ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη ενός ασύρματου δικτύου αισθητήρων χρησιμοποιώντας το Zephyr RTOS

Διπλωματική Εργασία

Αντώνιος Σιούτας

Επιβλέπων καθηγητής:	Σπύρος Λάλης
Συνεπιβλέποντες καθηγητές:	Νικόλαος Μπέλλας
	Αθανάσιος Κοράκης

Βόλος 2020

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Development of a wireless sensor network using Zephyr RTOS

Diploma Thesis

Antonios Sioutas

Supervisor: Spyros Lalis

Co-supervisors: Nikolaos Mpellas
Athanasios Korakis

Volos 2020

Acknowledgments

First, I would like to thank my supervisor Spyros Lalis for his guidance in carrying out this thesis. I would also like to thank my family that always been there for me. Finally, I would like to thank my friends and coworkers for their support.

Υπεύθυνη Δήλωση Περί Ακαδημαϊκής Δεοντολογίας και Πνευματικών Δικαιωμάτων

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο Δηλών
Αντώνιος Σιούτας
29/9/2020

Περίληψη

Τα ασύρματα δίκτυα αισθητήρων αποτελούν ένα σημαντικό κόμματι του διαδικτύου των πραγμάτων και λύνουν το πρόβλημα της παρακολούθησης περιβαλλοντικών συνθηκών χωρίς να χρειάζεται η φυσική παρουσία του διαχειριστή. Αυτή η μέθοδος έχει αρκετά πλεονεκτήματα όπως ευελιξία και χαμηλό κόστος που την καθιστούν ελκυστική σε πολλές εφαρμογές. Καθώς το διαδίκτυο των πραγμάτων εξελίσσεται, αυξάνεται η ζήτηση για ασύρματα δίκτυα αισθητήρων τόσο για βιομηχανικές όσο και για οικιακές εφαρμογές. Για αυτό το λόγο η τεχνολογική πρόοδος στον αντίστοιχο τομέα έχει μεγάλη σημασία. Παρόλο αυτά, υπάρχουν αρκετές προκλήσεις, όπως η αξιοπιστία, ο ρυθμός μετάδοσης δεδομένων και η διάρκεια ζωής, που πρέπει να αντιμετωπιστούν, για να επιτευχθεί η επιθυμητή λειτουργία. Συνήθως κάθε περίπτωση χρειάζεται διαφορετική προσέγγιση, το οποίο κάνει τις εμπορικές λύσεις λιγότερο ελκυστικές λόγω της αύξησης του κόστους και της περιορισμένης ευελιξίας.

Το αντικείμενο αυτής της διπλωματικής εργασίας είναι η ανάπτυξη ενός συστήματος ασύρματων δίκτυών αισθητήρων, χρησιμοποιώντας τεχνολογίες ανοιχτού κώδικα. Η ανάπτυξη έγινε πάνω σε ένα λειτουργικό σύστημα βασισμένο στο Linux. Για την υποστήριξη των συσκευών από το λειτουργικό σύστημα χρειάζεται η ανάπτυξη των αντίστοιχων driver. Η επικοινωνία μεταξύ των συσκευών σχεδιάστηκε από την αρχή. Βασίζεται στο πρότυπο IEEE 802.15.4, πάνω από το οποίο χρησιμοποιείται ένα πρωτόκολλο δρομολόγησης τύπου δέντρου, που δεν απαιτεί τη γνώση των διευθύνσεων των ενδιάμεσων κόμβων για την αποστολή δεδομένων και μειώνει τον αριθμό των αποστολών μέσα στο δίκτυο. Επιπλέον, ένας μηχανισμός εξοικονόμησης ενέργειας μειώνει την κατανάλωση ενέργειας των κόμβων. Το σύστημα υποστηρίζει ένα σύνολο από εντολές που επιτρέπουν στο χρήστη να ελέγχει τη συμπεριφορά των κόμβων καθώς και την εύκολη εγκατάσταση τους. Το σύστημα που αναπτύχθηκε δοκιμάστηκε στο εργαστήριο για να επιβεβαιωθεί ότι είναι αξιόπιστο καθώς επίσης και για τη μέτρηση χαρακτηριστικών του δικτύου όπως ρυθμός μετάδοσης δεδομένων και κατανάλωση ενέργειας.

Abstract

Wireless sensor networks, a key part of the Internet of Things, solve the problem of measuring environmental conditions, without requiring physical presence of the owner. This method has many advantages, like flexibility and low cost, that make it popular to many applications. As the Internet of Thing grows, the demand for WSN in both industrial and home application increases. For this reason, the technological improvement for these networks has big significance. Apart from that, there are many challenges like reliability, high throughput and long battery life, that a wireless sensor network needs to overcome to achieve the desired outcome. Usually each application requires custom implementations to work, that makes the commercial solutions less appealing due to increased cost and limited flexibility.

This thesis describes the development of a system for wireless sensor networks, using open source technologies. As a platform, a Linux based real-time operating system is used to facilitate the system implementation. The development of hardware drivers is needed to integrate the hardware to the OS. The communication between devices is designed by scratch. It is based on the IEEE 802.15.4 standard, over which a tree-based routing protocol is used to make the network address free and minimize the network traffic. A synchronous sleeping mechanism is highly increasing the lifetime of each device to prevent any power related issues. The system supports a set of commands that enable the user to control the behavior of the network and permits easy deployments. The performance of the system is tested in the lab to verify the reliability and measure important network characteristics like throughput and energy consumption.

Table of Contents

Acknowledgments ... iii

Υπεύθυνη Δήλωση Περί Ακαδημαϊκής Δεοντολογίας και Πνευματικών Δικαιωμάτων ... iv

Περίληψη ... v

Abstract ... vi

Table of Contents ... vii

Abbreviations ... ix

Chapter 1 Introduction ... 1

Chapter 2 Background ... 3

2.1 IEEE 802.15.4 ... 3

2.1.1 Devices ... 3

2.1.2 Network Topology ... 3

2.1.3 Stack Architecture ... 4

2.1.4 Data Transfers ... 5

2.1.5 Frame Structure ... 7

2.2 CSMA-CA Algorithm ... 8

2.3 Zephyr OS ... 10

2.3.1 Threads and Scheduling ... 10

2.3.2 Data Passing ... 12

2.3.3 Timing ... 13

2.3.4 Logging ... 13

2.3.5 Network Stack ... 14

Chapter 3 User Model ... 16

3.1 Overview ... 16

3.2 Commands ... 16

3.3 Network deployment, node addition/removal, node grouping ... 17

Chapter 4 System Design 18
4.1 Routing Algorithm 18
4.2 Routing Gradient ETX 19
4.3 Frame Types 21
4.3.1 Data Frame 21
4.3.2 Network Frame 23
4.3.3 Command Frame 24
4.4 Command Subsystem 26
4.5 Duty Cycle 27
Chapter 5 System Implementation 29
5.1 Hardware 29
5.2 Software Architecture 31
5.2.1 MRF24J40 Driver 32
5.2.2 Network Layer 33
5.2.3 Protocol Layer 34
Chapter 6 Experiments and Evaluation 38
6.1 Power Consumption 38
6.2 Topology 39
6.2.1 Star Topology 39
6.2.2 Chain topology 42
6.3 Throughput 43
6.4 Various other tests 45
Chapter 7 Conclusion and Future Work 46
7.1 Summary 46
7.2 Possible extensions 46
References 48

Abbreviations

API	Application Program Interface
BE	Back-Off Exponent
CCA	Clear Channel Assessment
CRC	Cyclic Redundancy Check
CS	Carrier Sense
CSMA-CA	Carrier-Sense Multiple Access with Collision Avoidance
CTP	Collection Tree Protocol
ED	Energy Detection
ETX	Expected Transmission
FCS	Frame Check Sequence
FFD	Full Function Device
FIFO	First in First Out
GTS	Guaranteed Timed Slot
IEEE	Institute of Electrical and Electronics Engineers
LIFO	Last in First Out
LQI	Link Quality Indication
LQI	Link Quality Indicator
MAC	Medium Access Control
MCU	Microcontroller Unit
MPDU	Mac Protocol Data Units
MQTT	MQ Telemetry Transport
MSDU	Mac Service Data Unit
NB	Number of Back-Off Periods
OSI	Open System Interconnection
OTG	On the Go
PAN	Personal Area Network
PCB	Printed Circuit Board
PHR	PHY Header
PHY	Physical Layer
PPDU	PHY Protocol Data Units
PRR	Packet Reception Rate
PSDU	Physical Service Data Unit
RF	Radio Frequency
RFD	Reduced Function Device
RSSI	Received Signal Strength Indicator

SHR	Synchronization Header
SPI	Serial Peripheral Interphase
SRAM	Static Random-Access Memory
TCP	Transmission Control Protocol
THL	Time Has Lived
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Chapter 1 Introduction

Internet of Things is an emerging topic of technical, social, and economic significance that promises to transform our lives. It is an ecosystem of connected devices accessible through the Internet. Computers, smart devices like smartphones or TVs, wearables and sensors are examples of IoT devices. Especially sensors have a wide range of applications in industry and domestic use such as health care, security, agriculture, automation, industrial production and smart homes.

Wireless sensor networks refer to a group of distributed sensor nodes (nodes) dedicated to monitoring environmental conditions such as temperature and pressure, machinery maintenance, structural health for buildings and thread detection. The nodes are composed of a sensor element, responsible for converting the measured conditions to electrical impulses, a micro-controller, which is the brain of the node, and a radio transceiver capable of transmitting and receiving messages. The radio typically has a limited range of a few meters or tens of meters, depending on the technology and antennas used.

Nodes cooperatively communicate with a device called gateway that has the ability to transfer data to the Internet. Usually the target is to cover a large area, thus increasing the number of nodes in the network and the distance between them. Most times there are accessibility issues not allowing a person to attend and solve problems on site, such as missing nodes or nodes with exhausted battery. For this reason, the network must be capable of self-healing from communication failures. Also, it must be able to operate in low-power mode for an extensive period.

In order to communicate, all the devices in the network use a specific protocol. A network protocol is a set of rules that all the devices follow to communicate. Obviously, the network protocol plays a central role in the system’s ability to operate successfully even when there are individual node failures. Although there are many commercial solutions, most of them have limitations because of closed source policy that does not facilitate any optimizations or extensions.

In this thesis, a wireless sensor network is developed, using an open source operating system. A tree-based routing protocol is used to keep network maintenance related traffic as low as possible. To minimize energy consumption, nodes operate in two states, normal and low-power mode. The network can recover from communication and node failures. Also, the system can be easily deployed with minimum human intervention.

The rest of the thesis is organized as follows. Chapter 2 summarizes the technologies used to implement the system. Chapter 3 introduces the functional objectives. Chapters 4 and 5 describe the design and implementation of the system. Chapter 6 presents the experiments that were performed to evaluate the implemented system prototype. Finally, Chapter 7 summarizes the work and proposes possible extensions.

Chapter 2 Background

2.1 IEEE 802.15.4

IEEE stands for the Institute of Electrical and Electronics Engineers, a non-profit organization, founded in 1884 for the purpose of consolidating ideas dealing with electrotechnology [1]. IEEE produces over 30% of the world's literature in the electrical and electronic engineering and computer science fields, publishing well over 100 peer-reviewed journals and magazines [2].

IEEE 802.15.4 is a technical standard which defines the protocol and interconnection of devices via radio communication [3] [4]. This standard defines the Physical Layer (PHY) and the Medium Access sub-layer (MAC). Those two are the lower network layers of a Wireless Personal Area Network (WPAN).

2.1.1 Devices

IEEE 802.15.4 systems include so-called devices. A device taking part in a WPAN can operate as a full function device (FFD) or as a reduced function device (RFD). In turn, FFD devices can operate either as a coordinator or as a normal device. At least one FFD in the network should be a coordinator.

RFD devices communicate only with FFD devices, while FFD devices have no restriction. The basis for this distinction is that RFD intend for applications that are simple, while keeping low cost and complexity.

2.1.2 Network Topology

The standard supports two types of network topologies, the star topology and the peer-to-peer topology. A WPAN operates in a star topology must include a central node working as a PAN coordinator. The remaining nodes communicate directly with this central node as shown in Figure 2-1. Most of the times the central node is mains powered while the devices are battery powered. This topology can be useful for applications like home automation or PC peripherals.

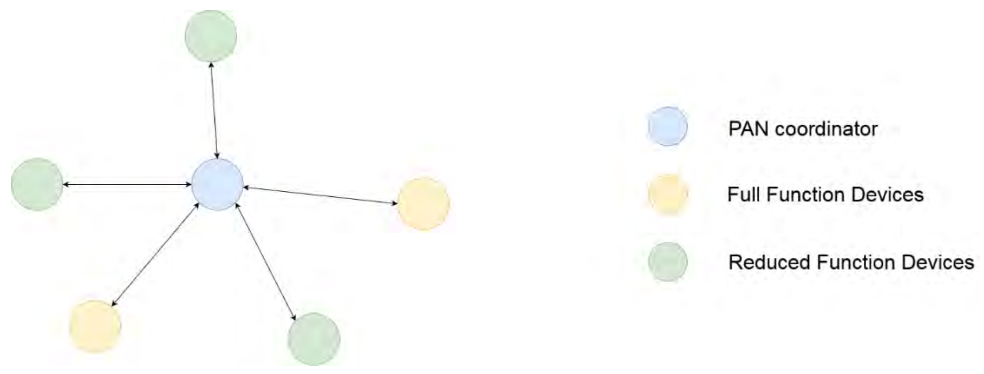


Figure 2-1: Star Topology.

In the peer-to-peer topology, nodes communicate with each other as long as they are in transmission range, shown in Figure 2-2. This topology includes a coordinator and both FFD and RFD devices. The peer-to-peer topology is the basis of more complex network structures like mesh networks.

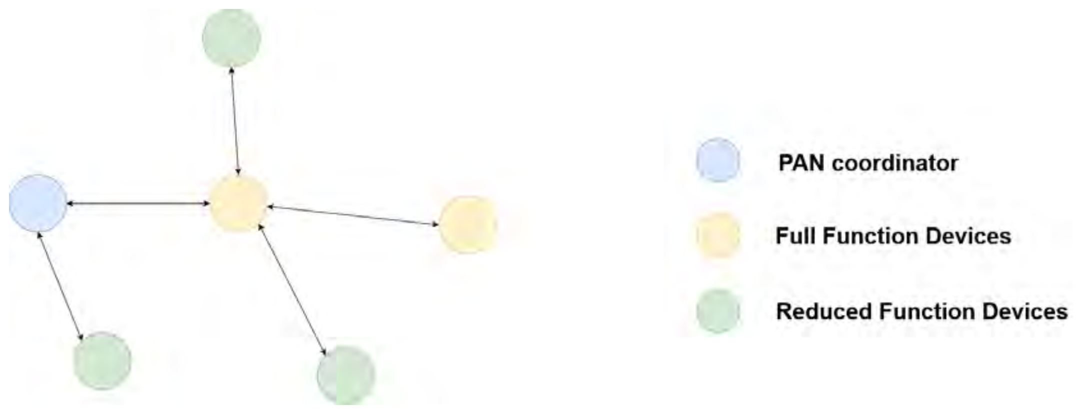


Figure 2-2: Peer-to-peer Topology.

In both network topologies, every device has a unique 64-bit identification number that is used for direct communication. This long address can be exchanged for a shorter 16bit address, allocated by the Coordinator.

In each independent PAN, a unique identification number is chosen by the Coordinator and is used to communicate within the network using short addresses or with another PAN.

2.1.3 Stack Architecture

The IEEE 802.15.4 stack architecture defined by several blocks called layers, as shown in Figure 2-3. The standard describes the two bottom layers (PHY and MAC).

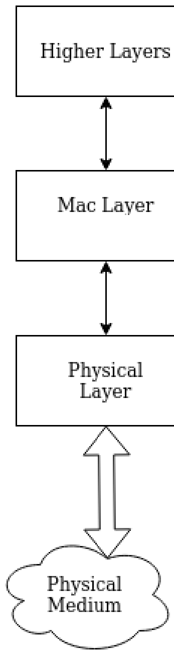


Figure 2-3: IEEE 802.15.4 stack.

The PHY layer contains the low-level control mechanisms of the radio transceiver. Each country has a license-free band available to use. There is also a worldwide license free band at 2400–2483.5MHz. The radio device must comply with one of those frequencies.

The PHY layer is responsible for transmission and reception of PHY protocol data units (PPDU), activation and deactivation of the radio transceiver, channel selection, energy detection (ED) within the current channel, clear channel assessment (CCA) and link quality indication (LQI) of received data units.

