University of Southampton

Faculty of Engineering and Physical Sciences

Electronics and Computer Science

A Low Power Sleepy Network Design for IPv6-based Environmental IoT Systems

by

Jiawei Lu

ORCID ID 0009-0003-4326-6722

Thesis for the degree of Doctor of Philosophy

Mar 2026

University of Southampton

<u>Abstract</u>

Faculty of Engineering and Physical Sciences

Electronics and Computer Science

<u>Doctor of Philosophy</u>

A Low Power Sleepy Network Design for IPv6-based Environmental IoT Systems

by

Jiawei Lu

Environmental Sensor Networks (ESN) are widely deployed for monitoring in remote and hazardous environments where energy is constrained. ESN nodes are typically battery-powered and are required to operate reliably for long periods. Benefiting from standardisation, the concept of Environmental Internet of Things (E-IoT) has emerged. Energy-constrained E-IoT systems commonly adopt duty-cycled MAC protocols such as ContikiMAC and 6TiSCH to reduce energy consumption. Although such protocols can effectively reduce the activation time of wireless transceivers, the unnecessary radio activation caused by periodic idle monitoring or frequent network maintenance cannot be further limited. Moreover, once the system is deployed, certain predefined parameters such as the communication rate become difficult to adjust in response to changing environmental conditions or application requirements.

In contrast, schedule-based sleepy networks are designed to maximise the sleep duration of all nodes to conserve energy, while only transmitting data and maintaining the network during scheduled communication windows. Although extended sleep periods introduce additional communication delays and limit the ability to handle bursts of data, sleepy networks are still suitable for long-term environmental monitoring in remote or hazardous areas. This thesis investigates whether schedule-based sleepy networks can achieve lower energy consumption than systems deploying duty-cycling MAC protocols, and how to optimise communication windows within multi-hop tree topology network to reduce energy consumption. Firstly, through observation of system activities and energy consumption measurements, this work estimates the energy consumption of a sleepy network system in a star topology and compares it with the same system operating under ContikiMAC and 6TiSCH.

Secondly, based on scheduling experiments in a tree topology network, this study determined the minimum time required for nodes to complete network maintenance and data transmission after wake-up. Furthermore, in this thesis, an optimised scheme for communication window is proposed, and a corresponding energy consumption model is established to evaluate system energy consumption and the impact of other factors on it (such as changes in communication rate, packet retransmissions, and increases in routing depth).

Table of Contents

List of Tables

List of Figures

Research Thesis: Declaration of Authorship

Print name: Jiawei Lu

Title of thesis: A Low Power Sleepy Network Design for IPv6-based Environmental IoT Systems

I declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University.
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as:

- Martinez, K., Lu, J., Kendall, S., Hart, J.K.: Testing Low Power IP-Based Sensor Networks a Forest Environment. In: AGU Fall Meeting Abstracts, pp. IN025-003. 2020.
- Lu, J., Martinez, K., Weddell, A. (2025). Understanding Energy Consumption in Real-World E-IoT Systems with CoAP, RPL, and 6LoWPAN. In: Bradford, P.G., Gadsden, S.A., Koul, S.K., Ghatak, K.P. (eds) Proceedings of IEMTRONICS 2024. IEMTRONICS 2024. Lecture Notes in Electrical Engineering, vol 1229. Springer, Singapore. https://doi.org/10.1007/978-981-97-4780-1_14

Signature: .... Date: Mar 2026

List of Acronyms

6LoWPAN	Low-Power Wireless Personal Area Networks based on IPv6
6P	IPv6 over IEEE 802.15.4e TSCH Operation Sublayer Protocol
6TiSCH	IPv6 over the Time Slotted Channel Hopping mode of IEEE802.15.4e
6top	IPv6 over IEEE 802.15.4e TSCH Operation Sublayer
ACK	acknowledgement
AMQP	Advanced Message Queuing Protocol
APIs	Application program interfaces
ASK	Amplitude Shift Keying
ASN	Absolute Slot Number
AUV	Autonomous Underwater Vehicle
BE	Back-off Exponential
BR	Border Router
BPSK	Binary Phase Shift Keying
CBOR	Concise Binary Object Representation
CCAs	Clear Channel Assessments
CoAP	Constrained Application Protocol
CON-CoAP	Confirmable CoAP
CSMA/CA	Carrier-sense multiple access with collision avoidance
D2D	Device-to-device
DAD	Duplicate Address Detection
DAO	DODAG Advertisement Object
DAO-ACK	DAO Acknowledgment
DDS RTPS	Distributed Data Service Real Time Publish and Subscribe
DHCPv6	Dynamic Host Configuration Protocol for IPv6
DIO	DODAG Information Object
DIS	DODAG Information Solicitation

List of Acronyms

DODAG	Destination Oriented Directed Acyclic Graph
EB	Enhanced Beacon
E-IoT	Environmental Internet of Things
ESN	Environment Sensor Network
ETX	Expected Transmission Count
EUI-64	64-bit Extended Format
FCS	Frame check sequence
FDMA	Frequency Division Multiple Access
FTSP	Flooding Time Synchronisation Protocol
GNRC	Generic Network
GSN	Global Seismographic Network
ICMPV6	The Internet Control Message Protocol for IPv6
IEFT	Internet Engineering Task Force
IFS	Interframe spacing
IIoT	Industrial Internet of Things
IoT	Internet of Things
IPHC	IPv6 Header Compression
ISM	Industrial, Scientific and Medical
JSON	JavaScript Object Notation
LIFS	Longer Interframe spacing
LLN	Low-power lossy networks
LoRaWAN	Long Range Wide Area Network
LPL	Low-power listening
LPWAN	Low-Power Wide Area Network
LR-WPAN	Low-rate wireless personal area network
LTE	Long Term Evolution
LWMAC	Light-weight MAC protocol
MAC	Medium Access Control

MAC-ACK	MAC acknowledgement
MCU	Microcontroller Unit
MHR	MAC header
MQTT	Message Queue Telemetry Transport
MRHOF	Minimum Rank with Hysteresis OF
MSF	Minimizing Scheduling Function
MTU	Maximum Transmission Unit
NA	Neighbour Advancement
NB-IoT	Narrowband IoT
ND	Neighbour Discovery
NDP	Neighbour Discovery Protocol
NS	Neighbour Solicitation
O-QPSK	Offset Quadrature Phase Shift Keying
OF	Objective Function
OF0	OF Zero
OSs	Operating systems
PDU	Protocol Data Unit
PHY	Physical layer
PSDU	PHY Service Data Unit
RA	Router Advancement
RAM	Random Access Memory
RDC	Radio Duty Cycle
ROM	Read Only Memory
RPL	IPv6 Routing Protocol for Low-Power and Lossy Networks
RS	Router Solicitation
RSSI	radio signal strength indicator
RT	Real-time
RTC	Real Time Clock

SAM R30 Xpro	SAM R30 Xplained Pro
SCHC	Static context header compression
SF	Scheduling Function
SHR	Synchronisation header
SIP	Session Initiation Protocol
SLAAC	Stateless Address Auto Configuration
SoC	System-on-chip
TCP/IP	Transmission Control Protocol/Internet Protocol
TDMA	Time Division Multiple Access
TLS	Transport Layer Security
TSCH	Time Slotted Channel Hopping
WPANs	low-power wireless personal area networks
WSN	Wireless Sensor Network
XMPP	Extensible Messaging and Presence Protocol

Acknowledgement

The years of PhD study passed in what now feels like an instant. This journey was filled with challenges and hard work, as well as moments of coincidence and chance. Along the way, one was required to transform from the inside out. At this moment, I have come to understand that while achievements are transient, the process itself is deeply meaningful, and that being ordinary does not mean being small.

I am deeply grateful to my parents, Jun Lu and Wenli Xu, for their great support and selfless devotion. I sincerely thank my supervisors, Kirk Martinez and Alex Weddell, for their guidance, patience, and insight. I am also grateful to my partner, Xiaobing Wang, for her unwavering encouragement and support. I heartily thank all those who have offered help, support and encouragement along this journey.

Chapter 1 Introduction

Over the past few decades, the development of the Internet and communication technologies has shifted from linking people to each other and online Web content toward the development of interconnected systems that involve massive numbers of physical devices. This shift has resulted in a new technological concept, the Internet of Things (IoT) [1, 2]. By integrating sensing, computation, and wireless communication capabilities, IoT enables physical objects to interact autonomously with their surrounding environments. Among IoT applications, environmental monitoring is an important field. IoT-based environmental monitoring systems can be deployed with large numbers of sensors construct a large-scale system. Such systems face several key challenges, including energy consumption, reliability and robustness, and ease of maintenance [2].

Advances in sensor miniaturisation and wireless connectivity have enabled the rapid deployment of Wireless Sensor Networks (WSNs), which provide a mature and effective solution for automated environmental data collection. When deployed in unpopulated or harsh environments, WSNs must be designed with consideration for adaptability and long-term reliability within the given environment. To address these challenges, Martinez et al. [3, 4] proposed the notion of Environment Sensor Network (ESN), a subtype of WSN specifically designed to withstand natural environmental conditions and support large-scale environmental. Furthermore, the development of low-power communication protocols and IP-based standards for sensor networks has further enabled Internet connectivity and interoperability of ESNs, making the ESN an open, heterogeneous sensor network compatible with various sensor nodes. Finally, the Environmental IoT (E-IoT) [5] emerges while gaining from standardisation, which is often lacking in ESNs.

1.1 Related Work and Current Challenge

In remote or hazardous environments, E-IoT nodes are required to operate for long periods under resource-constrained conditions, especially for energy. In such application scenarios, it is not always necessary for the network to continuously maintain reachability. However, how to minimize system energy consumption while satisfying basic data collection and network maintenance requirements remains critical. Therefore, when designing such systems, it is essential to focus not only on the network architecture and node distribution, but also on the energy consumption of the system. Based on this, a wide variety of energy-saving techniques have been proposed and applied to ESNs, including optimizing radio mechanisms, low-power-sleep/wake-up switching, energy harvesting, energy efficient routing and network architecture, and data reduction mechanisms [6]. While these approaches can improve energy efficiency, their advantages and limitations must be considered in overall system design, as discussed in Sections 2.6 and 2.7.4.

Generally, compared to sleep states and computational energy consumption, communication is a major source of energy consumption for nodes in WSNs [7, 8]. It has been identified that the energy consumption in the communication mode is much higher than the energy consumption in the sleep mode. The focus of radio optimisation techniques is to reduce the energy cost of radio transmissions [9-11], but not reduce the energy consumption by unnecessary radio activation. Similarly, through energy harvesting can provide nodes with the ability to harvest energy and extend their lifetime [12-14], but it does not reduce the overall energy consumption of the system.

In term of data-driven energy saving approaches, such as data reduction/compression [15-17], data aggregation [18], and data prediction [19], aim to improve communication efficiency and reduce communication energy costs by reducing the volume or frequency of data transmission. However, these technologies may also introduce additional complexity and cannot fully guarantee the accuracy of information [19, 20]. Event-triggered communication mechanisms further reduce communication demand by allowing transmissions only when sensed environmental changes exceed predefined thresholds, and have been widely applied in long-term disaster monitoring and early warning systems [21, 22]. However, in multi-hop event-driven networks, relay nodes typically require maintaining awake or frequently wake up to receive potential forwarding traffic. This continuous or periodic switching on of radio transceiver causes additional energy cost.

Although data-driven, event-triggered and energy harvesting technologies can significantly extend network lifetime, they primarily focus on enhancing transmission efficiency or energy supply rather than fundamentally reducing communication related network activity. This work considers sleep/wakeup approaches to be essential for the development of wireless sensor network systems, as it can efficiently limit the activation duration of communications. Existing classifications of sleep/wake-up schemes and duty-cycling mechanisms have been reviewed in the literature [6, 23]. However, commonly adopted asynchronous and synchronous duty-cycling protocols, such as ContikiMAC [24] and 6TiSCH [26], still require frequent radio activation for activities such as low-power listening, data transmission, and network maintenance, leading to additional energy consumption that cannot be further restricted after deployment [6].

Under this context, the design of schedule-based sleepy networks provides an approach, enabling all nodes (including child and parents nodes) to remain unreachable for extended periods of time and only carry out data transmission and network maintenance within predefined communication windows. Although such designs may introduce additional communication delays and may not adequately handle data bursts, sleepy networks are particularly suitable for long-term environmental monitoring in remote or hazardous area. In these environments, reducing the system's overall energy consumption and extending the network's lifetime can be more important than maintaining continuous network accessibility. However, existing studies have not clearly established whether sleepy networks with restricted communication windows can achieve higher energy efficiency than ContikiMAC- or 6TiSCH-based systems.

At the network layer, the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [25] is widely applied in constrained networks, and RPL-based routing protocols have been comprehensively surveyed in [26]. Although lost of enhancements to RPL have been proposed, including energy-aware objective functions [27-29] and QoS-oriented improvements [30, 31], these improvements often come at the expense of generality. Considering common applicability and generality, it remains unclear how a sleepy network using native RPL, without modifying default protocol parameters, can achieve improved energy efficiency. In particular, the minimum communication window duration required to maintain valid routing while minimising energy consumption has not been investigated.

Meanwhile, in terms of system energy analysis, simulation-based approaches have been an important way to analyse the performance of WSN systems. As an example, in the study of [32], WSN system performance was modelled and network lifetime was predicted by analytical modelling and simulation methods. While simulation can provide flexibility and usability, simulators that rely on simplified real-world systems can lead to inaccurate results due to the inherent complexity and diversity of WSNs [32]. Thus, assumptions must be made to accurately model such systems, and these assumptions need to be as realistic as possible to enhance the accuracy of the results. Currently, there are no studies on minimum communication windows for energy-constrained long-term environmental monitoring in sleepy networks with RPL. In addition, no studies have been conducted to estimate the energy consumption of such systems with optimised communication window lengths based on practical measurements.

1.2 Research Questions

1.2.1 Can a schedule-based sleepy network reduce overall energy consumption compared to duty cycling MAC approaches?

The duty cycling scheme reduces radio power consumption by periodically switching the transceiver between active and sleep states. Asynchronous schemes such as ContikiMAC can effectively reduce the duration of radio transceiver activation, though periodic channel assessments introduce additional overhead. Synchronous approaches like 6TiSCH can efficiently minimise idle listening periods and lower energy consumption, yet they cannot constrain network maintenance activities controlled and organised by other layer protocols (such as routing and neighbour information maintenance). Furthermore, radio activation within duty cycling schemes can support mechanisms such as low-power listening, data transmission, and protocol-controlled network maintenance (routing updates and neighbour discovery). These activities continuously consume system energy and cannot be further restricted once deployed. Unlike duty cycling MAC protocols that rely on frequent radio activation to perform channel assessment and maintain network availability, the sleepy network completes essential communications only within limited communication windows. Sleepy networks can effectively limit the activity time of transceivers, but at the cost of increased communication latency and reduced network reachability. However, the impact of limiting the communication time of a sleepy network on energy consumption and network maintenance still needs to be analysed experimentally.

The hypothesis *for this research question: For long-term operating and energy-constrained E-IoT networks, compared to duty cycling methods (specifically ContikiMAC and 6TiSCH), sleepy networks can achieve lower overall energy consumption by restricting all communication and network maintenance activities within predefined communication windows.*

To investigate this hypothesis, for sleepy network system, this study measured and characterised the actual energy consumption of different activities in various states (e.g., sleep, sensing, and communication). Then it estimated the energy consumption of same system under the conditions that applying duty cycling schemes (ContikiMAC and 6TiSCH). Finally, this work illustrated design details of the sleepy network and the system energy stack.

1.2.2 How can energy consumption be reduced by optimising communication windows within a tree topology? and what is the resulting energy consumption model?

While extending sleep duration is an effective means of reducing energy consumption, in multi-hop networks it introduces additional overhead associated with network maintenance, routing updates, and forwarding responsibilities after nodes wake up. Through analysis of deployment experiences in the literature, this thesis has identified some challenges introduced by sleepy network mechanisms in Section 2.7 and these can be summarised as follows:

1) In sleepy networks, communication windows may be over-allocated to address uncertainties arising from contention, network maintenance, and data exchange delays.
2) If numbers of nodes simultaneously wake up and attempt to rebuild the network and transmit data at the start of a communication window, this may cause radio conflicts, retransmissions, and packet loss, leading to energy waste.

3) Conversely, if a request-based communication mode is applied, where parent nodes or base stations query child nodes for data during wake-up periods, certain child nodes may remain radio-active for extended periods awaiting requests.

In a tree topology, the above issues are further amplified because parent nodes must carry the data forwarding responsibilities of their dependent child nodes.

The hypothesis *for this research question: By optimising the communication window design under different wake-up strategies in sleepy networks (sequential and simultaneous wake-up respectively), network maintenance and data transmission can be restricted to a minimal but sufficient communication window which can further reduce the overall energy consumption.*

This thesis investigates the minimum time required for network maintenance, routing information updates, and data transmission after node wake-up, and develops communication window optimisation solutions for different wake-up strategies within tree topologies. Based on the optimised scheme, this paper derives an energy consumption model to analyse the impact of communication rates, transmission success rate, and topological complexity on system energy consumption in a sleepy network with a tree topology. In this work, the data delivery rate is defined as the ratio of the number of successfully transmitted data packets to the total number of packets intended for transmission. Meanwhile, the success rate is defined as the ratio of successfully delivered packets to the total number of transmission attempts, including retransmissions.

1.3 Thesis Structure

This thesis is divided into five chapters, each chapter will describe and analyse some aspects of the research problems. This thesis focuses on understanding and optimising low power sleepy network scheduling for IPv6-based environmental IoT systems.

Chapter 2: Literature Review

This chapter first overviews the developments of IoT and E-IoT, subsequently detailing the network protocols and standards commonly applied across the physical layer to application layer within E-IoT systems. It further reviews operating systems widely discussed in WSNs and analyses the design of monitoring systems deployed in different environments. Finally, it discusses existing energy saving techniques and the design of low-power sleepy networks, focusing on energy saving, design trade-offs, and protocol selection.

Chapter 3: Design and Implementation of Sleepy Networks in an E-IoT System

This chapter designs a sleepy network based on a star topology using standardised low-power protocols and a schedule-based sleep/wake-up mechanism. The current consumption and duration of various network activities in a star topology, including read sensor, store data, CoAP transmission, and network maintenance (ICMPv6, RPL, 6LoWPAN control messages) are measured in the deployed a test scenario. In addition, the system applied IEEE 802.15.4 MAC, 6LoWPAN, RPL and CoAP, and this chapter also estimates the energy consumption of system applying MAC protocols such as IEEE 802.15.4, 6TiSCH and ContikiMAC based on the current consumption measurements. The experimental results show that the network maintenance activities have some regular pattern, and the optimisation of the communication window scheduling in the sleepy network can effectively reduce the system energy consumption.

Chapter 4: Optimization and Energy Profiling of Low Power Sleepy Network Design for Environmental IoT System

Using a tree topology, this work further investigates the radio activity patterns of nodes that wake up after a sleep duration longer than the default RPL lifetime (3 minutes), focusing on the characteristics of the neighbour discovery and routing updates activities. Through experiments and scheduling control, this chapter proposes an optimisation solution for the communication window under different wake-up strategies (sequential and simultaneous). This work also analyses the time cost of different types of radio activities and models the energy consumption of the system applying the optimised communication window solution. Finally, this chapter presents a methodology for predicting the impact of different transmission rates and topological complexity on the system's energy consumption. These evaluations can help with energy consumption prediction during the development of similar sleepy networks.

Chapter 5: Conclusion and Future Work

The final chapter summarised the contributions of this thesis and the limitation of this study, while also providing suggestions for possible future research directions as well as new opportunities. This chapter summarised the design of a low-power sleepy network for IoT systems over IPv6 and highlights the possible impact of the optimisation strategy for the sleepy network scheduling scheme.

Chapter 2 Literature Review

This chapter reviews the development history of ESN and E-IoT, and presents the relevant network protocols, standards, and OSs relevant to those concepts. This chapter also reviews examples of practical ESN deployments in diverse environments, analysing and summarising common design considerations including data sampling, communication, and energy-saving strategies. Based on the above observations, this chapter provides a detailed review of the energy-saving technologies involved in relevant deployments. It also summarises the design concepts of sleepy networks for E-IoT systems and the differences between it and traditional duty-cycling based approaches.

2.1 The Internet of Things

The concept of the Internet of Things refers to the integration of the internet into physical objects, enabling such objects to become addressable entities within the internet. Typical IoT devices integrate environmental sensing, embedded processing, and communication capabilities, enabling devices to continuously sense the physical world and automatically respond to environmental changes [33]. Accordingly, the Internet of Things emphasises the continuous access to information of the environment through the integration of physical objects with the internet and improving remote interaction between users and these objects.

As early as 1999, Kevin Ashton coined the term "Internet of Things" [1, 2]. Several well-known attempts in the complex real-world environment, such as smart city, has shown people a fascinating vision of the future smart world. Smart City improves the ability of the city to provide reliable services to citizens in the face of various complicated problems such as congestion and injustice [34]. As an example, the processing smart city project, #SmartME, aims to transfer Messina municipality (Italy) into a smart city to enhance interaction between citizens and government by sharing/analysing every valuable physical object based on one large Open Data Platform [35-37]. A. Bassi et al. have listed potential future applications on utilizing IoT technology [38], for example, intelligent logistics, the smart home, smart factory, and E-healthcare (telecommunications operators as O2 UK invested in advance on the relevant industry) [39].

2.2 Environmental Sensor Networks and Environmental Internet of Things

Wireless sensor networks (WSNs) are a special low-power personal area networks and it are also important component of IoT [40, 41], and It consists of many low power sensor nodes capable of sensing, processing and wireless communication. Since nodes have limited resources, especially power resources, the design priority of such systems is to balance the energy efficiency and extend the lifetime of nodes. Additionally, as network scale increases, system performance is further constrained by data collection and aggregation capabilities at sink nodes [42].

Some studies focused on the simulation of the performance of WSN such as [43] have contributed to the development of sensor network research and encouraged researchers to deploy systems in the real world (e.g. [44]) to identify the actual performance compared with simulation results. However, for unpopulated/geohazard-prone areas, designing real-world WSN systems require high reliability and adaptivity. Such WSN systems deployed in hazardous or remote environments need to pre-consider the possible challenges such as communication interference from terrain, lack of maintenance and resources limitations (e.g. power supply).

Therefore, when applying a WSN to monitor the environment in a target remote/hazardous area, the design of the WSN needs to consider the specificity of the deployment environment and the detailed needs of monitoring objectives. Martinez et al. [6] proposed the notion of Environmental Sensor Networks (ESNs) which is a subtype of WSNs specifically designed to carry out continuous sensing of the natural environment. ESN was proposed to monitor and understand the environment, which requires system design to consider environmental challenges and deployment details [3]. To enable ESNs to become a more open and heterogeneous network, various low-power protocols and IP-based standards need to be considered for integrating into the network design to achieve fuller Internet connectivity. Low-power protocols standardised for sensor networks and IP-based standards have been summarised by Nikoukar et al. [45]. Additionally, ESN emphasizes utilizing the large-scale deployed sensors to collect environmental data near real-time to format a global environmental monitoring system that can help researchers to investigate, model, and predict the ecological changes of hazardous regions [46]. Digital Earth [47] is an example of globalized monitoring system notion. For a globalized environment researching and monitoring, the ESN system can integrate various sensor data and different distributed sensing systems for systematic analysis and multiple access. Hart and Martinez [46] and Stacey and Berry [48] indicated two fundamental resources for earth system science for environmental analysis: remote sensing system and earth observation system. Furthermore, ESN should consider integrating into the environment to reducing the uncertain influence (such as ecological changes and pollution) caused by system deployment and system deployment should be limited in size and lightweight [46].

