a single node. However, whenever the node has multiple preferred parents with the same Rank, it may consist of multiple nodes. To select a single preferred parent OF0 makes use of the criteria specified by the RFC [5]:

- Prior to selecting a router as the preferred parent, a RPL implementation should validate the connectivity and suitability of the router.
- A router that offers connectivity to a grounded DODAG Version should be preferred over one that does not.
- A router that offers connectivity to a more preferable root should be preferred.
- The parent that allows obtaining the lesser resulting Rank for this node should be preferred.
- The preferred parent that was in use already should be preferred.
- A router that has announced a DIO message more recently should be preferred.

When selecting a backup feasible successor, the OF performs the following checks [5]:

- The backup feasible successor must not be the preferred parent.

- The backup feasible successor must be either in the same DODAG Version as this node or in an subsequent DODAG Version.
- Along with RPL rules, a Router in the same DODAG Version as this node and with a Rank that is higher than the Rank computed for this node must not be selected as a feasible successor.
- A router with a lesser Rank should be preferred.
- The backup feasible successor that was in use already should be preferred.

The MRHOF Objective Function looks for shortest paths, according to a given metric that measures a "distance" between two nodes. Since small fluctuations, if the computed metric might induce frequent topology changes, hysteresis is used to prevent excessive changes in the network topology. The OF metrics are additive and are advertised by the DIO messages.

Grounded and floating DODAGs

In DODAG construction and upward route, there are two kinds of DODAGs. These are grounded and floating DODAGs. A grounded DODAG offers connectivity to those hosts required to achieve the application goals. A floating DODAG is not associated with any particular activity, and in most cases, it simply provides routes to nodes within the DODAG. The Figure 4.6 illustrates the floating and grounded DODAGs.

Since LLN should are, by definition, low-power, loop detection should not be carried out regularly, but only when required by topology changes. To this extent, RPL's implements data-packet based on-demand loop detection. Since adding additional headers to packets that have to be transmitted is not expensive, standard messages also carry the information required to update the network topology. However, since data traffic can be infrequent, maintaining a routing topology that is constantly up to date with the physical topology requires additional messages and can waste energy.

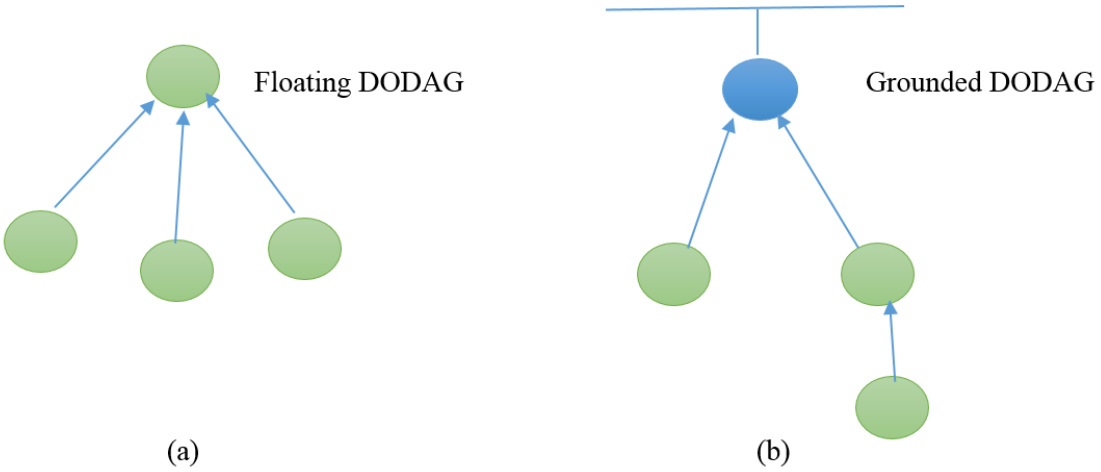


Figure 4.6: (a) Floating DODAG; (b) Grounded DODAG

Unfortunately, this additional exchange of packets cannot be avoided because it is fundamental to maintain the topology when data packets are not exchanged in the network or when they are transmitted with a too small frequency. Typical LLNs exhibit variations in physical connectivity that are transient and do not significantly affect exchanged traffic, but that is expensive to track from the control plane. Since these infrequent changes in connectivity need not be addressed by RPL by sending specific messages until there is data to send, RPL does not update the topology in this case. During data transmission, RPL uses RPL Packet Information [4] to update the network topology.

RPL Packet Information is an external mechanism to access and transport some control information within the data packets, which is used by RPL routers to process the routing information and to help maintain the routing topology. RPL Packet Information enables the association of a data packet with an RPL Instance and the validation of RPL routing states. The RPL Packet Information, that is contained within the data packets, is used to perform loop detection.

RPL tries to avoid creating loops when changes in topology occur. The standard defines a node as greedy if it attempts to move deeper (increase Rank) in the DODAG Version in order to increase the size of the parent set. If an RPL node is too greedy, the nodes attempt to optimize for additional parents beyond its most

preferred parents. This results in an instability in the DODAG network [6], [5]. Figure 4.7 illustrates a greedy behavior in a DODAG. To avoid instabilities in the DODAG Version, RPL disallows greediness for nodes that joined the DODAG Version, using RPL packet Information to avoid loops.

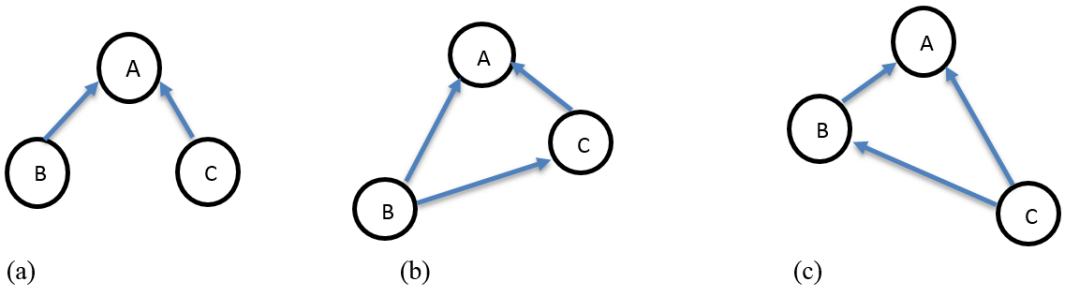


Figure 4.7: Greedy DODAG parent selection

In Figure 4.7(a) Node A is the DODAG parent for Node (B) and (C) while in Figure 4.7(b) Node A is a DODAG parent for Node B and C, and also Node C is a DODAG parent for Node B, in Figure 4.7(c) Node A is a DODAG parent for Node B and C, and also Node B is a DODAG parent for Node C.

Let us consider the DODAG illustration in Figure 4.7, and let's consider the effects of greedy nodes on the DODAG generation. Let' suppose that node (B) is allowed to leave and rejoin the DODAG with a different Rank. The objective function may then favor (B) taking both (A) and (C) as parents, as illustrated in Figure 4.7(b).

Now Node (B) is deeper than both Nodes (A) and (C), and Node (B) is satisfied with two DODAG parents. However, node (C) can now follow the same procedure: it's willing to receive and process a DIO message from Node (B) (against the rules of RPL), and therefore can decide to leave the DODAG to rejoin it at a lower Rank. To do so, it takes both Nodes (A) and (B) as DODAG parents. Now Node (C) is deeper than both Nodes (A) and (B) and is satisfied with two DODAG parents. Since node (B) is greedy, it will again leave and rejoin at a deeper Rank, to again get two parents. The process will repeat, and the DODAG topology will obscillate between Figure 4.7(b) and Figure 4.7(c) until the nodes count to infinity and restart the cycle again.

This endless loop can be avoided through mechanisms defined in RPL. Once nodes (B) and (C) have selected (A) as a preferred parent, they ignore DIO messages from nodes with equal or higher Rank, because such nodes are possibly in their own sub-DODAGs.

RPL uses an algorithm to construct the DODAG called Distributed Algorithm Operation. This algorithm constructs the topology as follows: first, a mote is configured to be DODAG root with associated DODAG configuration, which is used to distribute configuration information for DODAG operation through the DODAG. Second, motes advertise their presence (including affiliation with a DODAG, routing cost, and related metrics) by sending link-local multicast DIO messages to all-RPL-nodes. The other motes listen for DIOs and use their information to join a new DODAG (thus, selecting DODAG parents), or to maintain an existing DODAG, according to the specified Objective Function and Rank of their neighbors. In this thesis Objective Function Zero is used [4], [5]. Motes provide their routing table entries if they are in Storing mode or Non-Storing mode, as specified by the DIO message. Motes that decide to join a DODAG can select one or more DODAG parents as the next hop for the default route.

4.1.2 Downward route and Destination Advertisement

To establish Downward routes, RPL nodes send Destination Advertisement Object (DAO) messages Upward. The DAO contains the destination information. RPL nodes transmit a DAO Upward to propagate a destination information along DODAG after joining a DODAG. The RPL nodes use the destination information of the DAO to create a routing table for the Downward routing. Routes support multiple flows: the P2MP flow, which is used to send packets from the DODAG root towards the leaves, and the P2P flow, which allows sending messages from nodes towards the DODAG root (or a node ancestor) through an Upward route, and then towards the message destination through a Downward Route.

RPL supports two modes of downward routing traffic, Storing and Non-Storing modes. Storing mode is fully stateful. This means that nodes store downward routing tables for their sub-DODAG. At each hop, a node examines its routing

table to decide the next hop of the message (see Figure 4.8(a)).

In Non-Storing Mode, which is fully source routed, nodes do not store downward routing tables. Instead, all non-root nodes found in the DODAG sends all the traffic to the root (see Figure 4.8(b)). The DODAG root makes all decisions relevant to establish downward routing paths. The DODAG root uses a source routing (the route starts from the root mote) when sending Downward packets. In the source routing, the packets include a routing information from source to destination [4].

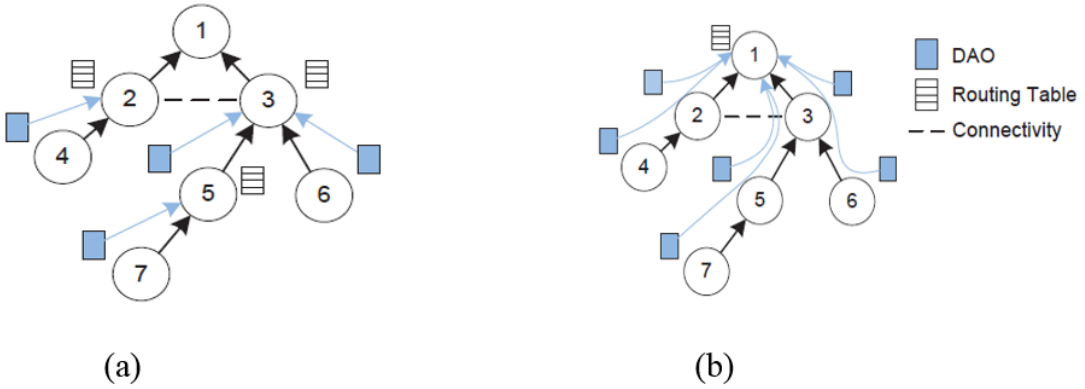


Figure 4.8: Storing(a) and Non-storing(b) mode with the RPL Downward routing (Source: http://slideplayer.com/slide/8727418)

4.1.3 Routing Metrics and Constraints Used by RPL

Routing metrics are used by routing protocols to compute the shortest path. RPL supports constraint-based routing, i.e. constraints may be applied to both links and nodes. Link and nodes not satisfying the given constraints are pruned from the set of candidate neighbors. The routing algorithm, therefore, looks for a constrained shortest path.

The set of routing metrics and constraints used by a RPL deployment is advertised along the DODAG that is built according to the Objective Function, the routing metrics and constraints are advertised in the DIO message. RPL may be used to build DODAGs with the goal of maximizing reliability by using the link reliability metric to compute the "best" path or to use the energy node characteristic as a node constraint when building the DODAG [4], [7].

In OpenWSN, RPL implements link routing metric. The routing metric and constraint is carried within a DODAG metric container object Figure 4.9. DODAG metric container option is present in DIO or DAO messages and it is used to report the routing metric in a DODAG. OF specifies the objectives used to compute the constrained path and it is decoupled from the routing metrics and constraints used by RPL [4].

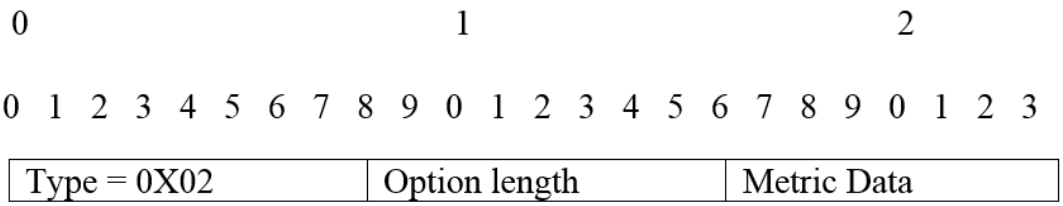


Figure 4.9: Format of DODAG metric container (Source: [4])

Object Function Zero (OF0) is used in OpenMote B and on this thesis for routing metric. In RPL OF0 is used for the nodes to join a DODAG Version that offers good enough connectivity to a specific set of nodes, or to a larger routing infrastructure for with there is no guarantee that the path will be optimized according to a specific metric [5]. The OF0 in OpenMote implementation computes the rank of a mote with respect to the DODAG root, then it selects the best preferred parent and feasible successors for a given instance from all the candidate neighbors. When a new DIO message is received, the OF is evaluated using the information obtained from the DIO message to update the DIO message itself. The new DIO message is then sent to the mote neighbors. The Wireshark log in the figure 4.10 shows that OpenMote B uses OF0.

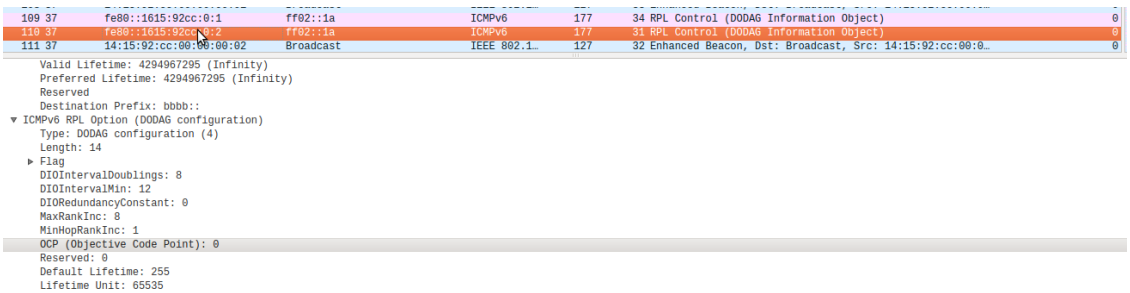


Figure 4.10: DIO message showing OF0 identification on Wireshark log

4.2 RPL Instance

One or more DODAG roots can be associated with a single RPL Instance. The roots can operate independently, or might be connected through a network that does not necessarily behaves as a LLN. RPL Instances are identified from the RPLInstanceID. These identifiers can be either Global and Local. Global identifiers are used for one or more coordinated DODAGs, while local ids always correspond to single DODAG instances. Global RPLInstanceID must be unique to the whole LLN and there can be up to 128 Global instances in the whole network. The RPLInstanceID is encoded in 1 octet. The first bit indicates whether it is Global or Local.