The MAC layer is responsible for the transmission and reception of MAC protocol data units (MPDU) across the PHY data service, beacon management, guaranteed timed slot (GTS) management, channel access, frame validation, frame acknowledgment and security mechanisms.

2.1.4 Data Transfers

Communication between devices follows one of three types of data transfers: (i) device to coordinator, (ii) coordinator to device and, in peer-to-peer topology, (iii) data transfer between two normal devices. The procedure for each data transfer depends on whether the network supports beacon transmissions.

If beacon transmission is enabled, a device first waits to hear a beacon and then sends a data request using the slotted CSMA-CA algorithm. The request should be acknowledged by the coordinator and then the data are transmitted using the CSMA-CA algorithm. If beacons are not enabled, devices send data using an unslotted CSMA-CA algorithm. Figure 2-4 shows both cases.

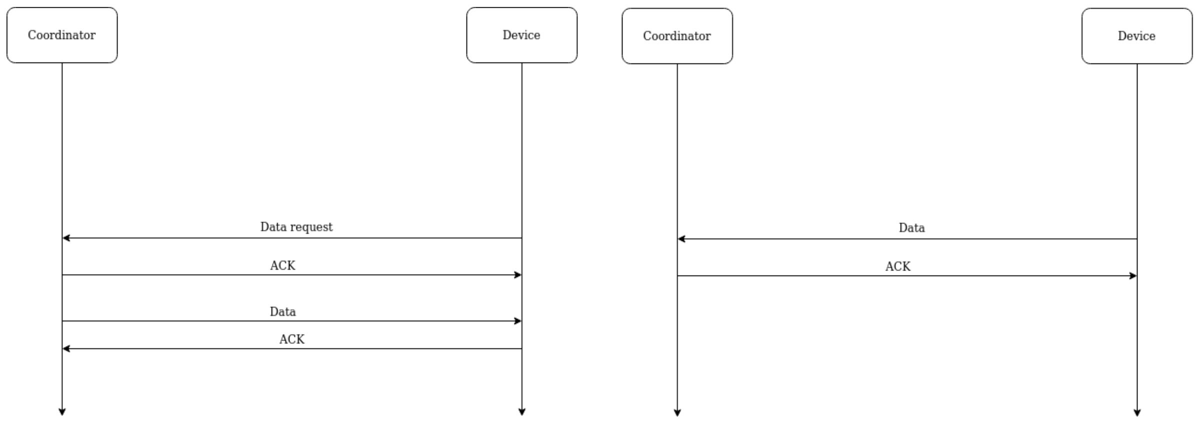


Figure 2-4: Transmission from a device to the coordinator with beacons enabled (left, disabled (right).

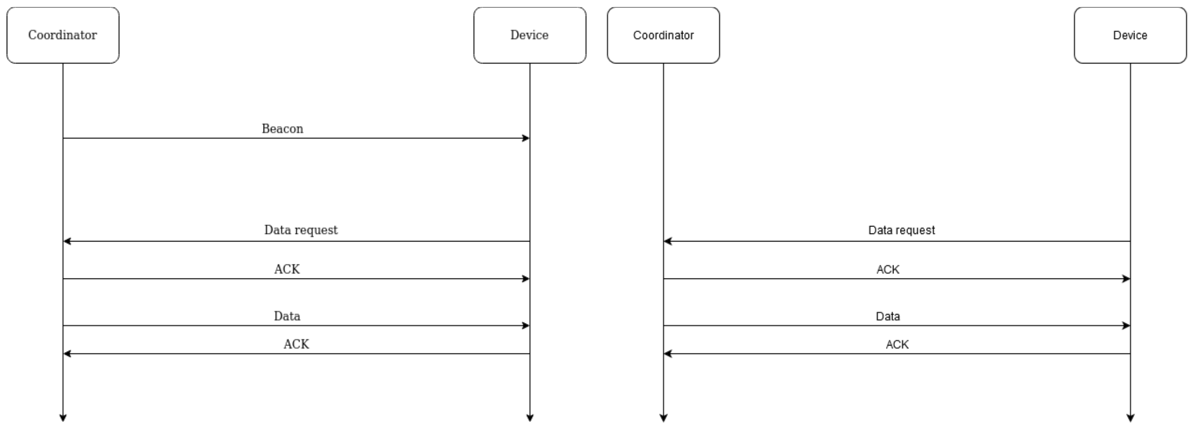


Figure 2-5: Transmission from the coordinator to a device with beacons enabled (left, disabled (right).

If the coordinator wants to send data to a device in a beacon-enabled network, it first sends a beacon, indicating that data is pending. The device then sends a data request using the slotted CSMA-CA algorithm. The coordinator sends an acknowledgement, followed by the data, using the slotted CSMA-CA algorithm. The device finally sends an acknowledgement. If beacons are not enabled, the procedure is the same without the beacon message, and the slotted CSMA-CA algorithm is replaced by the unslotted CSMA-CA algorithm. Figure 2-5 shows both cases.

2.1.5 Frame Structure

The IEEE 802.15.4 standard defines four frame types: (i) the beacon frame used by the coordinator, (ii) the data frame, (iii) the acknowledgement frame, used for confirmation of frame reception and (iv) the MAC command frame.

In each case, the MAC layer forwards the data unit to the physical layer. The latter forms the PHY Protocol Data Unit (PPDU) containing the synchronization header (SHR), the PHY header (PHR) and the Physical Service Data Unit (PSDU). The SHR contains the preamble sequence, used for symbol synchronization, the start of frame delimiter indicating the start of the physical header, the frame length of the PSDU and the PSDU containing the Mac Protocol Data Unit (MPDU), shown in Figure 2-6. Figure 2-7 to 2-10 show the MPDU for each frame type.

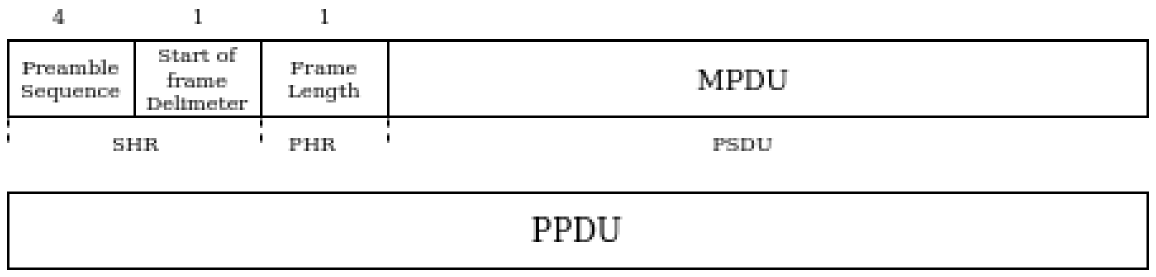


Figure 2-6: The PPDU structure.

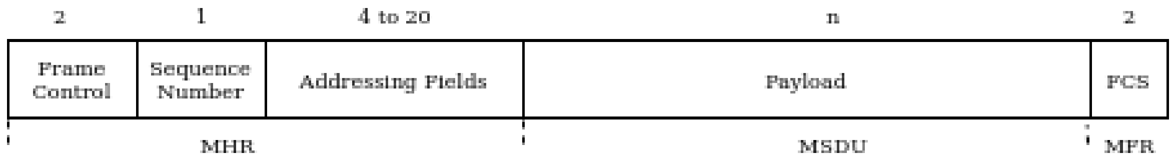


Figure 2-7: The MPDU structure of the data frame.

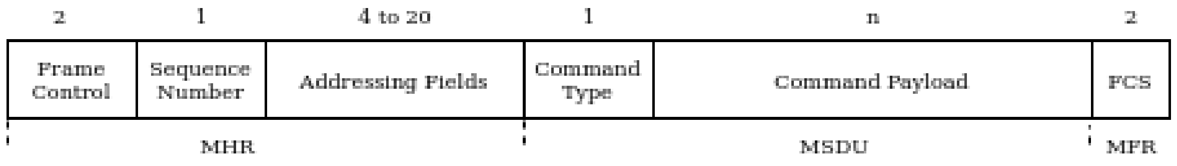


Figure 2-8: The MPDU structure of the command frame.

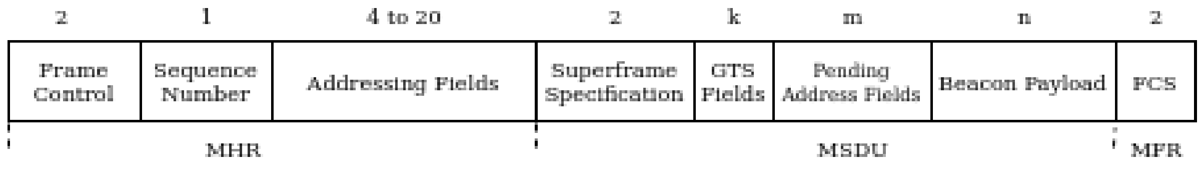


Figure 2-9: The MPDU structure of the beacon frame.

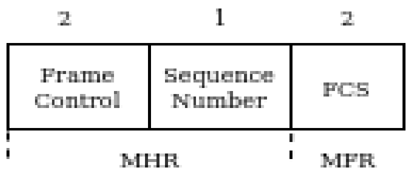


Figure 2-10: The MPDU structure of the acknowledgement frame.

The frame control field provides relevant information about the frame. The sequence number, which increases by one for each new frame, serve as the unique identifier. The addressing fields have the source and destination address, and the frame control determine the length. All the payload fields contain the data of the packet. The Frame Check Sequence (FCS) field is the cyclic redundancy check (CRC), used for error detection. The super frame specification, the pending address and the GTS fields provide information relevant to beacon enabled networks.

2.2 CSMA-CA Algorithm

The Carrier-sense multiple access with collision avoidance algorithm decides if the RF medium is ready to transmit a packet. Prior to transmitting, the node listens if someone else is transmitting. If so, the node waits for a random period before trying again. If the channel is free, the node sends the packet. Figure 2-11 shows the unslotted version of CSMA-CA algorithm. The node maintains two variables, the back-off exponent BE, which means how much time the node has to wait and the number of back-off periods NB that the algorithm was required to wait. NB is initialized to zero. After a delay, the node assesses the channel and if it is not idle, increases NB and tries again.

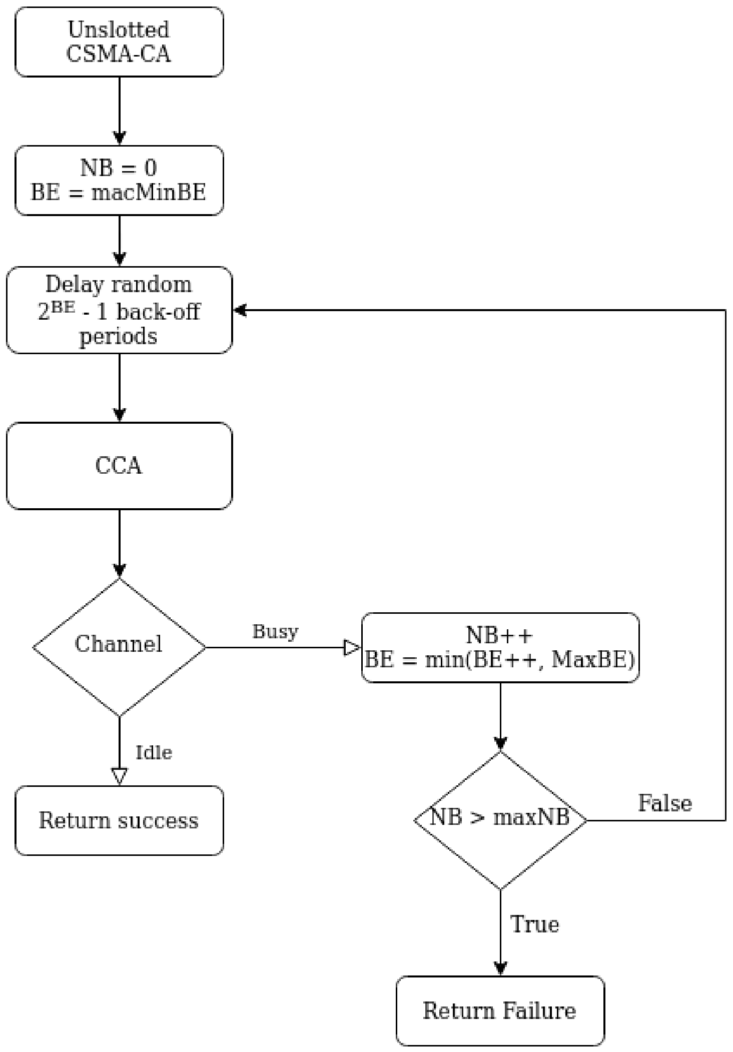


Figure 2-11: CSMA-CA algorithm.

There are three techniques, used by the CSMA-CA algorithm, to assess if the channel is clear [4], [5]. The first, called energy detection (ED), reports that the medium is busy if there is a signal above a threshold. The second, called carrier sense (CS), considers that the medium is busy when it detects a signal with the modulation and spread characteristics of IEEE 802.15.4. This signal level may be below the ED threshold. The third technique is a combination of the previous ones and reports that the medium is busy only upon detecting a signal above the ED threshold with the modulation and spread characteristics of IEEE 802.15.4.

The energy detection refers to the ability of the receiver to detect energy level present on the current channel based on the noise floor, ambient energy and interference sources. The threshold should be at most 10 dB greater than the specified receiver sensitivity. The carrier sense refers to the ability of the receiver to detect the IEEE 802.15.4 signal preamble.

2.3 Zephyr OS

Zephyr OS [6] [8] is a real time operating system based on Linux, suitable for embedded devices. The neutral policy and small memory footprint of Zephyr make it perfect for IoT applications. It is an open source project, written in C language and support many boards and peripherals. The source code is available in GitHub [7] along with a command-line tool called West, which provides a multiple repository management system.

2.3.1 Threads and Scheduling

The Zephyr kernel services support two types of thread, cooperative and preemptible threads. Each thread can be in ready or unready state and has a priority level. The kernel supports a virtually unlimited number of thread priority levels, as shown in Figure 2-12. A thread with negative priority level is cooperative. Once it becomes the current thread, it remains the current thread until it explicitly makes an action through which control is given to another thread, as shown in Figure 2-13. Positive priority level means that the thread is preemptible and execution can be interrupted by other ready preemptible threads with higher priority or cooperative threads, shown in Figure 2-14.

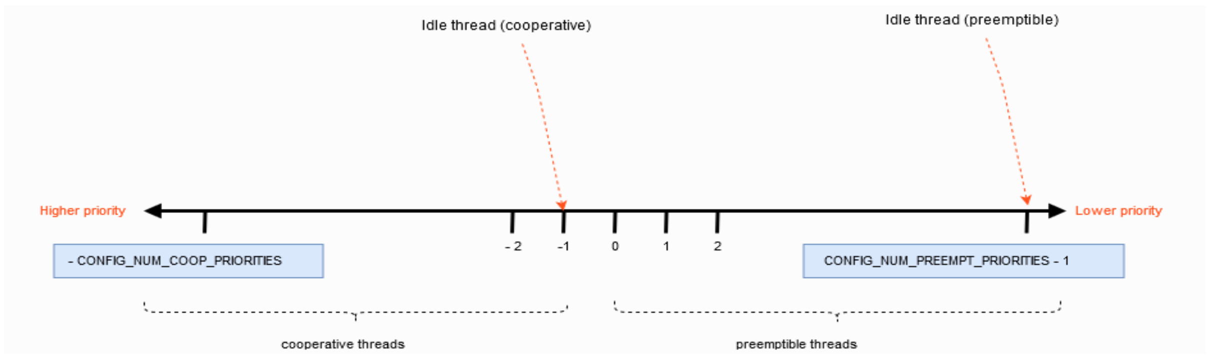

Idle thread (cooperative)
Idle thread (preemptible)
Higher priority
Lower priority
- CONFIG_NUM_COOP_PRIORITIES
-2
-1
0
1
2
CONFIG_NUM_PREEMPT_PRIORITIES - 1
cooperative threads
preemptible threads

Figure 2-12: Priority level of threads.