A number of ESN systems have already been widely deployed in the real world for applications such as water level monitoring [49, 50], target tracking[51], and landslide warning [52]. As another examples, the Mountain Sensing [53, 54] and Glacsweb [55, 56] are two implemented ESN projects for monitoring environmental changes in a remote and hazardous environment. Those real-world implementations have allowed the system to update/access data periodically and require node and whole system to be energy efficient by turning off the radio for most of the time as the battery replacement in those scenarios is quite challenging [57]. To further reduce energy consumption, the Glacsweb project designed a sleepy network using custom protocols (e.g. communication protocol) that requires the system to turn into a deep sleep mode most of the time and wake up to communicate in a fixed time window daily.

The concept of the Environmental Internet of Things (E-IoT) [5] has led to network designs that allow interconnection of cross-vendor devices and integrate with the Internet to enhance interoperability. Hart and Martinez [46] have comprehensively introduced and discussed all the aspects of the E-IoT. To withstand various real-world environmental challenges when designing networks, there are some aspects that should be considered such as the collaboration of different low-power protocols and technologies. E-IoT should be a heterogeneous system allowing the integration of various sensor types and cooperation on lower power communication standards such as 6LoWPAN and CoAP, and the system should be Internet-accessible [46].

When developing remote E-IoT systems, it is critical to apply various low-power protocols and efficient system energy saving strategies to improve system lifetime and reliability [3][6]. Meanwhile, seamless access to the Internet should also be considered for obtaining RT information. In addition, sensor nodes can apply certain data optimization strategies (optimizing the data format), such as autonomy in deciding or filtering the data, to transmit as much data as possible and reduce the power consumption to a certain extent. Related literatures have introduced and summarised the application of data reduction model in WSN [19, 58]. Additionally, system energy consumption can be reduced by efficiently scheduling tasks to operate and switching the node between low-power mode and operating mode. Specifically, communication module can switch to low power listening mode or be completely turned off to save energy when there is no further requirement for data transmission. Also, processing unit can be scheduled to entry sleep/deep sleep mode to save power.

2.3 Network Protocols and Standards in E-IoT

Low-power Lossy Networks (LLNs) often consisting of devices constrained by limited resources such as power and memory size. ESNs is also an LLN that are specialized in collecting, transmitting, and processing data from environment. The deployment scale of nodes within such networks is usually considerable, with limited resources (particularly energy). To support long-term environmental monitoring and internet connectivity under these constraints, consideration should be given to efficient network management, communication protocols, and network architecture.

Traditional ESNs gather the detected data to the server for the user without the direct accessibility from the user side to the remote nodes. The main differences between ESN and E-IoT systems are that E-IoT emerges while benefiting from standardisation which is often lacking in ESNs, and E-IoT allows bidirectional communication where users can access data through the Internet [57]. Trading off the maintaining cost, deploying cost and continuous accessibility of the system has promoted the development of low power networking development, which enable nodes to have a longer life-time. As an example, real-world applied projects like Glacsweb [55] can tolerate network outage for months, reminding the researchers to pay close attention to the low power design of the system.

From the application layer to the physical layer, various low-power protocols and standards such as CoAP, 6LoWPAN, ZigBee, LoraWAN can be applied to construct the LLNs stack. Figure 2.1 shows the protocol stack for low-power networks in IoT. For achieving minimized maintenance, simple data interoperability, and a longer system lifetime for ESN, the system design should consider low power and IP-based protocols and standards such as 6LoWPAN, RPL, and CoAP. The participation of those protocols makes the ESNs can evolve in a more standardised system which is E-IoT. This section reviewed low-power network protocols and standards, such as 6LoWPAN, RPL, CoAP, ZigBee [59], and LoraWAN [60]. Additionally, MAC layer mechanisms specifically designed for energy-efficient operation were introduced and discussed, including ContikiMAC [24] and IEEE 802.15.4e TSCH (6TiSCH) [61]. Besides, the selection of protocols at different layers of the network stack for the system considered in this work, including a comparative analysis of existing technologies and their respective advantages and limitations, will be discussed in section 2.7.

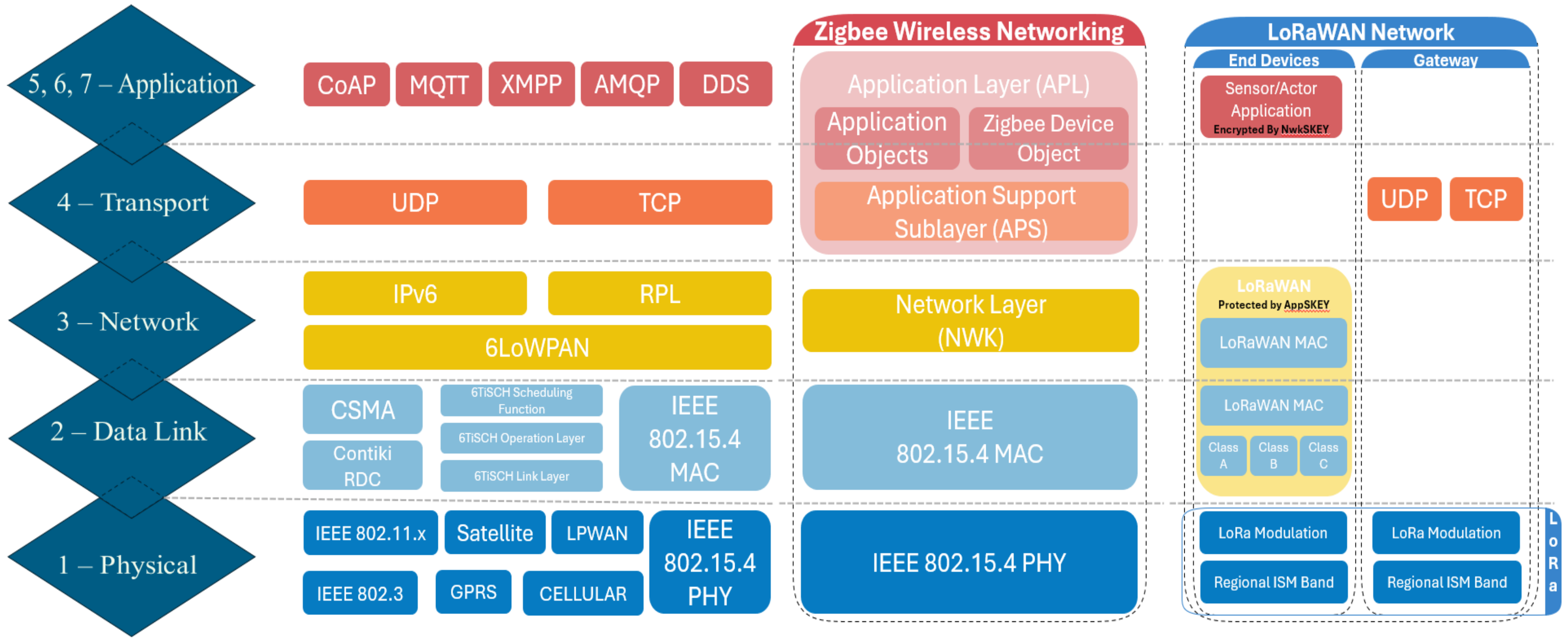


Figure 2.1. Protocol Stack for Low-Power Networks in IoT (part of the figure is adapted from [62] and [60]).

2.3.1 Physical Layer

As a radio protocol standard, IEEE 802.15.4 has standardised the regulation of the physical layer and media access layer which aims to enable devices to send data with low data rate (ranging from 10 kbps to 1Mbps) [63]. IEEE 802.15.4 physical layer has defined the regulation of the radio transceiver operation in unlicenced frequency band including 868/915 and 2450 MHz band. To accommodate unlicensed frequency bands in various regions and different modulations into the standard, IEEE 802.15.4 has defined the channel numbers and channel page to correspond to relevant frequency band with specific modulation method [64]. Commonly, radio frequency used in WSN is based on Industrial, Scientific and Medical (ISM) band with 2 options: 2.4 GHz and sub-GHz [65]. For the sub-GHz band, popular frequency applied in transceiver radio are: 915 MHz (North America) [66], 868 MHz (Europe) [54], 433 MHz [67], 315 MHz [68], and 173 MHz [69]. Typically, lower-frequency radio signals experience lower path loss and thus support longer transmission distances. However, such benefits come at the expense of lower data rates. Also, considering the Fresnel Zone, the Fresnel zone must be relatively clear to avoid interference, especially zone 1. For the same transmission distance, lower frequencies result in wider Fresnel zones than higher frequencies, which also means that lower frequency radios are more sensitive to the obstruction of terrain, plants, or buildings in the area. To balance the communication range, data rate and the influence of the obstructions, this work will use the 868 MHz frequency band (European area), of which the centre frequency is 868.3 MHz with channel number equal to 0.

2.3.2 Link Layer Protocols

This subsection will discuss IEEE 802.15.4 MAC, ContikiMAC (RDC layer with CSMA as MAC protocol) and 6TiSCH stack (6TiSCH working with 6top (6P) and 6TiSCH MSF together). Specifically, ContikiMAC is designed as an asynchronous protocol. Generally, the asynchronous mechanism may increase the probability of collision thus reduce the packet delivery rate [70] [71].

2.3.2.1 IEEE 802.15.4 MAC

IEEE 802.15.4 MAC is the underlying MAC specification for devices to run in low-power mode to save power [72]. By using the CSMA/CA as the MAC mechanism, the IEEE 802.15.4 protocol [73] was proposed with energy-efficient options but still comes with limitations in the nature of MAC [74]. Existing research indicates that both the reliability and packet delay of the IEEE 802.15.4 MAC layer can be further improved [75].

Unslotted (nonbeacon-enabled) and slotted (beacon-enabled) CSAM/CA are the two channel access mechanism types used in IEEE 802.15.4 low-rate wireless personal area network (LR-WPAN).

According to IEEE 802.15.4-2011 [76], devices should to wait a number of random backoff slots before transmit the data, and also need to wait for a recalculated random backoff slots if the channel is not idle. The CSMA/CA algorithm is illustrated in Figure 2.2. Back-off Exponential (BE) is defined to calculate the random number of backoff period units that device needs to wait before transmission. If the maximum number of fallbacks allowed has not been reached, the transmission is returned as successful, otherwise the transmission fails. Equation 2.1 below can be used to calculate the random BE period (the default minimum BE is 3, maximum BE is 5, unit of backoff period is 20 durations of symbol period) [76]:

$$BE\ period = random(0, 2^{BE} - 1) * Unit\ of\ backoff\ period \qquad 2.1$$

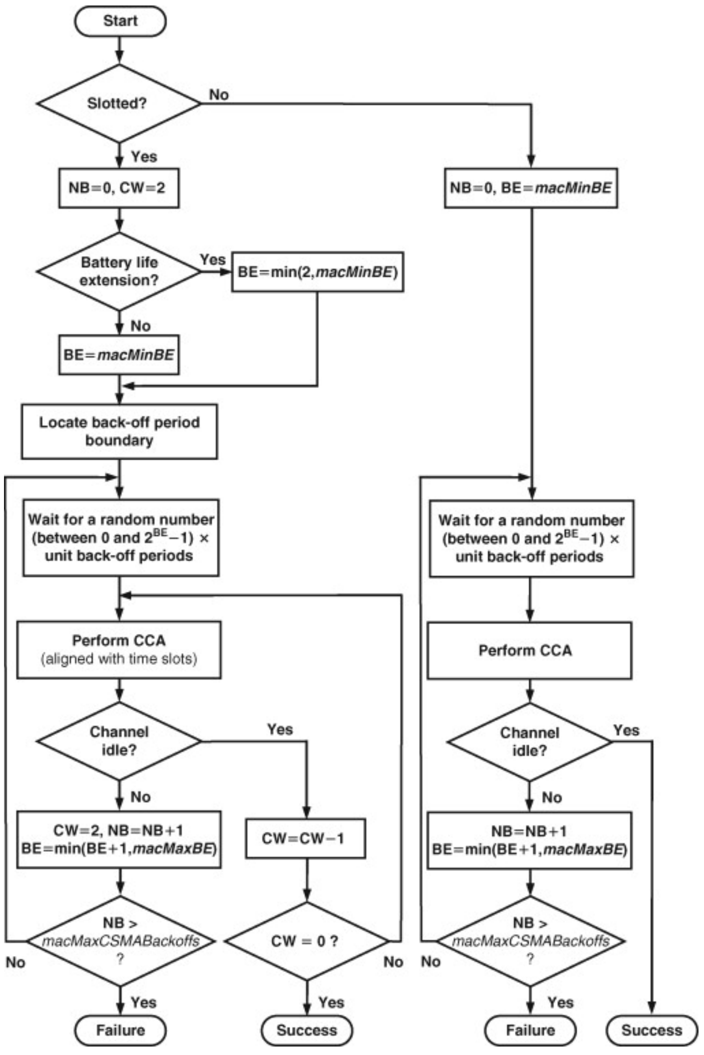

Start
Slotted?
No
Yes
NB=0, CW=2
NB=0, BE=macMinBE
Battery life extension?
Yes
BE=min(2,macMinBE)
No
BE=macMinBE
Locate back-off period boundary
Wait for a random number (between 0 and 2^BE−1) × unit back-off periods
Wait for a random number (between 0 and 2^BE−1) × unit back-off periods
Perform CCA (aligned with time slots)
Perform CCA
Channel idle?
Yes
No
Channel idle?
Yes
No
CW=2, NB=NB+1 BE=min(BE+1,macMaxBE)
CW=CW−1
NB=NB+1 BE=min(BE+1,macMaxBE)
NB > macMaxCSMABackoffs ?
No
Yes
CW = 0 ?
No
Yes
NB > macMaxCSMABackoffs ?
No
Yes
Failure
Success
Failure
Success

Figure 2.2. CSMA/CA Algorithm: the number of back-offs (NB) and the contention window (CW) length (Source: [77] – permitted by Elsevier, licence No. 6012100082438 (Apr 18, 2025)).

2.3.2.2 6TiSCH

In order to improve MAC performance in dense deployment scenarios (e.g., industrial applications) and real-time performance, IEEE 802.15.4 was extended to form IEEE 802.15.4e TSCH [78, 79]. The concluded limitations of IEEE 802.15.4 are mainly two aspects: the insufficient performance reliability caused by the application of CSMA/CA and the shortage at collision avoiding problem.

6TiSCH is a standardised protocol based on TSCH. The stack architecture adds a sublayer named 6TiSCH Operational (6top) between 6LoWPAN and IEEE 802.15.4 MAC (TSCH MAC) [80]. 6TiSCH supports multi-hopping topology by utilizing RPL and enables IPv6 through 6LoWPAN [81]. A light-weight secure joining process in 6TiSCH is designed by integrating the link-layer security mechanisms with a secure joining protocol using the CoAP [81]. Time Division Multiple Access (TDMA) and Frequency Division Multiple Access (FDMA) have provided the mechanism for supporting timeslot division and frequency slot division. By combining and referencing TDMA and FDMA, the TSCH has enabled channel hopping for enhancing reliability and time synchronisation (both frame-based and ACK-based synchronisation are supported) for networking. With the development of wireless control and increasing accuracy requirement of interoperability for industrial IoT, 6TiSCH has been proposed to enable seamless access between devices and the Internet. The Enhanced Beacon (EB) is designed as a MAC control message for nodes synchronisation which contains the TSCH parameters (Absolute Slot Number (ASN), slotframe size and channel number, etc.) and sends in the specific cell between the source node and the destination node. Transmission between nodes is scheduled and reserved in a specific cell (slot_offset, channel_offset) in one single slotframe, which is computed by channeloffset value (which is equal to the row position of TSCH schedule in a slotframe and the maximum value is equal to the available number of frequencies, 16 in 2.4 GHz band) and slotoffset (which is equal to the column position of TSCH schedule in a slotframe and the maximum value is 101 slots in this work with 10ms for each slot) [82].

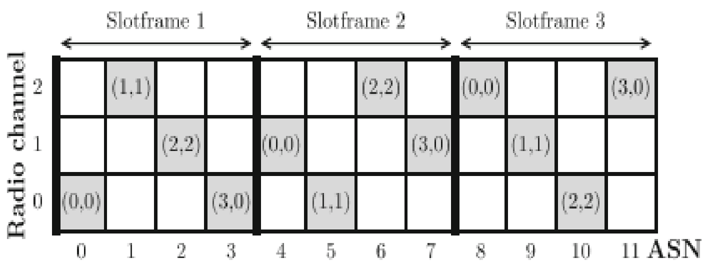


Figure 2.3. Fundamental TSCH Schedule example [82].

Figure 2.3 illustrates the fundamental TSCH Schedule example. The cell can reserve a node for sleeping and listening and make it critical to make the number of cells stay minimum. 6TiSCH has defined the schedule regulation/component, optimised schedule for other protocols to refer (such as routing protocol), and communication security function [83].

2.3.2.3 ContikiMAC

For achieving the power-efficient of low-power networking, the radio should be turned off as long as possible without affecting the packet exchanges. However, nodes in sleep mode can affect synchronising and packet exchanges. Thus, asynchronous MAC mechanism can potentially be an efficient solution for MAC control in low-power networking. ContikiMAC, as an asynchronous mechanism, is designed for easy implementation for low power devices to avoiding unnecessary radio power consumption by applying timing and fast sleep optimization [24]. ContikiMAC are designed to optimise the power consumption further. ContikiMAC uses a Radio Duty Cycle (RDC) and CSMA [24]. It applies a duty cycling mechanism (8 Hz by default), waking up nodes for transmission after two successive Clear Channel Assessments (CCAs) are identified [24]. The Mountain Sensing [54] project used ContikiMAC as it saves power while making the network appear to be always alive.

As a duty cycling mechanism, it contains two driver interfaces, RDC driver interface and CSMA driver interface. Two RDC protocols are provided in ContikiMAC are low-power probing protocol and LPL protocol. Compared with another duty cycling mechanism, X-MAC, ContikiMAC spends 1.78% time on average for listening, receiving, and transmitting packets, while X-MAC requires 7.83% of the total time when RPL is applied [84]. CSMA is used as the MAC layer, and the RDC layer is used to control the radio [85] . The callback function would be used to control a message whether it should be deleted from the queue as successfully transmitted or controlled by CSMA to indicate the packet should be retransmitted again. Packet sending is repeatedly done, and the receiver will wake up and start receiving the package after two successive CCAs are identified as negative (incoming packet existed) according to the radio signal strength indicator (RSSI) thresholds [84]. The receiver will execute fast sleep action if CCA is identified as positive (RSSI lower than decided thresholds) with no incoming packet in the space to save power. Figure 2.4 illustrates the data receiving after waking up with two CCAs be identified as negative and ACK transmission procedure in ContikiMAC.

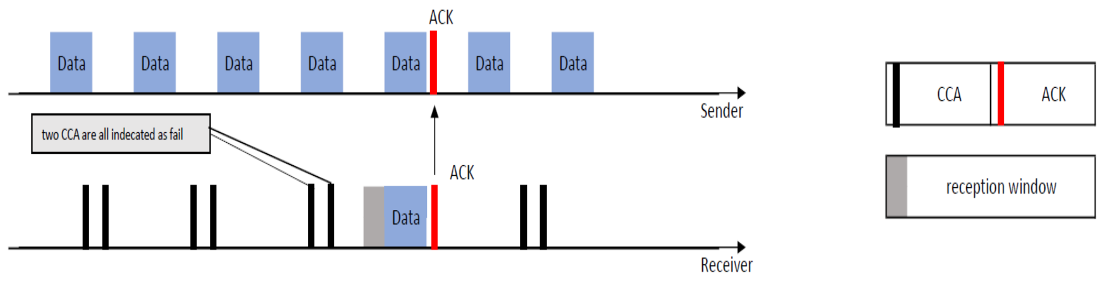


Figure 2.4. Data Receiving and ACK Transmission Procedure in ContikiMAC (adapted from [24]).

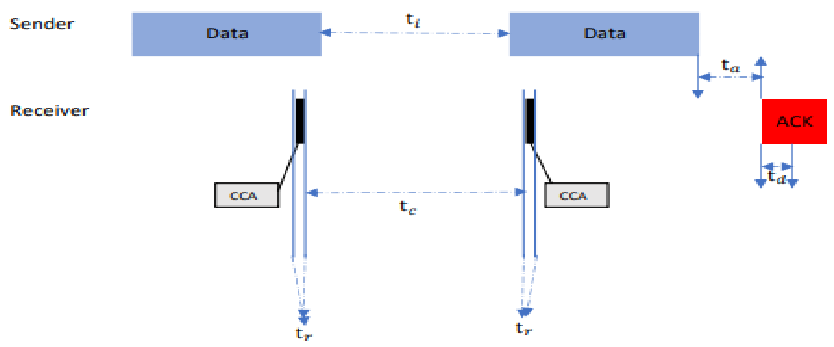


Figure 2.5. Timing Mechanism of ContikiMAC (adapted from [24] and [84]).

ContikiMAC has been designed with an advanced timing mechanism (as illustrated in Figure 2.5) to enable power-efficient functionality. Simply, there are five time parameters that should be considered for timing: time interval between data packet t_i, the time interval of CCA t_r (definition is various depend on the transceiver), the time interval between two CCAs t_c, the time between ACK and receiving a data packet t_a and time of ACK successful detecting t_d. Default value of t_d (equal to 12 symbols which are 12* and $12 * \frac{4}{250}\, ms$) and t_d (equal to $10 * \frac{4}{250}\, ms$) is 0.192 ms and 0.16 ms as defined in IEEE 802.15.4, respectively. And in Contiki 2.5, t_i is 0.4 ms, t_c is 0.5 ms and data packet length is 0.884 ms. To conclude, there are three rules that should be followed as listed below:

i. Data packet length should be bigger than $2 * t_r + t_c$ for two successively CCAs can be detected as negative (lower than RSSI threshold).
ii. Also t_c should be bigger than t_i in case that the second CCA is detected as positive.
iii. t_i should be bigger than $t_d + t_a$ to ensure that ACK can be received before the conflict with the next incoming packer at the sender side.

2.3.3 6LoWPAN

Low-Power Wireless Personal Area Networks based on IPv6 (6LoWPAN) can enable the wireless connection of all possible physical objects by applying IPv6 for addressing problems caused by limited IPv4 address pool, which has already been applied in the IP-based ESN system such as the mountain sensing project [53] [54]. 6LoWPAN is dedicated to adapting to the LLNs network, and IPv6 is proposed to release the stress of address limitation in IPv4. However, the significant difference in the size of IPv6 Maximum Transmission Unit (MTU) and IEEE 802.15.4 MTU challenges the 6LoWPAN. Thus, the working group proposed RFC 6282 designed 6LoWPAN with reliable compression ability of IPv6 Header, IPv6 Extension header, and UDP header [86]. 6LoWPAN is like an adaptation layer, by providing header compression and segmentation mechanisms, it makes IPv6 suitable for networks based on IEEE 802.15.4. Normally, lower layer MTU is smaller than IPv6 MTU (1280 bytes for IPv6 and 127 bytes/octets for IEEE 802.15.4. More importantly, the reduction of the header in the 6LoWPAN packet provides more space for payload [87].