Local RPL Instances are always single DODAGs whose root owns the corresponding DODAGID. The DODAG unilaterally allocates the local RPLInstanceID (see Figure 4.4). Local RPLInstanceID can support up to 64 Local instances per DODAGID. A local RPLInstanceID is autoconfigured by the mote that owns the DODAGID and it must be unique for that DODAGID. According to the standard, the DODAGID must be a reachable IPv6 address, and it must be used as the endpoint of all communications within that Local instance. This thesis is focused on Local RPL Instance [4].

Data packets flow within the RPL network includes RPLInstanceID as part of the RPL Packet Information. Whenever a message sent from outside the RPL network reaches the ingress router, the router fills the RPLInstanceID field of the packet, which is then forwarded through the RPL network.

4.3 RPL routing requirement characteristics

The Routing Over Low power and Lossy network (ROLL) Working Group in 2009 published documents that describe unique routing requirements for Low Power and Lossy networks (LLN) for Industrial [8], Commercial (Building) [9], Home [10], and Urban [11] networks. These requirements can be subdivided for the above representative's types of traffic, in terms of resources, path diversity, their convergence time, heterogeneous routing, security, and other requirements.

The routing protocol must support P2MP and MP2P traffic pattern services and a bidirectional link between two motes in the network. The routing protocol should be implementable in resource-constrained devices and for battery-powered nodes. For this reason, power consumption must be low to achieve a long battery lifetime, therefore, the routing protocol battery-powered motes should provide no more than 1% of duty cycle on their sleep phase and at least the device should have five years of service lifetime [8], [9], [10]. The routers must be able to provide alternative routes for successful and reliable packet delivery. The packet delivery ratio must be $>$ 99 percent including retransmissions and for Lossy links, retransmissions must be not more than three [13].

In Industrial applications, the routing protocol must converge after the joining of new nodes within a few minutes, and after the re-establishment of a node or Lossy connectivity within tens of seconds [8]. In home automation, in case of instabilities of application devices (e.g., for instance, when motes are in motion) the convergence time must be within 0.5 seconds if no nodes have moved and 4 seconds if nodes have moved [11]. The routing protocol should consider characteristics of the node, such as power budget, memory, and sleeping interval.

The routing protocol should route via mains-powered nodes if possible [10]. The routing protocol also must be able to generate different routes with different characteristics for different flows to assure that mission-critical applications do not defer while less critical applications accessing the network can be managed with less timing requirements. Last but not least the routing protocol and its configuration have to contribute to prevent attacks, and to support message integrity of nodes from manipulating routing functions in the routing decision process security [35].

4.4 ICMPv6 RPL control message

Nodes running RPL exchange signaling information to setup and maintain the DODAG. This information is exchanged as a new type of ICMPv6 message called the RPL Control Message. A RPL control message is identified by a code, and the code identifies how the message is structured and how to interpret its content. Most

RPL control messages are related to a single link, except for the DAO / DAO-ACK messages in Non-Storing mode [4]. The message exchange is managed using unicast address over multiple hops, because DAO message is sent to the specific known node and thus uses global or unique-local address for both source and destination addresses. For the rest of RPL control messages, the source address is a link-local address, and the destination address can be either a link-local unicast address or an all-RPL-nodes multicast address. The all-RPL-nodes multicast address is a specific address with a value of ff02::1a (see Figure 4.20). A RPL control message consists of ICMPv6 header followed by a message body. The RPL control message base diagram is illustrated below.

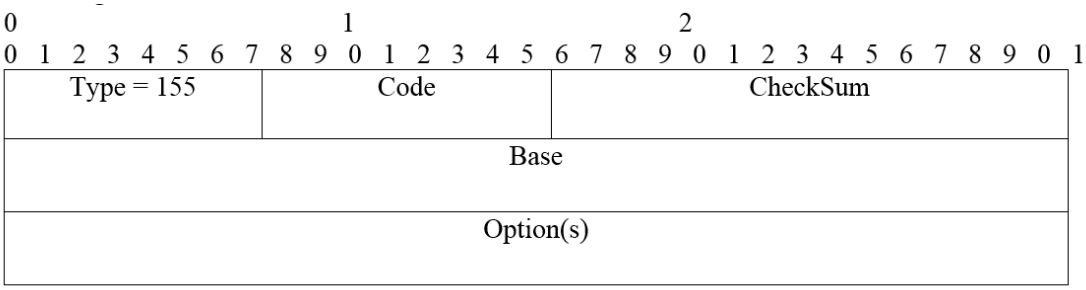


Figure 4.11: RPL control message

The RPL control message is an ICMPv6 information message with the 'Type' field set to 155. The RPL control message type is defined by the 'Code' field. The RPL control messages are: Code=0x00 DODAG Information Solicitation (DIS), Code=0x01 DODAG Information Object (DIO), Code=0x02 Destination Advertisement object (DAO), and Code=0x03 Destination Advertisement Object Acknowledgment (DAO-ACK) [4].

Messages containing unknown code fields must not be processed, and have to be directly discarded, without generating any response message.

The checksum field should be set to zero for RPL security operations. For the other packets, it's computed from the packet contents (including the security headers). Figure 4.11 shows an RPL control message.

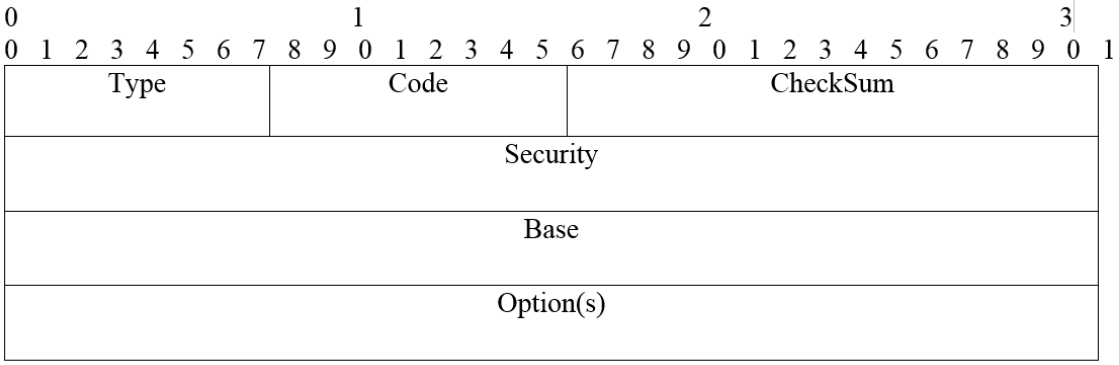


Figure 4.12: secure RPL control message

To provide additional integrit, authenticity, confidentiality and delay protection, each RPL message has a secure variant. The security information is situated between the checksum and the base fields, as shown in Figure 4.12.

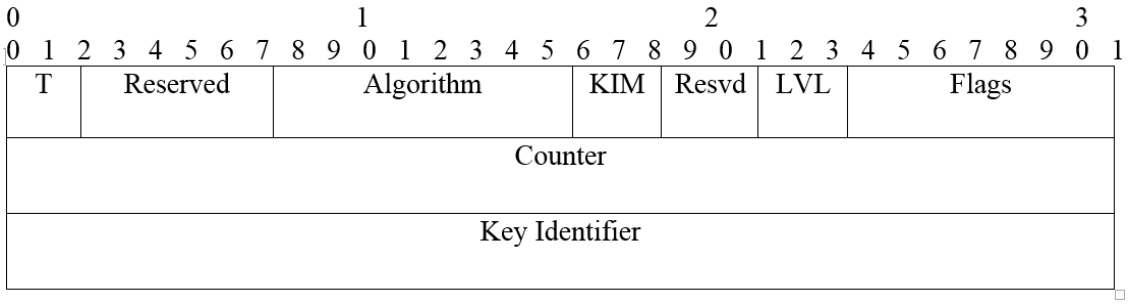


Figure 4.13: Security Section

4.4.1 DODAG Information Solicitation (DIS)

DIS is a RPL control message and it is used to solicit a DIO from a RPL mote, in the same way as the router solicitation mechanism specified in IPv6 Neighbor Discovery. Router Solicitation originates from hosts only, these messages are sent to detect the presence of routers on the link. Routers respond to these messages by sending Router Advertisement. Router Advertisement messages originate from routers, these messages indicate the presence of the router on a link. These messages contain essential parameters along with IPv6 prefix of the network.

A mote may use DIS to probe its neighborhood for nearby DODAGs. As shown in Figure 4.14 the format of DIS base object fields is: 8 bit unused field reserved for flags, followed by a Reserved field that consists of 8 unused bits. The "options" part allows the DIS to carry PAD1, PADN and Solicited Information. PAD1 and PADN may also be present in DIO, DAO and DAO-ACK messages. PAD1 option is usedto specify that a single octet should be inserted as padding, while PADN is used to insert two or more octet of padding into the message (to enable options alignment). Solicited Information option is used by a mote to request DIO messages from a subset of neighboring motes.

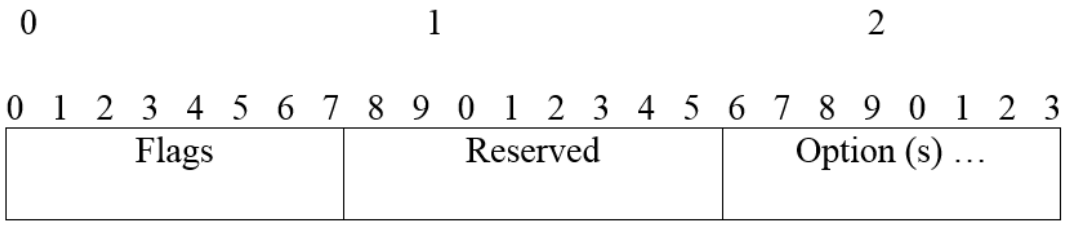


Figure 4.14: DIS base object

Flags and Reserved fields are initialized to zero by the sender and contain an 8 bits reserved field. A Secure DIS message follows the format in Figure 4.12, where the base format is the DIS object shown in Figure 4.14.

4.4.2 DODAG Information Object (DIO)

The 1 byte code in the ICMPv6 header is used to differentiate between different sub-types. The DIO message, used to build the DODAG, is called the DODAG Information Object (DIO). DIO carries information that allows a mote to discover an RPL Instance, learn its configuration parameters, select a DODAG parent set, and maintain the DODAG [4]. The main fields of DIO message include Version Number, DODAGID and RPLInstanceID. The DODAGID must be a routable IPv6 address associated to the DODAG root. The format a DIO base object is shown in the Figure 4.16.

All the motes in the network regularly issue a DIO which serves as a heartbeat to indicate their rank. The sending rate of DIO messages is regulated by the Trickle Algorithm [12]. The algorithm detects graph inconsistencies, for example, due to motes that join or change position in the network, to decide when to multicast DIO messages. To this extent, the algorithm defines a trickle time, that denotes the interval between two consecutive DIO messages. The trickle time is dynamically updated to increase or decrease the message rate.

As the network stabilizes, the interval between DIO messages is increased, resulting in fewer messages sent through the network. When inconsistencies are detected, the nodes reset the trickle time and send DIOs more often. Using this mechanism, the frequency of the DIO messages depends on the network stability. One of the main advantages of the Trickle Algorithm is its simplicity: the implementation of the algorithm does not require excessive computational power. This aspect is quite relevant for LLNs. The fundamental mechanism of the trickle timer is shown in Figure 4.15.

The algorithm runs for a defined interval, which is based on three parameters. These are the minimum interval size, 'Imin', which is defined in units of time (e.g., milliseconds, seconds). For example, a protocol might define the minimum interval as 100 milliseconds. The maximum size interval size, 'Imax', which is described as the number of doublings of the minimum interval size (the base-2 log(max/min)). For example, a protocol might define 'Imax' as 16. If the minimum interval is 100 ms, then the amount of time specified by Imax is 100 ms * 65,536, i.e., 6,553.6 seconds or approximately 109 minutes.

Finally, the algorithm defines a redundancy constant 'K', which is greater than zero. Trickle Algorithm also uses three additional variables. These are current interval size 'I', a time within the current time interval or the time for sending message 't', and a counter 'C'.

When the algorithm starts its execution, the algorithm sets 'I' to a value in the range of [Imin, Imax], and it begins the first interval. When the interval begins, the trickle algorithm resets 'C' to 0 and 't' is selected randomly within the interval [I/2, I). When the Trickle algorithm hears a transmission, if it is "consistent", the

counter 'C' is increased by one. At time 't', The trickle algorithm transmits a DIO message, when the counter 'C' is less than the redundant constant 'K'.

When the interval 'I' expires, the Trickle doubles the length of the maximum interval size. At this point, if the new interval length 'I*max' is greater than the time specified by Imax, the trickle algorithm sets the interval length 'I' to the time specified by 'Imax'. If the Trickle algorithm hears a transmission that is "inconsistent" and the interval 'I' is greater than 'Imin', it resets the trickle timer. To reset the timer, the trickle algorithm sets 'I' to 'Imin', and starts a new interval as in step 2.

If 'I' is equal to 'Imin' when the trickle algorithm hears an "inconsistent" transmission, it does nothing. The algorithm is illustrated in Figure 4.15. The trickle algorithm can also reset its timer in response to external "events" [12].

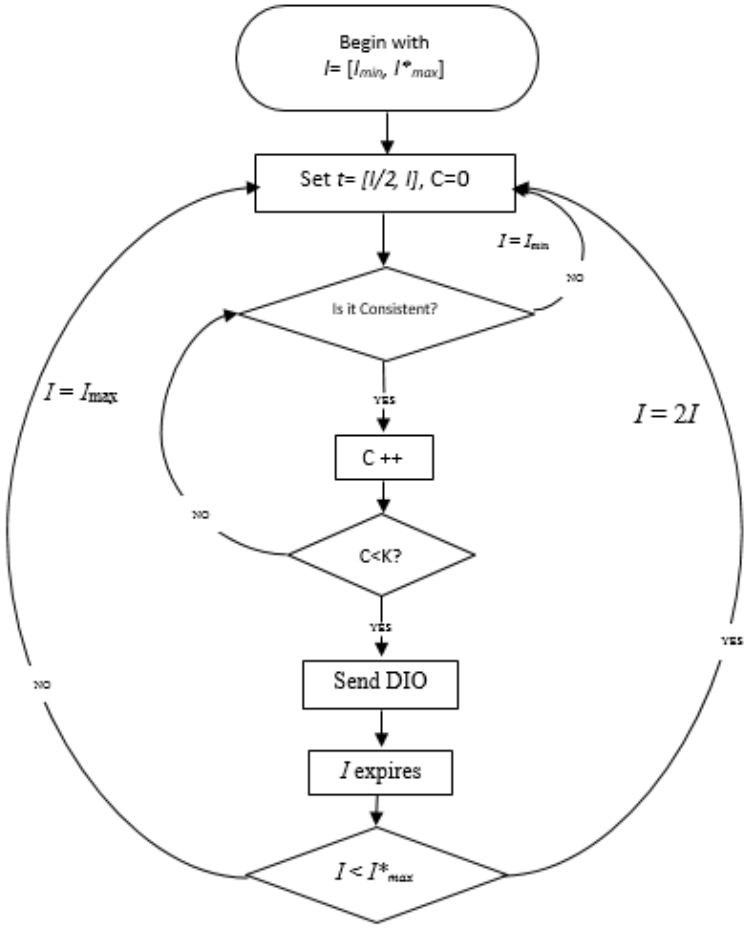


Figure 4.15: The trickle time process of the trickle algorithm (Source: [51])

The only time the Trickle algorithm transmits DIO message is when the C>K condition of the algorithm illustrated in the above Figure 4.15 is true. This means there is an inherent delay between detecting an inconsistency (shrinking I to Imin) and responding to that inconsistency (transmitting at time t in the new interval). This is intentional. Immediately responding after an inconsistency is detected can cause a broadcast storm, where many nodes respond at once and in a synchronized fashion.