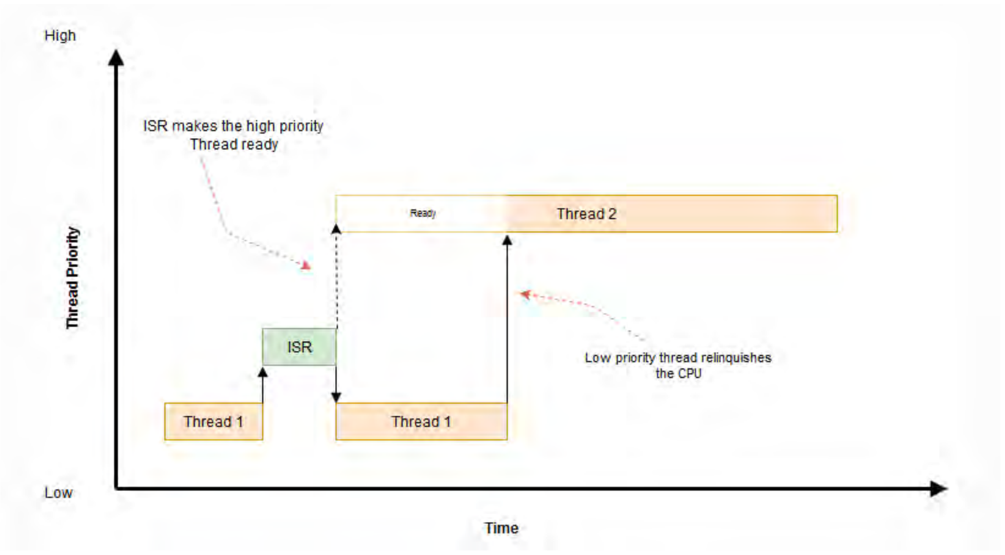

High
ISR makes the high priority Thread ready
Ready
Thread 2
Thread Priority
ISR
Low priority thread relinquishes the CPU
Thread 1
Thread 1
Low
Time

Figure 2-13: Cooperative thread scheduling.

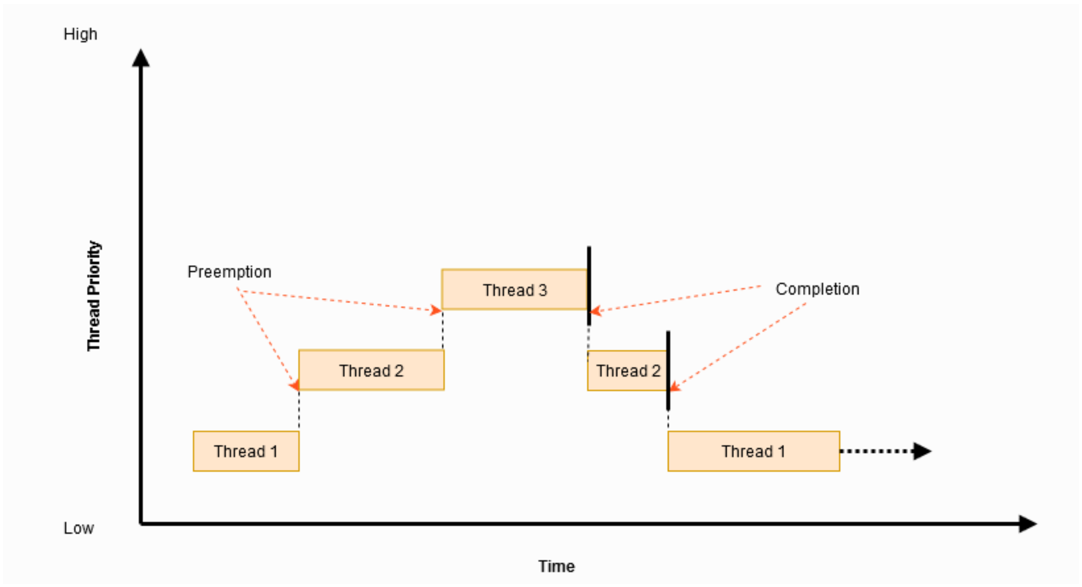


Figure 2-14: Preemptible thread scheduling.

The kernel also supports so-called work queue threads, dedicated to the processing of specific work items added to a work queue. Work queue threads are typically used by interrupt handlers to offload non-urgent processing.

The kernel includes semaphores and mutexes, used for thread synchronization. A thread can “give” a semaphore in order to allow other threads to enter a critical region, and “take” a semaphore in order to enter a critical region, as shown in Figure 2-15. Semaphores maintain a counter that increases each time when the semaphore it is given and decreases each time it is taken. When it is zero, the semaphore does not allow any threads to enter.

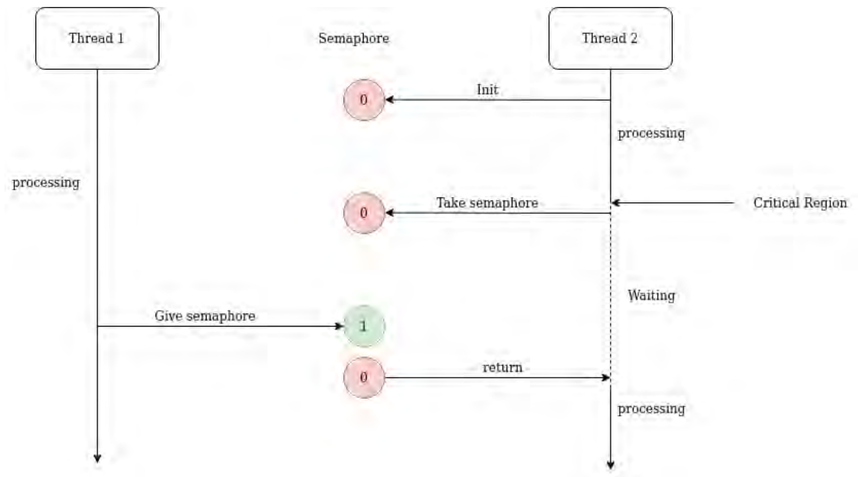


Figure 2-15: Thread synchronization using a semaphore.

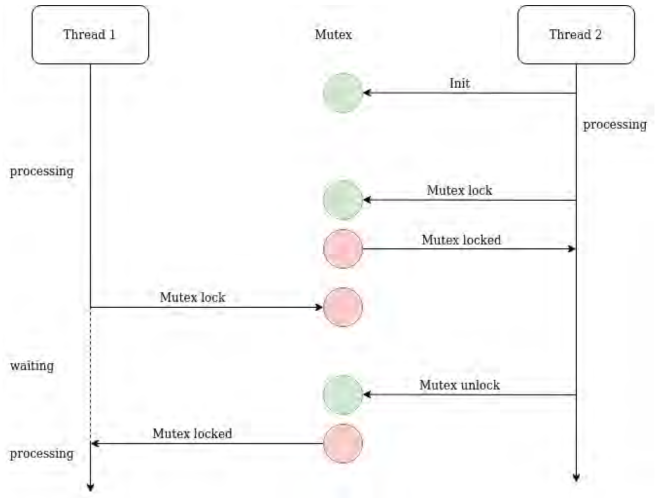


Figure 2-16: Thread synchronization using a mutex.

Mutexes provide exclusive access to a critical region. A thread can lock a mutex, if unlocked, to enter and then unlock it, shown in Figure 2-16. Else, if the mutex is already locked, the thread is blocked until the mutex is unlocked by another thread.

2.3.2 Data Passing

The operating system supports several data structures to pass data between threads. The FIFO object implements the traditional first-in first-out queue. Data flows from one direction to another, maintaining their initial order. The LIFO object supports data passing in the reverse order. There are also pipes that allow threads to send byte streams either synchronously or asynchronously. The ring buffer object is a circular buffer where data stored in FIFO order. It is used for sending small byte streams or raw data. Figure 2-17 presents all the above structures.

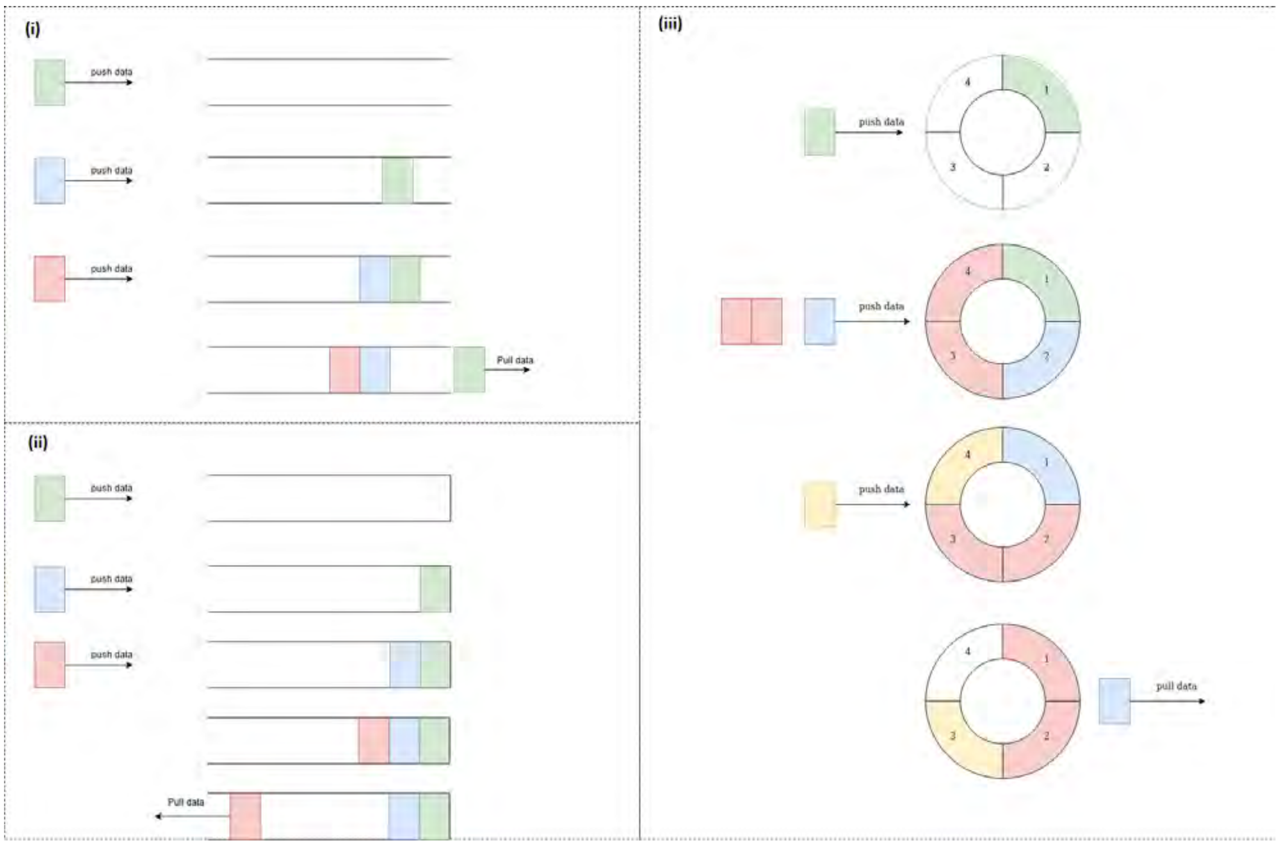


Figure 2-17: Push and pull data from (a) FIFO, (b) LIFO and (c) ring buffer.

2.3.3 Timing

Timer is a kernel object used to measure time based on the kernel's clock. When a timer reaches the specific limit, the handler function takes control by interrupting the execution of the current thread, similar to interrupt functions. Timers can measure time in a range of milliseconds and used in cases that high precision is a requirement.

2.3.4 Logging

The operating system provides a logging interface. Log messages pass from a front end and are processed by the back end. There are four severity levels of logs, information, warning, error and debugging. The default configuration has the UART as a back end and output contains the time, the relevant string along with the thread and function that recorded it.

2.3.5 Network Stack

The network stack comprises different layers, and each one offers services to the others as shown in Figure 2-18.

The Network Application is the top layer. It can access the application-level protocol libraries to send or receive data. This layer can also access the Network Management API to configure network parameters.

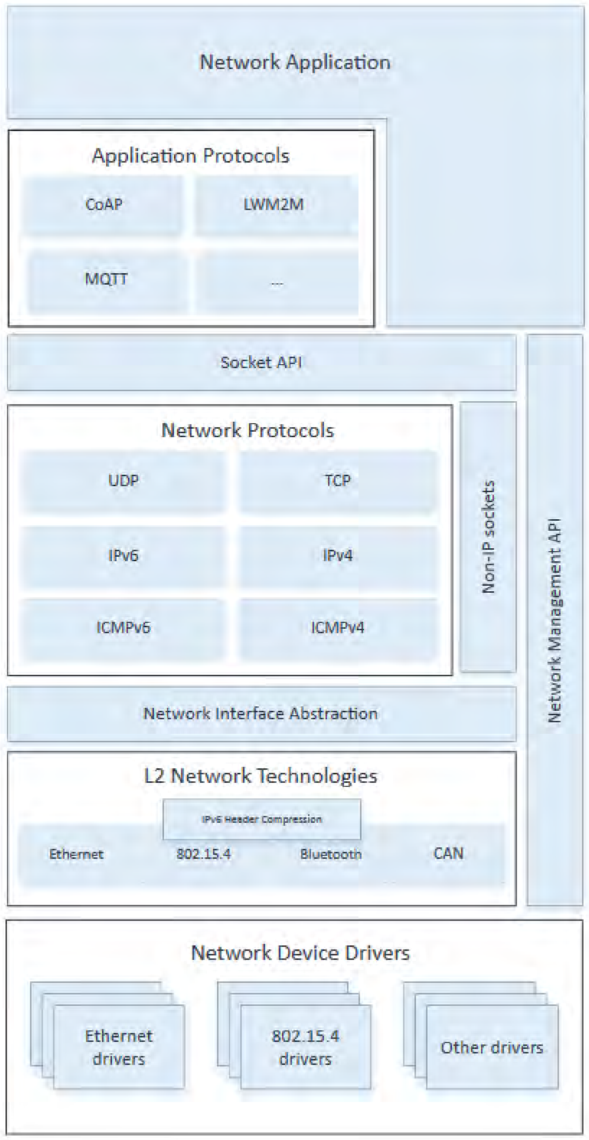


Figure 2-18: Network Stack.

The Network Protocol layer implements different popular protocols. It is divided to two sub-layers: Application-level protocols like MQTT, and the Core-level protocols like TCP and UDP.

The L2 Network Technologies layer hides the complexity of different networking technologies from the upper layers. It provides a generic API that includes basic functions for sending/receiving data, enabling/disabling traffic over the network interface, and retrieving information regarding the capabilities of the hardware like multi-cast or promiscuous mode. Includes protocols like Ethernet, Bluetooth and IEEE 802.15.4.

The Network Device Drivers is the bottom layer. It includes the driver implementation for the supported hardware and handles the physical data send and receive.

Chapter 3 User Model

3.1 Overview

The user can communicate with the coordinator to receive information and configure/control the WSN, using a command-line program. The user sees the real time measurement of each node. Alongside the measurements, the user sees information about the existence of weak links between them. This feature gives the ability to find weak points in the network.

3.2 Commands

The user is able to send commands to the nodes, to change the configuration settings and see relevant information about the links between them. The target can be either a specific node or all the nodes in the network. After the command request, the user gets a response with the requested information or an error message for wrong input.

The complete list of the commands can be seen below:

Command	Description
TX power level	Change the transmission power level of the radio. There are four available options: low, medium, high and highest.
Average Sampling	Change the number of samples the node collects.
Weak link report	The nodes report along with data the existence of “child” nodes with quality below the threshold. The evaluation uses the LQI system of the radio module, when they receive data frames from their child nodes.
Short address	The frames contain an address of 16-bit length or 64-bit length. Changing to 16-bit addresses increases the available data length.
ID	Change the network ID.
Enable/disable routing for different ID	The nodes can forward frames regardless of the Network ID
Enable/disable aggregation for data frames	Reduce the traffic by concatenating frames wherever this is possible
Enable/disable Duty Cycle	Enable the transition to low power mode.
Discover Network	Discover all the nodes in the network and all their links and ETX values

Discover Node	Discover the communication path of a node and their ETX values
Change routing timer	Change the timer to send network frames.
Reset ETX Value	Force a node to reset the ETX value. This may change its parent node.
Clear routing table	Clear the routing table of a node.
SP	Change the sleep period of the network
WT	Change the wake period of the network.
OS	Read the time until the end of the current sleep period.
OW	Read the time until the end of the current wake period.

3.3 Network deployment, node addition/removal, node grouping

To deploy the network, the user, first, has to execute the command-line tool and then has to power on the coordinator and connect him to the PC. Changing the sleep period is optional, but a small sleep period leads to faster synchronization. Afterward, the user should power on the devices. All of them are synchronized by the coordinator or by other synchronized devices and find the optimal communication path. Once the network is ready, the user can change again the sleep period in order to meet the requirements of the particular application.

Adding new nodes is easy, as they can be synchronized by other nodes directly, if they are awake. If the network is configured to operate with a large sleep period, the user should activate a new node near the coordinator (to avoid long synchronization delay and battery depletion). Removing a node affects temporarily the nodes relying on him, but they automatically find alternative paths.