2.3.3.1 6LoWPAN: Features and Advantages

The scalability of 6LoWPAN is one significant advantage, supporting extensive IoT deployments that can include thousands of devices. Its scalability is achieved through efficient routing protocols and the ability to segment the network using routers and border routers. By using these mechanisms, 6LoWPAN can maintain high performance and reliability even as the network size increases, making it an ideal choice for large-scale applications such as smart cities and industrial automation. Interoperability is an important feature of 6LoWPAN, particularly because of its use of IPv6. This feature ensures seamless integration with existing IP networks, allowing devices within the 6LoWPAN network to effortlessly communicate with devices on other networks. This capability helps to create large interconnected systems that can work together to enhance the overall functionality and reach of IoT solutions. The ability to interoperate with other IP-based systems is critical to the development of an integrated IoT ecosystem that can leverage existing infrastructure and technologies. In environments where bandwidth is a limited resource, the efficient use of bandwidth becomes critical. 6LoWPAN's header compression mechanism significantly reduces the overhead in data packets, freeing up more bandwidth for actual data transmission. This efficiency is particularly beneficial in scenarios where maintaining high data throughput is essential, such as in industrial monitoring and environmental sensing applications. By maximizing the use of available bandwidth, 6LoWPAN ensures that data is transmitted quickly and reliably, even under constrained conditions. Energy efficiency is an important consideration considering that many of the devices in a 6LoWPAN network are typically battery-powered. 6LoWPAN has been designed with a variety of features, including duty cycling and low-power listening, to minimise power consumption. These features help to extend the battery life of the device, ensuring that it can run for long periods of time without frequent charging or battery replacement. This energy efficiency is critical for devices deployed in remote or inaccessible areas where maintaining power is a challenge. Security is an integral aspect of 6LoWPAN, with robust mechanisms to ensure data integrity, confidentiality, and authentication. These security measures are essential to protect the network from a variety of threats, including eavesdropping, data tampering and unauthorised access.

2.3.3.2 Network Architecture and Components

The architecture of a 6LoWPAN network is designed to facilitate efficient and scalable wireless communication. It consists of three fundamental elements: the host, the 6LoWPAN router, and the 6LoWPAN border router [88]. The host is the end device that communicates within the network, performing tasks such as sensing or actuating. The 6LoWPAN router manages packet forwarding within the network and connects different network segments, ensuring that data can travel efficiently from one part of the network to another. The 6LoWPAN border router serves as a gateway between the 6LoWPAN network and other IP-based networks, enabling seamless integration and communication across different network domains. Figure 2.6 below gives detailed information about 6LoWPAN network architecture.

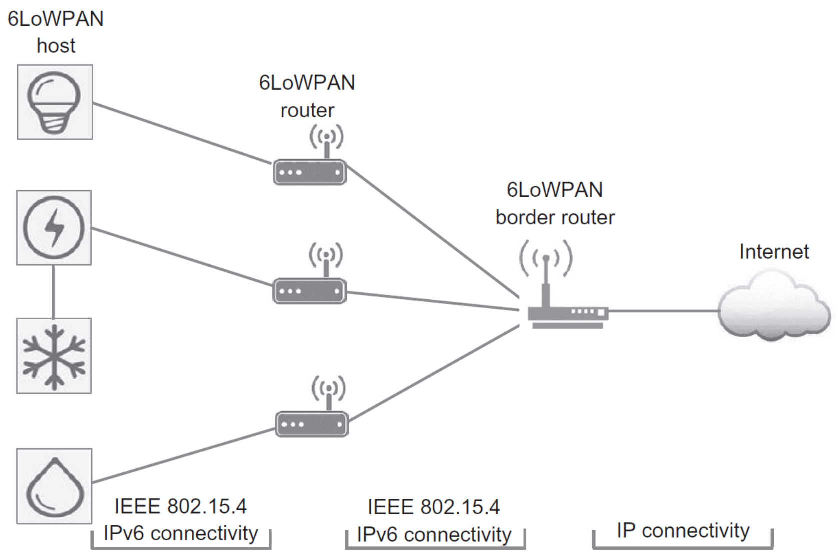


Figure 2.6. 6LoWPAN network architecture [89].

2.3.4 ZigBee, LoraWAN, NB-IoT and Mobile Network

ZigBee has already been applied in a wide range of application scenarios and it is designed to target a scenario that requires low-energy, low data throughput, and stable communication [90]. ZigBee architecture includes four layers, PHY layer, MAC layer, network layer, and application layer. network layer in ZigBee has the following functions: (a) providing routing and node addressing; (b) communicating and data exchanging with application layer; (c) network topology maintenance and (d) ensure network security [90]. IEEE 802.15.4 mainly defines PHY and MAC layer, network and application layer are designed by ZigBee standard with essential security features [77].

Low-Power Wide-Area Network (LPWAN) standards such as Long Range Wide Area Network (LoRaWAN), Sigfox and Narrowband IoT (NB-IoT) are formulated for long-distance communication with low power consumption and low data rate features and mature standards, and LoRaWAN have already been applied in the industry [91]. For example, LoRaWAN is targeting a communication range between 5-15 km with a battery lifetime up to an estimated ten years theoretically, and gateway/base station can support a massive number of items with specific topologies [91]. Furthermore, IEFT has standardised a lightweight adaption layer for LPWAN, which is called static context header compression (SCHC) and fragmentation, to enhance the adaptability of IPv6 for LPWAN under strict capacity limitations [92]. By applying SCHC [93] solution, LPWAN standards like NB-IoT, Sigfox and LoRaWAN can work over IPV6 [92]. However, unlike 6LoWPAN, nodes in LPWAN especially in Lo-RaWAN, cannot communicate with each other because of the network architecture design [89].

2.3.5 IPv6 Routing Protocol for Low-Power and Lossy Networks - RPL

The IPv6 Routing Protocol for Low-Power and Lossy Networks (LLNs), commonly known as RPL, is a specialized protocol designed to meet the routing requirements of LLNs. These networks, characterised by their low power, limited resources, and lossy communication links, are common in IoT applications. RPL is defined by RFC 6550 [25] and provides a robust framework for establishing and maintaining efficient network topologies in challenging environments.

2.3.5.1 DODAG

The key design of RPL is the Destination Oriented Directed Acyclic Graph (DODAG). The DODAG is a key component that organizes the network into a tree-like structure with a single root node. As the fundamental structure of, DODAG is essential for routing efficiency and ensuring reliable data transmission across the network. In the context of an ESN system with large-scale sensor node deployments, RPL facilitates packet hopping among nodes, constructing, and maintaining low-power routes. Referring to RFC 6550 'RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks (LLNs)' [25], RPL is explicitly designed to fulfil the routing requirement of LLNs. One important terminology is Objective Function (OF) which defines all the routing metrics, especially the method of formatting DODAG, and OF also indicates the method to select root node. There is three fundamental traffic flow method; (a) Multipoint to Point; (b) Point to Multipoint and (c) Point to Point. OF selects the optimal path defined by different routing costs and Rank value (position indicator). Besides, RPL would allow the node to communicate with surrounding nodes and select a root to join DODAG by exchanging the Internet Control Message Protocol for IPv6 (ICMPv6) messages: DODAG Advertisement Object (DAO), DODAG Information Objective (DIO), DODAG Information Solicitation (DIS) [94]. Four RPL instances introduced by RFC 6550 [25] are (a) Single DODAG with one root; (b) Multiple DODAG; (c) Single DODAG that uses a virtual root on backbone network; and (d) a combination of (a) (b) and (c) dependent on a real scenario.

OF is defined for indicating the rule of DODAG formation and decide the parent/child routing table by related objective value (Rank and Path Cost) computing. There are two different OFs: OF Zero (OF0) and Minimum Rank with Hysteresis OF (MRHOF) [95-97]. OF0 is used can determine the DODAG's parent/child set and preferred parent for each node by computing and comparing rank values. MRHOF uses the metrics of the DIO metric container (such as Expected Transmission Count

(ETX)) for finding the optimal-metric path (such as minimum-ETX) or use the metric to compute rank value instead.

2.3.5.2 Neighbour and Routing Registration Process: applying RPL in 6LoWPAN

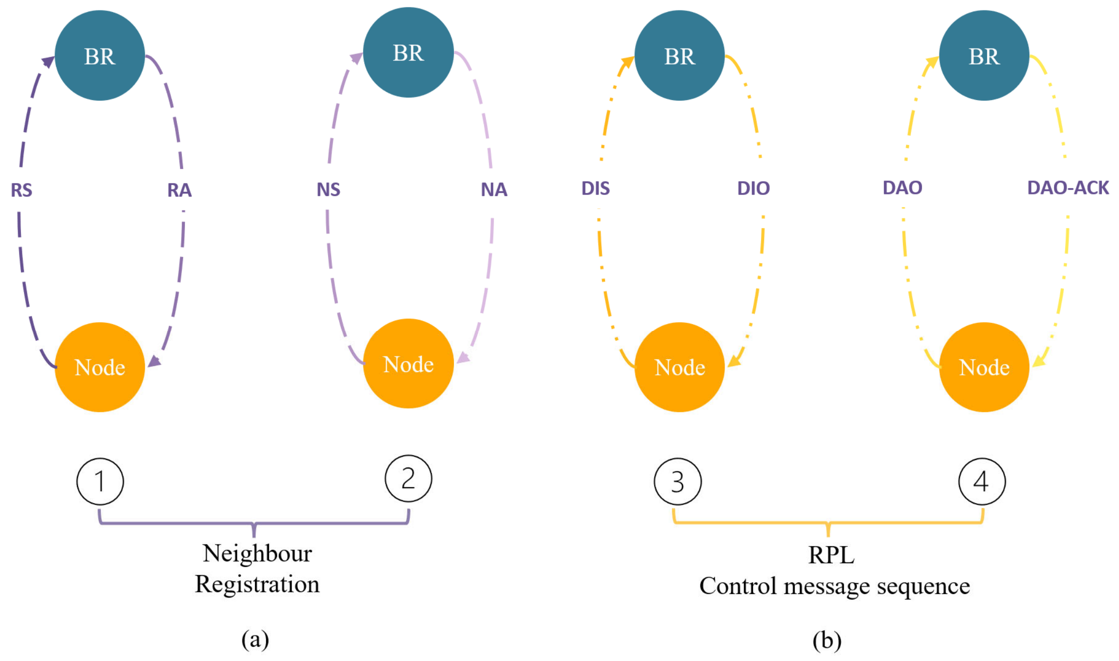


Figure 2.7. Node registration process applying with RPL and 6LoWPAN: (a) Neighbour Registration Process (adapted from Figure 3 of [98]); (b) RPL Control Message Sequence (adapted from Figure 2 of [99]).

The typical node joining process (also called registration process) when applying RPL in 6LoWPAN networks is illustrated in Figure 2.7. During the Neighbour Discovery (ND) address registration process a sequence of control message exchanging between node and border router as followed [98]. The node initially issues a Router Solicitation (RS) for network configuration information. Then, the border router will send a Router Advancement (RA) message in response to receiving RS. Following to this, the node sends the Neighbour Solicitation (NS), and the BR receives it and sends the Neighbour Advancement (NA) in response. Next, RPL control messages will be exchanged to complete the network registration process [99]. Firstly, within the routing information registration process, node will send a DIS message to request to join the DODAG and the BR will reply to the DIS with DIO messages which help BR to multicast trickle timer information. After node synchronise its routing information following with the DIO message, node will issue a DAO message upward to BR through any possible parent node to announce its existence and reachability. To reply to the DAO message, BR will send a DAO Acknowledgment (DAO-ACK) message to the joining node through the routing table to finish the registration process. The explanation and information contain in the packets mentioned during the node registration process is listed in Table 2.1.

Table 2.1. The explanation of radio packets during the node registration process.

Packet	Description	Information Contained (with options)
Router Solicitation (RS) [100]	Request the router to send RA immediately (unscheduled send).	Sender's link-layer address.
Router Advancement (RA) [100]	To advertise the presence of the node with prefix information.	Router lifetime, Neighbour Unreachability Detection (NUD) information, prefix information, MTU, source link-layer address, etc.
Neighbour **Solicitation (NS) [100]**	Request the router to send the link-layer address of the neighbour and reachability.	The target address (IPv6) of the solicitation (BR), source link-layer address, etc.
Neighbour **Advancement (NA) [100]**	In response to NS to announce its link-layer address change.	The IPv6 address of the source and destination, the target address of the solicitation (BR), etc.
DIS [25, 101]	To request and find DODAG from nearby neighbours, and it can also multicast to all nodes to reset the trickle timer for DIO transmission.	Flags and DIS Options.
DIO [25]	To advertise node with the information of RPL instance and node will select reasonable parent node according to DIO message.	RPL Instance ID, DODAG ID, Rank & trickle timer of the DODAG root, Mode of Operation, prefix information option, DODAG Configuration, routing information, etc.
DAO [25]	To inform the reachability of the node and will transmit upwardly to the parent node by unicast.	RPL Instance ID, DODAG ID, RPL Target, etc.
DAO-ACK [25]	Sent by the parent node in reply to the receiving of a DAO message.	RPL Instance ID, DAO Sequence, status of completion, DODAG ID, etc.

The router node will start constructing the topology by sending the DIO message to neighbour nodes. The nearby nodes within the router's communication range will respond to this DIO message to announce that the corresponding node will join the DODAG or not. If a node is outside of the radio range of the DODAG parent but within the communication range of the parent's child, this out-of-range node A will send DIS for neighbour solicitation. And nearby node can send DIO to let node A join with the DAO response from a node after. After all, this DAO from node A will be transmitted to the DODAG parent [96].

2.3.5.3 RPL Instance and Trickle Timer

RPL allows for the creation of multiple DODAG instances, which can coexist and provide redundancy or serve different applications. The Trickle Timer is a mechanism used to control the frequency of DIO messages, balancing the trade-off between network overhead and responsiveness. The exchange of the RPL control message is periodic and controlled by the RPL "Trickle Timer" which has four parameters, Minimum Interval (I_{min}), Maximum Interval (I_{max}), Redundancy Constant (K), and DIO Timer.

RPL's design is characterised by its DODAG structure, objective functions, and efficient traffic classification, which makes it a powerful and versatile routing protocol for low-power and lossy networks. Its ability to manage traffic via ICMPv6 messages, support different modes of operation, and adjust the frequency of activity using a trickle timer ensures that it meets the diverse and demanding needs of IoT applications. Through continuous improvement and adaptation, RPL remains a key component in the development and deployment of efficient IoT networks.

According to RFC 6550, the RPL networking construction will start with building a DODAG. DIO message will be transmitted at the beginning which includes DODAG information (especially trickle timer information). The minimal interval of exchanging DIO message will affect the RPL construction especially the DODAG construction time, and this will critically determine the minimum time cost to start the CoAP packet hopping.

2.3.6 The Constrained Application Protocol (CoAP)

The constrained application protocol (CoAP) was designed to support the communication between nodes and the internet using the RESTful paradigm [102]. CoAP is a lightweight application-layer protocol designed for constrained devices and low-power lossy networks and enables RESTful communication between constrained devices and networks [102]. It supports node-to-node (in same or different constrained networks) and node-to-internet interactions over low-power and lossy networks by enabling a direct RESTful paradigm [102]. As an HTTP-like application protocol designed for these resource-limited systems [103], CoAP has low complexity as it uses UDP, not connection-based TCP protocol. While UDP lacks built-in mechanisms for error recovery or retransmission of lost packets CoAP provides these features. The CoAP message format is shown in Table 2.2 below.

Size (bits)	**2**	**2**	**4**	**8**	**16**	**0-64**	**0 or more**	**8**	**If any**
Field	**Ver-sion**	**Type**	**Token Lengt h**	**Code**	**Mes-sage ID**	**Token**	**Options**	**End of options Marker**	**Payload**

Table 2.2. CoAP message format (adapted from [103]).

To enhance packet reliability, a CoAP transaction can optionally wait for an acknowledgement (ACK) from the receiver and retransmit the packet after a timeout. The inclusion of a 2-byte Message ID in the CoAP header allows the sender to identify ACKs to avoid duplicate transmissions. RPL can effectively extend the range of the network through packet forwarding. Previous studies have focused on improving the efficiency, applicability, and operation of 6LoWPAN, RPL and CoAP by simulation [103], and have concluded that limited power is the main challenge of WSNs based on these protocols [104]. CoAP message includes an 8-bit unsigned integer that indicates the message and response type, and the request method code (GET, POST, PUT and DELETE). Furthermore, RFC 8323 [105] proposes CoAP to support TCP, Transport Layer Security (TLS), and Web Sockets.

2.3.7 Other Application Protocols

As introduced by Rayes and Salam [63], in addition to CoAP, there are other application protocols that have been widely used in IoT contexts, such as Extensible Messaging and Presence Protocol (XMPP), Message Queue Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP), Session Initiation Protocol (SIP), and Distributed Data Service Real Time Publish and Subscribe (DDS RTPS). As defined in RFC 6120 [106], XMPP is designed to address the needs of applications (proximity) to exchange messages in real time with low latency. The instantaneity of XMPP enables the protocol to be used in real-time scenarios such as multimedia calls, alarm notifications, and device configuration [107]. AMQP, as well as XMPP, are open messaging protocols based on TCP [108]. Moreover, AMQP is been initially applied in financial industry to provide reliable point to point message exchange [63] [108]. Same as XMPP and AMQP, MQTT is another TCP based application protocol. And MQTT is a publish/subscribe messaging protocol which need a broker in the network to receive the data from publisher(client) and then report the data to the subscriber(base station/cloud). Compared with COAP, MQTT is TCP/IP based which has more network over heading than UDP [109], and MQTT has higher power consumption and resource requirements [110]. However, the developed MQTT-SN based on UDP which reduces network overhead can be potentially more powerful than CoAP in the application of resource limited IoT [109]. However, MQTT-SN relies on a broker-based publish-subscribe model and does not support message fragmentation [111]. A summary of application protocols discussed above is listed in Table 2.3. CoAP provides advantages in enhancing network connectivity and interactivity with the Internet, as it uses a RESTful architecture. As a short conclusion, the simple UDP-based application protocol, CoAP, will be applied in this study due to its suitability for ESNs.

Table 2.3. A summary of application protocols (adapted from [63]).

Protocol	Transport	Main application scenario	RESTful paradigm
CoAP	UDP	Resource constrained IoT	Yes
XMPP	TCP	Instant message exchange	No
AMQP	TCP	Financial services	No
MQTT	TCP	Enterprise telemetry	No
DDS RTPS	UDP	IP telephony	No
MQTT-SN	UDP	Resource constrained IoT	No

2.4 Operating System: Contiki, TinyOS, LiteOS and RIOT-OS

Due to the limited resources of devices, the operating systems (OSs) running on them are often constrained in functionality. Supporting complex protocols such as IPv6, 6LoWPAN, and CoAP can therefore be challenging. Widely accepted OSs like TinyOS [112], Contiki [113], LiteOS [114], Zephyr [115] and RIOT-OS [116] [117] are all mature platforms that have already supported several IoT functionality. The requirements for an IoT OS (e.g. modularity, energy management and protocol support) have been discussed in previous studies [118-120]. These studies also discuss the advantages and limitations of well-known OSs such as TinyOS, Contiki and RIOT-OS. There are a several additional articles that have summarised the features of various IoT OSs [121-125], and some of the key features the project is focusing on will be discussed next to summarise the real-time operating system selection strategy for this project. Referring to those publications, here are some key design characteristics summarised therein that will be further analysed: (a) kernel architecture and standardised interfaces; (b) RT behaviour; (c) supported programming languages; (d) required memory space; (e) diversity of IP protocol support and (f) diversity of hardware support.

As summarised by Qutqut et al. [126], Contiki has modest requirements, needing less than 2 kB of Random Access Memory (RAM) and less than 30 kB of Read Only Memory (ROM). As a more lightweight option, TinyOS requires less than 1 kB of RAM and less than 4 kB of ROM, and this make it suitable for resource-constrained environments. However, LiteOS has a relatively high requirement of about 4 kB of RAM and 128 kB of ROM. Also, Zephyr project documentation [115] has indicated the minimum ROM footprint is 2-8Kb with many features and services disabled (e.g., network stack, file system, etc.). In contrast, RIOT-OS requires lightweight ROM (about 5kB) and RAM (about 1.5kB) space, which is quite close to the minimum requirements of TinyOS.

Secondly, communication protocol support and network connectivity are key considerations for this project. Fundamentally, the real-time operating system applied in this project should be able to support at least communication standards such as IEEE 802.15.4, especially in the sub-GHz radio band. Support for 6LoWPAN is also one of the main requirements. In addition, to address the potential coverage and routing challenges in this project, widely used routing protocols (e.g., RPL) should be considered to provide reliable and efficient network connectivity. Furthermore, heterogeneous device support and interoperability should be also considered when selecting the OS. OSs for IoT applications should be able to support different devices/boards/components manufactured by various brands to ensure compatibility. In this context, a popular solution is to design standardised interfaces and application program interfaces (APIs) that allow different brands of devices to communicate and work together in a portable and efficient manner [124] [125]. Table 2.4 lists the partial MCU support for different OSs. To implement these OSs in hardware, a minimum requirement of RAM and ROM space is required for the different OSs. The modularity of OSs is also important which allows the operating system to simplify hardware access, protocol integration and development processes through standardised interfaces (e.g., POSIX) and APIs [119]. Table 2.5 lists a comparison of design characteristics between OSs as discussed above.

Table 2.4. Microcontroller Unit (MCU) supports of different OSs. (○) full support or par. al support, (×) no support.

Operation System	*AVR*	*MSP430*	*ARM*	*ESP32*	*RISC V*
***Contiki* [127]**	○	○	○	×	×
***TinyOS* [128]**	○	○	○	×	×
***RIOT-OS* [129]**	○	○	○	○	○
***LiteOS* [114] [125]**	○	×	○	○	○
***Zephyr* [125] [130]**	○	×	○	○	○

Table 2.5. IoT OSs comparison: (√) full support, (Δ) partial support, (×) no support.

Operation System	*Programming Language support [123]*	*Programming Model [121, 124, 125, 131]*	*Modularity [124]*	*RT Behaviour [121]*	*6LoWPAN [121, 123]*	*RPL [121]*
Contiki	C	Event Drive, Protothread	Δ	Δ	√	√
TinyOS [128]	nesC	Event Drive, TOS Threads	×	×	√ (Tiny6LoWPAN)	√ (TinyRPL)
RIOT-OS [122]	C and C++	Message-based, Multithreading	√	√	√	√
LiteOS [114]	C and C++	Event Drive, thread	Δ	Δ	Δ	×
Zephyr [115, 132]	C and C++	Message-based, Multithreading	√	√	√	√

RIOT is a good platform choice for this work given its developer-friendly design and high level of modularity. Specifically, RIOT-OS supports various protocols that are essential for the experiments related to this topic. It not only provides comprehensive support for IPv6 over 6LoWPAN but also implements the RPL through its Generic (GNRC) network stack [124]. Full support for C programming, multithreading, and RT aspects make RIOT the better option for this design than other OSs.

However, RIOT does not support protocols (e.g. 6TiSCH) itself but is able to adapt to and collaborate with other network stacks to provide implementation. As an example, RIOT integrates the OpenWSN network stack [133] via the internal packet system that will allow calls to the OpenWSN library, which was released in Dec.2020 [134].

2.5 Characterising of Existing System Designs from Real Deployments

To address the requirements and properties for ESNs, this section will provide a comprehensive analysis of various ESN projects documented in the published literatures. Also, this section will analyse and category the goals, designs and deployments of these projects and attempt to summarise the design considerations of the ESN project using characteristic parameters. As a specialized type of WSN, ESNs require targeted design of low-power networks based on an understanding of the environment [4]. It should also provide specific task-oriented services and be able to interact with the user and the development environment like generic WSNs [135].