By making responses follow the Trickle algorithm (with the minimal interval size) and randomly in selecting the transmission times of the messages, a routing protocol can benefit from Trickle's suppression mechanism and scale across a huge range of node densities [4].

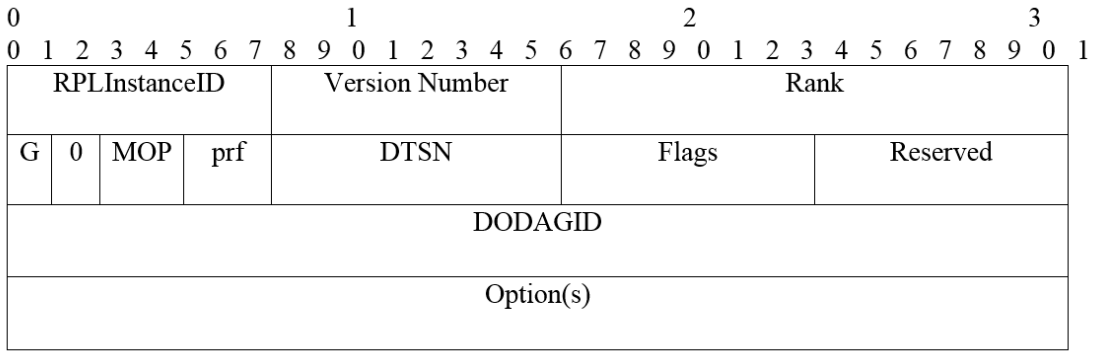


Figure 4.16: DIO base object

In the DIO base object, the 'G' flag means Grounded, and it indicates whether the DODAG advertised can satisfy the application-defined goal. If 'G' flag is set, it means that the DODAG is grounded. If it is not set, it means that the DODAG is floating. MOP (Mode of Operation) is used to identify the operation mode of the RPL Instance.

MOP is encoded in MOP 0, 1, 2, 3. MOP value 0 indicates no downward routes maintained by RPL, which means that destination advertisement messages are disabled and the DODAG maintains only Upward routes, MOP value 1 indicates the operation is done in Non-storing mode, MOP value 2 indicates the operation is held in storing mode without multicast support and MOP value 3 indicates that

the operation is held in storing mode with multicast support [4]. The MOP and 'G' flag are illustrated in Figure 4.17.

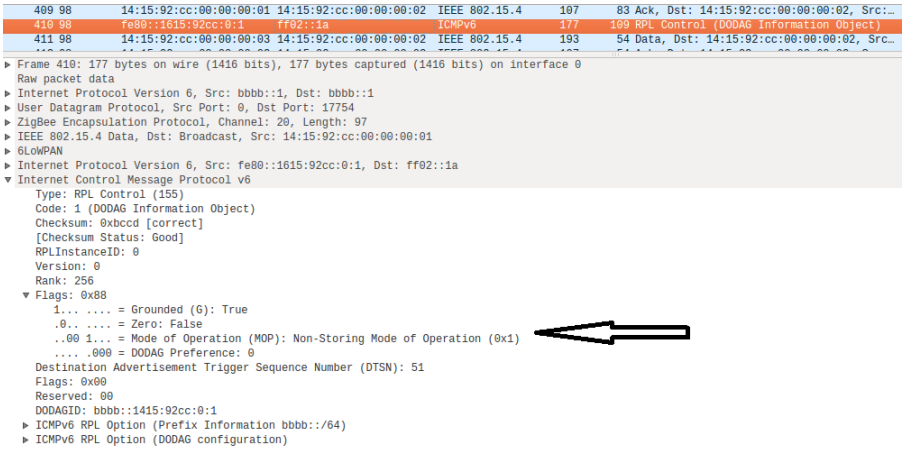


Figure 4.17: MOP and Grounded flag captured in Wireshark

DODAGPreference (Prf) is a 3-bit unsigned integer and defines how preferable the root of this DODAG is compared to other DODAG roots belonging to the same instance. Version Number is an 8-bit unsigned integer. The DODAG root sets this field to the value specified by the DODAGVersionNumber. Rank is a 16-bit unsigned integer containing the Rank in the current DODAG of the node sending the DIO message. RPLInstanceID is an 8-bit field set by the DODAG root and identifies the RPL instance the DODAG belongs to. Destination Advertisement Trigger Sequence Number (DTSN) is an 8-bit unsigned integer is used as part of the procedure to maintain Downward routes and is set by the mode issuing a DIO message. Flags and Reserved are 8-bit unused fields, that are set to zero by the sender and ignored by the receiver. Finally, DODAGID is a 128-bit IPv6 address that uniquely identifies the DODAG, and is set by a DODAG root[4]. DIO control messages have the same structure of RPL control message, with a base object like the one shown in Figure 4.16.

4.4.3 Destination Advertisement object (DAO)

The DAO is used to propagate destination information along the DODAG. P2MP is enabled by piecewise source routing: when issuing a message for a specific node

in the network, the LBR prepends the sequence of nodes that need to be traversed to get to the destination. To learn this sequence, each node in the network is asked to transmit a Destination Advertisement Object (DAO) to the DODAG root. In Storing mode, the DAO message is sent in unicast by the child to the selected parent(s). In Non-Storing mode, the DAO message is sent in unicast to the DODAG root [3]. The format of DAO base object is shown in Figure 4.18.

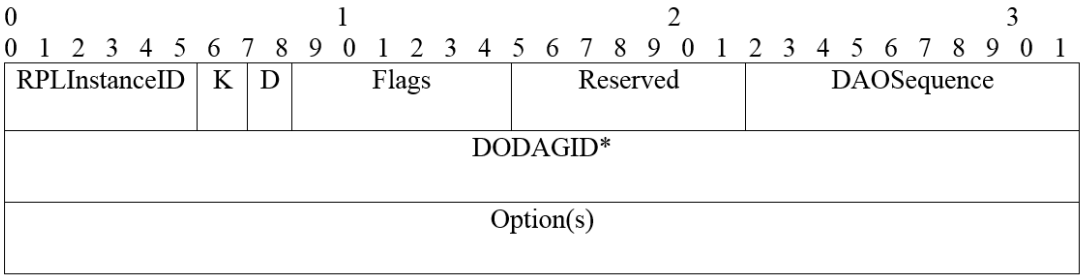


Figure 4.18: DAO base object

RPLInstanceID is an 8-bit field indicating the topology instance associated with the DODAG, as described for the DIO, the 'K' flag indicates that the recipient is expected to send a DAO-ACK back, the 'D' flag indicates that the DODAGID field is present. This flag must be set when a local RPLInstanceID is used, the field (Flags) consists of 6 unused and reserved bits. The field must be initialized to zero by the sender and must be ignored by the receiver, and 'Reserved' is an 8-bit unused field. DAOSequence is incremented at whenever a DAO message is generated from a node. The corresponding DAO-ACK messages use the same value. DODAGID (optional) is a 128-bit unsigned integer that uniquely identifies a DODAG, and is set by the DODAG root. The flag 'D' indicates whether the field is present or not. Typically, this field is used to identify a DODAGID associated with an RPLInstanceID whenever local RPLInstanceIDs are used [4].

4.4.4 Destination Advertisement Object Acknowledgment (DAO-ACK)

The DAO-ACK message is a unicast address packet sent by a DAO recipient to reply to a unicast DAO RPL control message. The format of DAO-ACK base object is shown in the Figure 4.19.

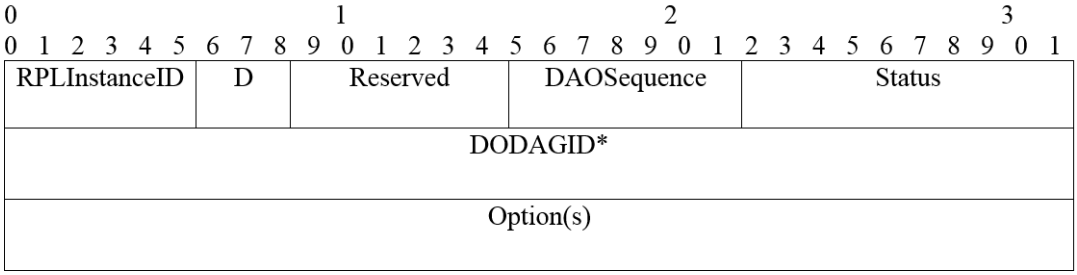


Figure 4.19: DAO-ACK base object

RPLInstanceID is an 8-bit field indicating the topology instance associated with the DODAG, as described for the DIO, the 'D' flag indicates that the DODAGID field is present. Typically, this would only be set when a local RPLInstanceID is used. 'Reserved' is a 7-bit field, reserved for flags. DAOSequence is copied by the recipient from the DAO message that triggered the DAO-ACK, to associate the acknowledgment to the correct DAO message. 'Status' indicates the completion of the DAO-ACK control message accepted or rejected. Status 0 is defined as unqualified acceptance in this specification. The remaining status values are reserved as rejection codes. As for DAO control messages, the DODAGID field is optional [4].

4.5 Topology Construction and Message Exchange

The topology construction process starts at the root mote, which is configured by the user who controls the system. If a mote is configured to act as a root, it starts sending the topology information with the new information to its neighboring peers using DIO messages. If the mote is a leaf mote, it simply joins the graph by requesting the DIO message using DIS messages and does not send any DIO message, to

avoid forming loops which cause network instability (see Section 4.1.1) The neighboring peers will repeat this process and do parent selection, route addition and graph information advertisement using DIO messages.

The root mote with its IPv6 address multicasts the DIO RPL control message to the leaf motes using all-RPL-multicast address ff02::1a as shown in the Wireshark log in Figure 4.20. The motes send DIO packets with a period between 2 to 3 seconds. The state of the motes changes frequently due to new motes that try to join in the network. Therefore, the root mote sends the topology information with the new information using DIO messages to its neighboring motes. This process builds the graph from the root mote to the leaf motes. The leaf motes can send a data packet into the root the graph by just forwarding the packet to its immediate parent. This model represents a MP2P (Multipoint-to-point) forwarding model where each node of the graph is reachable toward the graph root.

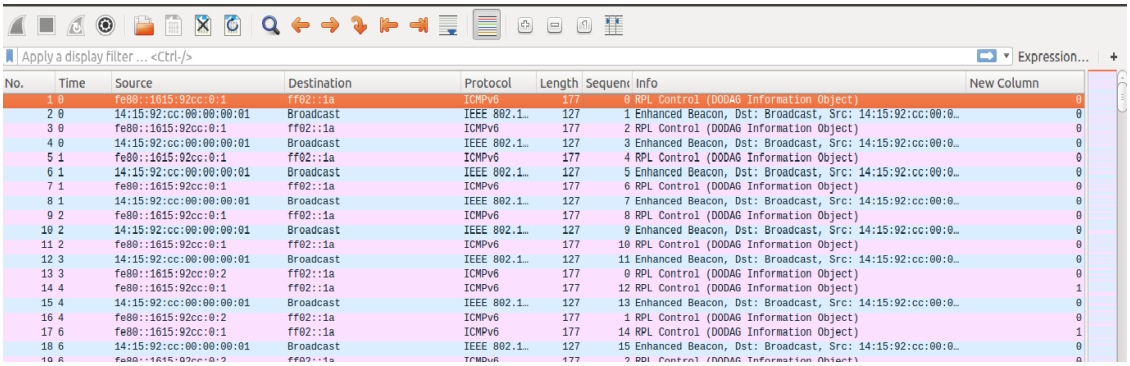


Figure 4.20: The-all-RPL-multicast address of RPL control message DIO from root mote

The routing choice is based on the used metrics and constraints defined by the OF. Object Function Zero (OF0) metric is used for routing metrics. Each mote computes its own rank when there is an inconsistency in the network, the Trickel Algorithm runs (see Figure 4.15). Then the neighboring motes in the network, when they receive DIO message, compare their state with the new DIO message and updates their states. The leaf motes will send DAO message to the DODAG root or their immediate parent.

The following Figure 4.21 illustrates emulated motes starts to build the graph

with their neighbors and contains five motes as seen from the terminal. The web based interface simulator OpenSim starts on the browser using the address http://127.0.0.1.8080/. One of the motes in the simulator was configured manually to be the root mote, then the motes start to discovering their neighbors using DIS control messages and join the network using DIO control messages. As seen from the Figure 4.21, Mote 2 and Mote 3 are synchronized to use the offset at "slotoffset 0" following the standard IEEE 802.15.4e. After they join the network, to maintain the network all the motes multicast a DIO control message except the leaf motes according to The Trickle Algorithm (see Figure 4.15). For every DIO message, the leaf motes will send to the root mote a Non-storing DAO message which uses a unicast address. The DAO control message, as reported by the OpenSim emulator, is shown in Figure 4.21.

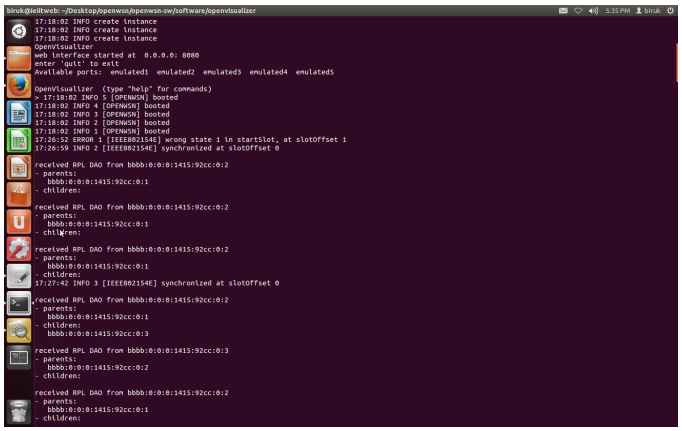


Figure 4.21: Neighboring Discovery and Non-Storing DAO message

The Wireshark log of Figure 4.22 shows a transmission of an Echo message from the root mote to a child mote. When the root mote starts pinging the child mote, the root mote sends an *echo request* message to the child mote and the child mote sends an *echo reply* message to the root mote. At this time an inconsistency is detected due to the exchange of messages between root mote and child mote through the network. Then, the Trickle algorithm reacts by sending DIO messages to the motes in the network using local communication. Motes will check their own states with a Trickle message and update their states accordingly. Everytime when inconsistecies occur in the network, a DIO message will be multicast to the child and leaf motes,

as shown in the Figure 4.22.