The user may form groups of nodes and control them without affecting all the nodes. This can be achieved by changing the identification number of a few nodes. Then, by sending a single command, the user can change, for example, the sampling rate of the entire group of sensors. Although the nodes with different identification number receive and forward the command, they will not execute it as they do not belong to the group.

For applications that require large data transfers, the user can enable the option to use short address (16bit) to give nodes the largest payload size. For small data transfers, the user can enable packet aggregation to improve the network performance (note that all data packets have the same destination).

Chapter 4 System Design

4.1 Routing Algorithm

The routing algorithm is based on the Collection Tree Protocol [9]. The entire network operates like a tree, with the coordinator being the root of the tree. Each node is connected with a node above and closer to the root, as shown in Figure 4-1. The data flows from the leaves to the root. Nodes generates routes to the root using a routing gradient. The CTP algorithm is address free, meaning that nodes do not send packets to specific destinations, but choose only the next hop.

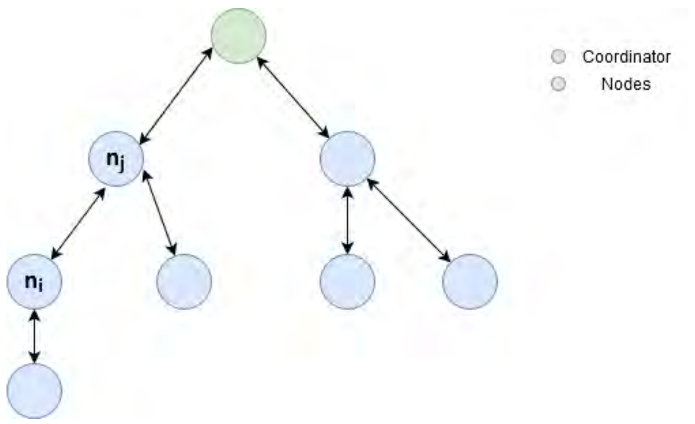


Figure 4-1: Tree based WSN.

The CTP uses expected transmission (ETX) as the routing gradient. The ETX metric is a measure of the quality of the path between two nodes. The ETX value of a path varies from 1, which indicates a perfect transmission, to infinity, which indicates a non-functional path. The root has always ETX value of 0, while the other nodes take the ETX value of their parent node plus the ETX of the link to the parent node. For example, assuming node n_i has node n_j as its parent, as shown in Figure 4-1, the ETX value of n_i is calculated as: $ETX_i = ETX_j + ETX_{link}$. If more than one candidate nodes exist, CTP chooses the node with the lowest ETX value.

As shown in Figure 4-2, the three nodes initially have an invalid ETX value, so all of them send a local broadcast to their neighbors (4-2i). The root node receives the frames for the first two nodes and replies with another local broadcast (4-2ii). The two nodes after the reception of the root's broadcast message, calculate their ETX value and send another broadcast message to inform their neighbors (4-2iii). The last node receives the messages and calculates his ETX value (4-2iv).

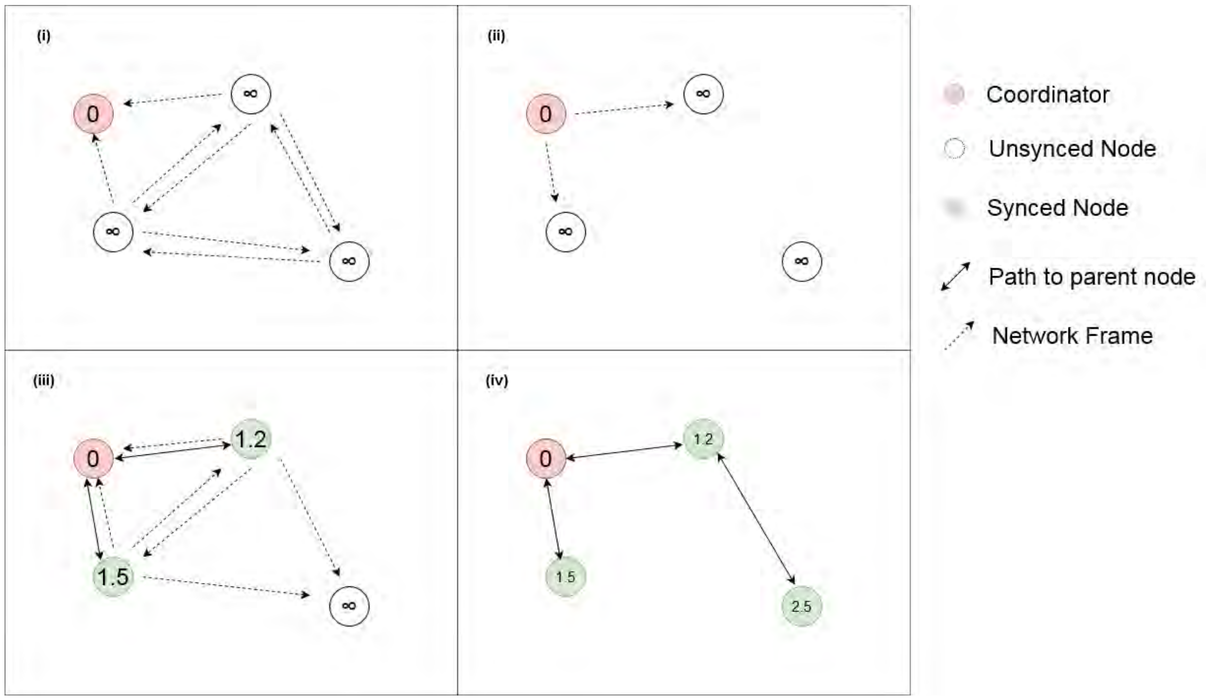


Figure 4-2: The node's procedure to estimate their ETX value.

4.2 Routing Gradient ETX

As mentioned, the node's ETX value is based on the ETX value of its parent plus the estimation of ETX value for the link between them. The ETX link value measures the performance of the link between two nodes. The estimation [9]–[11] is based on the packet reception ratio of the data and network frames and can be summarized in Equations 1 and 2.

$$ETX_{link} = \frac{ETX_{link} + PRR_{ack}}{2} \tag{1}$$

$$ETX_{link} = \frac{ETX_{link} + \frac{1}{PRRrouting}}{2} \tag{2}$$

The ETX value changes when a node forwards data frames or command replies to the parent node, following Equation 1. The PRR_{ack} is the packet reception rate and is measured every five packets following Equation 3. The numerator is the number of transmitted frames and the denominator is the number of the acknowledged frames. This method is based on a method proposed by Woo [12].

$$PRR_{ack} = \frac{\#packet\ send}{\#packet\ ack} \qquad (3)$$

When all the frames are not acknowledged, then the PRR_{ack} is the total number of the fames plus one. Figure 4-3 shows the estimation of PRR_{ack}, while transmitting frames to the parent node. Green squares represent acknowledged frames and red squares not acknowledged frames.

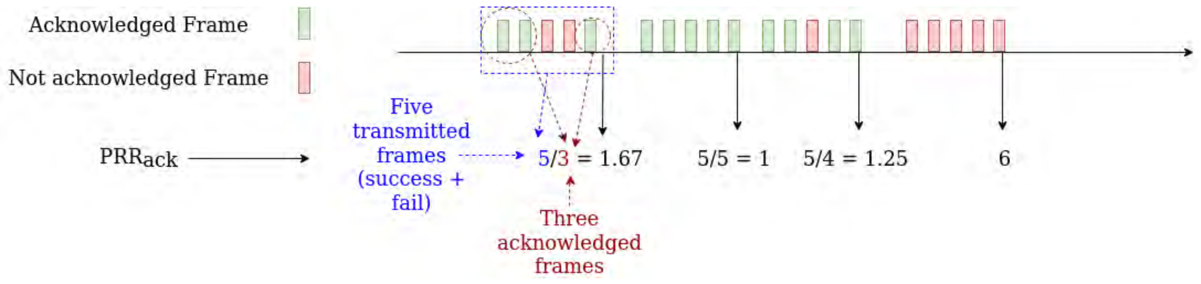


Figure 4-3: PRR_{ack} estimation.

The ETX value changes when a node receives network frames, following Equation 2. The $PRR_{routing}$ is the network packet reception rate and is measured every two received network frames, following Equation 4 proposed by Woo [12]. Nodes keep information about the $PRR_{routing}$, for every node that sent a network frame. The average of the previous value and the number of the received network frames divided by the total number of network frames that the node sent is used to estimate the $PRR_{routing}$. The number of network frames that a node missed can be calculated by keeping the sequence number of the received network frames.

$$PRR_{routing} = \frac{PRR_{routing} + (\frac{\#received\ NF}{\#total\ NF})}{2} \qquad (4)$$

First, we calculate the division $(\#received\ NF\ /\ \#total\ NF\ )$ every two received network frames (from the same node) and then we calculate the $PRR_{routing}$, that is the average with the previous result. Figure 4-4 shows the estimation of ETX value.

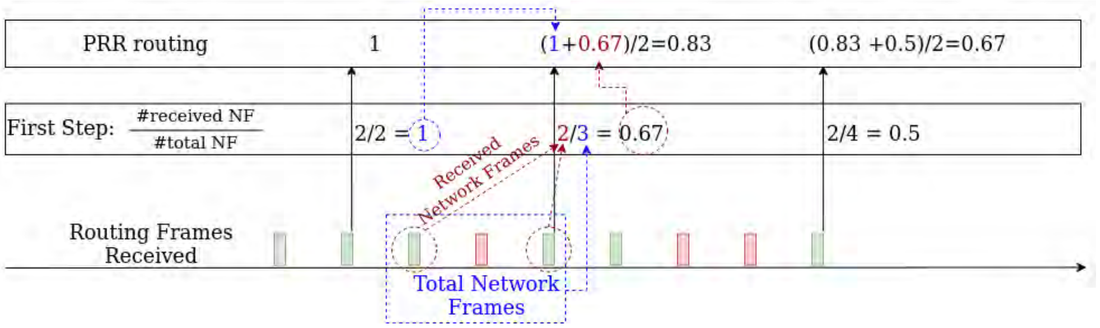


Figure 4-4: $PRR_{routing}$ estimation.

The ETX_{link} is the average of the previous value of ETX_{link} and either the PRR_{ack} or the inverted $PRR_{routing}$ (Equation 1 and 2). Figure 4-5 shows the ETX_{link} estimation.

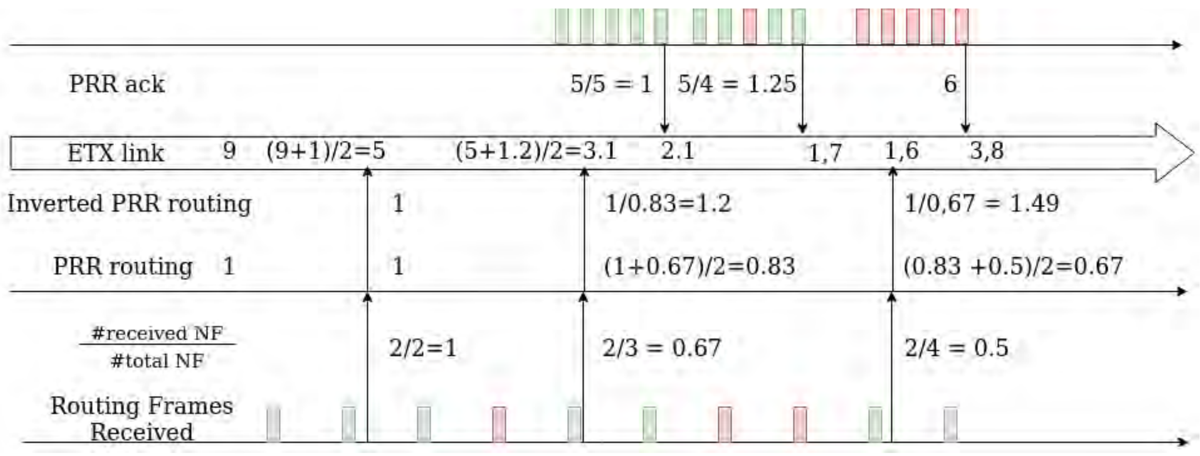


Figure 4-5: ETX link estimation.

4.3 Frame Types

There are 3 types of frames in this protocol, the data frame, the network frame, and the command frame. These are discussed in more detail in the following.

4.3.1 Data Frame

The data frame is used to transfer data to the parent node, using a unicast transmission. When a node receives a data frame, this is forwarded to its parent node. Figure 4-6 shows the structure of the data frame. The first two bytes are the frame control bytes and exist in every frame type.

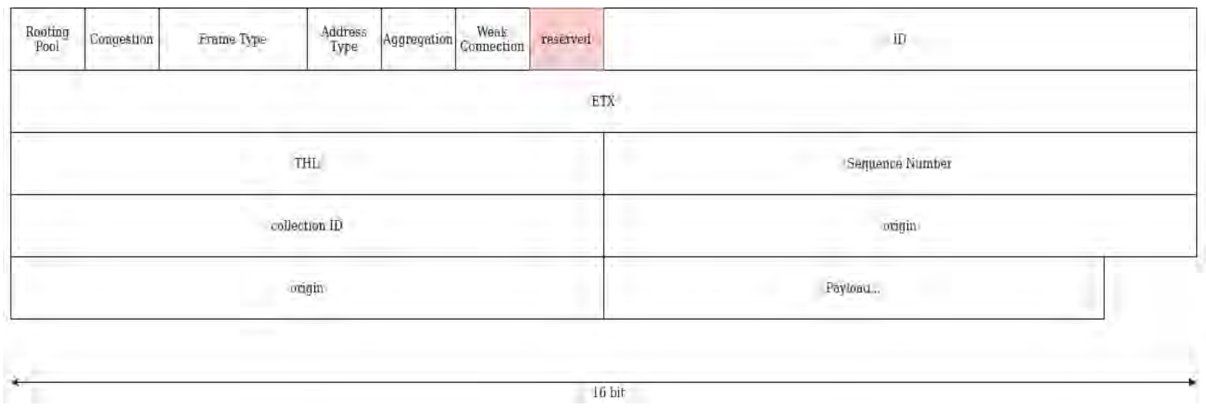


Figure 4-6: CTP Data Frame.

The role of each field is briefly as follows:

- The **Rooting Pull** bit allows the node to request routing information from its neighbor nodes. When a node receives a frame with the routing pull bit set, it must transmit a network frame.
- The **Congestion** bit informs the neighbor nodes that the node is forced to drop a packet.
- The **Frame Type** bits denote the type of the frame.
- The **Address Type** bit denotes whether the address has 16bit or 64bit length.
- The **Aggregation** bit indicates the existences of a piggy-back frame.
- The **Weak Connection** bit informs the coordinator that there is an inferior quality link along this route.
- The **Reserved** bit is currently not used.
- The **ID** byte is the network ID of the source node.
- The **ETX** bytes contain the routing gradient of the source node. When a node forwards a data frame, it replaces this information with its own ETX value.
- The **THL** (Time Has Lived) byte is the number of hops for this route. When a node generates a data frame, it sets this to zero. Every other node in the routing path increases it by one.
- The **Sequence Number** byte is an identification for this data frame. The origin node increases it for every generated data frame.
- The **Collection ID** byte serve as an identification for the type of data embedded to this frame.
- The **Origin** bytes declare the origin node. The length of the address depends on the address type.
- Finally, the **payload** contains the actual application-specific data.

The ETX and THL are the only fields that change during the forwarding. The origin and sequence number fields denote a unique origin packet and along with THL denote a unique packet instance within the network. The nodes keep a small cache with information (origin, sequence number, THL) about the frames they forward.

There are some ground rules that every node has to follow when receives a valid data frame. First, the ETX value inside the packet should always be greater than the receiving node's ETX value, else there is a network inconsistency and the receiving node should transmit a network frame before forwarding this frame. This rule comes from the fact that the ETX value produced by the parent's node ETX plus the link ETX. The link ETX value cannot be zero, therefore a node's ETX value is greater than the parent's. Secondly, if a node receives a data frame and the tuple (origin, THL, sequence number) exists in the cache, then it has to drop the frame as this is a duplicate (it has already been received at an earlier point in time). Thirdly, if a node receives a data frame and the tuple (origin, sequence number) exists in the cache, but the THL value is different, then there is a routing loop (the node has already received and also forwarded this packet in the past. To solve this problem, the node transmits a network frame and delays the transmission of this frame.

4.3.2 Network Frame

The network frame provides routing and duty cycle information to nodes, and it is a local broadcast transmission. Figure 4-7 shows the structure of the network frame.