To analyse the ESN system design, this work will refer to the study of Borges et al. [135] where a comprehensive taxonomy of characterisation parameters is used to analyse and evaluate the system. Six categories of characterisation parameters are summarised in Borges et al.'s approach, namely service, communication and traffic, service component, network, node, and operation environment [135]. Additionally, the data delivery model of the system can be classified into four different driven approaches, event-driven, query-driven, continuous transmission, and time-driven [135, 136]. In particular, this work will focus on the time-driven mode, in which sensing data are collected from the environment at a certain data sampling rate periodically.

Sensor nodes can collect various types of data from the deployed environment and send the data to one or more receiver ends, which will then send the data to a designated server [136]. Generally, sensor node is composed of multiple components, including but not limited to sensor, processor, transceiver, positioning system, and power supply unit [136]. ESNs can assist people in detecting environmental changes by configuring numbers of nodes in a targeted physical area to sense the surrounding environmental parameters, and then aggregating the sensed data to stations with processing capabilities through technical approaches. In general, ESN applications will involve all kinds of subject areas based on task requirements, which typically increases the system complexity. Furthermore, with the increasing number of nodes in the network, the power consumption of the entire system is non-negligible, which also causes challenges for system lifetime and maintenance in resource-constrained environments. On the other perspective, this has further validated that it is necessary to improve the system energy efficiency to realize an efficient ESN system. The system must minimize energy consumption while maintaining sufficient sensing and communication capabilities under assumptions of realistic access requirements and power availability. In this section, the system design needs to firstly identify the system technical details of the various published ESN projects, including sensor types, monitored environments, applied protocols, etc. To analyse and category the existing literature, this section will characterise and summarise these ESN systems based on the system features listed in Table 2.6.

Table 2.6: Characterisation parameters explanation of ESN systems (adapted from [135]).

Characterisation parameters	Description
Data delivery model	Event-driven, query-driven, continuous transmission, and time-driven
Monitoring target	Specific environment that system is designed to monitor
Sensor type	All the sensors that have been integrated into the system
Data sampling rate	The rate at which sensors read data from the environment
Communication rate	The rate at which nodes transmit data to the server or sink node
Operating system	The operating system that the project has used
Network protocol	The network protocols involved in the different protocol stack layers of the system, including application layer protocol, transport layer, network layer, data link layer, and physical layer
Power management	Approaches applied in the project to manage power consumption and harvest energy from environment

2.5.1 Artificial Environment Monitoring

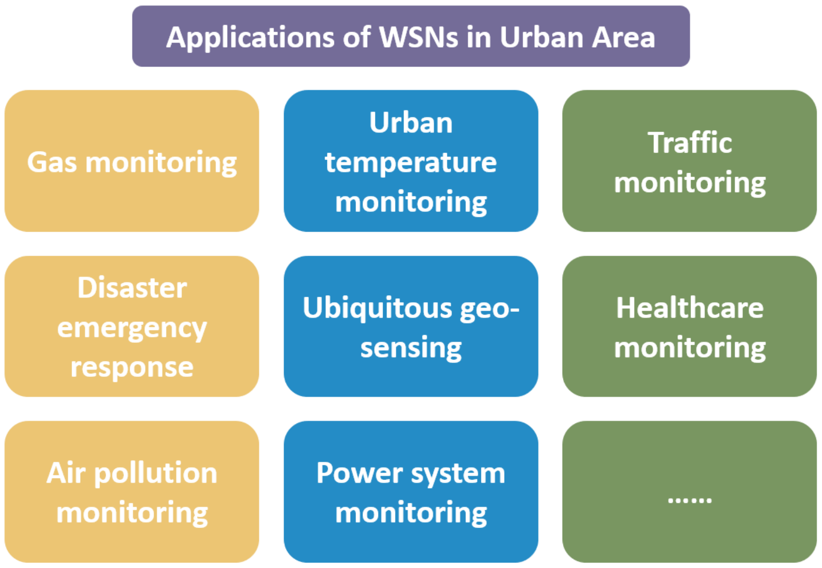


Figure 2.8. Applications of WSNs in urban area (adapted from [137]).

In artificial environments such as urban areas, indoor spaces, and controlled ecological areas like greenhouses and botanical gardens, deploying ESN systems can typically support decision-making and management by sensing the environment. The diversity of environmental monitoring in urban areas allows for the application of ESNs at various scales to help city management systems respond in a timely manner and minimize resource wastage, such as monitoring of hazardous gases and temperature changes [138, 139], traffic management [140], and energy saving in indoor lighting [141, 142]. In terms of the system applications in urban areas, Rashid and Raimani have summarised as illustrated in Figure 2.8 [137].

As an example, OutSense targets was designed to monitor the environmental influence of traffic on urban air quality in Sofia, Bulgaria [138]. The design of this project is based on two assumptions. Firstly, it can operate continuously when supplied with a constant power source. alternatively, when powered by batteries, it takes data samples periodically (every 10 minutes) and delays data transmission to the network until a connection becomes available (e.g. Wi-Fi), subsequently entering sleep mode after data transmission [138]. This design demonstrates that system energy savings can be achieved not only through sleep modes, but also by taking advantage of tolerance for delays to transmit data only after satisfying specific conditions. By contrast, an urban air quality real-time monitoring system deployed in Taipei City utilises CC2420 wireless transceivers to aggregate data to a gateway based on the IEEE 802.15.4 protocol standard and ZigBee specifications [139]. In this project, the gateway was configured to send message through GSM module to server. Meanwhile, the operation of sensor nodes is divided into different stages (networking, configuration, data collection, and sleep stage. Its energy-saving mechanism is to allow nodes to enter sleep mode after periodic data transmission (every 10 minutes). The two urban air quality monitoring systems reflect differing design assumptions based on requirements, including varying expectations regarding infrastructure availability and data timeliness.

As another example, a WSN based urban environment monitoring system using LoRa technology was deployed in Medellín, Colombia [143]. LoRa technology is characterised by long range and low power consumption, making it easy to cover large areas. Compared to the previous two projects, this LoRa-based monitoring deployment utilises data processing at the node end. The system operates at a relative-high sampling frequency (every 7 seconds), but the data can be processed using Kalman filtering and down-sampling techniques before transmission.

Compared to outdoor environments, the deployment of WSN systems in indoor areas can take more advantages on utilising various infrastructure resources (such as power and networking). Intelligent Workplace (IW) at Carnegie Mellon University [144] is a typical case of indoor environmental monitoring and control applications. The design and implementation of the IW system are described and summarised in [141, 145, 146]. The system was designed to be integrated with the infrastructure and IT systems in the building. IW system aimed to monitor and manage various factors of the working environment in the building, including heating, ventilation, air conditioning, lighting and energy use, hence adaptively controlling the environment to improve occupant comfort and work efficiency. This system can take advantage from uninterruptible power supply and the support of IT systems to provide real-time environmental data for seamlessly control, which is fundamentally different from other resource-constrained ESN systems.

In urban environment monitoring applications, except for some disaster early warning systems, other projects usually only require the system to record environmental data for a long period of time and usually do not need to respond in a timely manner. Compared to urban monitoring, artificial ecological environments such as greenhouses and botanical gardens typically require customised monitoring of growth processes for specific targets, and may sometimes require the implementation of early warning or environmental control measures. Rustia et al. applied a system integrated with environmental sensing and image processing to monitor the insect pests [147]. This project is characterised by its two different data types: environmental data and image data. These two different data types have separate requirements in terms of processing capacity, data sampling and transmission rates. For ESNs, another design approach for energy saving could be to dynamically adjust the frequency of data collection and transmission based on the environment and requirements. As a similar example, Ripple-1 project proposed an adaptive sampling strategy, where the project guides sampling through a strategic sensor deployment strategy to minimize the dependence on a dense sensor deployment [12]. The nodes were designed to measure environmental data through soil moisture sensors with sampling frequencies ranging from once per minute to once per hour or tens of hours (depending on external weather conditions and previous moisture values). Furthermore, the ENS system can adopt different operational strategies based on awareness of the resources (for example, adjusting operating modes according to the available energy sources). In another example, the Sensor Web project [148] proposed a system, which used a coordination mechanism supporting multi-hop communication (with a 916 MHz radio). Node batteries can be charged via solar power and support switching to a low-power operating mode when battery levels are low.

Table 2.7 analyses and characterises these ESN Applications in Artificial Environments. The analysed ESN systems deployed in artificial environments demonstrate that: 1) When real-time data transmission is not required, the system can save energy through periodic or delayed transmission; 2) When infrastructure and power resources are available, the system can reliably support continuous or real-time data transmission; 3) The system can reduce data transmission volume through technical approaches (such as data processing and adaptive sampling) to save energy.

Table 2.7. Key Features and System Characteristics of ESNs in Artificial Environment.

ESN project case	OutSense [138]	Urban Air Quality Monitoring [139]	Urban air pollution Monitoring [143]	Intelligent Workplace (IW) [144]	Automated Greenhouse Insect Pest Monitoring [147]	Ripple-1: Wireless Smart Sensor Network [12]	Sensor Web [148]
Monitoring Target	Urban Areas	Urban Areas	Urban Areas	Indoor Areas	Greenhouse	Botanical Garden	Botanical Garden
Sensor type	Gas (CO_2, CO, NO_2, O_3, O_2), temperature, humanity, and pressure sensor	Light, temperature, humidity sensors, CO Concentration Sensor, and WS-2316 weather station module	Air quality indicators (e.g., PM2.5, PM10, etc.) and hazardous gases (e.g., CO, CO_2, O_3, etc.)	HVAC, electrical usage, and outdoor environmental conditions sensors, temperature, CO_2, motion, and others	Camera, AM2301 Sensor, and BH1750 light sensor	Soil moisture sensor	Air temperature, soil temperature, relative humidity, O_2 level, soil moisture, light level
Data Sampling rate	Continues or Every 10 minutes	Every 10 minutes	Every 7 seconds	Real-time	Every 5 mins (Environment); every 10mins for im-ages	Per minute to once per hour or tens of hours	Every 5 minutes
Data Transmission Rate	Transmit all the unsend data once it has connected to Wi-Fi	Every 10 minutes	every 7 seconds (from node to gateway)	Continuous data stream with real-time processing capabilities	every 5 mins (Environment); every 10mins for images	Defined by base station, could be various	Every 5 minutes
Operating System	Linux	TinyOS	NA	central building management system	Linux	NA	A commercial microcontroller
Network Protocol	Standard Internet protocol, Wi-Fi, GPRS (can be added)	IEEE 802.15.4 PHY and MAC, ZigBee, and GSM	LoRa	Seamless integration with existing IT infrastructure	UDP and HTTP POST	ZigBee, IEEE 802 15.4 PHY and MAC, and 3G	Multi-hopping, Sensor Web protocol (coordinated with 916 MHz radio)
Power Management	Uninterruptible power supply or battery powered, entry Sleep mode after transmission	Battery-powered, entry sleep stage after transmission and wake up as scheduled	Battery-powered with solar panel charging, turn off radio after sending data	Uninterruptible power supply	Uninterruptible power supply, low-power-sleep/wake-up switching	Battery powered with solar panel charging, low-power-sleep/wake-up switching	Battery powered with solar panel charging, low-power-sleep/wake-up switching

2.5.2 Natural Landscape Environment Monitoring

In contrast to the artificial environment applications discussed in the previous subsection, ESN deployments in complex landscape environments (e.g., mountainous areas, forests, wetlands, etc.) are typically constrained by complex natural terrain, climatic variability, deployment coverage and transmission distance. These factors impose fundamentally different design assumptions, particularly with respect to transmission distance, communication reliability, and long-term energy sustainability.

At small spatial scales, ESNs can be developed to monitor micro-environmental variations around specific natural targets. The Redwood's life monitoring system deployment [149] is an example of this deployment type. This project was designed to obtain spatial gradients of temperature, humidity, and light levels that affect redwood trees, and it helped to study the relationship between changes in environmental factors and changes in tree height. Data were sampled and transmitted every five minutes using a 433 MHz radio. Nodes rely on TinyDB [150] to filter and aggregate data, subsequently transmitting it to the gateway with the support of MintRoute [151] following data collection. Regarding energy-saving considerations, this project applied a duty cycling approach and pointed out that the duration for each network awake is 4 seconds. Besides, nodes are battery-powered and capable of solar charging. Although solar charging technology is available, this approach essentially combines sensing with communication activity, resulting in frequent radio activations at the node level. By contrast, it is expected that communication energy consumption could be further reduced by buffering sampled data and transmitting information with higher payload utilisation during a constrained, lower-frequency communication window.

Moreover, Light Under Shrub Thicket for Environmental Research (LUSTER) [152] project was proposed to monitor the micro-environment of shrub thicket. LUSTER employed high-frequency sensing (default sampling every 1 second) and relied on LiteTDMA to regulate transmission and sleep schedules. On the other hand, the transmission interval in LUSTER is closely linked to the network topology and varies according to changes to the number of child nodes. The nodes in the LUSTER project are configured by the MAC protocol controlling the transmission and sleep states of the nodes. In addition, when data collection and transmission were not taking place, the sensor nodes would enter sleep state and only wake up for a scheduled time slot. It is notable that as the number of child nodes increases, the data receiving duration for parent nodes extends. However, child nodes within the project continuously listen to the channel whilst other nodes are transmitting, resulting in energy waste.

In ESN deployments within large-scale natural landscapes (e.g. mountains, deserts, island, etc.), system design may also need to consider environmental diversity and the extension of communication range and monitoring coverage. The Mountain sensing project [53, 54] integrated Internet protocols into a low-power WSN architecture to support long-range, multi-hop communication over 3.5 km using 868 MHz radio. By combining 6LoWPAN, ContikiMAC, RPL, UDP, and CoAP on Zolertia Z1 nodes running ContikiOS, this project enabled data sampling rate of every 20 minutes and communication rate of every hour as schedule. This project used energy harvesting and the duty-cycling based ContikiMAC protocol to conserve power, but successive channel assessments after the radio switched on still resulted in unnecessary energy consumption.

As another example, the CiROCCO project [153] was deployed to continuously monitor environmental parameters in harsh desert ecosystem. This system applied Bluetooth Low Energy (BLE) and LoRaWAN, combined with solar energy harvesting, to continuous collecting and transmitting data from nodes to gateway. Similarly, the Bidong Island deployment [154] applied long-range LoRa communication to transmit data across a 23 km sea link. Although such designs effectively address coverage and environmental challenges, the relatively frequent or even continuous data transmission needs to rely on energy harvesting or infrastructure support, otherwise it would cause issues for the sustainability of deployment.

Table 2.8 analyses and characterises these ESN Applications in Natural Landscape Environments. Overall, these deployments demonstrate that ESN designs in natural landscape environments need to consider the challenges caused by complex natural terrain, climatic variability, deployment coverage and transmission distance. In system design, duty-cycling based low-power MAC protocols (such as ContikiMAC) and energy harvesting technologies have been widely implemented. Duty-cycling based low-power MAC protocols can optimise energy usage, and energy harvesting can extend system lifespan, while neither can eliminate unnecessary network activity. Furthermore, as many such systems are deployed in areas where frequent maintenance is difficult, the ability to adaptively adjust settings after deployment also need to be considered. In other words, for ESN deployments in resource-constrained remote areas, key considerations include remote management capabilities (IP-based system architecture), optimisation of communication frequency and unnecessary overhead, and power management.

Table 2.8. Key Features and System Characteristics of ESNs in Natural Landscape Environment.

ESN project case	Redwood's life monitoring WSN system [149]	LUSTER [152]	Mountain sensing [53] [54]	CiROCCO [153]	Bidong Island environment monitoring [154]
Monitoring Target	70m redwood tree	shrub thicket	Mountain	Desert	Island weather point
Sensor type	SHT11 temperature and humidity sensor and two S1087 photodiodes	Light sensor, expandable to other ecological parameters such as temperature, humidity, CO_2, and soil moisture	temperature, soil moisture, strain gauge, water level, rain gauge sensors	air temperature and humidity, the meteorological characteristics of wind and other climate parameters	DHT22 temperature sensor or YL-69 soil moisture sensor
Data Sampling rate	every 5 minutes	every 1 second (default)	every 20 minutes	continuous	Every 6 seconds
Data Transmission Rate	every 5 minutes (communication time cost is 4 seconds)	570 ms (32 child nodes) to 260 ms (1 child nodes)	Every 1 hour	continuous	Every 6 seconds
Operating System	TinyOS and TASK software	TinyOS	ContikiOS	N/A	Arduino
Network Protocol	TinyDB and MintRoute, GPRS	IEE802.15.4 radio, LiteTDMA MAC	IEEE 802.15.4, 6LoWPAN, ContikiMAC, RPL, UDP, CoAP	BLE, LoRaWAN, Wi-Fi, LTE (4G/5G), L-Band satellite communication, MQTT	LoRa (915MHz)
Power Management	Battery powered with solar panel charging, duty-cycling based low-power-sleep/wake-up switching	Battery powered (Node) Battery powered with solar panel charging (base node), low-power-sleep/wake-up switching	Battery powered with solar panel charging, duty cycling protocol	Battery powered with solar panel charging, low-power-sleep/wake-up switching	Battery powered

2.5.3 Inland Water and Ocean Environment Monitoring

In this subsection, ESN deployments for inland water and ocean environments will be analysed. Compared with land-based deployments, ESN applications in water and ocean environments often operate under distinct operational constraints, such as specific physical environments and communication requirements. In inland water environments, The Reactive Environmental Monitoring WSN system [155] was deployed to measure soil moisture. This project was raining event triggered, and system would respond to the rainfall while adjusting the sensing frequency. The data sampling rate would be increased to every 10 minutes if rain is detected, and the data sampling rate would drop to once a day if the event is not detected. To detect the rainfall, rain node deployed in this system would check the rain gauge every 12 seconds (controlled by SMAC) with the threshold of 1mm rain (Note that every trigger was reset a 2-hour timer). Within this system, the event-triggered mechanism enables the network to reduce unnecessary data communication during inactive periods. This selective data transmission activation is energy-efficient for systems monitoring specific events, but for long-term monitoring, it only allows for low-frequency collection of environmental changes when no events are triggered. For long-term environmental monitoring applications, where communication latency is acceptable, minimizing communication frequency remains an important design consideration for improving energy efficiency.

In contrast, ESN deployments in ocean environments must address different challenges. The Underwater Sensor Network project [156], demonstrated how communication limitations in underwater environments can be overcome by combining static sensor nodes with mobile platforms (an autonomous underwater vehicle (AUV) called Amour AUV and Starbug AUV). In this deployment, static Aquaflecks nodes collected environmental and visual data at fixed rates, while data transmission is broadcast with delay when the storage threshold is reached or transmitted via optical communication when the mobile AUV accesses the node. This design effectively overcomes environmental challenges but also introduces additional complexity and transmission delays.

As another case, Sensor Exploration Apparatus utilizing Low-power Aquatic Broadcasting system (SEA-LABS) [156], was proposed to monitor the health of coral reefs in shallow marine environments and surrounding environmental conditions. SEA-LABS does not rely on standard OSs , instead using custom task scheduling to manage transitions between sleep, sensing, and transmission states. The system prioritises energy efficiency over real-time data transmission, restricting communication to occur only after each data collection cycle. This project designed a sensor node, called Programmable Oceanic Devices (PODs), equipped with temperature, light, pressure, conductivity and pH sensors. The data collection and transmission rate were controlled by designed task scheduling algorithm (network load requirements are typically less than 1 percent). Instead of using TinyOS, this project applied custom scheduling algorithm to control and switch the operation mode of the node among sleep, data processing, and data transmitting mode. Nodes were battery-powered and would transmit data only after each data sampling slot, otherwise they would enter sleep mode to save power. This project demonstrates that algorithmically controlled sleep/wakeup switching can also achieve low network load rates, thus improving system energy efficiency.

Features and system characteristics of ESNs in inland water and ocean environment are tailed in Table 2.9 below. In such systems, it has indicated that system energy efficiency can be improved by combining sleep/wake scheduling with other technical approaches, such as event-triggered or delayed transmission. However, these approaches typically increase system complexity, reduce data timeliness, or increase dependence on specialised hardware, which needs to be balanced against deployment scenarios and requirements.

Table 2.9. Key Features and System Characteristics of ESNs in Inland Water and Ocean Environment.

ESN project case	Reactive Environmental Monitoring WSN [155]	Underwater Sensor Network project [156]	SEA-LABS system [156]
Monitoring Target	Groundwater Mound	Underwater	Oceanic Environment: coral reefs
Sensor type	Echo-20 dielectric sensors and Echo rain gauge	Pressure sensor, temperature sensor, and a CMUCam camera	Temperature, light, pressure, conductivity and pH sensors
Data Sampling rate	every 10 minutes (rainfall) once a day (no rain) every 12 seconds (rain detected)	every 10 minutes	Controlled by designed task scheduling algorithm (network load requirements are typically less than 1 percent)
Data Transmission Rate	Event-triggered: Event alert (1mm of rain measured) Event ends (2 hours after last alert)	Optical communication: Depend on duty cycling mode of AUV Acoustic communication: when the amount of data collected exceeds the storage limit	
Operating System	TinyOS	TinyOS	Custom task scheduling algorithm
Network Protocol	SMAC, GSM, 433 MHz radio	Data Mule, Optical communication, Acoustic communication	half-duplex slotted-Aloha, 900 MHz radio
Power Management	Battery powered, low-power-sleep/wake-up switching, event-triggered transmission, adaptive sampling	Battery powered, low-power-sleep/wake-up switching	Battery powered with solar panel charging, low-power-sleep/wake-up switching

2.5.4 Disaster Monitoring

Real-time monitoring systems for various types of disasters can help humans to warn the occurrence of disasters early and thus reduce the threat of disasters to human lives and the possible economic losses. ESN system can be placed in disaster-prone areas to monitor potential disasters and provide early warning. In the case of earthquakes, tsunamis and floods, ESNs application can provide early warning by detecting data on unusual changes in the environment.

For disasters requiring rapid response and having an extremely short warning window, ESN deployments typically prioritise continuous monitoring and real-time data transmission. As an example, the self-organizing seismic early warning information network (SOSEWIN) [157, 158] has already wide implemented in different countries, including Türkiye, Italy, and French. SOSEWIN sensor nodes applied in this project sampled seismic signals at rates up to 100 Hz. System network was designed to rely on communication technologies such as WLAN (2.4 GHz and 5 GHz), Optimised Link State Routing (OLSR), SeedLink Protocol, to ensure low-latency and high-throughput data transmission. Additionally, Incorporated Research Institutions for Seismology (IRIS) Global Seismographic Network (GSN) system [159, 160] has been deployed to provide worldwide seismic data for earthquake and tsunami warning. Both earthquake monitoring ESN projects were designed with high-precision environmental sensing and real-time low-latency data transmission capabilities. This demonstrates that for earthquake early warning ESNs, the system priorities are to satisfy reliability and real-time performance rather than energy efficiency.

Tsunami monitoring systems, while also requiring timely data delivery, typically incorporate sleep/wake methods and event-triggered mechanisms to reduce energy consumption, given that deployment locations may be remote from land. The Deep-ocean Assessment and Reporting of Tsunami (DART) [161, 162] system maintains a default sampling rate and transmission rate for hydrological information during non-event triggering periods. Conversely, in event-triggered mode, both the duration of sampling and the data transmission rate are increased. This design satisfies the requirement for long-term environmental monitoring while enhancing system energy efficiency through its sleep/wake mechanism. Meanwhile, under the event-triggered mechanism, the system is enabled to respond rapidly to abnormal environmental changes.

Similarly, flood monitoring ESNs also emphasise long-term environmental monitoring and monitor the environment at a higher frequency during rainy periods. Remote water-level monitoring system (RWMS) [163], was proposed to monitor water levels change and alert flood in China's largest freshwater lake, Poyang Lake. The system collects water level data hourly and transmits it via GPRS/GSM. Monitoring frequency can be dynamically adjusted through the collaboration of the data centre and the data releasing module. This design demonstrates that ESN systems can conserve energy by maintaining moderate sensing and communication rates during stable periods, while still allowing post-deployment adjustments of parameters, such as data sampling and transmission rates, to accommodate changing monitoring requirements.