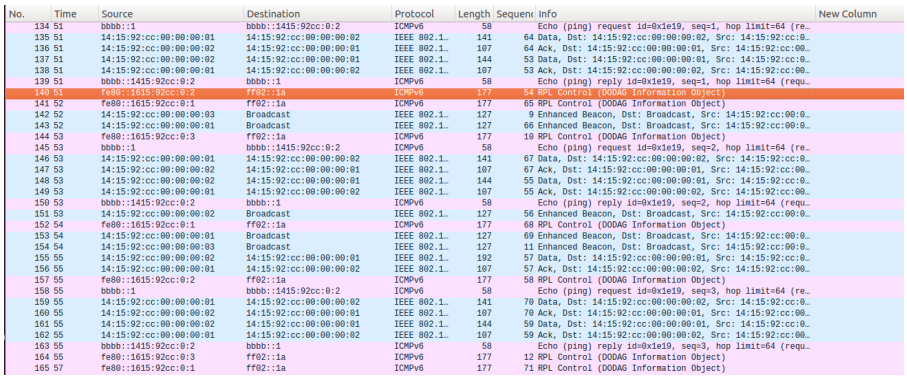


Figure 4.22: DIO messages after echo reply

During pinging, the terminal displays which mote has received the echo request. This is showed in the Figure below.

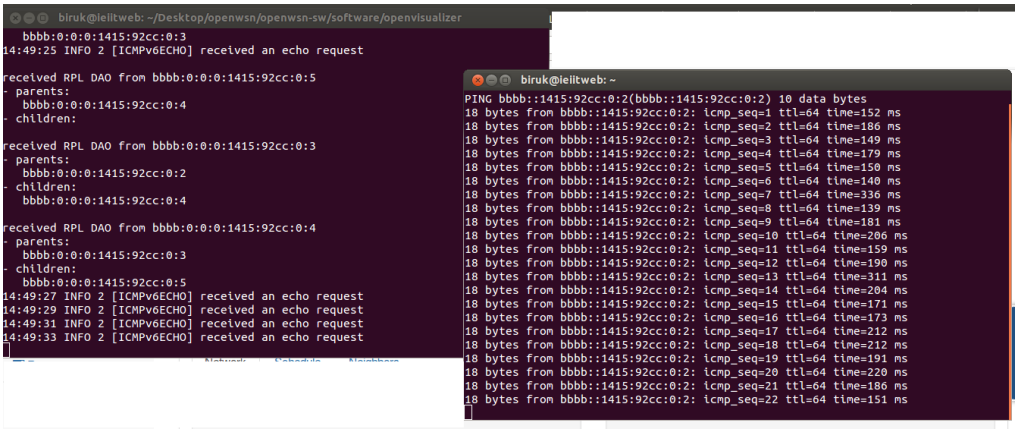


Figure 4.23: Terminal display during pinging and status of motes

To build the DODAG on the simulation, mote 0001 is selected as a DODAG root by the administrator. The root mote sends the graph information to its neigbours, and the rest of the motes join to the network using DIO message. The DODAG uses an OF0 as objective function to build the routing by finding the nearest grounded root. The DODAG graph construction with five motes in OpenSim simulator is illustrated in Figure 4.24.

Wireless links may be affected by weak signal reception, interference or other phenomena. At this point, by considering the above Figure 4.24, the idea is to analyze

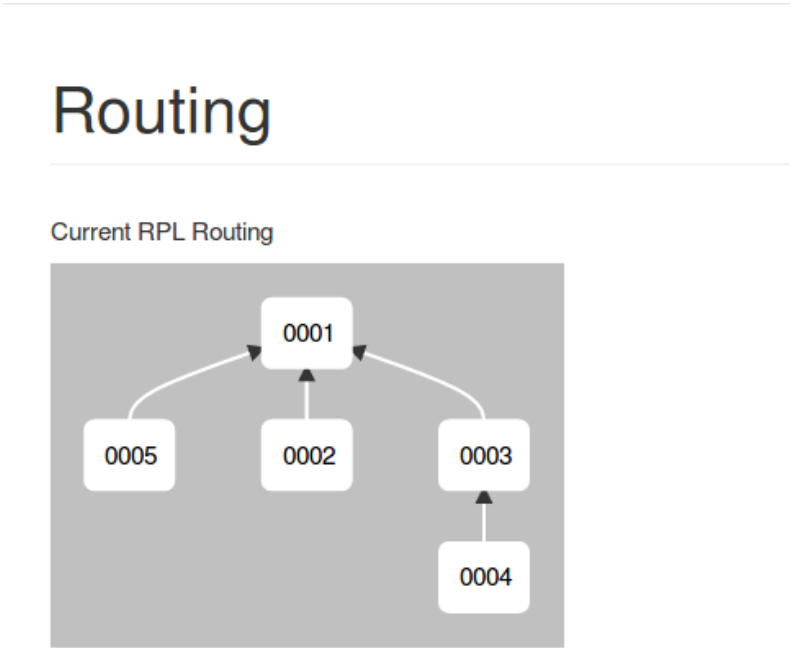


Figure 4.24: Five Motes DODAG

how much time will take the RPL routing to re-establish the connection between nodes, if the link between them is disconnected. From the above DODAG graph the root mote starts pinging mote 0004 and after some time, the link between mote 0003 and 0004 was disconnected manually from the simulation. After disconnecting the link manually, as shown in the Wireshark log in Figure 4.25, the root keeps sending echo requests, but the link that connects mote 0003 to mote 0004 is not available and the originator does not obtain any response from the destination mote until it rejoins the network and connects with mote 0003.

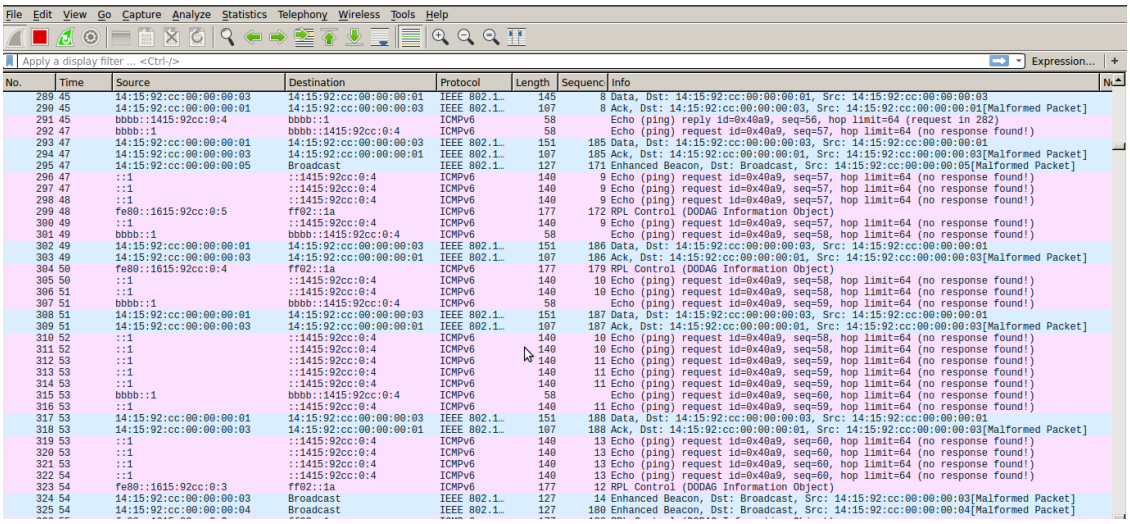


Figure 4.25: Link disconnection display captured in Wireshark

After 43 seconds, the disconnected Mote 0004 rejoins the network and connects with root mote using the RPL metrics defined in the OF, and starts replying to the echo request which was sent by the root mote. It also changes its routing, in fact Mote 0004 was reached by the root mote through Mote 0003, but now Mote 0004 makes its route directly to the root mote. Also the DODAG Rank is changed from rank two to rank one, and the DODAG Version changed from the DODAG version reported in Figure 4.24 to that reported in Figure 4.26.

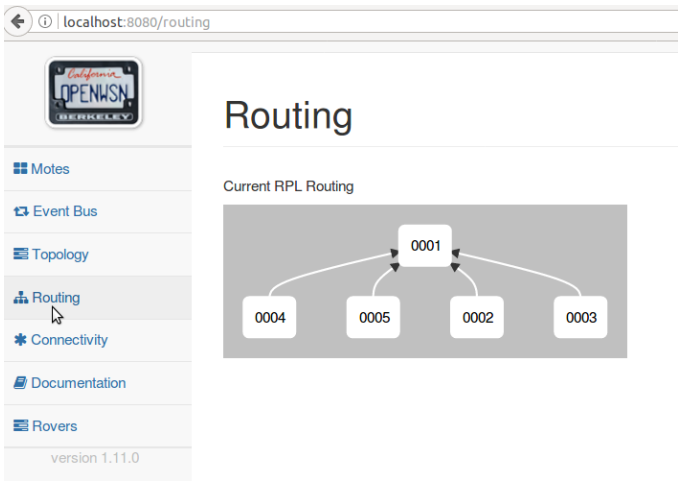


Figure 4.26: Routing displaying Mote 0004 rejoining to the network

Chapter 5

Simulation and Execution Platform for OpenWSN

5.1 Overview simulation Platform

Computer simulations are used to test the performance of protocols in different network scenarios, and assess their performance and practical values. There are three main types of simulation: the Monte Carlo Simulation, Trace-Driven Simulation, and Discrete-Event Simulation. Discrete-event simulation is widely used in wireless sensor networks, and communication networks in general, due to the simplicity of simulating lots of jobs running on different sensor nodes. This simulation can list pending events which are executed in order. Each event can modify the state of simulated devices, and generate other events that are added to the list [46].

The global variables, which describe the system state can represent the simulation time. This type of simulation includes input routines, output routines, initialization routines, and trace routines. In addition, it provides dynamic memory management, which can add new entities and drop old entities in the model. Debugger breakpoints are usually provided in discrete-event simulation, thus users can check the code step by step without disrupting the program operation. Instead, Trace-Driven Simulation provides different services. This kind of simulation is commonly used in a real system [45].

There are several simulation tools used to simulate open source low-power WSNs. These simulation tools can be categorized into two types: simulators and emulators. A simulator is universally used to develop and test protocols of WSNs especially in the beginning stage of their designs. The cost of simulating thousands of nodes networks is very low and the simulation can be finished within a very short execution time, which is typically smaller than the real time. The tool which makes use of real firmware and software to perform the simulation is called an emulator. Emulation can combine both real software and firmware implementation. Emulator relies on real code, thus it may provide more precision performance [47]. Usually, the emulator has high scalability, which can emulate numerous sensor nodes at the same time. Some of the simulation tools used in WSNs are OpenSim, NS-3 [48], OMNeT++ [49], Cooja [52].

5.1.1 OpenSim emulator

OpenSim is an event-driven emulator which is part of the OpenWSN project developed in University of California, Berkeley. The OpenSim environment combines elements from OpenWSN firmware and OpenWSN software. Emulated firmware motes are compiled as a python extension module OpenWSN creates an instance of the resulting class for each emulated motes and run them within OpenVisualizer.

The OpenVisualizer software is able to gather the debugging information on events, which are transfered by the hardware motes by means of a serial port. To run OpenVisualizer some necessary elements must be installed. These elements are different libraries and dependencies. In particular, they include Python (which is a popular programming language), PySerial, PyDispatcher, PyWin32 (Python Extension for Windows), Scons (used for tool execution), and TAP for Windows (IPv6 tunnel drive) used for Windows only.

On Linux, TUN/TAP is already included in the operating system, and OpenVisualizer configures it on the fly. The host computer needs to have `gcc` installed to be able to compile the firmware as a Python extension module. To compile the firmware, the compiler will need to have access to the Python.h header file. The Python header files should be present by default on Windows. On Linux, the

Python-dev package needs to be installed.

OpenSim is a part of OpenVisualizer. OpenSim is allowed to emulate a mote to generate data and inject to the event bus. The Event Bus provides the messaging framework. Specific components implement services like a connection of wireless motes via a serial connection, and external user applications via an IPv6 TUN interface. A simulated network behaves exactly the same as a network with a real device. Figure 5.1 shows how the emulated motes communicate with the EventBus.

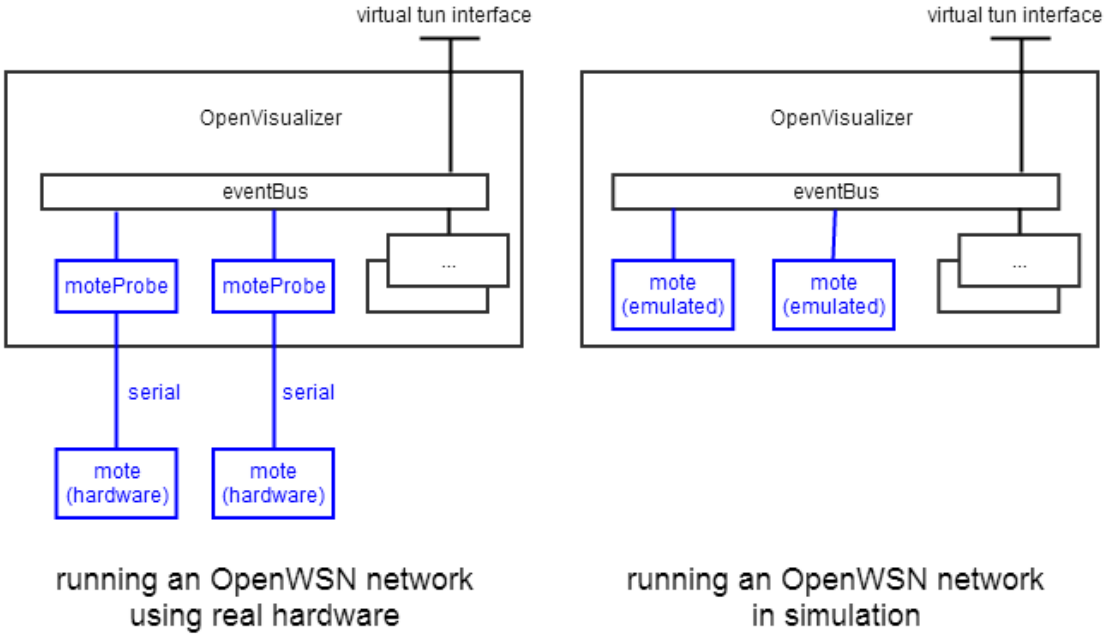


Figure 5.1: Hardware and Simulation architecture (Source: [53])

When the OpenSim simulation is running, these emulated motes communicate with the rest of the OpenVisualizer architecture. OpenSim allows users to simulate an OpenWSN network without physical devices and emulate a full network in python and C programming languages OpenSim is compatible with Windows and Linux. Multiple emulated motes are able to run concurrently by using the OpenSim simulation framework [53].

5.1.2 NS-3 simulator

Network Simulator 3 (NS-3) is a discrete event network simulator written in C++, with optional Python programming language. It was implemented using an Object Oriented structure, providing a modular environment in which various models could reuse the existing code for networking protocols, and networking stacks. Simulation scripts can, therefore, be written in C++ or in Python. NS-3 allows scientists to analyze Internet protocols and large-scale systems in a controlled environment.

It is an open source simulator targeted primarily to network researchers and educational use. NS-3 is not compatible with NS-2 (its predecesssor) and it is weak in MAC a physical layer development support. NS-3 supports large-scale systems. Since NS-3 generates pcap packets trace files, Wireshark can be used to analyze traces and it is compatible with Linux, Mac and Windows operating systems [48].