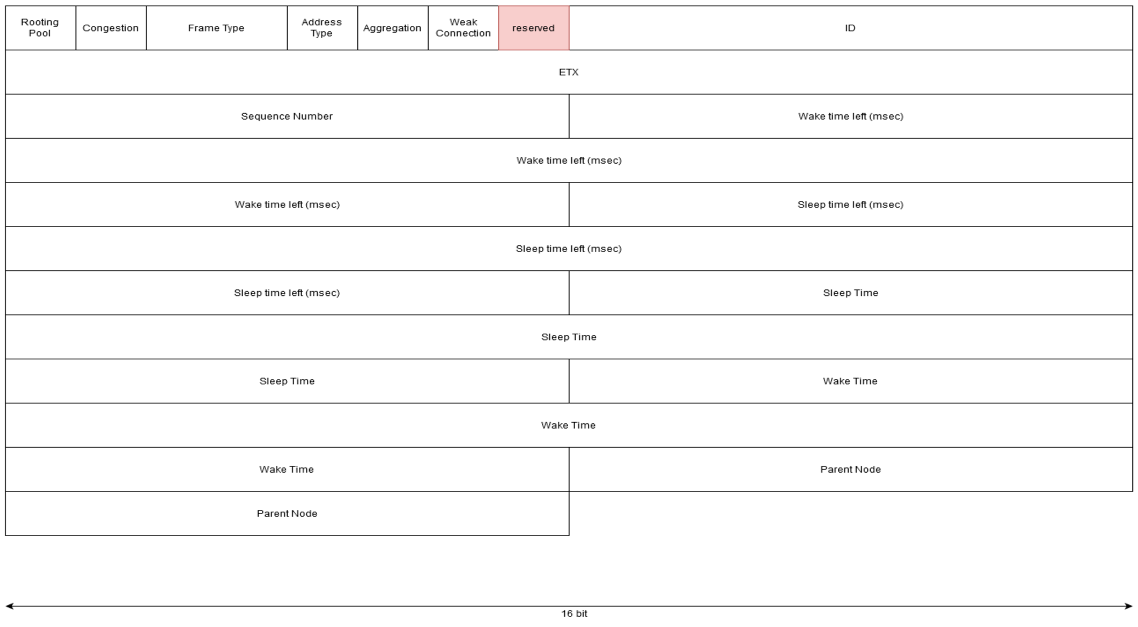


Figure 4-7: CTP Network Frame.

The first four bytes are similar to the data frame. The role of all other fields is briefly as follows:

- The **Sequence Number** byte serve as an identification for this network frame. The origin node increases it for every generated network frame.
- The **Wake Time Left** is the time until this node enters the low-power mode.
- The **Sleep Time Left** is the time until this node enters the normal-power mode. Since all the synchronized nodes cannot receive or send frames while operating in low-power mode, this field contains the sleep period of the current cycle. On the other hand, the coordinator remains active and uses this field to inform a new node about the time that he must wait until the network will be in an active state.
- The **Sleep Time** bytes declares the next cycle configuration for the sleep period.
- The **Wake Time** bytes declares the next cycle configuration for the wake period.
- The **Parent** bytes contain the address of the parent node.

Each node needs to receive at least one network frame to join the network. When a node receives a network frame, it compares the Sleep Time and Wake Time values with his own. If the configuration has changed, the node must transmit a network frame to notify the other nodes.

When a node receives a network frame, it must update his routing table and calculate the new ETX value. If this is significantly less than the old ETX value, transmits a network frame to notify the other nodes.

When a node receives a network frame with the routing pull bit set, it transmits a network frame. Nodes use an exponentially increasing timer to transmit network frames. If for any reason a node has to transmit a network frame, it resets the timer.

4.3.3 Command Frame

The command frame is used to send commands and receive the result. The coordinator is the only node able to send new commands.

When a node receives a command frame, it compares the target address with his own. If it is the same, the node executes the command and generates a command reply. Else, the node transmits it with local broadcast. The command replies are transmitted to the parent node using unicast transmission, like a data frame. Figure 4-8 shows the command frame structure.

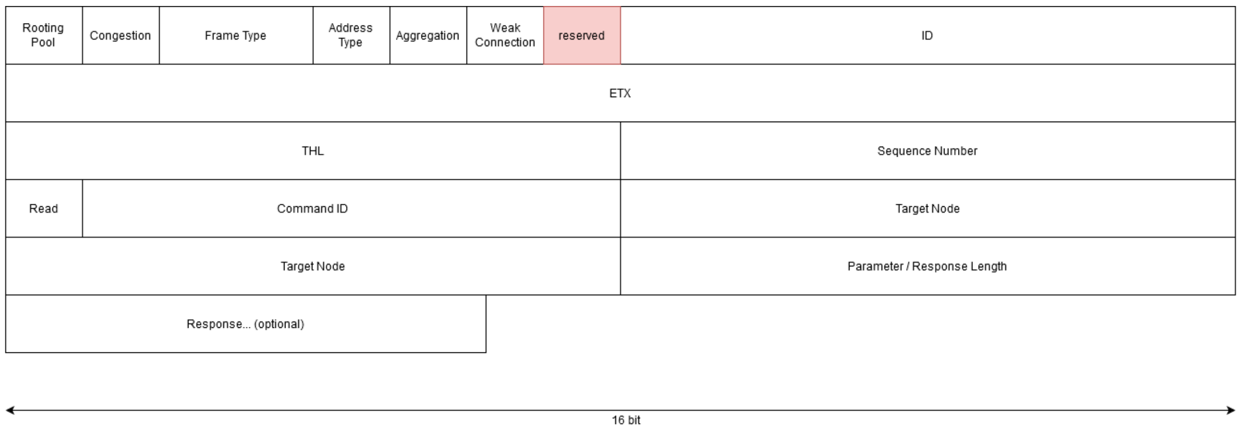


Figure 4-8: CTP Command Frame.

The first four bytes are the same as the data frame. The role of all other fields is briefly as follows:

- The **THL** (Time Has Lived) byte is the number of hops for this path. When a node generates a command reply, it sets this to zero. Every other node in the routing path increases it by one. If this is a command request, this field is zero.
- The **Sequence Number** byte serve as an identification for this command frame. The coordinator increases it for every generated command frame.
- The **Read** bit denotes if the coordinator request to set or read the value of the command register.
- The **Command ID** byte contains the command identification.
- The **Target Node** bytes are the address of the target node.
- The **Parameter** byte contains the command's parameter. If this is a command reply, this field contains the length of the reply.
- The **Response Payload** bytes contains the command reply. The command request does not use this field.

There is a small cache to recognize and ignore duplicate command frames using the sequence number for command requests and THL along with the sequence number for command replies. When a node receives a command request, it updates the ETX value of the origin node in the routing table. When a node receives a command reply, it increases the THL value, updates the ETX value inside the frame with his own, and then sends it to its parent node.

After each transmission, there is a delay to prevent self-interference. The delay depends on the transmission time. For a transmission time p, the delay is a random number between $(1{,}5p, 2{,}5p)$.

4.4 Command Subsystem

The commands sent by the user are transferred to the coordinator using the UART. The coordinator is responsible for verifying that the received data compose a valid command, a parameter and the target address. The target may be a specific node or all the nodes.

There are two types of command frames the command request and the command reply. The command request is used to send a command to the target node. The command reply is used by the target node to send a response back to the coordinator.

The command request is transmitted with broadcast and contains the sequence number, the read field, the command ID, the target node, and the parameter. The command reply is transmitted with unicast and contains the THL, the sequence number, the read field, the command ID, the target, the response length and the response. Both instances contain the frame control byte, the ID and the ETX field. To distinct the two instances, the first bit of the command ID is 0 for command request and 1 for command reply.

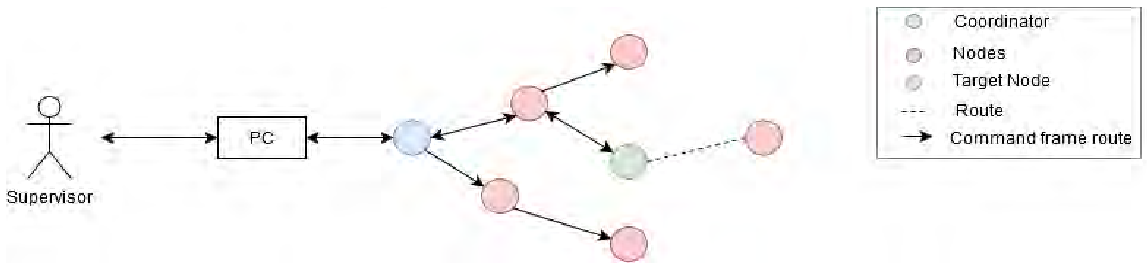


Figure 4-9: Command transmission from the user to the device.

Figure 4-9 shows the command transmission from the user to the target node and the command reply back to the coordinator. The command from the user is received by the coordinator and the command frame is constructed. Then it is transmitted with broadcast from the nodes. The target node receives the command request, generates the command reply and sends it with unicast to the coordinator.

4.5 Duty Cycle

The coordinator oversees the operation of duty cycle and is the only node that can change the sleep time and wake time. All other nodes try to synchronize their clocks, to maintain a global network state. Figure 4-10 shows the coordinator's process.

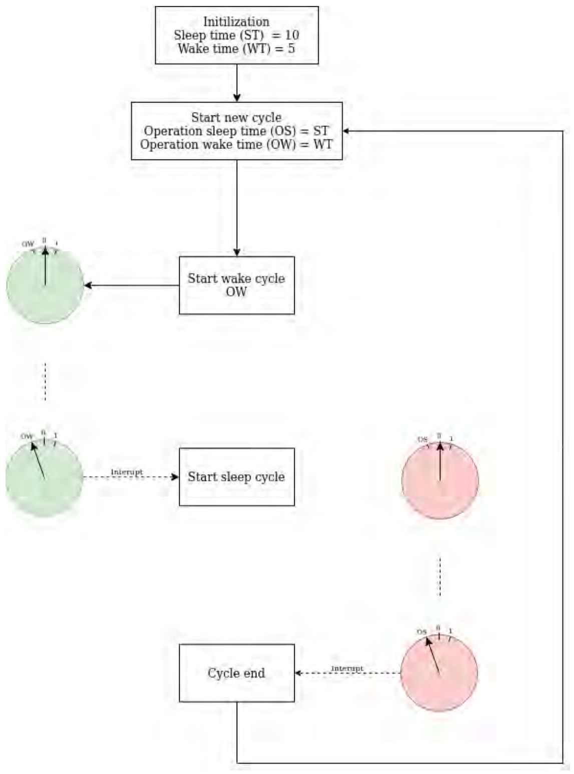


Figure 4-10: The Coordinator's duty cycle.

Four variables are used to drive system operation. The wake time (WT) and sleep time (ST) are the times for the new cycle. The operational wake time (OS) and operational sleep time (OS) are the times of the current wake and sleep period. At the start of the cycle, OS and OW take their values from ST and WT, then a timer counts for OW time. At the end a handler function changes the state to sleep state, and the timer counts for OS time.

The coordinator sends periodically network frames during the wake network state, containing the ST and WT for next cycle, the wake timer value until the network state changes and the current OS value.

A node synchronizes when receiving one network frame from the coordinator or any other synchronized node. This is done by setting the timer values as temporary wake and sleep period, and the WT and ST for the next cycle.

Synchronization of a node during sleep network state is possible only by the coordinator. A new node that joins the network, sends a network frame requesting routing information, and the coordinator replies with another network frame containing the ST, WT, the wake timer value, that is zero as the network is in sleeping state and the sleep timer value. The new node uses the sleep timer value to wait until the start of the next cycle.

The clocks of the nodes may drift in time, possibly at a different rate. To compensate such drifts, nodes estimate the difference between their clocks and the Wake Time Left value that exists in every received network frame. If the difference is greater than a threshold, the nodes will correct their clock next time they receive a network frame from their parent node, by adjusting the remaining wake time. In this way, each node remains synchronized with its parent node.

Chapter 5 System Implementation

5.1 Hardware

The system includes two types of nodes, the coordinator and the ordinary nodes. All nodes have an RF module to communicate with each other. In addition, the coordinator needs at least one USB port to communicate with an external computer, such as PC, and an optionally second one, can be used to print runtime logs (the logging interface implemented by the OS uses the debugger's USB port).

The current implementation uses the NUCLEO-L496ZG platform [13], [14] as the coordinator, and the NUCLEO-L476RG and NUCLEO-F401RE platforms [15], [16] for the nodes. All platforms are supported by Zephyr OS. The radio module used in this implementation is the Microchip MRF24J40MAT-I/RM, which needs a suitable driver to be used with Zephyr OS.

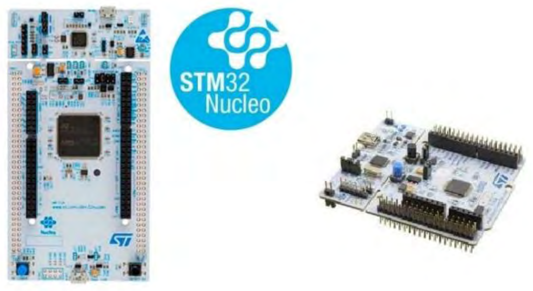


Figure 5-1: Left the Nucleo-L496ZG and right the Nucleo-L476RG.

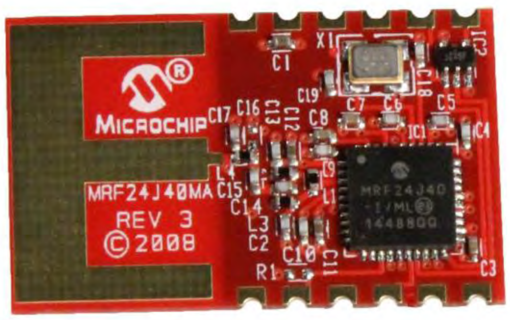


Figure 5-2: MRF24J40MAT-I/RM.

The NUCLEO-L496ZG board features an ARM Cortex-M4 based STM32L496ZG MCU, with 1MB flash memory and 320KB SRAM, USB OTG 2.0 ports and supports up to 20 communication interfaces. The NUCLEO-L476RG board features an ARM Cortex-M4 based STM32L476RG MCU, with 1MB flash memory and 128KB of SRAM and up to 18 communication interfaces. The

NUCLEO-F401RE board features an ARM Cortex-M3 based STM32LQFP64 package with 512KB flash memory and 96KB of SRAM. All boards have integrated the ST-LINK debugger/programmer for easy access.

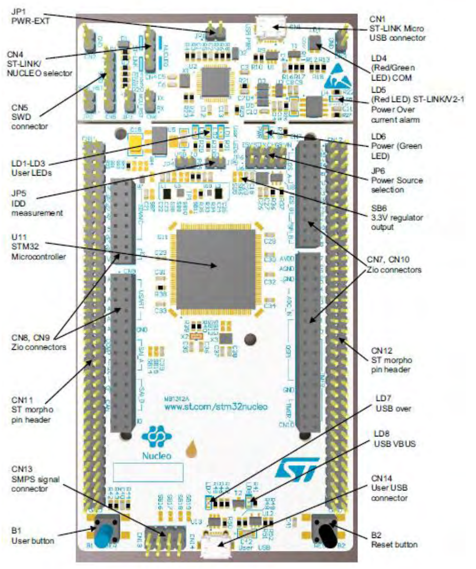
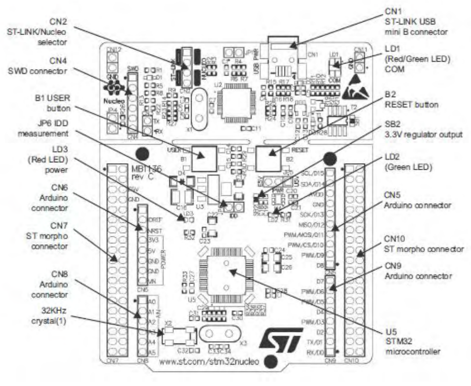



Figure 5-3: Nucleo Boards Layout.

The Microchip MRF24J40MAT-I/RM [5], [17] is a 2.4 GHz IEEE 802.15.4 compliant RF transceiver module. It is compatible with ZigBee®, MiWi™, MiWi™ P2P and Proprietary Wireless Networking Protocols. It has a small size (7.8 mm x 27.9 mm), integrated crystal, internal voltage regulator, matching circuitry and a PCB antenna, as shown in Figure 5-4. The module has a hardware CSMA-CA mechanism, an automatic acknowledgment response and an FCS check. It also supports all CCA modes and provides estimates of RSSI and LQI. It is also capable of automatic retransmission and features a hardware security engine.