Volcanic activity monitoring and eruption warning is also an important application area for ESNs. To monitor an active volcano in Ecuador, Reventador, an volcano monitoring ESN system was deployed [21]. This project typically transmits sampled data at fixed rate(every 10 seconds). However, when the number of triggered nodes exceeds the threshold, multiple nodes will perform bulk data transmission. Time Synchronisation Protocol (FTSP), MintRoute, Deluge protocol, and Fetch Protocol were applied in this system supported by TinyOS.

The dry conditions in forests and mountainous areas that are prone to wildfires. In addition, mountainous and forest areas typically cover an extensive area, and the location of wildfires is typically random and unpredictable. As a result, wildfire monitoring typically requires long-term early warning monitoring over relatively large physical ranges and at certain densities. As an example, Remote and In-Situ Monitoring for Biodiversity and Environmental Areas (RIMBAMON) system [22] applies periodic sampling and permits nodes within a certain region to perform continuous monitoring according to base station requirements. In contrast, the forest fire surveillance system deployed in South Korea [164] requires continuous environmental monitoring and alerts only when events are triggered.

According to the analysed ESNs based disaster early warning system, the frequency of data collection/transmission of the system tends to be tightly correlated with the level of the detectability of the disaster (Check Table 2.10 below for more details). Earthquake early-warning systems typically require continuous sensing and low-latency communication, often relying on uninterruptible mains power to ensure reliability. Besides, other early warning systems for natural hazards mostly apply event-triggered mechanisms to monitor the possibility of disasters over long-term in a low-power approach. Also, switching node to low power mode and energy harvesting solutions are also valuable to consider to further enhance the system reliability especially in system lifetime prospective. However, for disaster monitoring ESNs, improvements in energy efficiency must be balanced against the requirements for data timeliness and communication reliability.

Table 2.10. Key Features and System Characteristics of ESNs in Disaster Early Waring.

ESN project case	SOSEWIN [157] [158]	Global Seismographic Network (GSN) system [159]	DART [161]	Remote water-level monitoring system (RWMS) [163]	Volcán Reventador sensor network [21]	RIMBAMON system [22]	Forest Fire Surveillance System [164]
Monitoring Target (Early warning events)	Earthquake	Earthquake	Tsunami and Tides	Floods	Volcanic Activity	Forest Fire	Forest Fire
Sensor type	3-Axis MEMS Accelerometers and geophones	Seismometers, auxiliary sensor, motion accelerometers, pressure sensors	Tsunami gauges, pressure sensors, acoustic modems and sensors, tilt sensors	water-level sensors	single-axis seismometer or three seismometers, omnidirectional microphone	temperature, light intensity, acoustic, acceleration and magnetic sensor	Temperature and humidity, light
Data Sampling rate	Real-time monitoring with sampling rate up to 100 Hz	Sensors use fixed sampling rates (from 0.5 Hz to 40 Hz). Only motion accelerometers can rise to 100 Hz following event triggering	Every 15 seconds (Triggered - increased to 60s data) every 15 minutes – tidal report	Every 1 hour (will increase during rainfall)	100 Hz	Every 30 minutes	Real-time monitoring
Data Transmission Rate	Continuous data handling	Continuous data handling	Every 6 hour Every 1 hour - Event	Every 1 hour(will increase during rainfall)	Every 10 seconds Transmit continuous 60s data - Event	Event triggered: Base station requests 5 readings	Send event alert when reading exceeds threshold
Operating System	OpenWRT (Linux)	NA	Embedded System	ZKOS [165]	TinyOS	TinyOS	TinyOS
Network Protocol	Optimised Link State Routing (OLSR), SeedLink Protocol, and WLAN (2.4 GHz and 5 GHz) communication	NA	Satellite communication	2.4 GHz ad hoc network, or the GPRS/GSM	2.4 GHz Radio, Flooding Time Synchronisation Protocol (FTSP), MintRoute, Deluge protocol, and Fetch Protocol	ISM Band communication	2.4 GHz radio, Minimum Cost path Forwarding (MCF)
Power Management	Mains electricity supply with optional battery backup, idle monitoring and event-triggered	Mains electricity supply	Battery powered, duty-cycling based transceiver on/off mechanism	Battery powered, low-power/detection mode switching	Battery powered	Battery powered with solar panel charging, low-power-sleep/wake-up switching	Battery powered

2.5.5 Arctic and Glacial Environment Monitoring

As a result of global warming, climate change in the Arctic and glacial regions has attracted the attention of a significant number of scientists. However, for in situ environmental monitoring, the environmental conditions in this region are hostile and many areas are unsuitable for long-term settlement. Therefore, for remotely monitoring in the polar regions, implementing ESNs to address the challenges of environmental monitoring under these extreme conditions has become a reliable option. As an example, SensorScope project [166] deployed several WSNs targeting a specific environment in Switzerland. As an example of harsh environment targeted deployment on the top of a rock glacier in Switzerland, this deployment was designed to monitor the microclimate of the rock glacier. The system deployed sensor nodes equipped with multiple environmental sensors and operated with a fixed sampling rate of every 120 seconds and the system network duty cycles is 10%. Although this approach can be useful for saving energy for long-term monitoring, it still relies on frequent radio activation, which may limit the system's lifetime.

To study extensive environmental changes, such as glacial movements and earthquakes for harsh arctic environments, ESNs have also been designed with highly constrained communication schedules. The Glacsweb project [55, 56] was developed to investigate the results influence that climate change might have on glacier movement in Briksdalsbreen, Norway. In this deployment, heterogeneous sensor nodes were programmed to collect data only during predefined timeslots and to communicate within one daily communication window. This design, which activates communications according to a strictly limited schedule, effectively avoids the energy consumption caused by frequent radio activation. However, it introduces new challenges. Firstly, once all nodes' radios are activated simultaneously, how long will it take for the nodes to complete the autonomous network construction to allow data transmission. Moreover, it is foreseeable that multiple nodes initiating communication simultaneously will cause collisions, retransmissions, and data loss. Secondly, if the base station queries data from child nodes as designed, certain nodes may experience excessively long wait times. Solutions to these issues have not yet been investigated.

Martinez et al. also proposed and deployed the geophone wireless sensor network (another subsystem alongside the Glacsweb project) [69] to investigate the link between sub glacial processes and climate change. In contrast to periodic environmental sensing, the sensor node only sampled data at 512 Hz frequency when it detects an earthquake event. Meanwhile, data transmission was scheduled at the same daily communication window same as the main Glacsweb system. This design effectively suppresses redundant data collection and transmission, and it minimised unnecessary radio activation. However, this design still depends on precise event detection mechanisms and the reliable coordination with surface node.

Similarly, the geoPebble System [167] was deployed on the Greenland and Antarctic ice sheet is aimed to collect the natural earthquake data to investigate its relationship with Ice lakes drying up and glaciers disintegrating. This system introduced multiple operational modes, including operation modes, burst mode, continuous mode and sleep mode for each sensor node. While such multi-mode operation improves flexibility and balances varying data sampling requirements with energy efficiency for long-term operation, it also increases system complexity and introduces challenges on synchronisation and network management.

Taking this further, the Distributed Arctic Observatory's (DAO) research team proposed a system in Arctic Tundra and this system was deployed for about 12 months [168]. This project aimed to investigate the influence of climate changes on the environment of arctic tundra. The designed sensor node was called Observation Units (OUs), and it was designed to collect sensor readings every 30 minutes and to send data to central server once per day over LTE-M networks. Except sensing and transmitting mode, OUs were planned to entry deep sleep mode to save power. However, the battery power sensor nodes are not equipped with solar panel or other devices to harvest energy from environment yet.

As a short summary, Table 2.11 below lists all the system characteristics of the analysed cases in this subsection. In resource-constrained environments such as the Arctic and glaciers, ESN systems should be deployed with as much consideration as possible to reliable design and energy saving technologies. In terms of energy-saving technologies, the fundamental technology to consider is sleep/wake approaches, including timetable-based scheduling methods and duty-cycling based protocols. Additionally, the system may apply event-triggered mechanisms to minimise unnecessary data sampling and transmission, whilst energy harvesting (such as wind energy) is also a possible option. In harsh and resource-constrained environments, while energy efficiency remains a primary design consideration, system design still requires trade-offs between energy efficiency and data reception latency, as well as node accessibility.

Table 2.11. Key Features and System Characteristics of ESNs in Arctic and Glacial Environment.

ESN project case	SensorScope [166]	Glacsweb [55] [56]	Geophone Wireless Sensor Network [69]	geoPebble System [167]	Distributed Arctic Observatory (DAO) [168]
Monitoring Target	Rock Glacier	Glacier	Glacier	Greenland and Antarctic ice sheet	Arctic Tundra
Sensor type	Air temperature & humidity, precipitation, soil moisture, solar radiation, surface temperature, water content, and wind direction & speed sensor	Pressure, temperature, orientation (tilt in three dimensions), external conductivity and strain gauge sensor	Geophone, 3D accelerometer, digital compass, temperature sensor	Triaxial geophones, temperature and pressure sensors	Carbon dioxide level and temperature
Data Sampling rate	Every 120 seconds	Every 4 hours at 00:00, 4:00, 8:00, 12:00, and 20:00 in a day	Event triggered: sample data at 512 Hz when a seismic event is detected	Event triggered: sample data for 60s at 10 kHz or sample at 625 Hz if passive seismics is detected	Every 30 minutes
Data Transmission Rate	Every 120 seconds (network duty cycles is 10%)	Once a day at 16:00	Once a day at 16:00s	After sampling data	Once a day
Operating System	TinyOS	Linux	Linux	NA	MicroPython
Network Protocol	868 MHz radio, custom multihopping, custom self-organizing network, synchronous duty cycling MAC, anypath routing [169]	Customized packet-based communication protocol with error detection, 433 MHz and 173 MHz radio and GSM	GPRS or Wi-Fi based on IPv6 (surface station), 868 MHz (surface node) or 173 MHz radio (under glacier node)	Wi-Fi (2.4 GHz)	LTE-M networks
Power Management	Mains electricity supply with optional battery backup, low-power-sleep/wake-up switching	Battery powered, low-power-sleep/wake-up switching	Battery powered, low-power-sleep/wake-up switching	Battery powered, low-power-sleep/wake-up switching	Battery powered, low-power-sleep/wake-up switching

2.5.6 Summary

In Section 2.5, by analysing and categorising deployed ESN projects, this work expects conclude the characteristic parameters (as listed in Table 2.6) of ESNs designed for environmental sensing. In this section, the relevant projects were classified into five categories, namely, examples of ESN deployments in artificial environments, natural landscape environments, inland water and ocean environments, arctic and glacial environments, and the deployments for disaster monitoring. The cases discussed above involve a variety of hardware platforms and sensors, all of which were selected to serve the specific target of environmental monitoring. Expanding to this, when the project's objective is to monitor the artificial environment, the system design needs to give more consideration to continuous monitoring of specific environmental parameters at a comparably high data sampling rate (or even in real time). In another situation, such as disaster monitoring, system should consider targeting at specific disaster and develop efficient early warning mechanism.

Although the analysed projects differ in their system design due to different objectives and design ideas, energy saving strategies have been applied in these projects to some extent. The energy saving technologies applied includes duty-cycling MAC protocols, event-triggered mechanism, schedule-based wakeup/sleep mechanism, data reduction, and energy harvesting. Among these technologies, the most fundamental energy-saving strategy is the sleep/wake up approaches. For example, in artificial environments, even if nodes are required to turn on sensing and communication functions fairly frequently, such as real-time monitoring, the system design should consider allowing the node entering low-power sleep mode after completing the task to save energy. Although some nodes have unlimited power supply, they still need to enter low-power sleep mode periodically to maintain reliability, save energy and extend device lifetime. Also, when the designer wants to deploy the network to an environment where human intervention (e.g., changing batteries or maintaining the device) is more difficult, the nodes are expected to enter a low-power sleep state after completing the scheduled tasks. Similar design ideas can be found in other deployment categories that have more resources constraints than the artificial environment.

Furthermore, ESN system design requires trade-offs in several aspects depending on the design objectives and the characteristics of the targets. For instance, in ESN deployments for disaster warning systems, low-latency data transmission should be prioritised over energy efficiency. Conversely, systems for long-term environmental monitoring can achieve energy savings by accepting delayed data transmission and reduced radio activation frequency. Thus, as a short summary, different ESNs have been designed to optimize energy consumption to maximize lifetime while maintaining system abilities of environment sensing and data transmission.

2.6 Energy Saving Technologies

For WSNs, the energy saving approaches can be classified into five categories, optimizing radio mechanisms, low-power-sleep/wakeup switching, energy harvesting, energy efficient routing and network architecture, and data reduction mechanisms [6]. This section will outline representative energy-saving methods, covering sleep/wake mechanisms and other non-duty-cycling techniques (including data reduction, aggregation and prediction, together with event-trigffgered mechanisms). This work considers sleep/wakeup approaches to be essential for the development of wireless sensor network systems, and this section will focus on the advantages and limitations of energy-saving methods in radio activity management.

2.6.1 Sleep/Wake Up Approaches

As the wireless transceiver consumes more power than other modules within the sensor node, and when no communication activity is present, it can be temporarily disabled to conserve energy [7]. Therefore, Consequently, disabling the radio transceiver during non-active periods has been widely recognised as an effective approach to reducing energy consumption in wireless sensor networks. Sleep/wake-up techniques build upon this principle by controlling radio state transition, and it can be categorised into topology control and duty cycling schemes [6].

Topology control can selectively active nodes and control the network structure to ensure connectivity of the network, which relies on the dense and redundant deployment of nodes. The solution proposed by Misra et al. [170] is an example of an energy efficient approach that inspired the idea of this work to control the energy consumption of the network, i.e., it is possible to control the mode of the nodes using algorithms. Specifically, since it is possible to algorithmically control the activation and inactivation of nodes covering a specific area [170], it is also feasible to algorithmically control the node's task scheduling and entering sleep mode after completing the task for sleepy network.

As summarised by Carrano et al. [23], duty cycling schemes can be classified into three main categories: 1) synchronous; 2) semi-synchronous and 3) asynchronous schemes. In synchronous schemes, system should enable all nodes within a cluster or even globally to turn on and off their radios simultaneously, ensuring that nodes can exchange data at the same time (e.g. RT-Link [171], Traffic-Adaptive MAC protocol (TRAMA) [172], and 6TiSCH). However, this scheme can potentially lead to increased waiting time due to communication conflicts, even when skewed scheduling (e.g. DMAC [173]) is used to shift wake-up times. In terms of semi-synchronous schemes, it introduces small virtual clusters, where nodes dynamically synchronise their schedules within their group (e.g. Sensor-MAC (S-MAC) [174] and Timeout-MAC (T-MAC) [175]). For Elected Cluster-head (e.g. Low-

Energy Adaptive Clustering Hierarchy (LEACH) [176]) and Random Duty Cycling scheme (e.g. Random Asynchronous Wakeup Protocol (RAW) [177]), they are generally suitable for larger, dynamic topologies but it will cost additional signalling. Similarly, mechanisms that utilize wakeup signals/beacons, such as receiver-Initiated transmission (e.g. Receiver-Initiated MAC (RI-MAC)) and on demand wakeup mechanism (e.g. Sparse Topology and Energy Management (STEM) [178]), could potentially bringing in additional energy consumption, increasing protocol overhead and system complexity. Preamble Sampling (e.g. XMAC [179]) or Low Power Listening (LPL) Technique, which requires nodes periodically listen to detect the channel activity while maintaining the node in sleep mode expect exchanging data.

Duty cycling scheme is widely adopted in WSNs projects [53, 180] which periodically switches the radio transceiver between active and sleep states to reduce energy consumption, at the cost of increased communication latency [181] [182]. In duty cycling scheme, asynchronous approaches may require nodes to wake up more frequently to achieve overlap with the wakeup slots of parent/child nodes. Synchronous approaches can effectively reduce idle listening and lower power consumption but introduce additional overhead [182]. One critical parameter in duty cycling schemes is the duty cycles, which describes the proportion of time a device/node‘s transceiver is active relative to the total operational period. Duty cycling protocols with lower duty cycles can reduce the system's daily energy consumption more efficiently. However, in actual deployment, duty cycling MAC protocols are usually configured prior to deployment. Even for protocols with extremely low duty cycles, adjustments made such as changing the data sampling rate after deployment will require the whole network to be re-coordinated or redeployed.

2.6.2 Non-Sleep/Wake Up Energy Saving Approaches

In WSNs, communication module is the primary source of energy consumption. The key focus of radio optimisation techniques is to reduce the radio energy cost required for data transmission. As summarised in [6], typical technologies include adaptive control of transmission power [9], cooperative communication [10], and energy-efficient modulation techniques [11]. Additionally, relevant technologies also include the low power communication module and low power embedded board. Such methods can reduce the average energy consumption during communication. However, they do not address some challenges in heterogeneous ESN network, particularly the to reduce and limit energy wastage caused by unnecessary radio activation. This study aims to optimise the system's overall energy consumption through network-level approaches and efficient scheduling of radio activity. Accordingly, radio optimisation approaches will not be investigated in this work.

The integration of energy harvesting technologies with wireless sensor networks (WSNs) enables nodes to acquire and store energy from renewable sources, thus addressing system energy constraints to a certain extent. Current WSN deployments widely considered harvesting from environmental renewable energy sources, including solar [12], wind [13], and other energy sources [14]. Sufficiently large batteries and energy harvesting techniques can indeed increase the theoretical lifetime of any system. However, energy harvesting depends on environmental power density, and changes in such density can have a fluctuating effect on energy harvesting. Secondly, energy harvesting as a charging solution cannot replace electricity. It does not reduce the overall energy consumption of the system, although it can provide nodes with the ability to harvest energy and extend their lifetime. Regardless of the availability of solar or other energy sources, the overall energy consumption of a system remains a key criterion in determining its performance and reliability.

As summarised in [6], by applying single-path routing protocol, relay nodes in the path will consume more energy than other non-relay nodes, as relay nodes need to forward traffic in addition to transmitting their own data. In contrast, multipath routing protocols aim to distribute forwarding tasks across multiple nodes to balance energy consumption within the network, though this typically increases routing and control overhead. Although efficient routing protocols Although efficient routing protocols are critical for WSN systems in extending network lifetime and enhancing energy efficiency. However, from the system design perspective, relying solely on efficient routing protocols is insufficient. When constructing the network protocol stack for WSNs, low-power protocols must be considered at multiple layers. through the collaboration of low-power protocols at different layers, the overall energy consumption of the network can be effectively reduced. This work will adopt low-power hardware and OS supporting low-power communication protocols and power management mechanisms.

Data reduction techniques represent an effective energy-saving approach in WSN applications. By processing and compressing data, this approach can effectively reduce the volume of data that needs to be transmitted, thus reducing communication energy consumption. Firstly, through data compression and encoding, more effective information can be carried within each data packet, and the efficiency of wireless transmission within the network can also be improved. For example, literature [15] proposes a data compression method based on encoding, which expresses discarded node's data within the data packet through the ordering of data from different nodes. Relevant works [16, 183] have reviewed data compression techniques in WSNs comprehensively. Additionally, in multi-hop networks, data can be processed at a single node or aggregated at intermediate nodes or sink nodes for centralised processing before forwarding. Such mechanism of data aggregation and processing is defined as in-network aggregation [18]. In-network aggregation reduces the amount of data transmitted through the network and reducing the overall communication energy consumption at the cost of additional computation at the aggregating nodes.

Other than the technical approach of reducing data volume through computational processing at nodes, data prediction is another commonly applied efficient approach. As introduced and summarised in [19], data prediction technology requires the deployment and maintenance of coherent prediction algorithms or models across nodes involved in prediction and decision-making. The prediction algorithm or model estimates sensor data based on predefined rules and compares it with actual perceived data. The node determines whether the prediction error exceeds the threshold based on predefined rules, and decides accordingly whether it is necessary to transmit the actual data.

In summary, data reduction and aggregation techniques can improve the efficiency of communication transmission and reduce the volume of data being transmitted. Also, data prediction can efficiently reduce the suppress transmissions when prediction error is tolerable. However, these technologies also potentially introduce additional complexity and cannot fully guarantee the accuracy of the information [19][24][196]. These data-driven approaches can be combined with network designs using sleep/wake up mechanism to further reduce the volume of transmitted information, hence reducing communication energy consumption. However, optimising the amount of transmitted data alone cannot minimise energy consumption caused by network and system activities such as radio activation, listening, transmission waiting, and network maintenance. It is therefore necessary to optimise the system to fundamentally reduce energy wastage due to unnecessary system and network activities.

Event-triggered mechanisms significantly reduce communication demands by initiating transmissions only when environmental variations exceed predefined thresholds, making them ideal for disaster early-warning (such as volcanic activity [21] and forest wildfires [22]) and target-detecting (such as animals [184])applications. In WSN systems applying event-triggered mechanisms, to construct multi-hop networks, nodes responsible for forwarding still need to continuously or periodically switch on the radio to successfully receive potential radio data. However, the energy consumption resulting from periodic listening can be optimised. As introduced in [185], although leaf nodes are not fully rely on LPL for periodical listening, mains-powered backbone nodes must continuously listen to data transmissions from leaf nodes. This approach significantly reduces the frequency of required data transmissions, thereby lowering leaf node energy consumption and maintaining low latency. However, a drawback of such networks is that relay nodes must remain powered for extended periods or at least be activated frequently.

2.7 Sleepy Network Design for E-IoT System

Sleepy networks require that nodes should remain in sleep mode for as long as possible to conserve energy, and their network design frequently employs schedule-based wake-up strategies. In brief, nodes only wake up during predefined communication windows to complete data transmission, while performing other tasks such as data sensing or remaining in sleep mode during the remaining time. Glacsweb project [55, 56] is typical of this design approach, with node communication activities activated only once daily at scheduled times. The primary advantage of this design is that overall system energy consumption can be reduced by increasing node sleep time and limiting the number of times the radio switches between on and off states. However, this mechanism also introduces additional challenges. Firstly, communication windows may be over-allocated to address uncertainties arising from contention, network maintenance, and data exchange delays. Additional, if large numbers of nodes attempt to transmit data at the start of the synchronous wake-up communication window, this may lead to collisions and retransmissions. Conversely, if the communication mode applies a request mechanism where parent nodes or base stations query data from child nodes during the wake-up period, some child nodes may remain in a radio-on state for extended periods awaiting requests. This results in unnecessary idle listening and waiting, consuming energy unnecessarily. However, for applications with lower latency requirements, if the minimum time needed to complete radio activities (including the exchange of network maintenance control messages and data transmission) can be accurately estimated, the wake-up timing and communication window of the Sleepy Network can be further reduced.

This work considers sleep/wakeup approaches are essential for the development of WSN systems. As reviewed in Section 2.5, a number of existing ESN deployments demonstrate that system designs often combine multiple energy-saving mechanisms. As an example, overall energy consumption can be further reduced by integrating the non-duty cycling approaches discussed in the last section with sleep/wake-up strategies. Nevertheless, even with the application of multiple energy-saving technologies, WSN systems still need to minimise unnecessary network activity wherever possible. During long-term deployments in remote or hazardous environments, power resource is often seriously constrained, making the reduction of system energy consumption a primary design objective. To achieve this, the fundamental requirement is to restrict unnecessary node activity, including frequent radio on/off switching, idle waiting before transmissions, retransmissions, and continuous network maintenance.