5.1.3 OMNeT++ simulator

OMNeT++ (Objective Module Network Test-bed in C++) is a discrete event modular network simulator, developed in C++. It is very simple to use, due to its clean design. OMNeT++ contains strong GUI libraries for animation, tracing and debugging. The simulator also has graphical tools for simulation building and result evaluation in real time.

OMNeT++ provides a hierarchical nested architecture. The modules are programmed in C++, and the GUI is created by using the Tk library. The modules are assembled into components and models by using a high-level language (NED). Modules communicate by sending messages. The main flaw of this simulator was the lack of available protocols in libraries when compared to some other simulators. OMNeT++ simulator has very good scalability and supports very large scale network topologies, and it is limited only by available memory on the user computer.

OMNeT++ simulator mainly supports standard wired and wireless communication networks. Without the proper extensions, the simulator lacks suitable protocols and energy models for WSN, since basic support is mainly for IP networks. Today, there

are several sensor network simulation frameworks based on OMNeT++. The Mobility Framework, for example, implements the support for node mobility, dynamic connection management and wireless channel model [49], [50].

Finally, OMNeT++ offers various frameworks for the deployment of RPL in a network. Additional available modules may be used to add support for energy consumption and mobility models, but might introduce compatibility problems . OMNeT++ is compatible with Linux, Mac, and Windows operating systems.

5.1.4 COOJA simulator

COOJA is a discrete event simulator written in Java, but allows the programming of a node in standard C programming language by using Java Native Interface (JNI). This enables the simulator to be easily extendable, and allows sensor node software to be written in C programming language. It was developed by the Swedish Institute of Computer Science. COOJA is primarily a code level simulator for networks consisting of nodes running Contiki OS.

Nodes with different simulated hardware and different software can be included in the same simulation. Code level simulation is achieved by compiling Contiki core, user processes, and special simulation glue drivers into object code native to the simulator platform, and then executing this object code from COOJA. COOJA is able to simulate non-Contiki nodes, which are implemented in Java, and act as a generic WSN simulator. COOJA can simulate sensor nodes at all levels of details. Additionally, nodes simulated at different levels of details can coexist and interact in the same simulation. This feature is called cross-level simulation.

COOJA offers simultaneous simulation of both low-level node sensors hardware and high-level network behavior. COOJA allows to change or replace all levels of the simulated system: sensor node platforms, operating system, radio transceivers, and radio transmission models. The simulator, however, due to cross level simulation and extendibility, is not very efficient. Simulation of many nodes with several interfaces each requires a lot of computational power. A test interface GUI is absent, thus making extensive and time-dependent simulations difficult. COOJA simulator

fully supports RPL and has relatively low efficiency for large-scale simulations. Scalability is up to 200-500 nodes for simulation and takes long processing time. The simulator supports Linux, Mac and Windows operating systems [52].

5.2 Configuration of OpenSim

OpenWSN can be installed and it is compatible with different operating systems. For this thesis, the configuration is done in Linux OS. In Linux, OpenWSN can be configured in simulation mode and integrated with simulated OpenWSN hardware devices. In simulation mode it allows to use the `ping` tool and it allows simulated devices to interact with CoAP. OpenSim is used to simulate an OpenWSN network without the physical device.

The first step is to create a directory "OpenWSN" on the desktop, then inside "OpenWSN" directory download updated OpenWSN firmware source code, software source code and python module from Github. The firmware source code is used to run motes which are emulated, the software code is used to run the simulation on a computer and the python module is used to implement the CoAP service. To run the simulation, OpenSim compiles a python extension and creates an instance class for each emulated mote. The simulation web-based interface is run from the Linux terminal as shown in the Figure below.

In OpenSim, OpenVisualizer is used to connect the OpenWSN network to the internet over a virtual interface, to control the status of the OpenWSN network and communicating with the mote. It works for both Linux and Windows OS and interacts with the motes connected to it over a serial port. It works for real devices and emulated motes. It includes a 6LoWPAN Low-power Border Router (LBR), i.e., it converts 6LoWPAN packets into IPv6 packets, thereby connecting the low-power wireless mesh to the Internet. LBR provides border router translation between IPv6 packets on the external network and 6LoWPAN packets on the LLN. It generates the events messages, which then may be handled by the RPL protocol.

OpenVisualizer has three types of user interfaces. These are a graphical user interface, a command-line user interface, and a web user interface. In this thesis, the

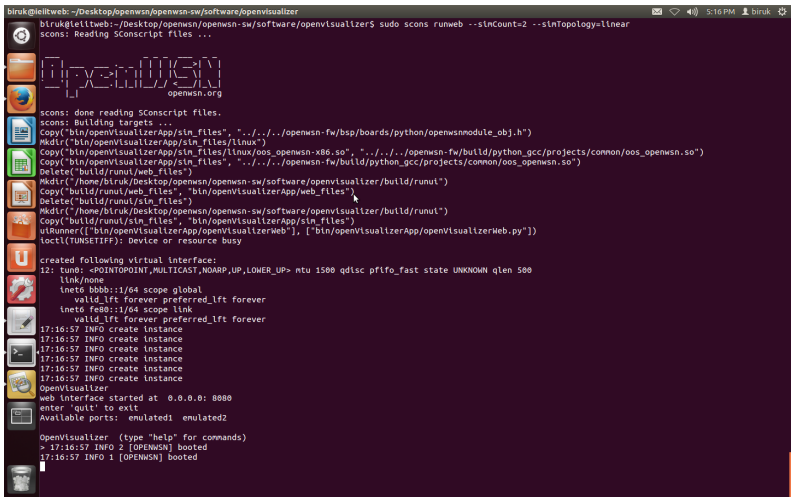


Figure 5.2: Starting OpenWSN from the command line

web-based user interface is employed. The web-based user interface provides relevant information about the OpenWSN network. This includes the internal states of each mote (neighbor table, communication schedule, routing table, etc.), the multi-hop routing structure, debug information and error messages generated by the motes. The OpenVisualizer also sends commands and data to the motes, especially packets coming from the Internet.

The next step to start the simulation is to install Python header files and Scons for compiling the firmware as a Python extension module, as a form of a shared library. When OpenVisualizer run with `sudo` the Python program will create a tun interface and an open source software construction tools help the OpenVisualizer to run on web user interface. The web interface in OpenWSN simulation can be displayed in the browser. In OpenWSN simulation to display the web interface http://127.0.0.1.8080/ address must be used, as shown in Figure 5.3.

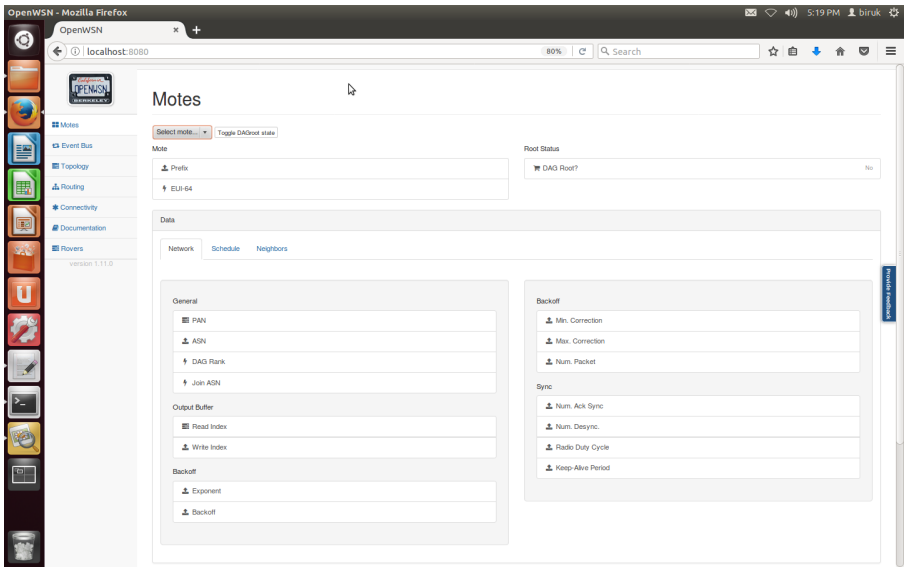


Figure 5.3: OpenMote Home page

As shown in the above Figure, the home page displays the DAG root state, an IPv6 address of the motes, status of DAG root, and other data. On the left side of the Figure some tabs related to Event Bus, Topology, Routing, connectivity, Documentation, and Rovers are shown.

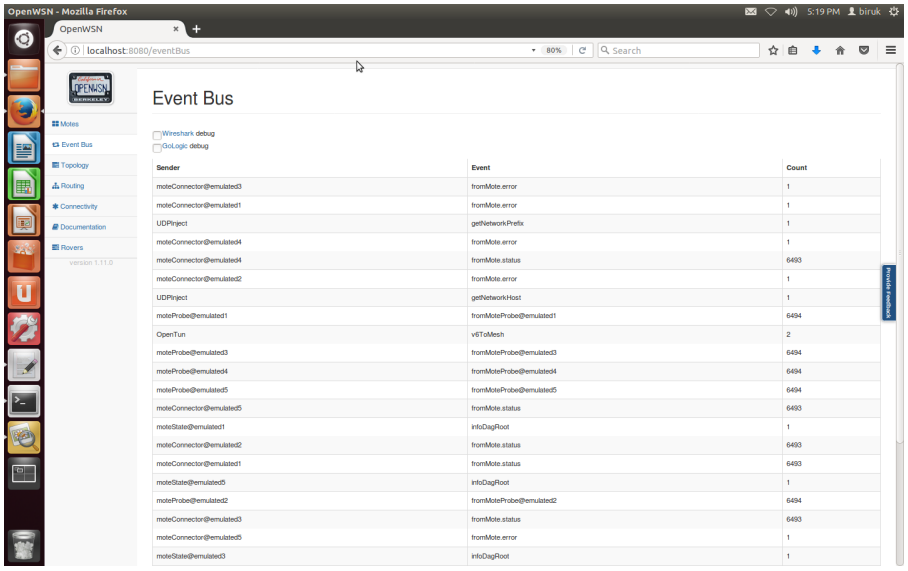


Figure 5.4: Event Bus of OpenWSN page

The components defined inside OpenVisualizer use a message to communicate with

each other. Event bus is used as a communication pipe between emulated motes. The Event Bus also provides to the simulation environment; sender information,i.e., a description of the kind of event and the number of events occurred.

In the Topology tab, the position of the motes and their links are displayed, as shown in Figure 5.5. In the simulation, using the computer mouse, it is possible to move the positions of the motes by left click on a mote. The user can also right click on the two motes to connect them with a wireless link, and it is possible to change the link quality by modifying the value of the Packet Delivery Ratio (PDR), which can be used to measure the performance of the network in a more realistic operating condition.

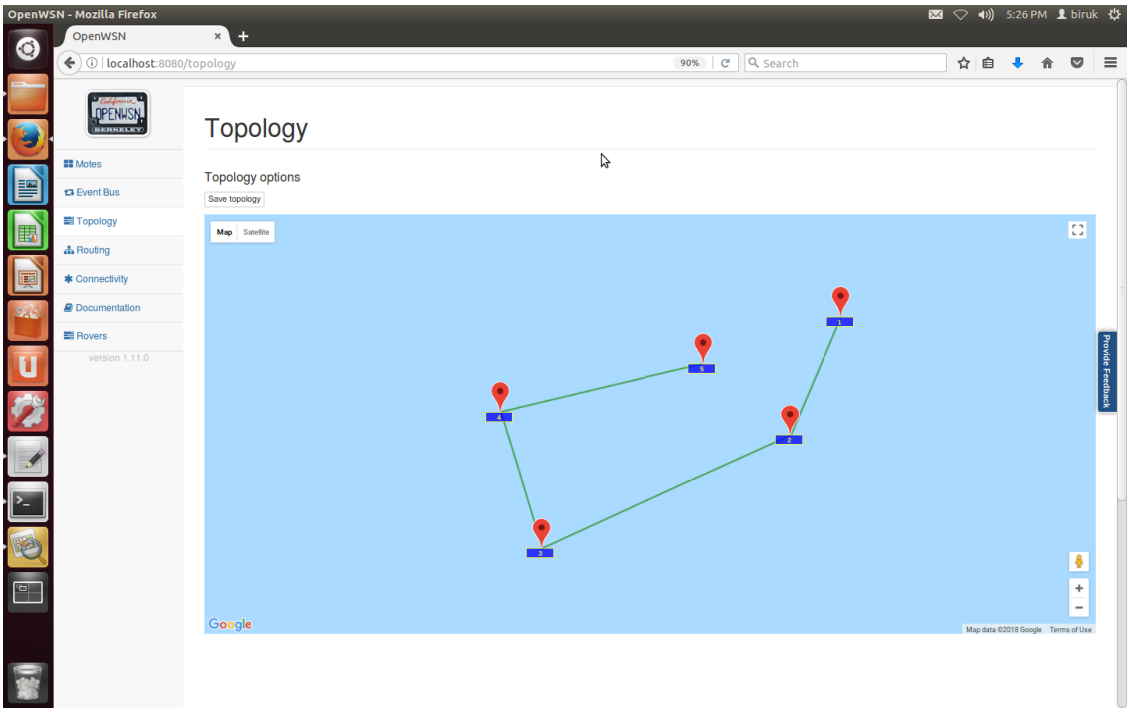


Figure 5.5: Linear topology view on OpenWSN simulation with 5 motes

The Routing tab shows the resulting RPL route in a given topology. The definition and generation of the new routing starts when a mote is Toggled as a DAG root in the Mote page.

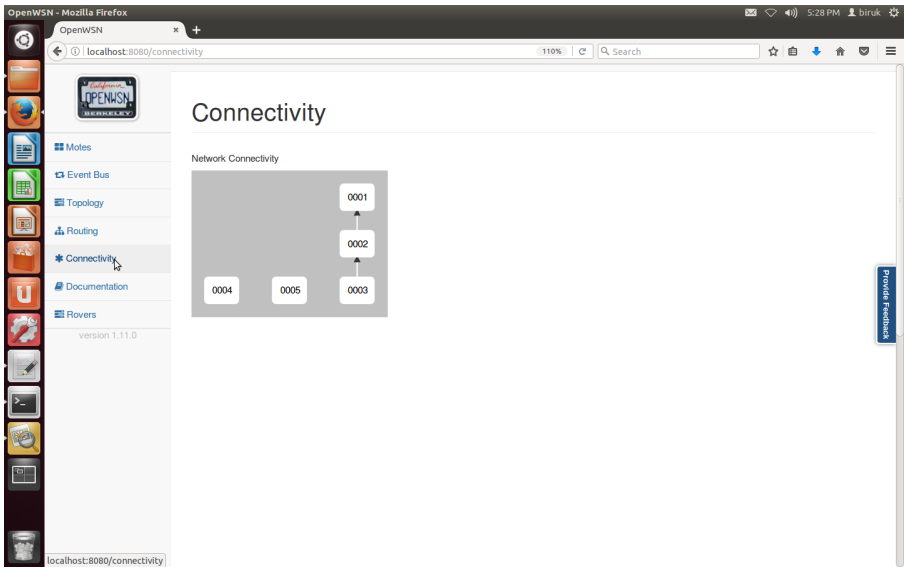


Figure 5.6: Connectivity Status in a linear topology consisting of 5 motes

The Rover tab is used to combine the OpenMote B hardware and the OpenWSN software. The OpenVisualizer monitors the internal state of motes. The state is presented through a web interface. Rover extends the OpenVisualizer by allowing motes plugged into different computers to remotely connect to it. Once connected, a user monitors and manages the motes exactly as if they were connected locally. Rover also allows the user to remotely reset or reflash the motes with any arbitrary firmware.