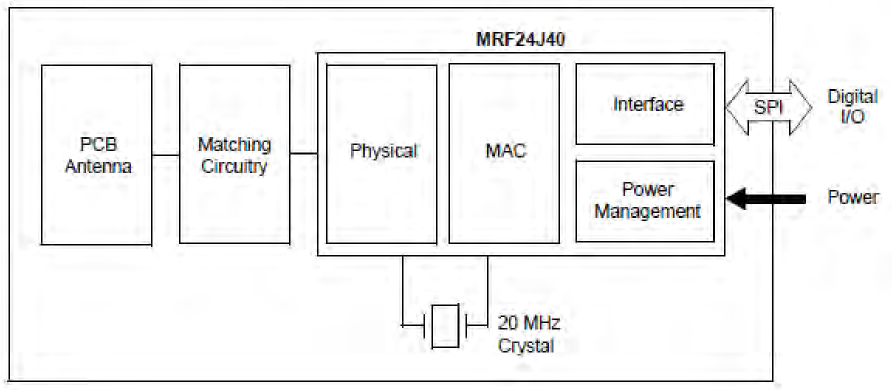


Figure 5-4: MRF24J40MA layout.

The radio module is connected to the nucleo boards via a 4-wire SPI interface and has 3 more pins, the interrupt, the wake and the reset, as shown in Figure 5-5. The interrupt pin is used to trigger the interrupt function of the driver. The wake pin is used to put the module to low-power mode and the reset pin to do a hard reset when this is necessary.

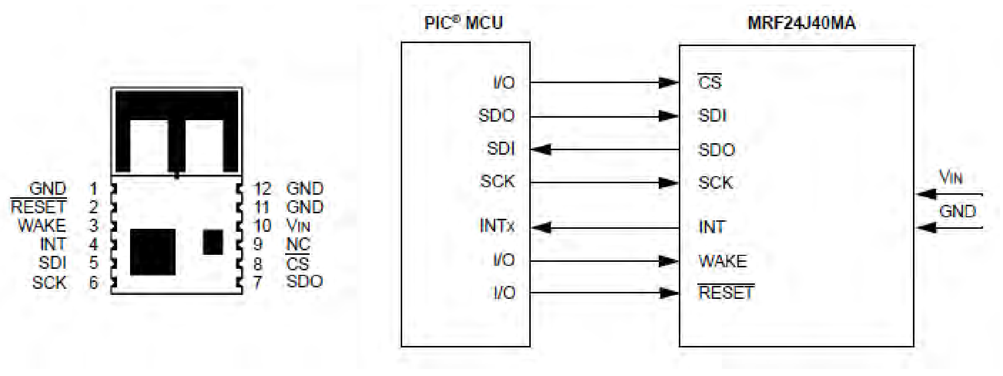


Figure 5-5: MRF24J40MA pin layout.

5.2 Software Architecture

The coordinator and the ordinary nodes both follow the same software architecture, which comprises three layers, as shown in Figure 5-6.

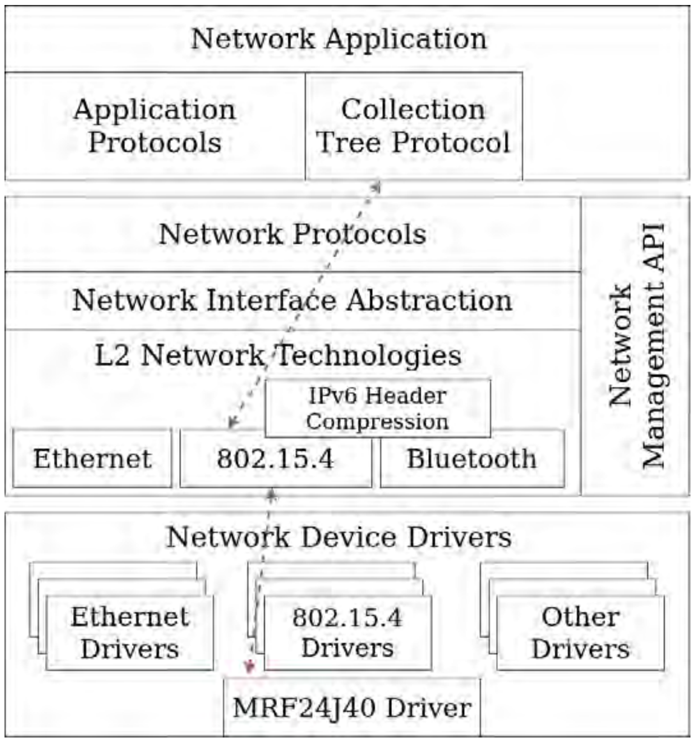


Figure 5-6: Software Architecture.

5.2.1 MRF24J40 Driver

The bottom layer includes the radio device driver that is responsible for communication with the radio device and the network layer. The driver is implemented using the C language, and the hardware is imported to the operating system using device-tree bindings and kernel configuration described with YAML language and Kconfig symbols.

The driver includes two preemptible threads, the first initializes and controls the radio device, and the second receives incoming frames and stores them temporarily. There is also a worker thread that triggered after the frame reception. The worker thread is responsible for verifying that the received frame is a valid frame, allocating the memory needed and notifying the network layer.

The radio driver also provides an API to network layer that includes the following functions:

- The **Transmit Function** transfers the packet to radio device and returns the result from the interrupt handler. The transmission is considered as failed if an expected acknowledgement is not received within the expected time window or the radio device reports that channel is busy.
- The **Start** and **Stop Functions** put the radio device in normal and low-power mode, respectively.
- The **Filter Function** changes the IEEE 802.15.4 configurations like the address or the PAN ID.
- The CCA Function performs an energy detection on the channel.
- The Get Capabilities Function reports the radio module specifications to the network layer.

Figure 5-7 shows the internal structure of the driver. Colored with green are the major components like thread and handler functions. Colored with blue are the relevant libraries, interfaces and data structures. Colored with red is the hardware.

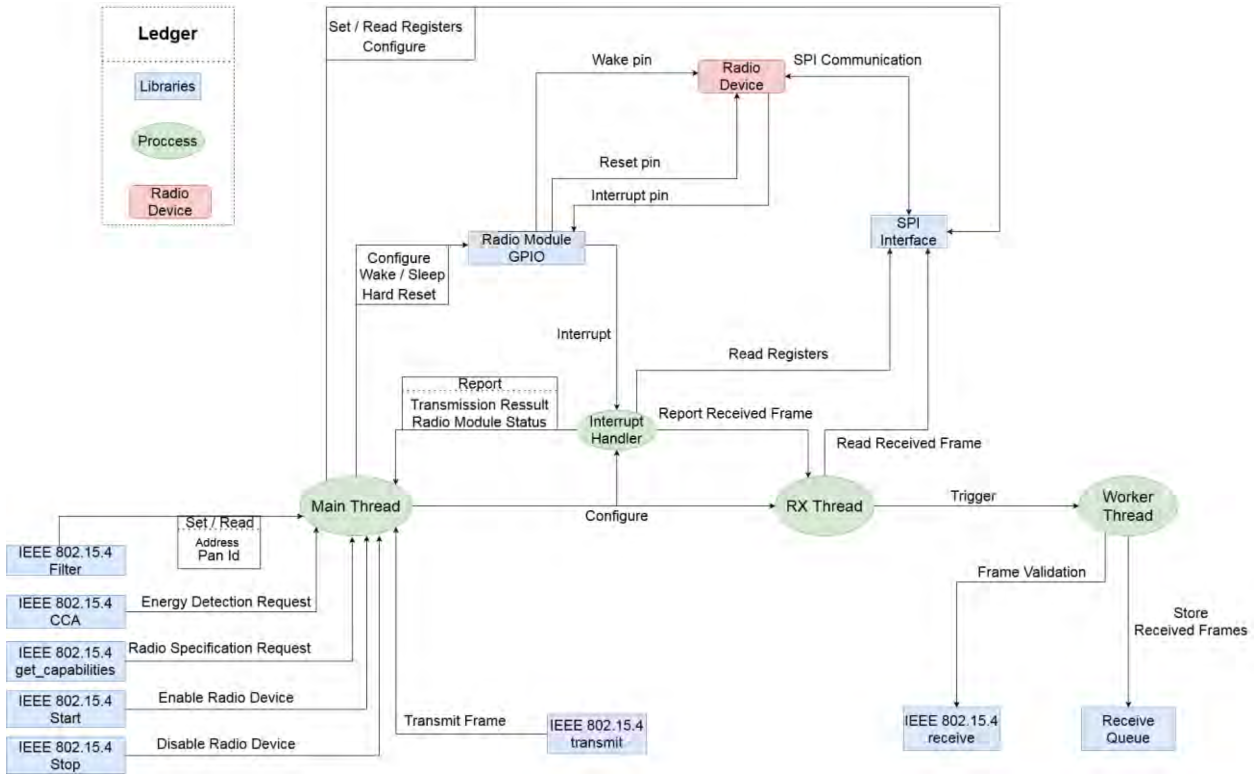


Figure 5-7: Radio Driver Internal Structure.

5.2.2 Network Layer

The Network layer includes the IEEE 802.15.4 drivers and the network interface. The implementation is based on the existing infrastructure of the operating system, with a few extensions in order to better support this particular type of network. The API consists of the following functions:

- The **Init function** is called from the protocol level and initializes the network interface and the IEEE drivers as a non-beacon IEEE802.15.4 network.
- The **Send function** is called from the protocol level to send a packet. It takes as a parameter the payload of the packet and creates the MAC header. Then the MSDU is passed on the driver layer.
- The **Receive function** is called from the driver layer when a new packet is received. The function checks and decomposes the packet. If it is a valid MAC frame, the functions push the packet to the RX FIFO queue.
- The **Enable function** can be used to enable or disable the network interface.

5.2.3 Protocol Layer

The protocol layer is responsible for the CTP routing algorithm, the duty cycle and either the data collection (normal nodes) or the communication with the PC (coordinator). This layer implemented using the C language. Figures 5-8 and 5-9 show the internal structure for this layer. It consists of three threads, described in the following.

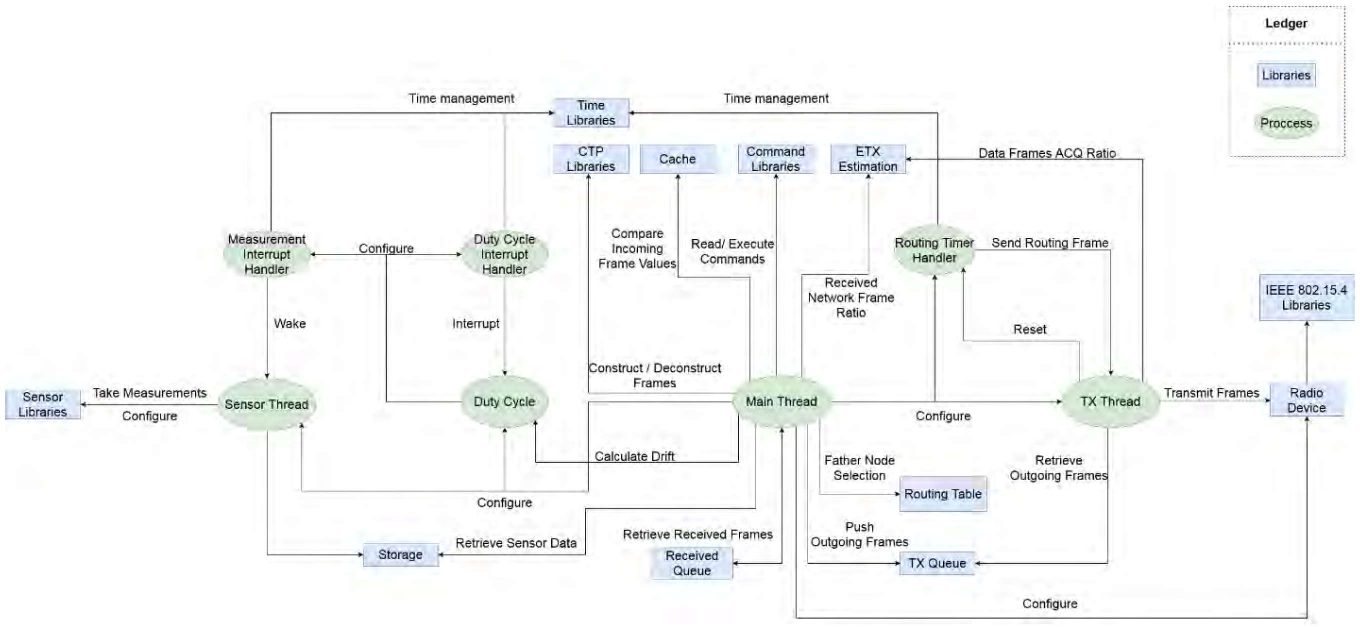


Figure 5-8: Internal structure of a normal node.

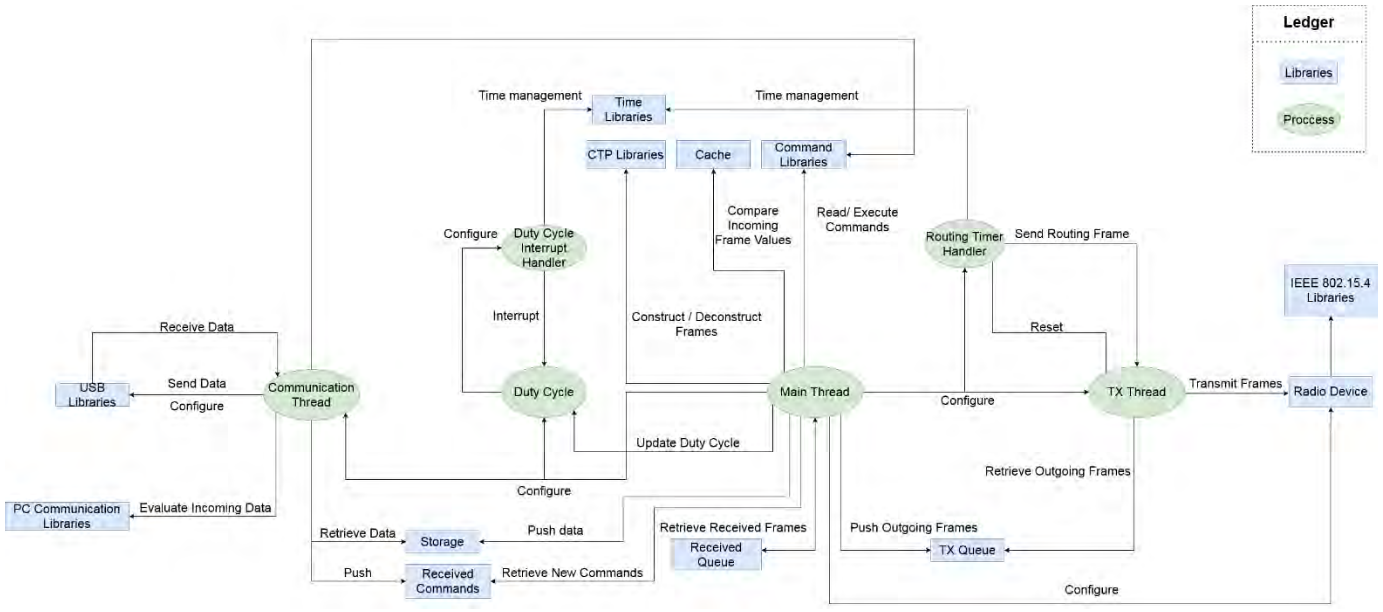


Figure 5-9: Internal structure of the coordinator.