This study considers that the sleepy network is potential to address these challenges by allowing all nodes within the network to enter a sleep state as longer as possible. As the network remains inaccessible outside predefined communication windows, this design inevitably introduces higher communication latency. However, the associated benefits of sleepy network are equally evident. It is possible to enable an energy consumption reduction if the duration of node activities can be kept short and tightly bounded. Existing literature has not adequately explored this aspect, and this study aims to address this research gap.

2.7.1 Sensor and Targeting Environment

In general, sensors applied in corresponding project are strongly correlated to the targeting environment/deployment location and the objectives of the project. For example, air quality and gas sensors are always deployed to monitor the air conditions of artificial environment such as urban area and indoor environments. Moreover, when the objectives of an environmental monitoring task involve some specific parameters such as geological target parameters and characteristic parameters for disaster early warning, particular or even specialised sensors will be selected. The hardware and software platforms chosen for the nodes must be guaranteed to be able to support the sensors on board. In addition, the increased number and types of sensors utilized can result in several potential problems, such as 1) the time a single node spends per sensing will increase which will affect the data sampling time cost, 2) the payload during data transmission will increase relatively and it can result in various effects on communication. Therefore, if a project's environmental detection target or the number of sensors equipped on a single node can be predetermined, the project design will be able to adjust the timing of each activation of the sensing activity and determine the time required for a data transmission according to the needs.

2.7.2 Data Delivery Mode

In terms of data delivery model, these models can be categorised into event-driven, query-driven, continuous transmission and time-driven. It can be identified that event-driven and query-driven model can be applied into disaster monitoring (e.g., volcanic activity and wildfires). In disaster monitoring or early warning systems, nodes can monitor the environment at a fixed data sampling rate and transmit data only when a particular environmental parameter exceeds a predetermined threshold. Depending on the design, user can also send requests to nodes asking for a response of sending environment data based on requirements, or schedule the transmission in a fixed time interval. Alternatively, when the hazard requiring early warning and leaves a short response window, system can continuously sample the environment and send data in real-time. Therefore, continuous transmission model is commonly used in real-time environmental monitoring applications to provide accurate and time-sensitive data on rapidly changing environmental parameters. To maintain a reliable data transmission, this model usually relies on mains electricity supply and communication methods with uninterrupted transmission capability, such as Wi-Fi, Ethernet, cellular networks, satellite links, and stable IT infrastructures.

The time-driven model is one of the most commonly used data delivery models for the projects analysed above. Characteristic of this model is that data is transmitted at a fixed or predetermined rate, ensuring consistent monitoring while optimising power consumption. A significant advantage of this model is that it reduces the energy costs associated with frequent activation of communication modules, as transmission schedules are pre-defined rather than real-time. By controlling data exchange frequency, network activity and energy consumption are also reduced to a certain extent. This work will focus on the time-driven mode which is therefore suitable for long-term environmental monitoring systems in which the regular collection of data is sufficient for analysis, such as glacier monitoring and climate observation networks. However, although predefined transmission schedules can help to reduce energy consumption, such schedules cannot be changed after the system has been set up. This means that if the frequency of sensor sampling and node communication needs to be changed in response to environment changes, a time-driven system may not be able to respond immediately, and all nodes need to be recovered for redeployment.

2.7.3 Data sensing and communication

The sampling and communication rate are also corelated to the energy saving of the network system. Generally, nodes are designed to complete predefined task (e.g. sensing, transmitting data) and then enter low power mode to save energy. Batteries, mains power, back-up batteries and solar panel charging are the four main solutions for powering nodes. For ESN project deployed in artificial environment, such as the environmental monitoring of the human living area and its surroundings, the system design is less constrained by the power supply and maintenance difficulties. As the infrastructure of power, communication, etc. are available in artificial environment, the environment in which the network end nodes are deployed can be characterised by the following features: 1) Power supply can be uninterrupted or battery operated, while solar charging is optional, 2) LPWAN technology is commonly considered for application in such cases, especially ZigBee and LoRa, 3) The data sampling and communication rate of the node can be relatively high or even continuous, if the deployment environment has sufficient energy sources to harvest energy. On the other hand, in resource-limited deployment environments, batteries and solar panel charging are widely applied to ensure power supply and extend lifetime as long as possible. Considering to the energy saving, nodes are usually designed to switch the operation mode between low-power sleep and wake up mode to optimise the energy usage in resource-constrained networks. Additionally, in the analysed resource-constrained long-term environmental monitoring projects, it is identified that the most frequent the data sampling rate is every 10 minutes, and transmission rate is ranging from every 30 minutes to once a day.

2.7.4 Energy Saving and Design Trade-offs

Sufficiently large batteries and energy harvesting techniques like solar power, can indeed increase the theoretical lifetime of any system. However, while energy harvesting can extend the operating time of a system, it does not reduce the energy consumption of the system. Regardless of the availability of solar or other energy sources, the overall energy consumption of a system remains a key criterion in determining its performance and reliability. If the system itself is not optimised for low-power operation, high energy consumption can still lead to power shortages during periods of low energy availability (e.g. cloudy days or nighttime). Therefore, it is important to consider energy efficiency when selecting network protocols in such systems to minimise power consumption while ensuring reliable communication and data transmission. By enabling low-power network protocols, it is possible to improve overall energy consumption efficiency, ensure system performance, and extend the lifetime, even in energy-constrained environments. This project will attempt to apply a variety of energy saving approaches to the network design, so that overall system energy consumption can be optimised whilst maintaining the performance of the sleepy network.

In terms of power management, sleepy network is controlled by algorithms that default the data sampling rate and communication rate, controlling nodes to activate only within predetermined activity windows. The time synchronisation is typically achieved using mechanisms such as Real Time Clock (RTC) devices, which ensures precise wake-up timing without requiring complex time synchronisation protocols. To address hardware limitations including clock drift and clock disorder, sleepy networks may re-synchronise and update system parameters by reserving periodic maintenance windows, enabling the network to adapt to changing requirements during long-term monitoring. Referring to the various ESN projects reviewed in section 2.5, this work considers that sleepy network designs are more suitable for sensing tasks where the monitored objects evolve slowly and communication delays are acceptable, such as long-term environmental change monitoring. In such applications, data collection is typically periodic, communications are predictable with tolerable delays. However, when deployed in the real world, some extreme scenarios may challenge the reliability of sleepy network, such as errors in node time synchronisation or unexpected node disconnections. For instance, when a node experiences a time synchronisation error, it may initiate communication at an unexpected time, thereby disrupting the scheduled communications of other nodes. Alternatively, when a node's energy runs out, the loss of that node may severely impact the existing topology and pre-established data transmission paths.

On the other hand, Sleepy networks can also been applied in disaster monitoring and early warning scenarios, such as long-term water level monitoring [163] and tsunami detection and alert systems [161]. However, for detecting early warning signals of disasters, the periodic data transmission mechanism based on fixed communication windows may fail to promptly reflect sudden environmental changes, thus introducing delays between event perception and reporting. Moreover, in large-scale deployed multi-hop networks, there is no guarantee that end-to-end routes remain available at any given moment if event-triggered data are collected at child nodes due to the unreachability of relay nodes outside communication window. Further, when disaster events generate a burst of data, the preset length-limited communication windows in sleepy networks may not be able to handle the large volume of data simultaneously generated by nodes within a short period. This can cause data collisions, increase delays, and potentially lead to data loss. For such scenarios, systems could adopt hybrid data transmission modes, as employed in the DART project [161], combining periodic and event-triggered approaches to enhance responsiveness to sudden incidents.

Moreover, although Section 2.6.2 reviewed data-driven energy-saving approaches, such techniques are not the primary focus of this work on sleepy network design. Data reduction, aggregation, prediction and event-triggering mechanisms may reduce the volume of transmitted data and thereby lower communication energy consumption. However, network behaviour impacts the overall system energy consumption, which cannot be fundamentally eliminated through data volume optimisation alone. Furthermore, data-driven approaches typically introduce additional system complexity and uncertainty, which further challenges the maintenance of remotely deployed ESN systems. Therefore, this study prioritises minimising unnecessary radio activation and communication activity as far as possible, while ensuring essential network maintenance activities are maintained. Data-driven energy-saving techniques may be integrated with the proposed system design to further improve energy efficiency in future work. However, this work does not currently consider introducing additional algorithmic complexity by optimising the volume of transmitted data and instead focuses on reducing unnecessary communication activation.

In term of sleep/wake up energy saving approaches, compared to sleepy networks, the duty cycling approach offers better flexibility in data transmission. The duty cycling MAC protocol maintains network availability by activating the wireless transceiver controlled by MAC protocol, which also allows for more timely data exchange. In practice, duty cycling schemes may be capable of achieving extremely low duty cycles. For example, the Mountain Sensing project using ContikiMAC demonstrate a duty cycle close to 1% [54]. If the duty cycle is reduced further to ultra-low levels, the energy cost associated with frequent radio state transitions may become less dominant. Furthermore, to overcome the limitations, researchers have proposed adaptive duty cycling MAC protocols, such as the Duty Cycle Learning Algorithm (DCLA) and the IEEE 802.15.4 Adaptive MAC Protocol (AMPE)

[186]. These protocols dynamically adjust parameters based on observed network conditions, and they provide improved responsiveness to traffic fluctuations and can further reduce unnecessary energy consumption under different workloads. Compared to sleepy networks, adaptive duty cycling protocols are more suitable for applications requiring timely data transmission or frequent interaction. However, these advantages come at the cost of sustained or frequent radio activity.

Frequent radio activation in the duty cycling scheme can support mechanisms such as low-power listening, sensing data transmission, and network maintenance (routing updates and neighbour discovery). These activities will continuously consume energy within the system and cannot be further restricted after deployment. Moreover, in long-term deployments within remote or hazardous environments, the duty-cycling-based system is typically pre-configured, making it difficult to change system parameters after deployment to respond to evolving monitoring requirements. Furthermore, it remains unclear whether sleepy networks can achieve better overall energy efficiency compared to duty cycling schemes. Therefore, it is necessary to compare the energy consumption of systems employing the same network stack when applying sleepy network and duty cycling schemes (ContikiMAC and 6TiSCH respectively) in next chapter.

2.7.5 Network Protocol Selection in ESN Systems

E-IoT system requires combining matched and suitable technologies to construct the application-based system. Thus, selection of proper protocols and technologies to collaborate with each other in the system has potential research value. This section discusses the selection of potential network protocols for E-IoT systems reviewed in Section 2.3, focusing on their suitability for low-power, IPv6 supports, and interoperability. In practice, customised algorithms and protocols are common in ESN deployments, such as the sensor web and sea-lab projects. These customised solutions can meet user requirements for system performance in specific deployment scenarios, but they also reduce reusability and increase system complexity. Therefore, this section will focus on the selection of standardised low-power network protocols for system design.

Systematically, real world E-IoT application are required to be able to face various environment especially the hazardous environments. Thus, essential system design should consider the implementation environmental challenge firstly to ensure reliable system performance alongside the designed system lifetime. To address the real-world E-IoT system implementation challenge, researchers has indicated numbers of key factors which will eventually affect the system performance, power consumption is one of the most critical elements that will decide the system lifetime and reliability [3].

Key requirements for this study include low power consumption, scalability, reliable communications and bi-directional communication capabilities. Low power consumption is critical because systems can be deployed in hazardous/remote areas where battery replacement is unpractical. This study wants to design an E-IoT system for environmental monitoring targeting at hazardous/remote areas. This system design wants to reduce the system power consumption by applying low power protocols and low-power-sleep/wake-up switching. By applying low-power protocols standardised for sensor networks and IP-based protocols [2], the system can enable full internet connectivity, therefore enhance the interoperability of the system. On the other hand, the objective of this study also involves exploring how to efficiently design network communications and schedules based on the applied low-power protocols. Thus, system should effectively reduce unnecessary energy consumption (e.g., reducing unnecessary operating modes and switching on of radio modules). Moreover, interoperability is critical for flexibility of the system which ensures the system works with a wide range of devices and technologies. Finally, bidirectional communication is also one of the design objectives, which allows users to interact with the network for remote configuration and control.

ZigBee, LoRaWAN and NB-IoT have been introduced in 2.3.4. More specifically, when discussing protocol compatibility with the Internet and IPv6, technologies and protocols such as ZigBee and LoRaWAN are lacking natively. To realize IPv6 works over ZigBee, one straight forward method is to add IPv6 stack on top of the ZigBee network layer, or it can apply dual stack approach of another. However, adding additional IPv6 stacks will further compress the size of the ipv6 packet (up to 54 bytes of data assuming 40 bytes header). Also, as summarised by Wang et al. [187], ZigBee over IEEE 802.15.4 does not support packet fragmentation, and it also need additional IPv6 stack to enable ZigBee over with IPv6. Topology wise, Sigfox and LoRaWAN are relied on the gateway to forward packet in a star topology, which limits direct device-to-device (D2D) communication and flexibility of the network. For environmental monitoring of remote/hazard area, this study needs a solution that does not rely on the GSM and LTE networks. Thus, NB-IoT will not be considered in the network for this project. As discussed before, there are two critical reasons that IPv6 should be enabled in the network of this project 1) to support flexible bidirectional communication between the network and the user and 2) to address the IPv4-based address shortage problem. As mentioned above, even though in later releases they all provided IPv6 support via new technologies, ZigBee, Sigfox and LoRaWAN are not natively support seamless Internet integration and IPv6 standard. ZigBee can support IPv6 by applying an additional IPv6 stack or dual-stack approach, which allows it to support IPv6 although it also increases the complexity and payload on the network.

On the other hand, 6LoWPAN can effectively enhances the IP connectivity over networks and compresses the header to increase the communication efficiency, which makes it a suitable option for resource-constrained networks. Besides, to achieve a seamless and scalable E-IoT network, the system must be based on IP. Besides, from the application layer to the physical layer, various low-power protocols can be applied to the system network stack. CoAP has low complexity, and it is lightweight as it uses UDP, not connection-based TCP protocol. To enhance packet reliability, a CoAP transaction can optionally enable the confirmable message that require receiver to reply with an acknowledgement within the timeout. Also, CoAP header contains "Message ID" which is to avoid duplicate transmissions. In terms of scalability, the low-power routing protocol RPL is an advantageous choice and has been used in combination with CoAP and 6LoWPAN in deployed projects.

2.8 Summary

In this chapter, the historical progression of IoT technology was reviewed at the beginning. The development of internet technologies, especially the technological advanced shifting from IPv4 to IPv6, enabled the connectivity of a massive numbers of devices. This chapter reviewed typically network protocols and standards that support the development of E-IoT systems.

Commonly, radio frequency used in WSN is based on ISM bands which are sub-GHz and 2.4 GHz band. To balance the communication range, data rate and the influence of the obstructions, it is optimal to operate the radio at 868 MHz frequency band for this work. Also, this literature review has shown that while non-IP solutions and LPWAN technologies can achieve low power consumption and long-range communication, they often rely on centralised architectures and lack native IPv6 support. In contrast, IP-based network stack built on 6LoWPAN combined with RPL and CoAP, provide a more flexible and interoperable foundation for E-IoT systems. However, the communication overhead generated by the protocols, such as listening and network maintenance, makes efficient communication scheduling particularly crucial. This chapter further applied eight characterisation parameters to analyse and describe the requirements and characteristics of ESN systems in five categories.

In terms of energy saving technologies, this work considers sleep/wakeup approaches are essential for the development of E-IoT systems. In existing research, duty cycling approaches have been widely discussed and applied which reduce energy consumption by periodically switching the radio module on and off. However, such methods typically encounter two evident limitations. Firstly, frequent radio switching introduces additional energy overhead and radio activities controlled by protocol are difficult to constrain precisely. Secondly, predefined parameters are difficult to adjust flexibly after system deployment even with ultra-low duty cycle, which limits the system's adaptability to environmental changes and application requirements.

By contrast, sleepy networks maintain nodes in a sleep state for most of the time by maintaining them unreachable outside predefined communication windows, theoretically offering greater energy-saving potential. However, it remains unclear whether sleepy networks can achieve better overall energy efficiency compared to duty cycling schemes, and this work will compare the energy consumption between them to fill this gap. On the other hand, sleepy network is suitable for WSNs deployed long-term in remote or hazardous environments, where system must minimise unnecessary node activity wherever possible, including frequent radio state transitions, pre-transmission idle waiting periods, retransmissions, and network maintenance. Moreover, the sleepy network design for E-IoT systems enables scalable bidirectional communication under IP-based standards, whilst allowing system parameters such as data sampling and communication rates to be modified after deployment via downward data flows. Other energy-saving approaches for WSNs have also been reviewed, including radio optimisation, energy harvesting, energy-efficient routing, data reduction, and event-triggered communication. Although practical system designs typically combine multiple energy-saving mechanisms, it remains essential to minimise unnecessary network activities.

On the other hand, the design of sleepy networks still faces several resolved issues. Using the Glacsweb project as an example, the synchronised wake-up mechanism applied in this project introduces new challenges: when numbers of nodes attempt to transmit data at the start of the communication window, it may lead to channel contention, collisions, and retransmissions. Furthermore, when base stations request data from child nodes, child nodes may need to keep their radios active for extended periods while awaiting queries, resulting in unnecessary idle listening and energy consumption. Nevertheless, the energy-saving potential of sleepy networks will depend on the precision of constraints on the duration of node activities. Existing literature has not deeply investigated this aspect, and this work aims to address the gaps.

Chapter 3 Design and Implementation of Sleepy Networks in an E-IoT System

As previously introduced, when considering the design and implementation of E-IoT, the system design should consider the collaboration of different protocols, technologies and allow bi-directional access between users and sensor nodes. When building such a system, it is important to consider the cooperation of the various hardware and protocols, to reduce/optimize the energy consumption of the system to extend the node lifetime. To achieve low power consumption, nodes can save energy by sleeping components such as MCUs, radio transceivers, etc. when they are not needed. Of the five energy saving approaches summarised and classified by Alsharif et al [6], this work will focus on the sleep/wake scheduling approach. Specifically, the sleep/wake scheduling method saves energy by adjusting the frequency and duration of nodes' wakeups to a certain ratio to complete a given task (e.g., sensing and packet sending/receiving/forwarding), and minimizing the possibility of node idling.

Subection 2.7.5 discussed network protocol selection for ESN systems used for remote and hazardous area environmental monitoring. The target system and network design should be IP-based, which ensures seamless integration with the Internet and future scalability. 6LoWPAN was chosen because it offers IPv6 support, effective header compression and low power consumption for resource-limited networks. RPL and CoAP will also be applied in this sleepy network design as discussed above. The selected protocols aim to enable Internet connectivity for the resources-limited devices in a large sensor network with power-efficient routing and enable the cooperation of IPv6 to IEEE 802.15.4. In terms of operating system selection, RIOT-OS was selected for its modular design, multi-threading support and compatibility with low-power IoT protocols. Potential OSs that can be applied were reviewed in section 2.4. RIOT-OS also has good modularity design and wide hardware/low power protocol support. RIOT-OS supports a wide range of development boards that support 868 MHz radio, including TI CC13xx series LaunchPad (with CC1100 transceiver), SAM R30 Xplained Pro (SAM R30 Xpro).

This work aims to design a low-power, long-range IPv6-based system, which supports standardised low-power protocols such as IEEE 802.15.4, 6LoWPAN and RPL. To address the design requirements, SAM R30 Xpro [188] was selected to develop the system which is based on a low-cost system-on-chip (SoC) ATSAMR30. Additionally, to design an E-IoT system with optimal energy consumption performance, scheduled communication would be designed to allow the system to transmit messages within a communication window and turn to sleep to save power after completing scheduled tasks. This chapter requires the construction of a schedule-based sleepy network system.

Considering the complexity of the system and the profiling and modelling of the system's energy consumption required in the work of Chapter 5, this system design is equipped with a single sensor that is common in analysed ESN projects - a temperature sensor.

It is noted that the control message for maintaining RPL routing table could affect the power consumption and will be discussed in this chapter. In addition, this chapter will also compare energy-efficient MAC protocols with the sleep/wake scheduling approach of sleepy networks, focusing on their power consumption differences when building environmental sensing networks. Part of the work presented in this chapter has been published in [189] and reproduced with permission from Springer Nature.

3.1 Design Problem and Scope

In the Sleepy Network design, if all nodes turn on their radios simultaneously, it is necessary to minimize synchronisation overhead by utilizing RTC devices. Additionally, since nodes do not generate any radio activity while sleeping, so before sending data after nodes wake up, it is necessary for the node to complete some radio activities to maintain network such as neighbour discovery and routing information updates [190]. While advanced MAC protocols and wakeup signals/beacons-based technology may reduce the energy consumption associated with frequent radio switching on, the focuses of this work are to determine how the duration of the sleep time affects the network activity when a node wakes up, and how sleepy the network can be, without additional overhead and complexity.

Therefore, the primary challenge in sleepy network design is to minimize communication conflicts in both synchronous and asynchronous scheduling. Schedule-based sleep/wake up schedule-based scheme allows nodes to enter an active state based on a predetermined schedule, without requiring additional wake-up signalling mechanisms and algorithm. To achieve this, it is essential to determine the minimum required communication time (for a fixed payload size) to avoid unnecessary communication overhead. In addition, the network design needs to ensure the timing of basic network activities such as routing information updates while ensuring minimum energy consumption. Thus, it is important to design an efficient schedule-based algorithm that balances synchronous and asynchronous wake-up strategies. This work will rely on IEEE 802.15.4 MAC and implement a time schedule-based sleep/wake-up mechanism. The test code used for star topology sleepy network design has been uploaded to GitHub for public viewing, and the link was provided in Appendix B.1.

3.2 Design and Implementation of Sleepy Networks for E-IoT

The objective of this section is to analyse the energy consumption of a sleepy E-IoT system using CoAP, RPL, and 6LoWPAN based on RIOT-OS. This study aims to explore the impact of different design choices on the system's lifetime and how to precisely schedule the sensing and communication activities while optimizing the energy consumption of the system. The system features can be split into three aspects: (1) user-configurable system, (2) energy-efficient sleep schedule, and (3) adaptive energy-efficient RPL configuration. This E-IoT system should be able to allow the user to configure and change the features of the system, such as communication and sensing rate, depending on their requirements.

When designing such an efficient sleepy network, the fundamental task is to ensure the system is energy efficient while delivering data reliably in environment monitoring. With efficient collaboration between system software and hardware, all the components should be able to reliably work with the sleepy network schedule. It is notable that, certain level of network maintenance activities controlled by different protocols need to be completed before data transmission, but the time overhead associated with such maintenance activities may conflict with the requirements of shorter communication window. Thus, by applying CoAP, RPL and 6LoWPAN in this system, it is essential to identify the possible system activities during sensing and communication slot and the corresponding energy consumption. Therefore, the essential task is to determine all the expected activities for the sensor node during sensing and communication state.

To be able to clearly analyse the network activities and measure the energy consumption of each activity within sleepy network system, this section focuses initially on a simple star topology rather than introducing multi-hop tree topologies. Under such constraints, this section measured and analysed the energy consumption of various activities involved in the sleepy network implementation.

3.2.1 Network Stack

In the previous section, an initial verification of the power consumption performance of a sleepy network based on scheduling was provided, and the results also showed the potential of sleepy networks. For a brief recap, 6LoWPAN, RPL, and CoAP are used in the system. Also, these low-power protocols allow devices to form a tree topology, reduce energy consumption, and ensure robust communications. The RPL standard [96] extends the range of the network and supports scalability, making it well-suited for constrained environment deployments. CoAP provides a simple RESTful request/response interaction model and supports features to make it possible for devices to reply to data requests as well as respond to configuration changes. This IoT network protocol stack is shown in Figure. 3.1 below.