Chapter 6

Result and discussion

The experiments were performed on a computer running Ubuntu 12.04 LTS. The simulations are based on Open Visualizer (OV). OV is a part of OpenWSN, it resides on the host computer and it provides communication and visualization functionalities by creating a IPv6 TUN (network TUNnel) interface for the motes' network. Wireshark was also used to analyze the experimental results obtained from the simulation. The results obtained from the simulation platform are quite variable because the code of the nodes is exactly that of a real device, and the only source of inaccuracy is related to the simulstion of the characteristics of the communication medium (which is an emulator (i.e. it runs exactly the same code executed in the real devices), behaves exactly like a real-life device. Simulations are carried out to analyze routing performance. we performed four experiments. The first one measures the RTT using Linear Topology with a variable number of hops between 1 to 4. The second one measures the latency by sending a huge amount of traffic and show the results using cumulative distribution functions. The third experiment measures the quality of the link between motes when changeing the packet delivery ratio from 30% to 100%. The last experiment analyzes how much time a mote will take to rejoin a network if the link between two nodes is disconnected.

6.1 Experiment 1: RTT with Linear Topology

This experiment measures the latency by determining the time required by a packet to travel from source to destination mote and back from destination to source mote, which is called "round-trip time" (RTT). The Minimum RTT, Average RTT, Maximum RTT and the Standard Deviation of the motes was analyzed to measure the latency in the network. We took into account four linear topologies (DODAG graphs) with a different number of motes. The first mote in all linear topology DODAGs was configured as a root mote, and the RTT test was implemented for each mote by sending ICMP packets from the root mote. The generation of such a kind of packets was obtained by means of the `ping` Linux utility. The DODAGs used for these experiments are illustrated in the Figure 6.1 below.

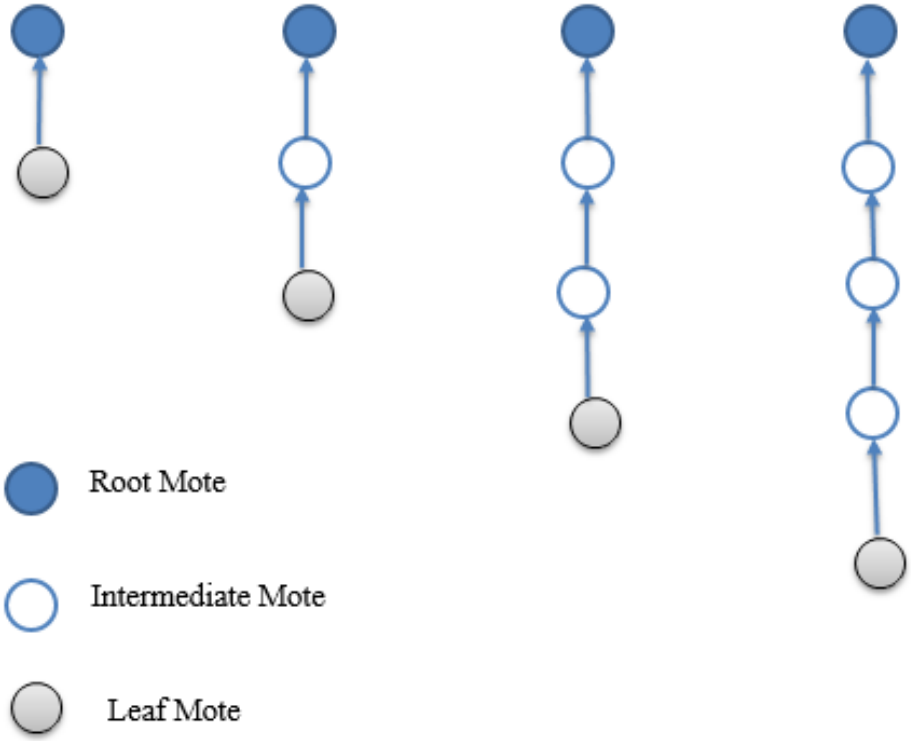


Figure 6.1: Linear Topology with 2, 3, 4, 5 motes respectively

The root mote sends for each child motes 500 sample packets. Each packet has a size of 10 kilobytes and two seconds waiting interval between sending each packet.

The Packet Delivery Ratio (PDR) was set to the value 100%. as a consequence, for this set of experiments, the network was ideal, i.e., packets have the certainity of being delivered to the destination mote.

As expected the topology which has only 2 motes (only one hop) obtain the minimum RTT (i.e., 54 ms). The minimum value of the RTT for the first hop with configurations with 3, 4 and 5 motes is 85 ms, 115 ms, and 148 ms, respectively. The same result considerations hold for maximum RTT, and Average RTT. The standard deviation also, as shown in the above Table 6.1, increases as the number of hops increases. These results implicate that the mote which is near to the grounded root shows better performance and increased determinism (lower standard deviation and variance).

In addition, commenting the experimental results obtained with the topologies of Figure 6.1 and reported in Table 6.1, the motes which have the same hop distance from the root mote have different values of measured latencies under Linear topologies characterized by a different number of motes. As shown in Table 6.1, in a linear topology with 3, 4, and 5 motes, the second hop minimum RTT is 158 ms, 236 ms, and 250 ms, respectively. Also, the maximum RTT for 3, 4, and 5 motes is 1085 ms, 1463 ms, and 1225 ms, respectively. Finally, the Average RTT for 3, 4 and 5 motes is 209 ms, 310 ms, and 336 ms, respectively. This is because of the amount of traffic increases as the number of motes in a network increase. This is due to DIO RPL control message that was multicast every time an event occurs in the network. In addition, there were the messages of the IEEE 802.15.4 protocol that are as observed in Wireshark logs. This will vary the value of the RTT for those motes that have the same hop distance from the root mote, but in linear networks characterized by an increase in the overall number of motes. Therefore, the amount of traffic that passes through the network has an impact on the quality of communication, even when the PDR is 100%. When a network has a large number of motes, the amount of traffic also increases, in addition to the ICMP packets that we sent to execute the experiments.

In the Figure 6.2 (a), (b), (c) and (d) the maximum RTT, the minimum RTT, the average RTT, and the standard deviation show that all the statistical indices related to RTT increase when the numder of hops between the root and the destination

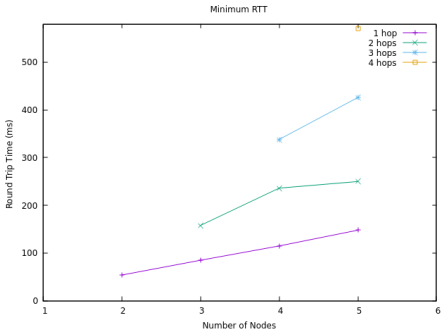
Number of Motes	Number of hops	Minimum RTT(ms)	Average RTT (ms)	Maximum RTT (ms)	Standard Deviation (ms)
2	1	54	74	210	17
3	1	85	123	381	33
	2	**158**	**209**	**1085**	59
4	1	115	160	600	46
	2	**236**	**310**	**1463**	100
	3	338	428	1536	111
5	1	148	219	1246	103
	2	**250**	**336**	**1225**	107
	3	426	561	1961	164
	4	571	720	2341	220

Table 6.1: Linear Topology with 2, 3, 4, 5 motes, respectively

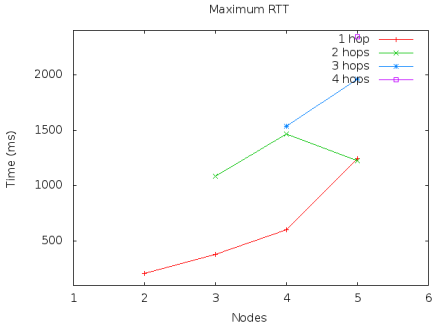
mote increase. Due to the linearity of the topology, the routing was also linear and it is not possible that the root mote can connect and communicate directly to the child motes. The root mote, when it tries to communicate with the last mote of the DODAG (which is more than one hop), should passes through the nearest child of the root mote as shown the above Figure 6.1.

Therefore, when the destination mote is far from the root, the time to get a response from the destination mote to the root mote is increased. For this reason, when the number of hops increases, the determinism of the network becomes lower and latencies increase, which means that a mote far from the root mote will have a higher variance and standard deviation, and also higher RTT. High RTTs can cause in the network long delays and high latency.

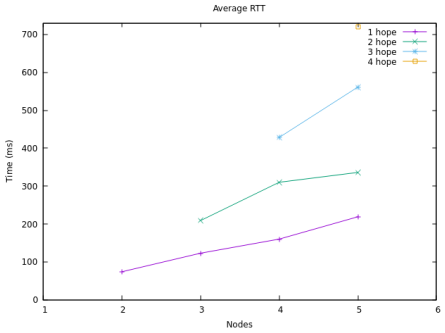
This affects the efficiency of the system, increases the response time and decreases the energy resource and durability of operated motes. In particular, high latencies and low determinism are not compatible with many control applications, espetially in the context of industrial application where determinism is of primarly importance.



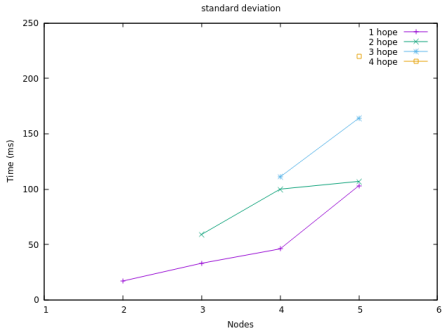
(a) Number of nodes vs Minimum RTT



(b) Number of nodes vs Maximum RTT



(c) Number of nodes vs Average RTT



(d) Number of nodes vs Standard deviation of the RTT

Figure 6.2: Experimental results on how the RPL is influenced by the number of hops in a Linear Topology

In conclusion, The motes in a network with Linear topology can communicate with a different communication quality that depends on their distance from the root mote. In particular, the communication quality decreases with the distance inversely proportional with the distance of the root mote.

6.2 Experiment 2: Latency

In this experiment we consider linear topology with 4 and 5 motes as shown in the Figure 6.3. The mote labeled 0001 (for both linear topologies in Figure 6.3 (a) and (b)) was configured as root, and the rest of the motes in both figures have a routing directed to the the root mote (i.e., mote 0001). Motes that are not directly

connected with root mote communicate with root mote through their immediate parents. In particular they use the Non-Storing mode (see Section 4.1.2).

(a) Four motes Linear Topology

(b) Five motes Linear Topology

Figure 6.3: The two Linear Topologies used in the experiment

The `ping` command executed in the PC (to which the root mote is connected through a USB cable) sends 7000 sample data packets with 100% PDR and two seconds interval between adjacent packets. The experiment analyzed and measured the round trip time using a linear topology contains 4 and 5 motes and which has 3 and 4 hops, respectively. The main difference from the previous experiment is the higher number of samples we used to statistically analyze the results. Using more samples allowed us to obtain results that have a higher statistical significance allowing us to plot the distribution of latencies. This is very important to investigate the characteristics of the network and to allow developers to correctly size it in the design phase.

As shown in Figure 6.3, the root mote is the first mote of the topology for both of them. The root mote pings mote 0004 (see Figure 6.3(a)) and mote 0005 (see Figure 6.3(b)). The results of the two experiments are represented in Table 6.2. In Figure 6.4 the Cumulative Distribution Function (CDF) of the RTT shows how the two configurations affect the communication quality.

Number of Motes	Minimum RTT(ms)	Average RTT (ms)	Maximum RTT (ms)	Standard Deviation (ms)
4	325	567	1423	206
5	571	991	3369	375

Table 6.2: Experimental results of the Round Trip Time for Linear Topology with 4 and 5 motes

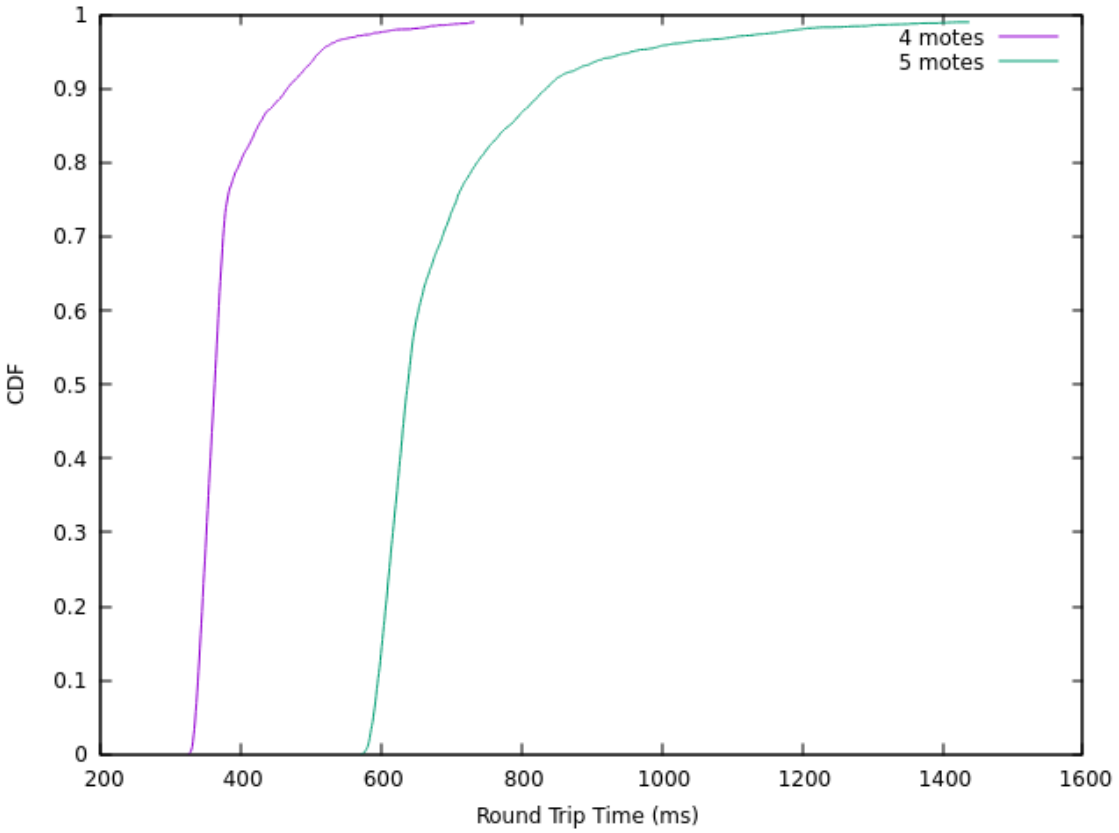


Figure 6.4: Cumulative Distribution Functions of Linear Topology with 4 and 5 motes

The experiment results of Table 6.4 show that the Minimum RTT, Average RTT, Maximum RTT and the Standard Deviation of the RTT of a DODAGs which has 4 motes is smaller than a DODAG composed of 5 motes. The DODAG with 3 hops (a network that has 4 motes) has a minimum RTT equal to 325 ms, a maximum RTT equal to 1423 ms, and an average RTT equal to 568 ms. The DODAG with 4 hops has the Minimum, Maximum and Average RTT of statistical indices referred

to the RTT equal to 571 ms, 3369 ms, and 991.203 ms, respectively.