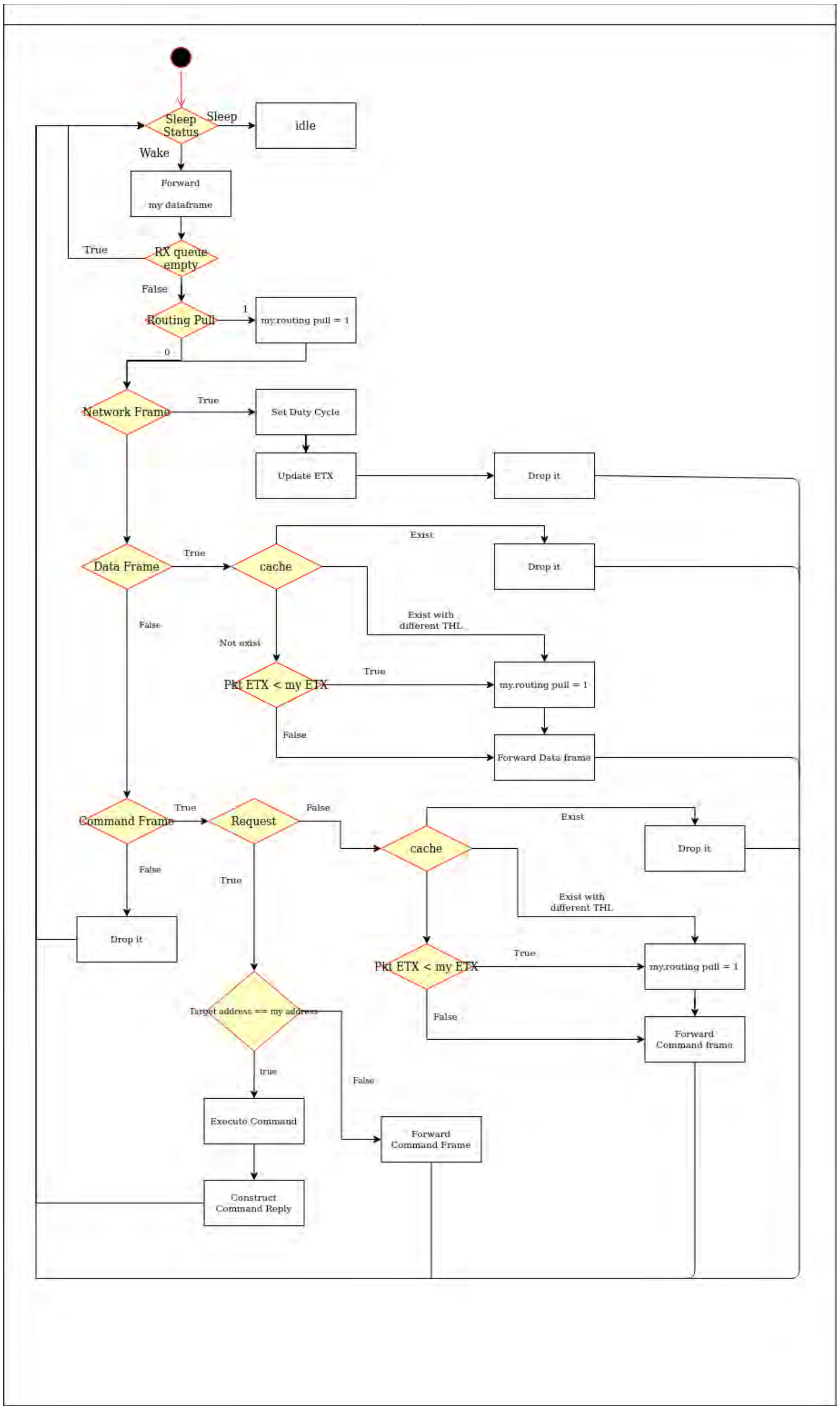


Figure 5-10: Protocol thread diagram.

The first thread implements the basic functions of the CTP protocol. It receives and decodes incoming frames from the lower layer, constructs new frames and stores them to the TX queue, maintains and synchronizes the duty cycle and executes received commands. Incoming frames are stored in a lower level FIFO. As shown in Figure 5-10, the protocol thread pulls the new frame and compares the THL, sequence number and origin node with corresponding values in cache. This procedure reveals if the new frame is a duplicate, or if there is a network inconsistency. For incoming data frames or command replies, the packets are added to TX FIFO to be forwarded from the TX thread. For incoming network frames, the thread compares the wake time left to find the drift and updates the duty cycle configurations (sleep time and wake time). For command frames targeting other nodes, the frames are added to TX FIFO. If the target is this node, the thread executes the command and a new frame (command reply) is constructed.

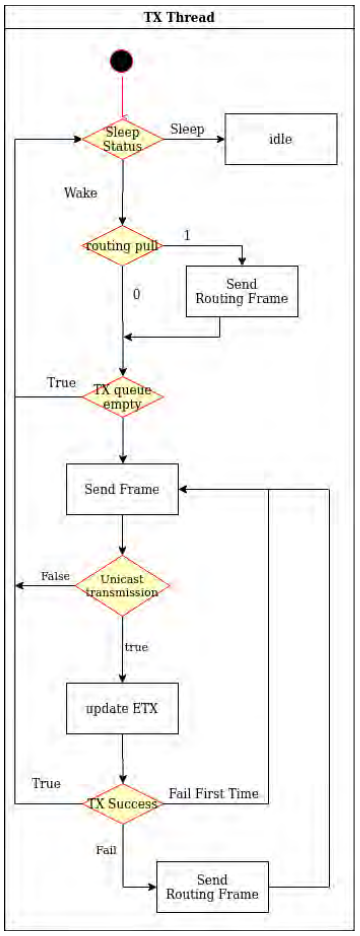


Figure 5-11: TX thread diagram.

The second thread or TX thread, shown in Figure 5-11, pulls frames from the TX FIFO and transmits them using the network's layer functions. For each unicast transmission, the ETX value is updated. If a unicast frame cannot reach the destination, the TX thread tries one more time. If it fails, transmits a network frame, waits for incoming network frames to update the parent node and then retries. To avoid transmissions at the end of the wake period, the thread stops transmitting new frames, approximately one second before the transition to low-power mode. When the network operates in low-power mode, only the coordinator is able to transmit network frames to synchronize new nodes.

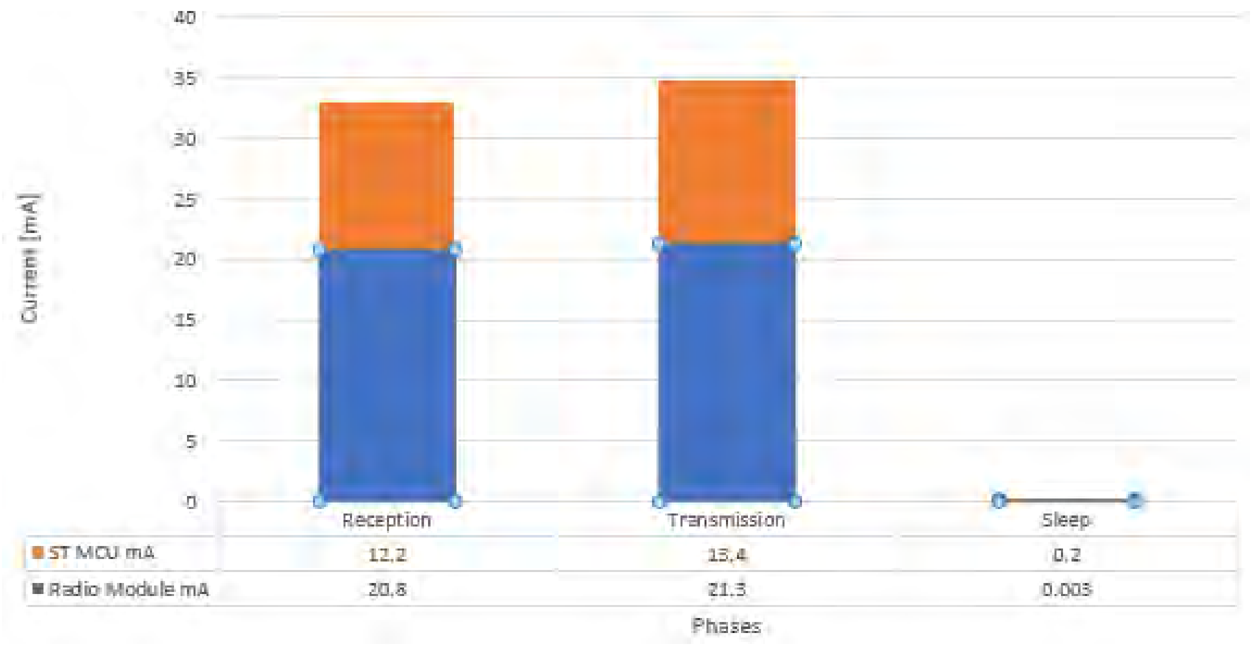
The third thread for a device that operates as an ordinary node is used to collect measurements from the sensors and store them in a queue. Then waits until the next cycle. It is active only for a brief period in each cycle. In the coordinator, the third thread is used to establish communication with the PC, via a UART, and to transfer incoming data and command replies. The incoming command request from the user are decoded and added to a command queue.

Chapter 6 Experiments and Evaluation

6.1 Power Consumption

The power consumption of a node can be broken down to the consumption of the micro-controller and the consumption of the radio module. To measure the consumption of the micro-controller, we remove the JP6 (IDD) jumper on the top side of the nucleo board and then connect an ammeter. To measure the consumption of the radio module, we connect the ammeter between the nucleo device and the radio device.

There are three execution phases, each one with different power consumption: (i) the RX phase, when the radio device waits for or is receiving incoming frames; (ii) the TX phase, when the radio device is transmitting frames; (iii) the sleep phase, when the radio device is not operating and the MCU enters a low-power mode. Figure 6-1 shows the power consumption for each phase.



	Reception	Transmission	Sleep
🟧 ST MCU mA	12.2	13.4	0.2
🟦 Radio Module mA	20.8	21.3	0.005

Figure 6-1: With blue the power consumption of the radio device in each phase. With red the power consumption of the nucleo board.

The power consumption for the sleeping period is relatively low and can be further improved, if the MCU enters the deep sleep mode

6.2 Topology

As power consumption depends on the total number of transmitted frames, the network topology is a significant factor for network performance.

6.2.1 Star Topology

First, an evaluation is conducted using a star topology, as shown in Figure 6-2. In this case, three nodes communicate directly with the coordinator.

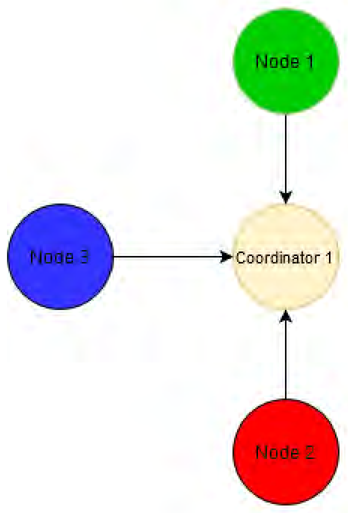


Figure 6-2: Network Topology.

We measure the traffic of each node in each wake period. Figure 6-3 shows the results. All nodes send at least one data frame in each wake period. The spikes are network frames send from the nodes to inform their neighbors about their ETX values. As time passes by and the network becomes stable, between the 7th and 21st wake period, the transmission of network frames becomes less frequent.

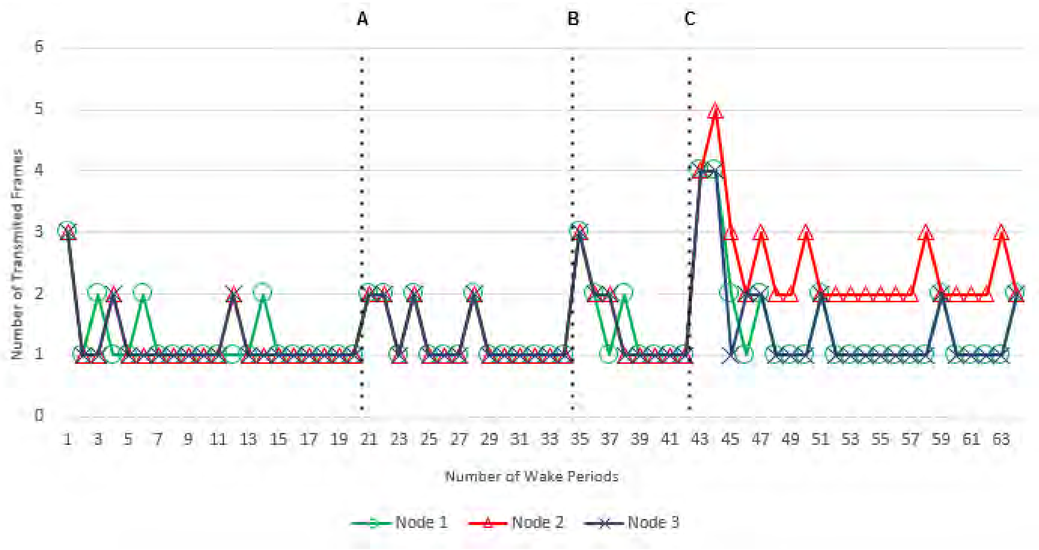


Figure 6-3: Total number of transmitted frames for each node.

At point A, we change the duration of the sleep period of the network, thus each node transmits network frames. At point B we change the duration of the wake period and, as expected, the nodes send network frames frequently. Finally, at point C, the old coordinator is replaced with a new one, which has different sleep settings, while the old coordinator is removed from the network, as shown in Figure 6-4 (i).

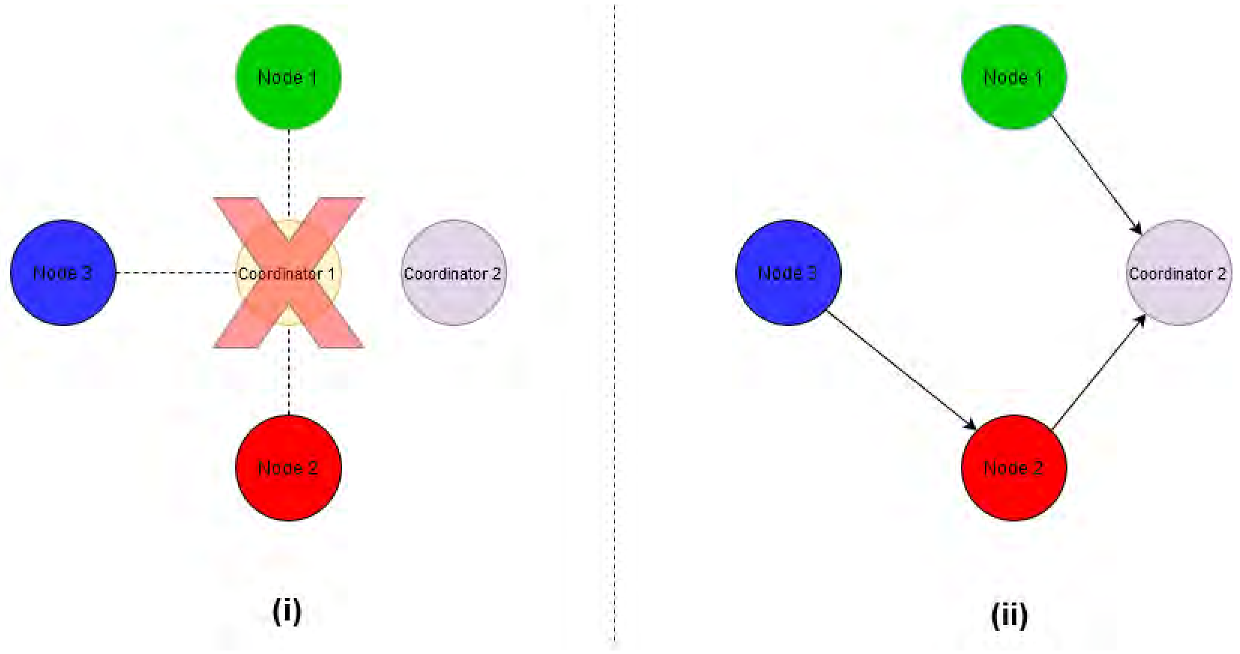


Figure 6-4: Network topology while replacing the coordinator.

The new coordinator transmits a network frame. All nodes take the new duty cycle settings from the new coordinator and transmit network frames to inform their neighbors. However, for a while, nodes 1-2-3 keep as their parent node the old coordinator because the last ETX value that each node had for the old coordinator is equal to the estimate for the new one. This changes as they are trying to send data frames and fail. The estimated ETX value increases, causing more network frames to be transmitted in order to find an alternative route. At this point, the estimate ETX value for the old coordinator is higher than the estimate for the new one, that causes the change of the parent node.

Interestingly, after nodes update their parent, node 3 selects node 2 as its parent node (instead of the new coordinator), thus it appears to be 2-hops away from the new coordinator, as shown in Figure 6-4 (ii). This can be explained as follows. When the node 3 requests additional routing information, it first receives the network frame from node 2, which it sets as its parent. Subsequently, it receives the network frame of the new coordinator, but the estimated ETX value is not low enough to change its parent from node 2 to the new coordinator. From that point, node 2 has to forward the incoming data frames of the node 3, which explains the increase in the number of frames sent by it.