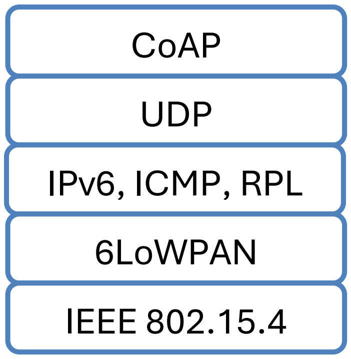


Figure 3.1. Network protocol stack (reproduced from [189]).

3.2.2 Experiment Setup

To verify the reliability of the implemented scheduling mechanism and observe the network activity, a simple experiment was performed in the lab. In this test, node was periodically switched between a sleep state and an active communication state, simulating the scheduled operation of a sleepy network. Both the sleep and communication durations were set to one minute. The purpose of this experiment is to measure the activities of nodes during different states, along with the current and duration of the associated activities. Taking Figure 3.2 as an example, it illustrates the SoC current level changes during the periodic switching between sleep and communication states of the node.

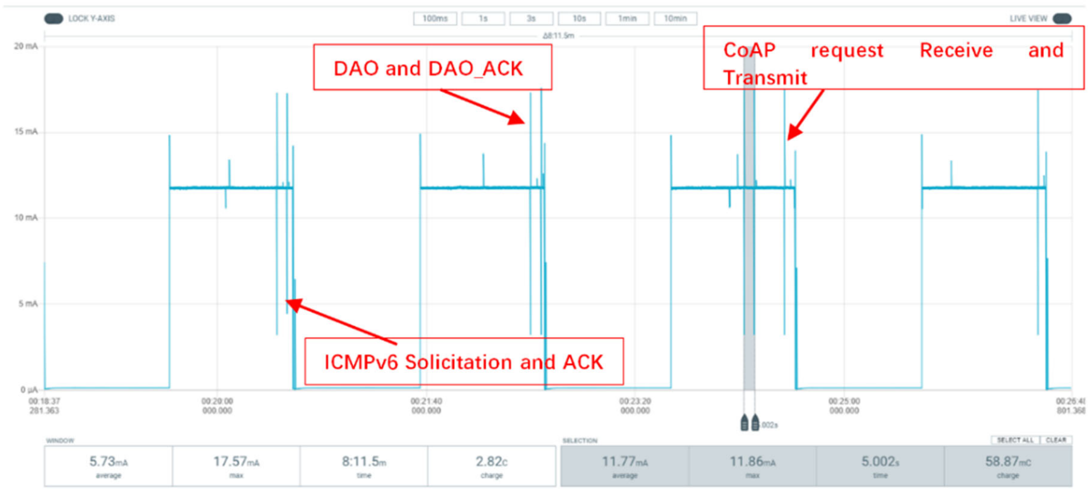


Figure 3.2. Current Measurement of SoC (reproduced from [189]) - The periodic sleep/wake-up switching period is 1 minute.

In this section, to evaluate the system energy consumption of sleepy network design, this study implemented sensor nodes integrated sensing, local data storage, and precise RTC to fully support real-world operation (demonstrated in Figure 3.3). In this experiment, Microchip SAMR30 Xpro [188] was selected as this hardware platform. It has an ultra-low power ATSAMR30G18A SoC which supports sub-GHz (868 MHz band) radio (AT86RF212B integrated transceiver). The embedded AT86RF212B transceiver was configured with channel number = 0 and channel page = 0 to obtain higher receiver sensitivity (BPSK-20kbps with -110 dBm) [188]. Also, due to the limited MCU flash memory which only has 256 kB, so an I/O1 Xpro SD card extension board was used to provide sensor data storage/buffering. Also, a DS18B20 Digital Temperature Sensor was used with its default 12-bit resolution supported by RIOT-OS (the conversion time is constrained by the driver leading to typically 750ms sensing time). For sleepy networks, the accuracy of RTC can affect the synchronisation and schedule timekeeping, as clock drift directly impacts wake-up synchronisation and the alignment of communication windows. While the embedded crystal of SAMR30 Xpro provides the functional RTC with an accuracy of ±20 ppm, to provide improved synchronisation reliability, DS3231 was applied to enhance the accuracy of synchronisation of wake-up and scheduling (±2 ppm from 0°C to +40°C, and (±3.5 ppm from -40°C to +85°C). By applying the DS3231, assuming that the working temperature is between 0°C to +40°C, the maximum time drift per week is about 1.2s. Compared with using the SAMR30 Xpro's built-in oscillator (about 10% as accurate as the DS3231from 0°C to +40°C), using DS3231 can enhance the synchronisation reliability, as the DS3231 provides enhanced accuracy across a wide temperature. It is also notable that the time reading of DS3231 RTC supported in RIOT-OS is second precision and epoch time is used for time synchronisation.

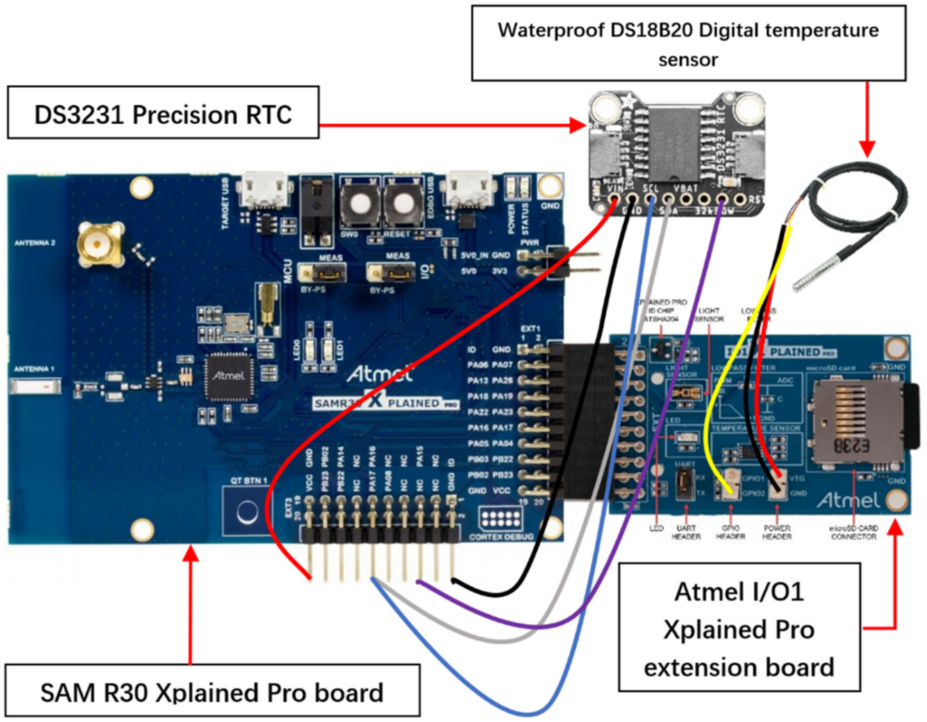

Waterproof DS18B20 Digital temperature sensor
DS3231 Precision RTC
SAM R30 Xplained Pro board
Atmel I/O1 Xplained Pro extension board

Figure 3.3. Node structure in this experiment (adapted from [191] [192] [193]).

The experiment configuration is shown in Figure 3.4, GNRC network stack, 6LoWPAN, RPL, and CoAP are all provided by RIOT-OS. Radio packets were captured by RIOT-OS sniffers to help correlate activity timings with current consumption changes, simultaneously measured using a Nordic Semiconductor Power Profiler Kit II and Agilent 34411A multimeters. The initial phase involved investigating the current consumption of the fundamental networking activities controlled by GNRC network stack and CoAP packet. Subsequently, the experiment enabled RPL to explore the characteristics and current consumption of the various RPL control messages. Finally, all the different activities in various states (e.g., sleep, sensing, and communication) were measured and repeated multiple times under identical configurations. In repeated experiments, the types of protocol-driven network activity under communication states are consistent. Although each experiment showed slight variations in time and current measurements, these were attributed to measurement noise and timing jitter. Overall, these variations did not affect the identification of network activities or the calculation of average energy consumption.

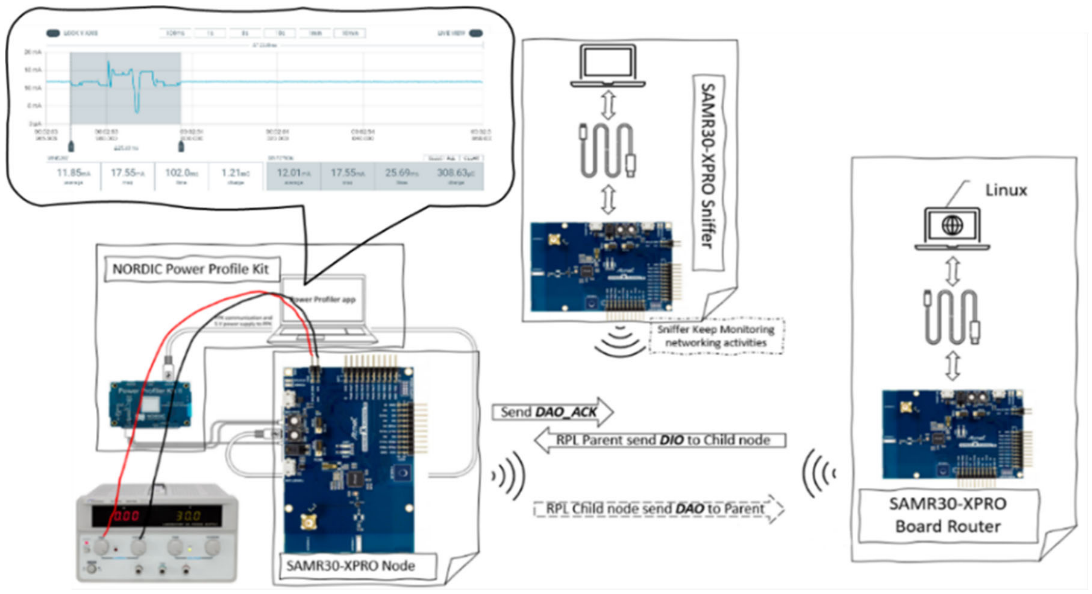


Figure 3.4. Example of system current measurement experiment (reproduced from [189]).

Figure. 3.5 shows an example of an RPL star topology DODAG. In the current implementation, nodes are firstly deployed in a star topology with one Border Router (BR) as the parent. This simplified topology allows all nodes to communicate directly with the BR without considering complex multi-hop routing structures. At the same time, starting from designing a system with a star topology can help this study to clearly analyse the radio activity patterns and efficiently design schedules. By applying this method, it is aimed to build and validate an initial sleepy E-IoT scheduling strategy based on real-world timing and power measurements of the system activities.

The exchange of the RPL control message is periodic and controlled by the RPL "Trickle Timer" which has four parameters:

(1) I_{min}: the minimal interval of DIO ($I_{min} = 3$ in this experiment which equals 8ms).

(2) I_{max}: the maximum time interval between DIOs ($I_{max} = 20$ in this experiment, approximately 2.33 h).

(3) K: the constant determines doubling the interval between transmissions ($K = 10$ in this experiment).

(4) DIO timer: Interval for DIO sending is 0, I_{min}, ..., I_{max}.

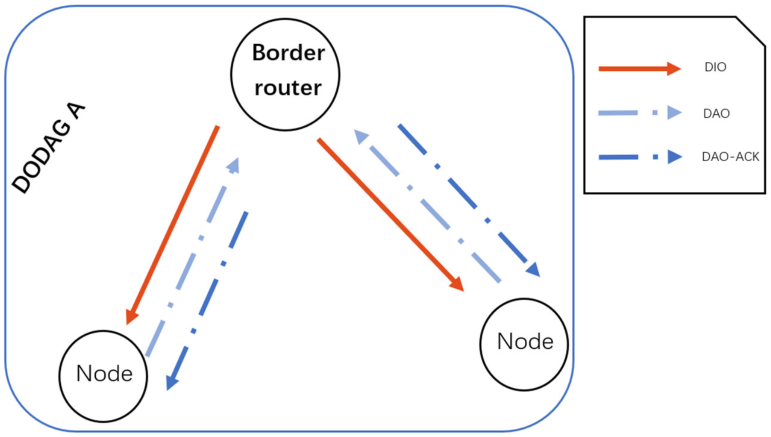


Figure 3.5. RPL node star topology-DIO & DAO exchange and DODAG (reproduced from [189]).

3.2.3 Schedule in Star Topology

A typical Sleepy E-IoT system schedule is designed to balance energy efficiency and data availability by structuring the transition between sensing, communication, and sleep states. As illustrated in Figure 3.6, the system follows a structured schedule based on the configured communication frequency. For different communication frequency case, If the system is set to communicate once per day, a communication window opens at the start of the cycle, followed by a sensing window. Once data is sensed and transmitted, the device transitions to a sleep state to conserve energy. When more frequent communication is required (e.g., four times per day), the system evenly distributes communication windows throughout the day (e.g., every 6 hours). After each communication session, the system switches to the sensing state and then enters sleep mode. If more frequent communication is required, for example the communication frequency is once per hour**,** the radio module activates at the beginning of every hour, transmits data, then powers down to minimize energy consumption.

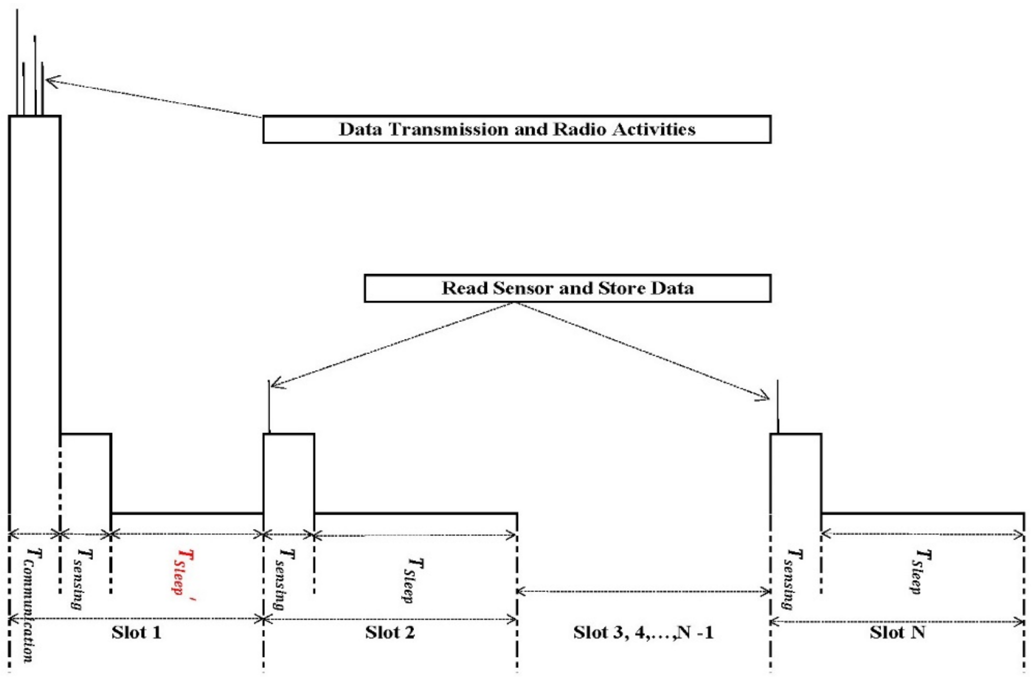


Figure. 3.6. Sleepy E-IoT system schedule overview (reproduced from [189]).

As Figure 3.6 shows above, the system will start to sleep after all the activities are finished. This design ensures that devices only wake up as scheduled according to the requirement, and it can reduce the unnecessary power waste while still maintaining sufficient data availability for the application. Additionally, System configurations such as the sensing rate and communication rate are changeable by sending a CoAP PUT to the nodes from the Linux PC. By effectively coordinating sleep schedules and communication intervals, this star topology-based sleepy E-IoT system can foundationally balance the energy efficiency and network requirement.

The nodes are synchronised by sending a CoAP PUT from the Linux PC to set the RTC of each node. This initial configuration allows users to accurately set the RTC timer for all nodes. Once the RTCs are configured, all nodes seamlessly enter the synchronised schedule at the commencement of the next hour. This synchronisation strategy supported by DS3231, compared with using the board's built-in oscillator, improves the synchronisation accuracy of every node to operate within the schedule. This synchronisation method ensures that the entire system functions operate cohesively and reduces potential risks in timing and activity. Currently, all the nodes are synchronised to wake up sequentially, where they take turns to communicate. Thus, in this experiment, the energy consumption caused by overhearing is minimised in the laboratory environment. In the next chapter, optimised synchronisation process will be explored to further facilitate the initial setup, fundamentally supporting the overall efficiency of the system.

3.2.4 Observation and Measurement of Node’s Activities

Two scenarios were considered: (1) a predetermined data sampling rate with one daily communication window, and (2) a predetermined data sampling rate with regular communication windows distributed daily. The BR was powered by an always-on Linux PC. Thus, the measurement focused on the energy consumption analysis of the individual nodes. In this experiment, the Linux PC sent CoAP GET requests to the node to retrieve their data. By correlating the packet capture results using the RIOT-OS sniffer with the measured current profiles, various network activities were identified, including RPL-related control messages, neighbour discovery messages, and CoAP messages. Next, this section accordingly presented the results observed and measured under the experimental setup described in subection 3.2.2. Firstly, Table 3.1 provides an overview of the involved activities for the sensor node. The observed network activity is driven by IPv6, ICMP, 6LoWPAN, RPL and CoAP.

Table 3.1. Node’s activities overview (reproduced from [189]).

State	Activity	Description
Sensing	Set system to sensing state	Node wake-up from sleep mode
	Read sensor and store data	DS18 temperature data read and stored on the board or SD card
	Sensing state	Node stays in IDLE mode and radio is off
	Set system to sleep state	Node enters sleep
Communication	Turn off the radio	Set transceiver to OFF
	6LoWPAN activity and Router Advertisement	Node sends 6LoWPAN and Router Advertisement packets to BR
	Neighbour Solicitation	Node sends Neighbour Solicitation to determine Neighbours
	Neighbour Advertisement	In response of receiving Neighbour Solicitation
	DIO	DIO control message activities of the node
	DAO-Tx and DAO_ACK-Rx	Node sends DAO message to BR and receives DAO_ACK from BR
	CoAP Response	Receive CoAP request and respond
	Communication	Node stays in IDLE mode with radio on
	Set system to sleep state	Node turns radio off and enters sleep state

Energy consumption was measured across three states: Sleep, Sensing & Communication. Specifically, during the sensing and data storage state, the node takes readings from the DS18B20 and stores the data. During the communication state, the focus of network activities is ICMPv6 control messages, DIO, and DAO/DAO_ACK managed by the GNRC network and RPL routing protocol. All activities' current consumption and time duration were measured and listed in Table 3.2.

In the experiments, the on board Xplained Pro Analog Module (XAM) was configured in the MCU and peripherals measurement mode, under which measures the current consumption of both the MCU and all connected peripherals simultaneously. As a result, the measured sleep current corresponds to the overall sensor node configuration rather than the nominal low-power sleep current of the SoC alone. In particular, the integration of external peripherals, such as the I/O1 SD card extension board, could introduce additional standby currents as these peripherals might not be optimised for low-power operation in sleep states. Although the measured sleep current was higher than that of the SoC itself, it reflects the current consumption of the fully configured sensor node. In the subsequent works, all experiments applied identical hardware and measurement configurations. And the energy consumption associated with sleep states will also be indicated separately, as communication energy consumption is not directly affected by sleep state. However, it will still affect the overall lifetime of the node.

Table 3.2. Average current and time duration of different sleepy E-IoT system activities (reproduced from [189]) (see Appendix C.3 for the corresponding raw data).

Activity Information	**Time Duration (s)**			**Current (mA)**		
	Average	**Min**	**Max**	**Average**	**Min**	**Max**
Neighbour Solicitation	0.144	0.139	0.153	13.848	13.714	13.973
Neighbour Advertisement	0.142	0.137	0.150	13.788	13.677	13.938
DIO	0.093	0.0895	0.0963	12.524	12.354	12.705
DAO-Tx & DAO_ACK-Rx	0.1395	0.134	0.1443	14.702	14.454	14.940
CoAP Response	0.130	0.127	0.132	14.832	14.713	14.962
Set system to sensing state	0.049	0.0481	0.0495	1.476	1.470	1.483
Set system to sleep state	0.048	0.0476	0.0490	1.475	1.464	1.487
Turn off the radio	0.048	0.0475	0.0487	7.679	7.597	7.774
Set system to communication state	0.095	0.094	0.097	6.523	6.410	6.617
Read sensor and store data	0.798	0.791	0.805	2.963	2.953	2.987
Communication state	NA			12.85	12.546	13.012
Sensing state	NA			2.85	2.814	2.879
Sleep state– Standby Mode	NA			0.132	0.127	0.142

All measurements listed in Table 3.2 were obtained from repeated experiments conducted under identical conditions, with each activity measured at least three times. For non-radio activities, additional measurements were taken dur to their controllability. The values listed in the Table 3.2 correspond to the average, minimum, and maximum recorded in the repeated measurements.

Due to measurement noise and timing jitter, the different measurements data of an activity were subject to varying variations. For current measurements, the maximum deviation from the mean did not exceed 0.3 mA. For time duration measurements, non-radio activities showed relatively small deviations from the mean (ranging from 0.83% to 2.1%), while protocol-driven radio activities showed larger deviations (ranging from 1.53% to 6.25%). This behaviour is expected, as radio activities involve packet processing and encapsulation, as well as MAC protocol- controlled mechanisms such as CSMA-CA channel assessment and backoff. Consequently, the measured time duration of radio activities should subject to greater deviation. Network activities, such as neighbour discovery, showed a slightly higher variability. This was attributable to their influence from frequent DIO activities and MAC-layer mechanisms. By contrast, deviations observed for DAO transmissions during initial network establishment and for DIO transmissions controlled by the RPL Trickle timer were comparatively small. Overall, the time measurement deviations of radio activities discussed above are protocol-driven, and the measurement results demonstrate that the experimental setup and measurement procedure are stable and reproducible.

3.2.5 Preliminary Energy Estimation and Comparison of Sleepy Network, ContikiMAC, and 6TiSCH

Based on the current measurements listed in Table 3.2, a preliminary estimation of energy consumption is performed in this subsection for a system applying ContikiMAC, 6TiSCH, and schedule-based sleepy network. To ensure comparability, below estimations are all based on system network stack assumptions that using 6LoWPAN, RPL, and CoAP to perform same communication requirements using a 3.3V power supply. Although the types and typical occurrence patterns of radio activities have been characterised in Section 3.2.4, a normalised activity model is adopted here to enable a simplified and controlled comparison. Except for DIO messages, all the other network maintenance activities occurred regularly. These include other RPL-controlled routing maintenance and neighbour discovery messages, which are initiated every minute. In the experimental observations, DIO messages were transmitted frequently and irregularly, and up to 22 DIO messages in a minute were recorded exchanging between two nodes for routing establishment and maintenance.

Specifically, for this preliminary energy comparison, it is assumed that one DIO, DAO, NA, and NS message is transmitted per minute, while a single CoAP message is transmitted once a day. This assumption was adopted to enable a simplified and controlled comparison between the sleepy network approach and duty cycling mechanisms. Besides, at this stage, sensing-related energy consumption was excluded, as it would be identical across all configurations under a fixed data sampling rate. The purpose of this estimation is to provide an initial comparison of the energy impact of different MAC-layer approaches rather than a comprehensive system energy evaluation. It should be noted that in the energy consumption calculation and modelling process after this subsection, the observed maximum DIO exchange count (22 times) was used to derive a conservative upper bound for energy consumption.