When the number of hops increases, there is an increase also in RTT. The RTT increases, because of the high number of transmitted packets: packets sent and received from the source (root mote) have to travel through the network to reach their destinations. This long delay results in higher latency. The higher traffic affects the performance of the network and decrease the network determinism (see Table 6.2).

The Cumulative Distribution Function in Figure 6.4 shows that, when the number of hops increases from 3 to 4, there is an increase on latency approximately around 250 ms. If we cross the CDF with an horizontal line place on the value 0.9, we can conclude that the 90% of the samples are transmitted with a delay lower than 400 ms in the 4 motes configuration, and a delay smaller than 800 ms for the 5 mote configuraton.

6.3 Experiment 3: Link quality with respect to Packet Delivery Ratio (PDR)

In this experiment is aimed at observing the performance of a DODAG network when the quality of the link between two motes is changed. The quality of the link can be tuned by setting the Packet Delivery Ratio (PDR). PDR is the ratio, computed for a specific link between two nodes, of the number of packets successfully received by the destination mote over the number of packets sent by the source mote.

Link quality unavoidably affects protocols and applications in wireless networks. The experiment uses two motes, as shown in Figure 6.5, connected linearly. The `ping` program executes in the PC generators, as we seen from the other experiments, echo request packets will be sent to mote 0002. Each experiment consists of 500 packets generated with a period of two seconds. The value of PDR was changed manually from 30 % to 100 % in the simulation, and the results were captured and analyzed.

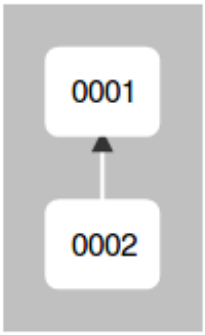
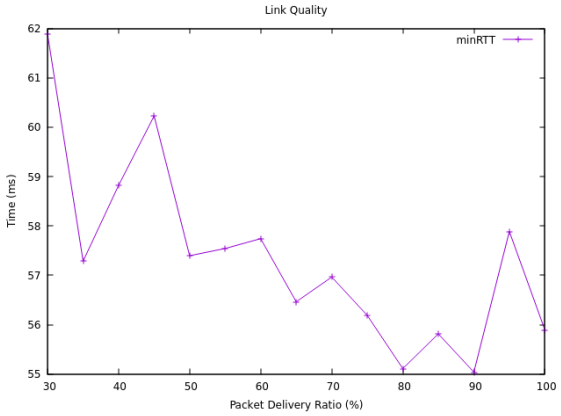
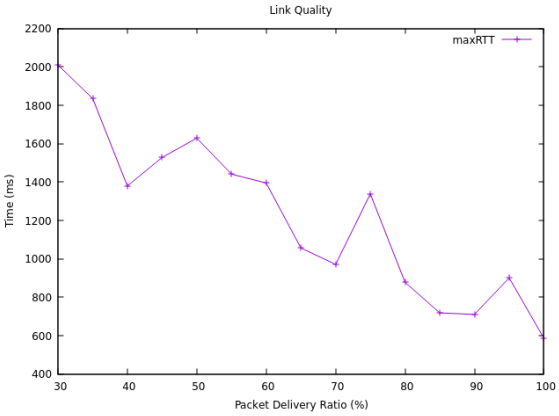


Figure 6.5: Linear routing topology with 2 motes used in the experiment

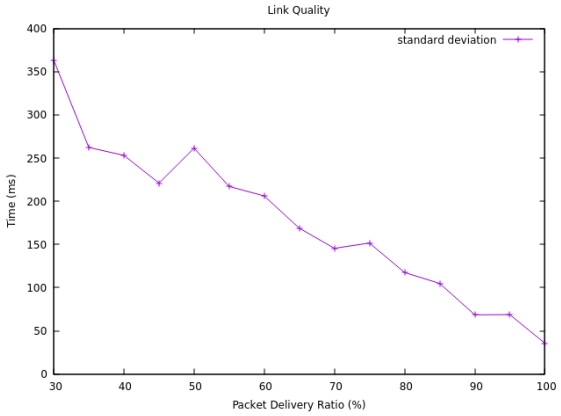
The results are shown in the following Figure 6.6.



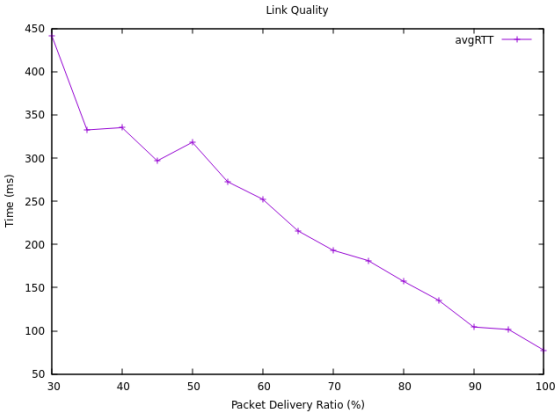
(a) Minimum RTTwith different PDR



(b) Maximum RTT with different PDR



(c) Standartd deviation with different PDR



(d) Average RTT with different PDR

Figure 6.6: Round Trip Time vs Packet Delivery Ratio

Figure 6.6 shows the RTT for the network when the quality of a link increases. The time required for a packet to travel from a root mote to specific destination mote and back again to the root mote (i.e., round trip time) is inversely proportional to the quality of the link (that is the Packet Delivery Ratio).

This behavior is similar for all the statistical quantities analyzed in Figure 6.6, i.e. minimum, maximum, standard deviation and average value. This relation can be explained observing that a packet that does not reach its destination is retransmitted.

The RTT increases because each retransmisstion, as stated by the TSCH technique, can only be scheduled in specific time slot, as a consequence, each time a retransmission is needed, the originator has to wait its scheduled time slots, with a consequent increase on the RTT.

As can be seen from the plots, the relation between the PDR and the RTT is linear. This can be easily observed in the plot of Figure 6.6 (d). In the plot related to the minimum, instead, there is not a linear dependence between the two quantities because, with a sufficient number of samples, there is a high likelihood that a packet can cross the entier network without experiencing any retransmisstion, also in the case of a network with low values of PDR.

The other statistical quantity analyzed in this experiment is the interaction between the Packet Loss Ratio and the PDR. It is important at this moment to point out that the Packet Loss percentage is computed at the application level, while the PDR at the link level (i.e., without traking into account retransmission).

As a matter of fact, a packet is lost at the application level when all its retransmissions (and the original transmission) are lost at the link level. Since in the OpenWSN implementation of the 6TiSCH protocol stack the number of transmissions of each data packet is 4 (i.e., 1 transmission plus 3 retransmission), the values of the packet loss percentage is affected only when all the 4 retransmissions at the link level are lost. The packet loss rate decreases when the packet delivery ratio increases. These results are shown in Figure 6.7.

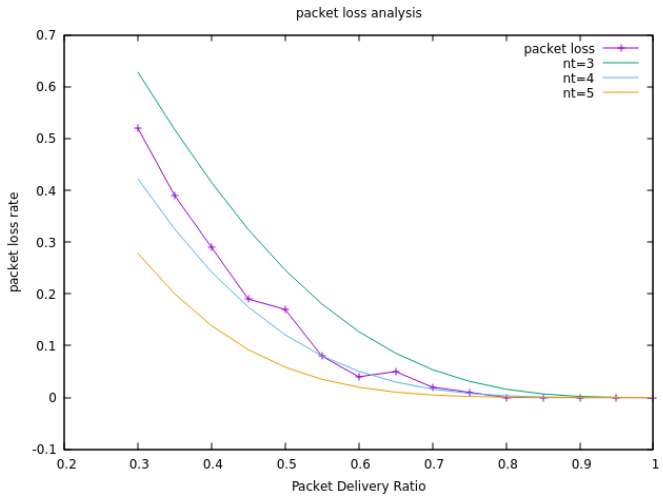


Figure 6.7: packet delivery ratio vs Packet loss

As expected, in a network with only two motes, when the link quality is better (i.e. high values of PDR), also transmission times are small, and motes have a higher probability to meet their communication deadline. If the Packet Delivery Ratio decreases, the rate of packet loss will increase. As we observed in Figure 6.5, the network performance with variable link quality has an impact on the quality of communication in terms of reliability between two motes.

When we have more than two motes in the network and the number of hops increases, the network performance decreases and as a consequence, the communication network becomes less reliable. As shown in Figure 6.7, we analyzed the correspondence between analytical and theoretical curves.

Since there is correspondence, we can also use the theoretical formulation to analyze the behavior of the network in the case the number of transmissions is 3, 4, and 5. These experimental conditions are identified with the labels nt=3, nt=4, and nt=5, respectively. The formula that links the PDR and the number of transmissions "nt" with the packet loss probability (PLP) is:

$$PLP = 2 \cdot (1 - PDR)^{nt} - ((1 - PDR)^{nt})^2 \tag{6.1}$$

In RPL the maximum number of transmission is 4 , consequently, a packetcan be

transmitted 4 times before being dropped.

6.4 Experiment 4: Rejoin Time

Basically, if RPL finds a better link or the communication on a link become bad because the link is interrupted, it changes the routing in order to use a different path by selecting its preferred parent to reach its destination mote. The process does not only change the preferred parent but it could change also the Rank of the mote.

RPL uses Rank to measure and indicate the hop distance of a mote from the root mote in the network. If the Rank of a mote is large, the mote will become far from the root mote. If the network is unstable, the route will change many times. When RPL selects a new path, it may not change the mote's Rank because the new route might have the same length.

The Last experiment measures how much time a mote took, when disconnected from the current network, to rejoin to the network again. As described in Chapter 4, when a mote is disconnected from the network, to rejoin the network again, it requests a DIO control message from its neighbors by sending a DIS control message. When a disconnected mote receives a DIO control message from a mote already in the network, this mote will be immediatly selected as a new parent. Finally, the disconnected mote replies to its new parent mote using a DAO control message.

Therefore, the time measured is between a mote is disconnection from the network and the instant the same mote rejoins the network by sending a DAO control message. This is done in a given mesh topology using the DODAG constructed by the RPL routing protocol. After constructing the DODAG tree graph, the root mote will be configured and the PC starts pinging one of the child motes with 100 packet samples. The packet size is 10 kilobyte and the interval time between the adjasent packet is 2 seconds. During the pinging time, one of the links was disconnected manually from the randomly topology constructed in the simulation by RPL.

Then, each measure was taken as the interval from the time the root mote sends an echo request message generated by the PC and the echo request message is lost in the network (i.e., it is not delivered to the root mote), and the time the disconnected mote rejoins the DODAG by selecting its preferred parent (using RPL control messages) and it starts replying to the requests sent by the PC. In the following, some examples show how the topology changes when a mote disconnectes from the network and rejoins to the network. These examples are illustrated and shown in the Figure 6.8.

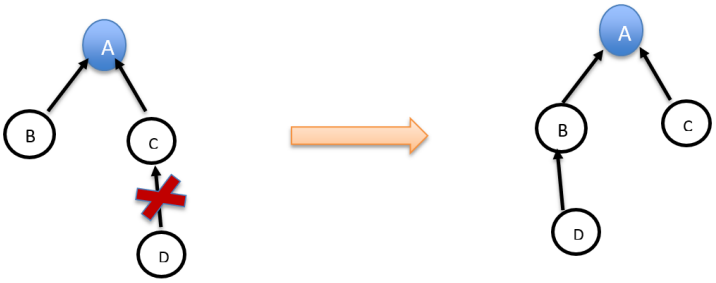
It is important for the routing protocol to react to changes in connectivity by rapidly reconfiguring the topology, while maintaining at the same time a low control overhead and power consumption.

For this experiment Wireshark is used to capture the packets exchanged in the network. In particular, wireshark is used to trace the sending time of the echo request packet and the presence of the coressponding reply, and measure the time taken by a disconnected mote to reconnect to the network. Experiments for each topology were repeated 10 times. The experimental results are shown in Table 6.3.

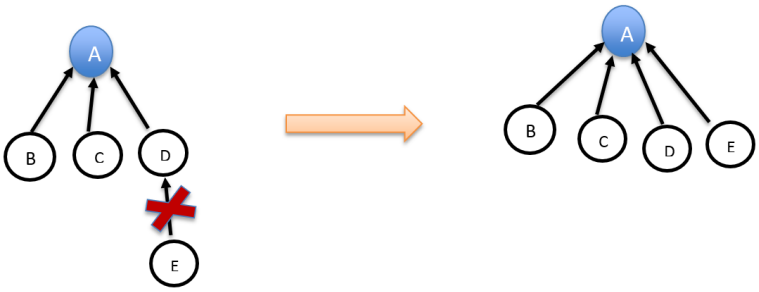
Number	3 motes Topology (s)	4 motes Topology (s)	5 motes Topology (s)
1	20	37	123
2	21	28	42
3	24	25	93
4	23	51	86
5	74	29	34
6	53	87	36
7	6	46	99
8	60	52	73
9	7	89	43
10	10	89	44
Minimum	6	25	34
Maximum	74	89	123
Average	29	53	67

Table 6.3: Time intervals between a mote disconnected from the network and rejoin to the network

(a) DODAG with 3 motes



(b) DODAG with 4 motes



(c) DODAG with 5 motes

Figure 6.8: Examples of how the topology changes when the link is disconnected and a mote rejoins the network

For not highly demanding Industrial applications, the routing protocol should converge within tens of seconds so that a device is able to establish connectivity to any other point in the network or quickly detect connectivity issues. Repeating the same experiments more times, we observed that the rejoin time varies. The rejoin time can range from few milliseconds under ideal conditions between closely spaced motes, to several seconds under different conditions between motes separated by a large distance (i.e. a high number of hops). The minimum time obtained for a 3

motes DODAG network is 6 seconds, for a 4 motes DODAG is 25 seconds and for a 5 motes DODAG it is 34 seconds.

The maximum time obtained for a 3 motes DODAG is 74 seconds, for a 4 motes DODAG 89 seconds and 123 seconds for a 5 motes DODAG network. The average times computed over 10 different runs of the simulation are 29 seconds for a 3 motes DODAG, 53 seconds and 67 seconds for a 4 and 5 motes DODAG networks, respectivelly. This result shows that when the number of hops increases, the time a disconnected mote needs to rejoin the network also increases. The analysis presents a small rejoin time for the motes with small hops. Those motes join quickly the network according to the routing protocol requirements, but an increase in the number of hops lead to longer rejoin times.

In fact, when the number of hops increases, the motes will wait a longer period to rejoin the network, resulting in a large number of packet dropped during the interval needed by the RPL protocol to establish a new route to the node. During this interval, communication with the disconnected mote is not possible. For this reason, the overal performance of the network decreases, because some motes are not reachable.

As discussed in Chapter 4, RPL (more in general 6TiSCH protocol stack) was designed for low power and lossy network wireless devices, to increase efficiency in Industrial, Commercial, Home and Urban networks. All these applications require to deliver information with low latency and to save the energy resources to decrease as much as possible the maintenance required by the motes after they are installed in the field.