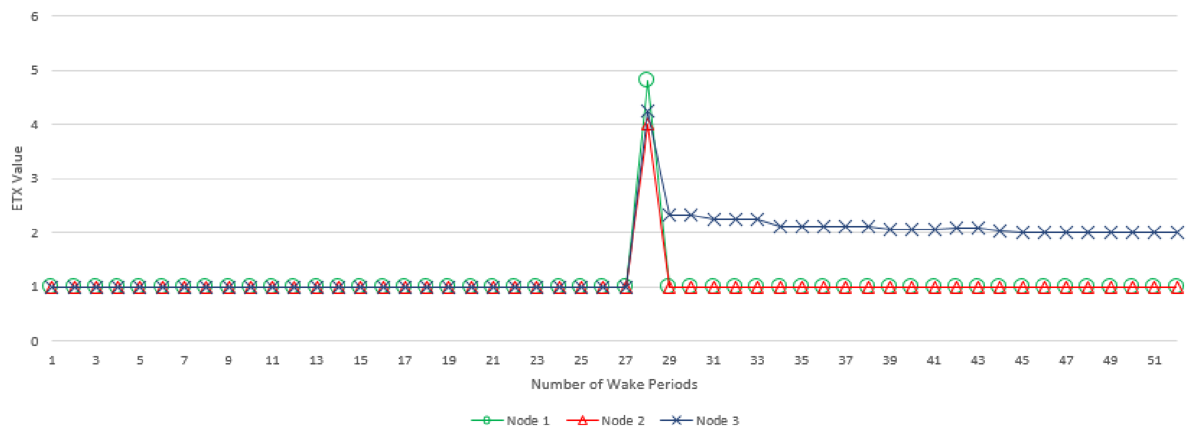


Figure 6-5: ETX estimation of each device. The spike is the moment that the coordinator is replaced. For a moment, the ETX value increases as they cannot send data frames. As they select the new coordinator as their parent node the ETX value decreases again. The Node 3 select another node as the parent node, which explains the higher ETX value.

The ETX value of each node is shown in Figure 6-5. Every node starts with ETX 1. Until the replace of the coordinator, there are no transmission failures, thus the ETX is stable. After replacing the coordinator, every node transmits network frames and synchronizes with the new duty cycle, without changing their parent node. All nodes maintaining an ETX value for the new coordinator that is equal to the old. The following failed data frame transmissions, causing the ETX value to grow (which is the estimated value for the old coordinator) and as a result, nodes request additional routing information. At 29th wake period, they change their parent and the ETX value becomes again 1 as there is no failed transmission to the new coordinator. For node 3 which chooses node 2 as a parent node, the ETX value fluctuates for a few cycles, because of the network frame reception rate, before it stabilizes to a higher value than other nodes due to the extra hop that is made to reach the new coordinator.

6.2.2 Chain topology

In the next experiment, we use a chain topology where each node has a different hop distance to the coordinator. The topology is shown in Figure 6-6.

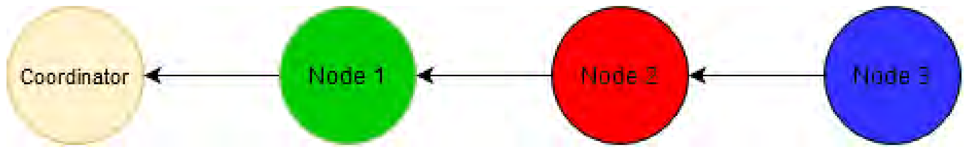


Figure 6-6: Chain topology.

In this case, each node must forward to the coordinator incoming frames sent by its immediately preceding nodes. The total number of transmitted frames for each node is shown in Figure 6-7.

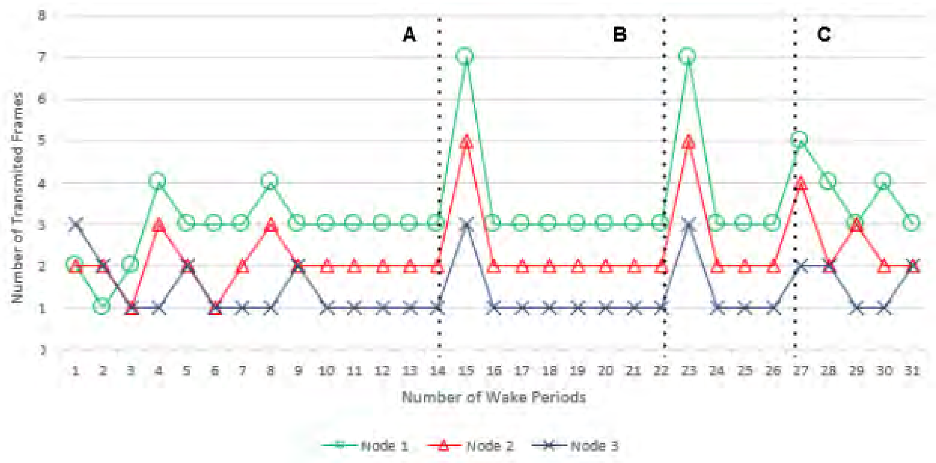


Figure 6-7: Total Number of transmitted frames.

As expected, the nodes closer to the coordinator transmit more frames. After synchronization, there are two small spikes, at 4th and 8th wake period, caused by network frame transmission. At the points A and B, we send a network command to change the sampling rate of all nodes. The purpose of the command transmissions is to evaluate the reaction of the network when they are sent. The command frames cause the two spikes at the points A and B, because they are forwarded using both broadcast (to forward the command request to the network) and unicast transmissions (to send the command reply to the coordinator). Between the 10th and 26th wake period, the nodes are not transmitting network frames, because the two command requests contain routing information. At point C, we change the sleeping period and the nodes send a network frame. Node 1 failed to transmit a data frame and requests new routing information, thus the total number of transmitted frames of the nodes in range of node 1, slightly increases too

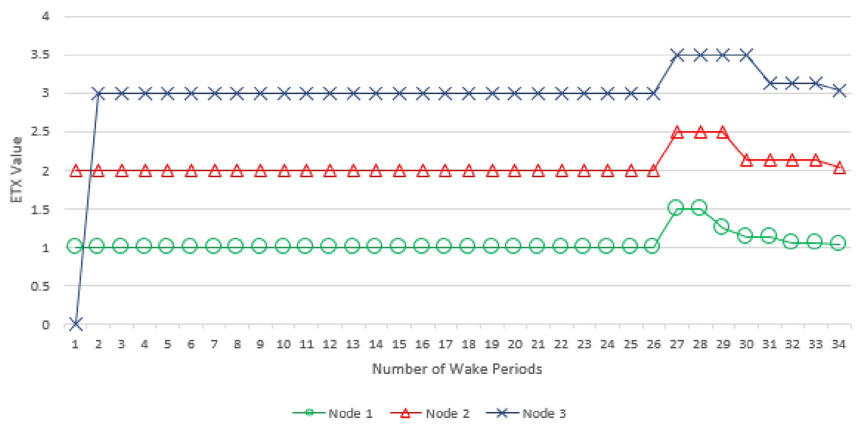


Figure 6-8: ETX Estimation.

This can be seen in the ETX value of each node in Figure 6-8. The ETX value of node 1 after the failed attempt to send the data frame, at 27th wake period, slightly increases, and subsequently the ETX value of the other two nodes, because node 1 exists in their route to the coordinator.

6.3 Throughput

In this experiment, the system throughput is measured. The topology includes two ordinary nodes in range of the coordinator, as shown in Figure 6-9. Each node tries to send the maximum number of frames in each cycle. To avoid unbounded memory usage/allocation, the nodes produce a new frame only after the successful transmission of the previous one. If a transmission is unsuccessful, the node tries to send it again.

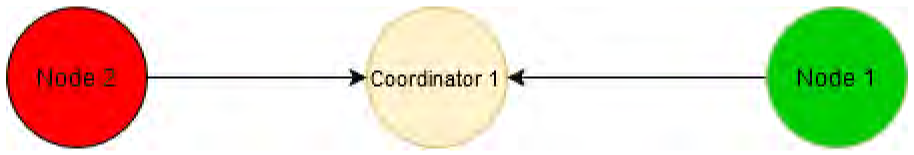


Figure 6-9: Test Topology

The produced frames are 128 bytes long with an application payload of 108 bytes. The rest 20 bytes of each frame are the IEEE 802.15.4 header, the CTP header and the FCS, as shown in Figure 6-10.

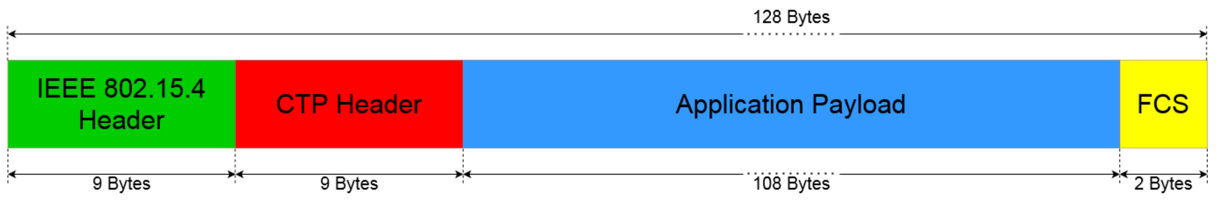


Figure 6-10: Data Frame Length.

The number of frames that can be transmitted depends on the wake period of the network, therefore the test was repeated for wake periods of 10, 15, 20 and 25 seconds. To calculate the system throughput, we use the payload length of the total number of frames send by the two nodes. Figure 6-11 shows the results.

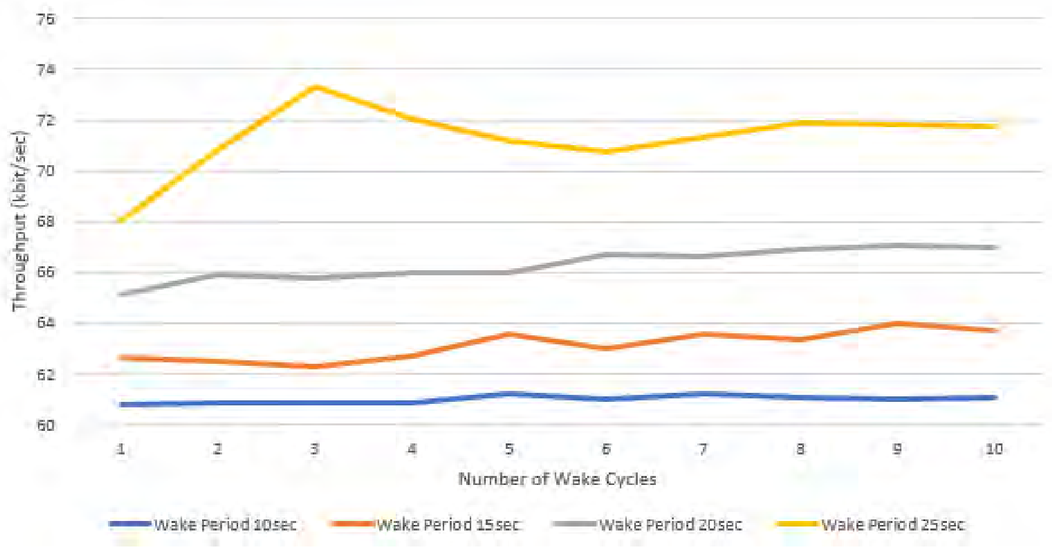


Figure 6-11: Throughput during different wake periods.

The results show that increasing the duration of the wake period generally leads to a small boost in throughput. This is because a small part of the wake period not used to send data but it is necessary for the system in order to maintain synchronization between the nodes, either by sending network frames or by preventing the nodes to send frames right before the transition to low-power mode. As wake period increases, the amount of time where the node is free to send data frames increases too and the overhead becomes less significant, leading to higher throughput.

To support networks with large traffic, the increase of the wake period can achieve the desire outcome but lead to higher power consumption. By using the wake periods mentioned before, a sleep period of an hour and a typical AA battery of 2100 mAh, the number of continuous operating days drops from 292 days for 10 seconds to 197 days for 25 seconds. The other two configurations last for 252 and 221 days respectively.

6.4 Various other tests

More experiments were performed to test the recovery process of the network, by replacing or removing nodes. In all cases, the rest of the node adapt to the changes and find another path. This proves that the protocol is robust.

The duty cycle was tested for various configurations of the wake and sleep period, over an extended period of time. All nodes and the coordinator had no deviation in their clocks.

In general, the number of the protocol-related frames is relatively low. In addition, the protocol is address free, meaning that there is no discovery process before data transmissions. During an experiment with two normal nodes and the coordinator over 360 wake periods, the total number of transmitted network frames is 40.

Chapter 7 Conclusion and Future Work

7.1 Summary

In this thesis, a system for WSN is developed as an alternative to other commercial WSN systems. The system is built using Zephyr OS and comprises 3 layers. The lower layer is the physical layer that communicates with the radio device to send and receive data. The middle layer is the MAC layer that uses the IEEE 802.15.4 standard to form and decode the transmitted frames over the medium. The top layer, which was developed as part of this thesis, implements the CTP protocol, a tree-based routing algorithm to select and maintain an optimal path for the data flow. Furthermore, the top layer implements a command subsystem to configure and control the nodes remotely, a duty cycle subsystem to improve the lifetime of the nodes, and a sensor subsystem to collect the relevant measurements. The radio module used in this implementation complies with the IEEE 802.15.4 standard. It offers the low-level control mechanisms like the CSMA-CA algorithm to sufficient use the medium, the FCS check and the automatic acknowledgment response to speed up the frame transmission process.

A series of experiments were performed to verify that the system is working under different topologies and duty cycle configurations. In addition, the recovery process of the nodes was tested to verify the reliability of the system. Furthermore, metrics like the power consumption, and the throughput of the system were measured to verify the performance of the network. The system prototype performed as expected in all cases.

7.2 Possible extensions

A graphical interface could be developed to present the node's data, communication statistics and routing position as well as to support the control of the coordinator, beyond the command-line tool that is used for this purpose in the current prototype.

The UART connection between the coordinator and the computer could be replaced by an Ethernet cable, and the data transfer could be implemented using a standard web protocol. This requires a new board for the coordinator node and a web-based interface for handling the incoming data. The advantage of this approach is that the coordinator could be directly accessed online, without requiring an intermediate computer.

The experiments show that a better memory handling may increase the performance of the network as there will be more memory available to handle more outgoing frames. During the throughput experiment, this problem introduced a difficulty to support large traffic (a new data packet is produced only after the previous one is successfully transmitted).

The procedure for maintaining alternative paths can be improved, if a garbage collector removes all inactive paths. This could reduce the number of network frames needed to find an alternative path because the routing table would have only recent entries.

Last but not least, data encryption and frame integrity should be integrated to the protocol to enhance security. The MAC layer offers a complete solution that could be integrated to the CTP protocol. The drawback is the reduction of the payload size in the frames, since extra bytes are needed to support the corresponding encoding and decoding procedures.

References

[1] J. Geier, *Wireless LANs, Second Edition*. 2001.

[2] “List of IEEE publications - Wikipedia.” https://en.wikipedia.org/wiki/List_of_IEEE_publications.

[3] E. Engineers, *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, vol. 13, no. 9. 2012.

[4] I. Standard and I. C. Society, *IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, vol. 2011, no. September. 2011.

[5] Microchip, *MRF24J40 Data Sheet*. 2008.

[6] “Zephyr Project | Home.” https://www.zephyrproject.org/ .

[7] “GitHub - zephyrproject-rtos/zephyr: Primary Git Repository for the Zephyr Project. Zephyr is a new generation, scalable, optimized, secure RTOS for multiple hardware architectures.” https://github.com/zephyrproject-rtos/zephyr .

[8] “Zephyr Project Documentation — Zephyr Project Documentation.” https://docs.zephyrproject.org/latest/index.html .

[9] R. Fonseca, O. Gnawali, K. Jamieson, and S. Kim, “The collection tree protocol (CTP),” *TinyOS TEP*, no. January 2006, pp. 1–7, 2006.

[10] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, “Collection tree protocol,” *Proc. 7th ACM Conf. Embed. Networked Sens. Syst. SenSys 2009*, pp. 1–14, 2009, doi: 10.1145/1644038.1644040.

[11] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, “CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks,” *ACM Trans. Sens. Networks*, vol. 10, no. 1, 2013, doi: 10.1145/2529988.

[12] A. Woo, T. Tong, and D. Culler, “Taming the underlying challenges of reliable multihop routing in sensor networks,” *SenSys’03 Proc. First Int. Conf. Embed. Networked Sens. Syst.*, pp. 14–27, 2003, doi: 10.1145/958491.958494.

[13] STMicroelectronics, “Getting started with STM32 Nucleo board software development tools,” no. January 2016, p. 22, 2016, [Online]. Available: www.st.com.

[14] A. Analysis and U. Manual, “User Manual User Manual,” vol. 3304, no. January, pp. 1–148, 2012.

[15] S. T. M. N.- Mb, “UM1724 User manual STM32 Nucleo-64 boards (MB1136),” *SpringerReference*, no. April 2019, p. 69, 2011, doi: 10.1007/springerreference_28001.

[16] R. B. J. Brinkgreve and S. Kumarswamy, “Reference Manual Reference Manual,” *Technology*, vol. 1, no. November, pp. 720–766, 2008, doi: 10.1093/cid/ciq238.

[17] M. Technology, “MRF24J40MA Data Sheet,” *Technology*, 2008.