The delay required by a disconnected mote to rejoin the DODAG network must be small because motes are placed in a network to deliver information or to do some specific task usefull for the application. If a mote does not complete the specific task within a required time (deadline), some applications, especially those used in industry, may not work work properly. To solve such kind of problems, and to increase operational efficiency in a network, we propose two methods: backup link or active redundancy path.

6.4.1 Backup Link Method

RPL protocol constructs a DODAG graph which is acyclic with all edges directed to the root and each source mote can use only one path for data transmission and switch to another path upon link failures. After topology construction, if one of the links is broken, according to RPL routing protocol specifications, it will take tens of seconds to minutes to reconstruct the new topology with the new route. Especially for industrial applications, these times are too long. The packets that are transmitted during the interval are dropped. Therefore, it is better to apply a backup link for the mote when it disconnects from the network. To use the backup link, we have to be sure that the time to reconstruct the new route and the new topology is lower than the time the RPL protocol requires to reconstruct the new route. We must also have to define when to use the backup link. A possible solution is switching to the backup link after a given number of frames have been lost. Heuristics can be used to select the optimal number of frames (that can be decided with some heuristic algorithm). The idea is illustrated in the Figure 6.9.

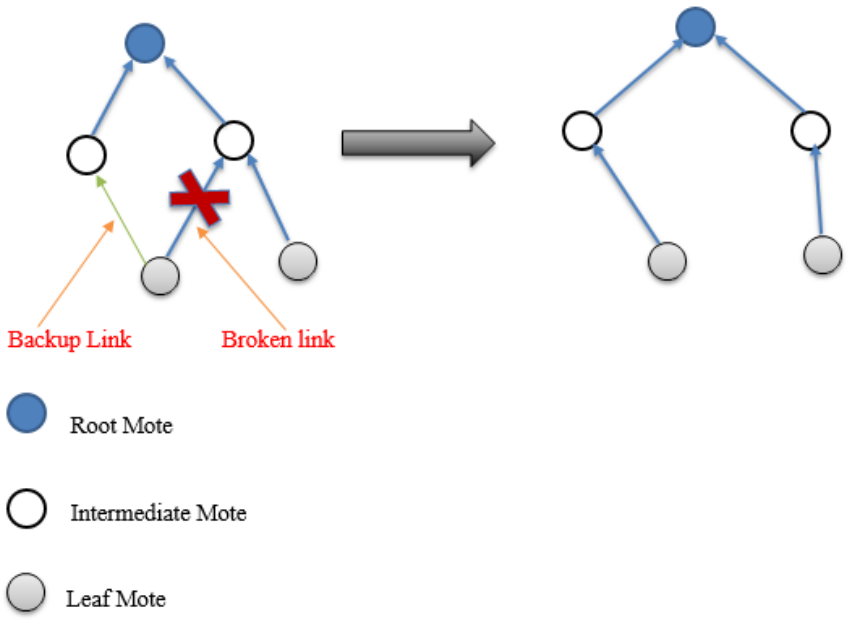


Figure 6.9: Reconstruction of the new network topology by using the backup link

When a link is broken between the leaf mote and the intermediate mote, the backup

link will be used and a new topology will be reconstructed to maintain reliable communication and to keep an uninterrupted data exchange in the network. It is important to have uninterrupted communication in WSNs for all applications, such as industrial ones, which require an immediate response.

6.4.2 Active Redundancy Path Method

The other method is active redundancy paths. Active redundancy plays an important role to maintain a communication in the network. This method consists in creating redundant routes in a network topology already constructed by RPL routing protocol using an improved version of the protocol aimed at generating two completely disjointed routes. The main idea is illustrated in Figure 6.10.

In Active redundancy systems, a source mote will send two copies of the same message (the purple and green color) as shown in Figure 6.10 to the destination using the two disjointed paths. The destination mote will take the message that arrives first. But, if one link is broken, one of the two copies of the message will arrive to the destination, and the network continues its communication without any loss and delay.

To be sure that at least one message arrives to the destination with a single point of failure in the route topology, we need that the two paths are completelly disjoint. The drawback of the proposed algorithm are a higher power consumption and a higher bandwidth usage compared with other algorithms not based on redundancy.

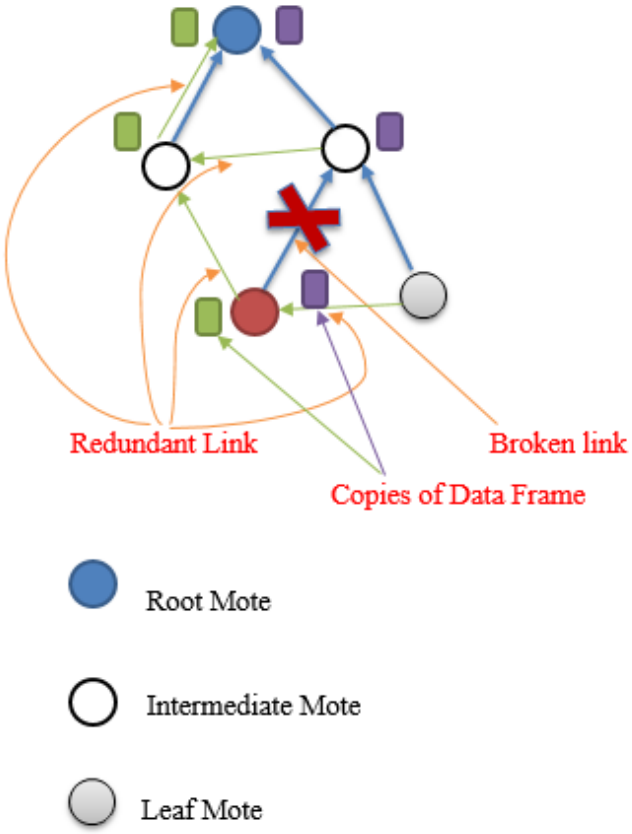


Figure 6.10: Active redendancy paths

Chapter 7

conclusion and future work

This final chapter presents some concluding remarks about the thesis, and makes suggestions related to possible improvements and future work.

The main topic of this thesis is the study of the RPL Routing protocol, with a particular attention devoted to the performance analysis of the quality of communication. The metrics used in the experimental campaign are the Round Trip Time (RTT) and the Packet Loss Percentage. These metrics were analyzed on a multitude of experimental conditions, aimed at checking a number of possible contexts of interest. RPL is the emerging routing standard for low-power and lossy networks (LLN), and was designed by the IETF ROLL Working Group to take into account the unique routing challenges posed by LLNs. RPL was designed in order to standardize and to incorporate the various independent and non-interoperable efforts, with the purpose of creating a routing protocol that would suit a wide variety of LLN scenarios.

In particular, RPL is a routing protocol which operates on top of the IEEE 802.15.4 standard, and it is designed for Low Power and Lossy Wireless Networks such as WSNs. Hence, RPL is optimized for collection networks with infrequent communication from the collection point to individual motes. RPL organizes a topology as a Destination Oriented Directed Acyclic Graph (DODAG), which is a directed graph whose edges are oriented as to avoid cycles. All paths are thus directed and terminate at the root mote. To construct the topology, RPL uses specific control

messages: DIO, DAO, DAO-ACK, and DIS. These control messages are used to achieve the final routing topology. Finally, RPL makes use of an Objective Function (OF) to estimate the distance between a node and the root mote. The outcome of the OF is used to build the appropriate network topology. The RPL protocol is used also in OpenWSN based motes, and it can be applied to different applications, including industrial applications that are characterized by stringent requirements in terms of timeliness and reliability.

Experimental results highlight that, when motes are placed far from the root mote, the RTT increases, casing a consistent delay in data packet transmission. As a result, latencies in the network are too high to be used in a number of application contexts, including industrial applications. The number of hops in the network, as well as the traffic, have a direct influence on latencies. In fact, the performance of the network decreases as the number of hops in the routing topology increases. The quality of a link in terms of Packet Delivery Ratio (PDR) plays an important role to improve the performance of a network. A poor link quality has a really negative impact in the overall performance of the network. Because of this, it can cause increases in packet loss and consequently it has a negative effect on both timeliness and reliability. As a recommendation, to cope with this problem, we proposed two methods, namely, backup link route and active redundancy path.

As future work, the RPL routing protocol should be analyzed also in large scale networks, and the performance should be measured with real devices and in more complex scenarios. In any case, results obtained in this thesis are a useful reference for network engineers. They provide a comprehensive analysis of the performance, in terms of latency and packet loss of a 6TiSCH network based on OpenWSN. Performance was obtained for different network configurations and experimental conditions. In addition, this is a good starting point for researches on more effective routing solutions for IIoT, but also for other application contexts.

Bibliography

[1] Daniel Minoli, *Building the Internet of Things with IPv6 and MIPv6. The Evolving World of M2M Communication*, John Wiley and Sons, 2013.

[2] J. Hui, Jp. Vasseur, Cisco Systems, *The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams*, IETF, RFC 6553, March 2012.

[3] *RPL*, accessed May 2018, [online], Available: www.openwsn.atlassian.net/RPL.

[4] T. winter, *RPL: IPv6 routing protocol for low-power and lossy networks*, IETF, RFC 6550, March 2012.

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

[6] J. Hui and Jp. Vasseur, *The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams*, IETF, RFC 6553, March 2012.

[7] Jp. Vasseur et al., *Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks*, IETF, RFC 6551, March 2012.

[8] K. Pister et al., *Industrial Routing Requirements in Low-Power and Lossy Networks*, Network Working Group, RFC5673, October 2009.

[9] J. Martocci et al., *Building Automation Routing Requirements in Low-Power and Lossy Networks*, IETF, RFC 5867, June 2010.

[10] A. Brandt et al., *Home Automation Routing Requirements in Low-Power and Lossy Networks*, IETF, RFC 5826, April 2010.

[11] Dohler et al., *Routing Requirements for Urban Low-Power and Lossy Networks,* Network Working Group, RFC5548, May 2009.

[12] P. Levis et al., *The Trickle Algorithm,* IETF, RFC 6206, March 2011.

[13] A. Conta et al., *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) specification*, Ntwork Working Group, RFC 4443, March 2006.

[14] *IEEE 802.15.4 Standard-2003, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LRWPANs)*, IEEE-SA Standards Board, 2003.

[15] K. Sohraby et al., *Wireless Sensor Networks: Technology, Protocols, and Applications*, John Wiley and Sons, Inc., 2007.

[16] Bachir, A., Dohler, M., Watteyne, T., Leung, K., *MAC Essentials for Wireless Sensor Networks. Communications Surveys and Tutorials*, IEEE. Vol.12, issue 2: p. 222-248, 2010.

[17] Gunn, M., Simon, G., Koo, M., *A comparative study of medium access control protocols for wireless sensor networks.*, Department of Mathematics and Computer Science, University of San Diego: San Diego, USA. p. 695-703, 2009.

[18] Akyildiz, I.F., Melodia, T., Chowdhury, K., *A survey on wireless multimedia sensor networks*. Computer Networks, vol.51, issue4: p. 921-960, 2007.

[19] Yick, J., Biswanath, M., Ghosal, D., *Wireless Sensor Network Survey*, Computer Networks, vol.52, issue 12: p.2292-2330, 2008.

[20] Nourhene Maalel, *Reliability in wireless sensor networks. Other.*, Universite de Technologie de Compiegne, 2014.

[21] M.H. Anisi, A.H. Abdullah, and S.A. Razak,*EnergyEfficient Data Collection in Wireless Sensor Networks*, Wireless Sensor Networks, vol. 3, 2011.

[22] S. Vaidyanathan and M. Vaidyanathan, *Wireless Sensor Networks- Issues and Challenges*, Information Systems: Behavioral and Social Methods eJournal, 2011.

[23] Sukhwinder Sharma et al,*Issues and Challenges in Wireless Sensor Networks*, International Conference on Machine Intelligence Research and Advancement, 2013.

[24] T. Zia and A. Zomaya, *Security Issues in Wireless Sensor Networks*, Proc. International Conf. Systems and Networks Communication (ICSNCÓ6), Nov. 2006.

[25] *Contiki OS*, accessed July,2018 [online], Available: http://www.contiki-os.org/

[26] *RIOT OS*, accessed July,2018 [online], Available: http://riot-os.org/

[27] G. Montenegro et al., *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, Network Working Group, RFC4944, September 2007.

[28] T. Kivinen and P. Kinney, *IEEE 802.15.4 Information Element for the IETF*, IETF, RFC8137, May 2007.

[29] Sahar Ben yaala, Ridha Bouallegue,*On MAC Layer Protocol towards internet of things: From IEEE802.15.4 to IEEE802.15.4e*, 2016.

[30] T. Watteyne et al, *Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement*, IETF, RFC7554, May 2015.

[31] N. Kushalnagar et al, *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*, Network Working Group, RFC4919, August 2007.

[32] T.watteyne et al., *OpenWSN: a standards-based low-power wireless development environment*, Jhon wiley and sons, Ltd, 2012.

[33] Z. Shelby et al., *The Constrained Application Protocol (CoAP)*, IETF, RFC 7252, June 2014.

[34] *Wireless Sensor Networks*, accessed May,2018 [online], Available: www.wikipedia.org/Wireless Sensor Networks.

[35] O. Iova, P. Picco, T. Istomin and C. Kiraly, *RPL: The Routing Standard for the Internet of Things... Or Is It?,* in IEEE Communications Magazine, vol. 54, no. 12, pp. 16-22, December 2016.

[36] Available: [online] www.openmote.com/

[37] Available: [online] www.researchgate.net/figure/WSN-Architecture

[38] Available: [online] http://sensor-and-networks.blogspot.com

[39] Available: [online] https://wirelessmeshsensornetworks.wordpress.com

[40] Available: [online] http://www.elprocus.com/

[41] Available: [online] http://www.microcontrollerslab.com

[42] Available: [online] http://www.scientechworld.com

[43] S.R. Jino Ramson, D. Jackuline Moni, *Applications of wireless sensor networks - A survey,* Proceedings of IEEE International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology, November 2017.

[44] Available: [online] https://en.wikipedia.org/wiki/TinyOS

[45] Raj Jain, *Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements A Survey of Wireless Sensor Network Simulation Tools,* Wiley Computer Publishing, John Wiley and Sons, Inc, 1991.

[46] *Discrete event simulation,* Available: www.wikipedia.org, Description: an introduction of discrete event simulation.

[47] Muhammad Imran, Abas Md Said, Halabi Hasbullah, *A Survey of Simulators, Emulators and Testbeds for Wireless Sensor Networks,* Information Techonology(ITSim), 2010 International Symposium in, June 2010, pp. 897-902.

[48] Available: [online] https://www.nsnam.org/

[49] *Omnet++,* URL: http://www.omnetpp.org/home/what-is-omnet, Description: a webpage introduced Omnet++.

[50] *Omnet++,* URL: http://en.wikipedia.org/wiki/Omnet%2B%2B, Description: an introduction of Omnet++ in wiki webpage.

[51] Xiyuan Liu et al., *Performance Analysis of Routing Protocol for Low Power and Lossy Networks (RPL) in Large Scale Networks ,* IEEE Internet of Tings Journal, vol. 4, No. 6, December 2017.

[52] *COOJA simulator,* https://github.com/contiki-os/contiki/wiki/, Description: An-Introduction-to-Cooja.

[53] *OpenSim emulator,* https://openwsn.atlassian.net, Description: OpenSim.

[54] E. Kim et. al. , *Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, IETF, RFC 6568, April 2012.