It is notable that the packets' duration in Table 3.2 are longer than the theoretical transmission times for the corresponding Physical Protocol Data Unit (PPDU) timing for BPSK 20 kbps in the SAM R30 Xpro board datasheet (0.4ms/bytes) [188]. This is because the measured duration represents the systemic activity time which also includes time cost for MCU processing/encapsulation, buffer reading, 6LoWPAN compression and fragmentation, MAC-layer framing, and channel access delay due to CSMA/CA check. Therefore, the measured radio activity durations are not directly applicable for analysis the energy consumption of ContikiMAC and 6TiSCH. Therefore, for ContikiMAC and 6TiSCH energy consumption estimation in this subsection, the time duration of radio packets in Table 3.2 has been estimated using PPDU timing for BPSK 20 kbps (Detailed in Table 3.3 Below).

Table 3.3. Radio packet size and duration used for estimating the energy consumption.

Radio Packet	Packet Size (Bytes)	PPDU Time Duration (ms)
Neighbour Solicitation	64	28.8
Neighbour Advertisement	48	22.4
DIO	79	34.8
DAO-Tx & DAO_ACK-Rx	96 & 32(ACK)	54.4
CoAP Response	81	35.6

Based on these observations and assumptions above, assuming the ContikiMAC is implemented, the daily energy consumption of the system can be estimated as following (referring to the parameters of the Mountain Sensing Project [54]). By applying ContikiMAC, nodes need to repeatedly send the packet until the two successive CCAs are received as negative at receiver side. In this project, in order to estimate the energy consumption of the sleepy network system if ContikiMAC is applied, it is necessary to preliminary estimated the LPL energy consumption of this system (refer to [54]). In the Mountain Sensing Project, it gives a 0.63ms radio on-time in every second indicating a 1% duty cycle (CC1120) and the parameters of ContikiMAC timing constraints and internal ContikiMAC settings are listed in Table 3.4. Similarly, refer to the SAM R30 Xpro datasheet, the t_r is about 0.626ms (radio state transfer between SLEEP mode and RX Listen State) [188]. Then the CCA_CHECK_TIME can be used in this estimation is 0.991ms (including 8 symbol periods for CCA measurement with BPSK-20kbps), which lead to a duty cycle of 1.58%. With the default RPL parameter settings, RPL will send at least one DIO packet and one DAO message. For the default ContikiMAC settings of an 8Hz channel check rate and two CCAs per cycle, the energy consumption estimation when applying ContikiMAC is at least 99.54J per day where sleep energy consumption is about 36.95J (assume devices in always on refer to [54]) under 3.3V power supply. However, this estimation has not count the influence of repeatedly transmission of the sender as the average latency in Mountain Sensing Project is range from 209.77ms to 258.33ms [54] which has indicated a success rate could be about 20%.

Table 3.4. Useful parameters of ContikiMAC timing constraints and internal ContikiMAC settings reproduced from Mountain sensing project (adapted from [54]).

Parameters	Descriptions	Values
$\boldsymbol{t_r}$	Time for a stable RSSI (CC1120).	0.462ms [194]
CCA_CHECK_TIME	CCA Check time – duration for each CCA.	0.63ms
CHANNEL_CHECK_RATE	The frequency of checking the radio channel per second.	8Hz
Duty Cycles	Percentage of time a node listens to the channel in a cycle.	1%
$\boldsymbol{P_{avg}}$	Average power consumption with 3.3V supply.	0.73mW

Table 3.5. Default Parameters of 6TiSCH on OpenWSN network stack in RIOT [195].

Default/Suggested Parameter	
SLOTFRAME_LENGTH	101
Slot Offset Duration	20ms
EB Probability	10% (averagely, 10 tries before sending EB) [196]
Maximum Setting up Time (synchronisation)	101*20ms*10=20.2s (for a matched transmitter and receiver channel)
MAX_NUMCELL	Likely value is 2^n where n is ranging from 0 to 4 suggested by Chang et al. [197]
RDC	$T_{TX/RX}$ for time length of communication at node side and router side
$\boldsymbol{I_{TX/RX/SLEEPING}}$ Current usage	For current usage of communication and sleeping and estimating of power consumption
Number of Node in single DODAG set	Ranging from 0 to 5 same as [30] in a tree topology

A similar daily energy consumption estimation can be calculated if 6TiSCH has been applied in this system. Technically, system should initially exchange routing and networking information before synchronisation using 6TiSCH. Table 3.5 lists the default/suggested parameter of 6TiSCH on the OpenWSN network stack in RIOT [195]. Assume that the parameters of 6TiSCH follow the suggested value in Table 3.5, with the extended Slot Offset Duration which is 20ms and check duration of 0.991ms. Also, the estimated current consumption of EB is 12.85mA with 10ms duration. Therefore, if one DIO, DAO, NA, and NS are transmitted every minute, and one CoAP packet is transmitted once a day, the daily energy consumption estimation is about 47.25J where sleep energy consumption is about 37.54J.

During the initial implementation of the sleepy network in this work, frequent radio activity was observed after the transceiver was turned on. However, it is usually easier to observe the transceiver idle state after turning on for about 20 seconds. Thus, to estimate the energy consumption of sleepy network system preliminarily, the communication window is set as 20s. Assuming the power supply voltage is stable 3.3 V and daily communication window is applied, then the daily energy consumption for this sleepy network is 38.48J per day where sleep energy consumption is about 37.63J. Through estimation, sleepy network can save 61.4% energy than applying ContikiMAC and save 18.6% energy than applying 6TiSCH, and the daily energy consumption of different states for system applying sleepy network, ContikiMAC, and 6TiSCH are illustrated in Figure 3.7 below.

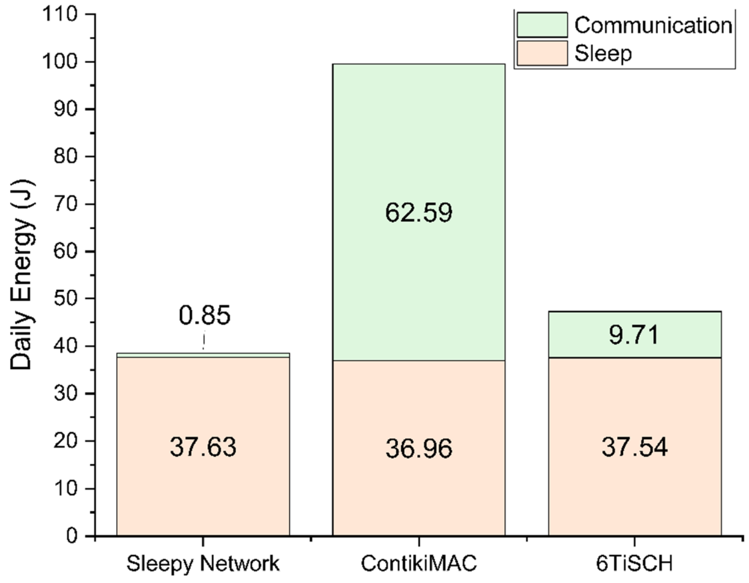


Figure 3.7. Daily energy consumption of different states for system applying sleepy network, ContikiMAC, and 6TiSCH.

Notably, the above energy estimations are based on the measured sleep current of sensor nodes in the experimental setup, rather than the low power sleep current of the SoC. In this context, Figure 3.7 illustrates that sleep energy consumption contributes a significant proportion of total daily energy consumption. In practice, many low-power wireless sensor nodes are designed to achieve microampere-level sleep currents. Assuming sleep current is so very low, the proportion of sleep energy consumption within daily energy usage would decrease significantly, while the impact of communication energy consumption on total energy would become more noticeable. Under these conditions, the impact of communication energy consumption can be observed more significantly. Assuming the sleep current is 1.86µA, which is consistent with the SAM R30 Xplained Pro board sleep current mentioned in Microchip's power profiling documentation [198].Then the daily energy consumption of different states for system with ultra-low sleep current was obtained as shown in Figure 3.8 below, and sleepy network can save 97.8% energy than applying ContikiMAC and save 86.5% energy than applying 6TiSCH.

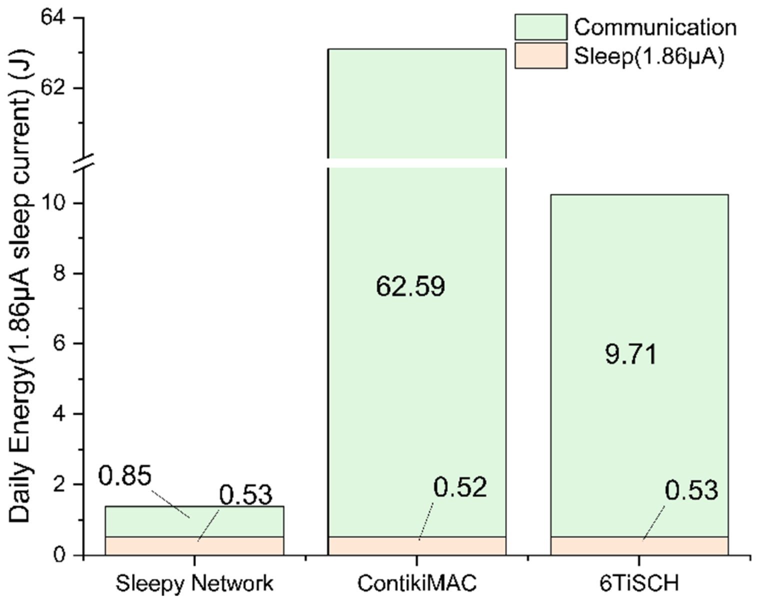


Figure 3.8. Daily energy consumption of different states for system applying sleepy network, ContikiMAC, and 6TiSCH, assuming an ultra-low sleep current of 1.86 μA.

As a synchronous scheme, 6TiSCH has a significant potential for intensive deployment in scenarios that require to avoid interference (e.g. industrial applications) due to its high reliability and good real-time performance [199]. The application of 6TiSCH can potentially benefit users who need real-time access to data, as it can support the node to transmit sensing data shortly in the next allocated transmission slot. But, in resource-constrained nodes, the system will consider minimising energy consumption by keeping the node in a low-power sleep state for a long time. In such conditions, extended periods of sleep cause nodes to re-establish routing and resynchronise 6TiSCH scheduling after waking up, thus introducing additional overhead.

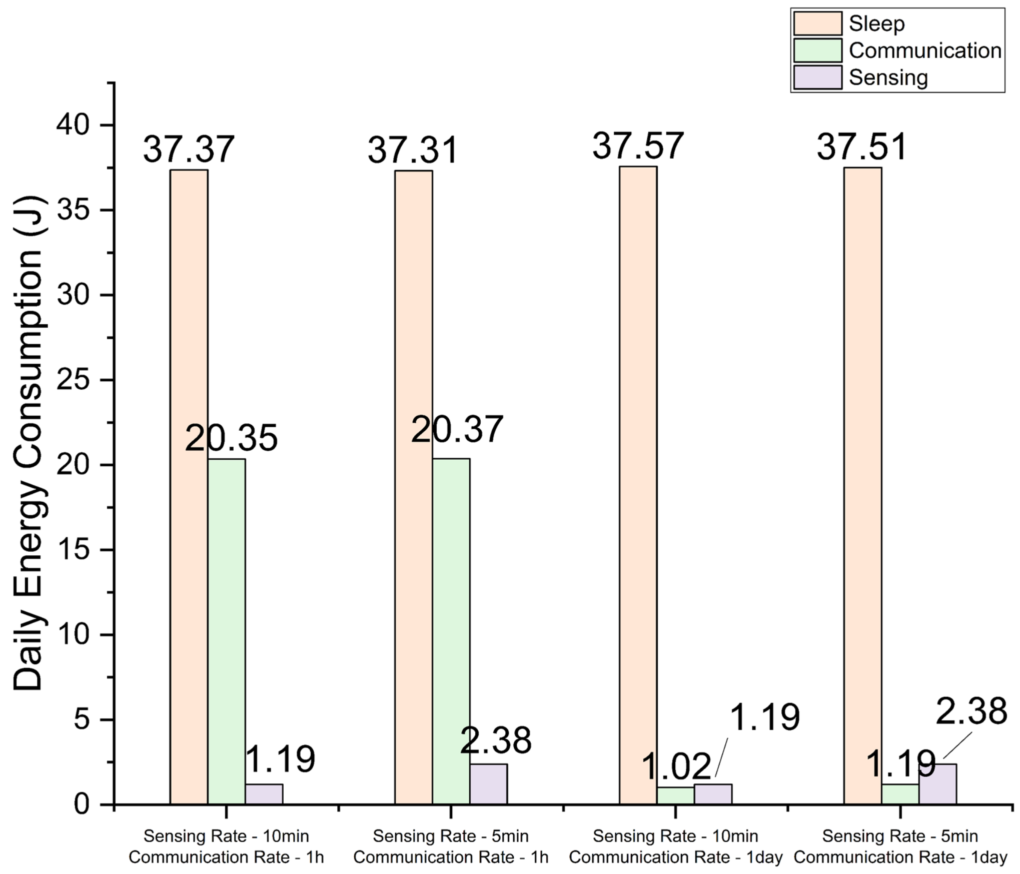
Unlike continuously running Industrial Internet of Things (IIoT) systems, where 6TiSCH can offer significant benefits, the low data transmission frequency in environmental monitoring applications makes it less efficient. For example, ESN systems such as GlacsWeb typically prioritise low power consumption over real-time performance. In these scenarios, the sleepy networking approach offers better energy-saving potential compared to duty cycling schemes. By contrast, 6TiSCH is highly promising for systems that prioritise high reliability and good real-time performance, such as industrial or continuously operating deployments. Besides, as a duty cycling based asynchronous mechanism, ContikiMAC can reduces unnecessary energy consumption and radio congestion [24], and it has good power consumption performance when applying RPL [200]. However, the estimation results above suggest that the sleepy network approach can potentially be more energy-efficient in situations where data is not required to be sent frequently.

3.2.6 Star Topology E-IoT System Energy Consumption Estimation and Evaluation

The time for the BR to periodically maintain the network was observed to be 60s including all the GNRC and RPL activities that cyclically construct the network. To begin with, the communication window defaults to 60s to allow all the network communication setups. According to the measurement, in a star topology, that 60s is longer than the necessary time overhead as the communication window can be shut after all the data is transmitted. Node can turn off the radio after the transmission is completed, and based on the observation, the periodic network maintenance activities, including GNRC and RPL updates, were typically completed within 20 seconds.

In the experimental configuration, each sensing event generated 18 bytes of data. Theoretically, each sensing event will create 18 bytes of data, and each CoAP packet can maximumly contain 97 Bytes of payload [201]. Thus, 5 groups of readings can be transmitted with one CoAP packet. In the subsequent analysis, the communication window was therefore sized by linearly extending the 20s baseline. Therefore, data exceeding the capacity of a single packet is assumed to be transmitted via multiple CoAP packets, with each packet having the same average transmission time and current consumption as the measured CoAP. Based on this approach, a 20s communication window was applied for 10-minute and 5-minute sampling rates with hourly communication, while once-per-day communication used 23.5s and 27.2s for 10-minute and 5-minute sampling rates, respectively.

Figure 3.9 shows the daily energy stack of different schedule examples. This result indicates that daily communication energy consumption has increased as the number of communication windows increases. Significantly, the dominant factor in energy use is the sleep state and communication state so any deployable system should minimise this. As a result, in the case of one communication window per day with a data sampling rate of every 10 minutes, the daily energy consumption is about 39.8 J and the estimated system lifetime is approximately 700 days (estimated using 2600mAh 18650 Lithium-Ion rechargeable battery, and the ratio of working voltage 3.3 V to the nominal voltage 3.7 V is applied).



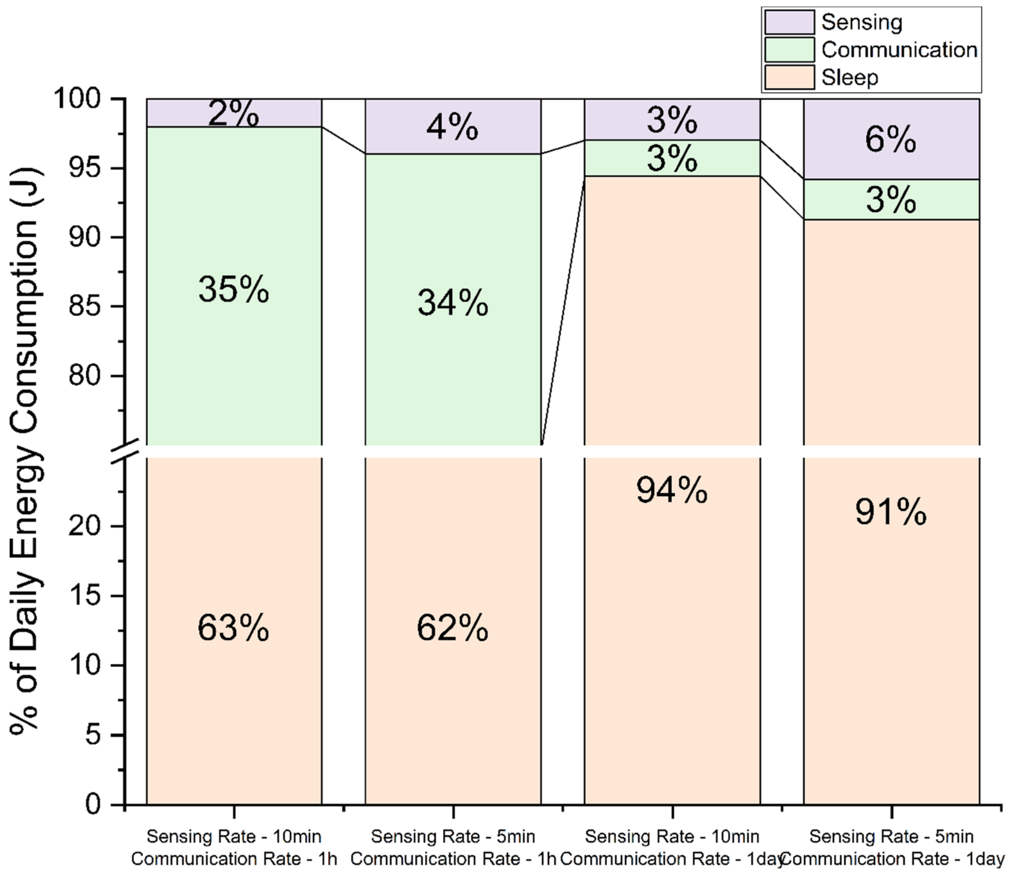


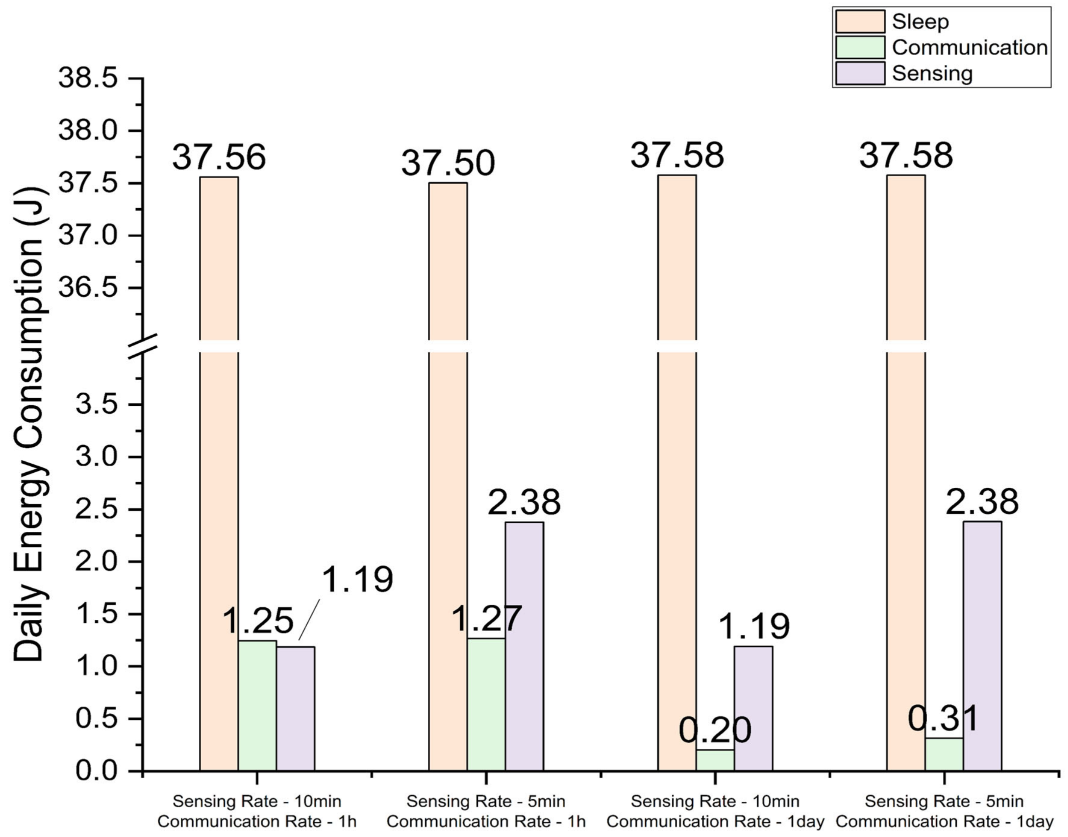
Figure 3.9. Daily energy consumption of different states (top) and energy stack (bottom).

A potential solution to further optimise the system energy consumption is to reduce the unnecessary communication window length. The following assumption is made to focus on evaluating the energy consumption related to data transmission through CoAP. This assumption can significantly reduce the communication window duration, thereby decrease active time of transceiver and enhance energy efficiency. In the initial stage following the establishment of the network and routing, short sleep durations can be possible to ensure successful early data transmission, as routing and neighbour registration information typically remains valid for a period. However, following an extended period of sleep, subsequent communication windows may experience transmission failures due to expired routing or neighbour information. In practical deployments, factors such as interpacket intervals, transmission delays, and channel access latency further impact the applicability of this assumption. However, the purpose of this assumption is only to analyse the energy consumption of nodes operating within strictly limited communication windows. This assumption enables this work to estimate the daily energy consumption of nodes after optimising communication windows and to conduct a preliminary evaluation of the potential energy-saving effects achievable through such optimisation.

Therefore, assuming N sensor readings will be taken before each communication window, the minimum time required for one communication window can be summarised in equation 3.1. T_{IDLE} includes the time cost to wake up the node and turn the radio on. T_{CoAP} is the time response cost for a single CoAP request. $T_{Rdio\_STANDBY}$ is the time cost for turning off radio after completing data transmission. Adding up the minimum communication activity and sensing activity (sensing every 5/10 minutes and communication every hour) implies a 2s communications window could safely be used.

$$T_{communication} = T_{IDLE} + ROUNDUP\left(N * \frac{1}{5}\right) * T_{CoAP} + T_{Radio\_STANDBY} \qquad 3.1$$

Here it can make similar assumptions for different communication rates. If the communication is scheduled once per day, the value of $ROUNDUP(T_{communication} + T_{sensing})$ is equal to 5s (sensing every 10 minutes) and 9s (sensing every 5 minutes). The estimated results illustrated in Figure 3.10 indicated that, applying the optimised prediction of communication window length efficiently reduced the communication energy consumption. Theoretically, optimizing the system energy consumption by minimizing the time overhead of network maintenance such as RPL control message, is potentially valuable. In next chapter, it is worth to reduce the communication time overhead by focusing on minimizing unnecessary RPL control messages while ensuring basic RPL reconstruction requirements.



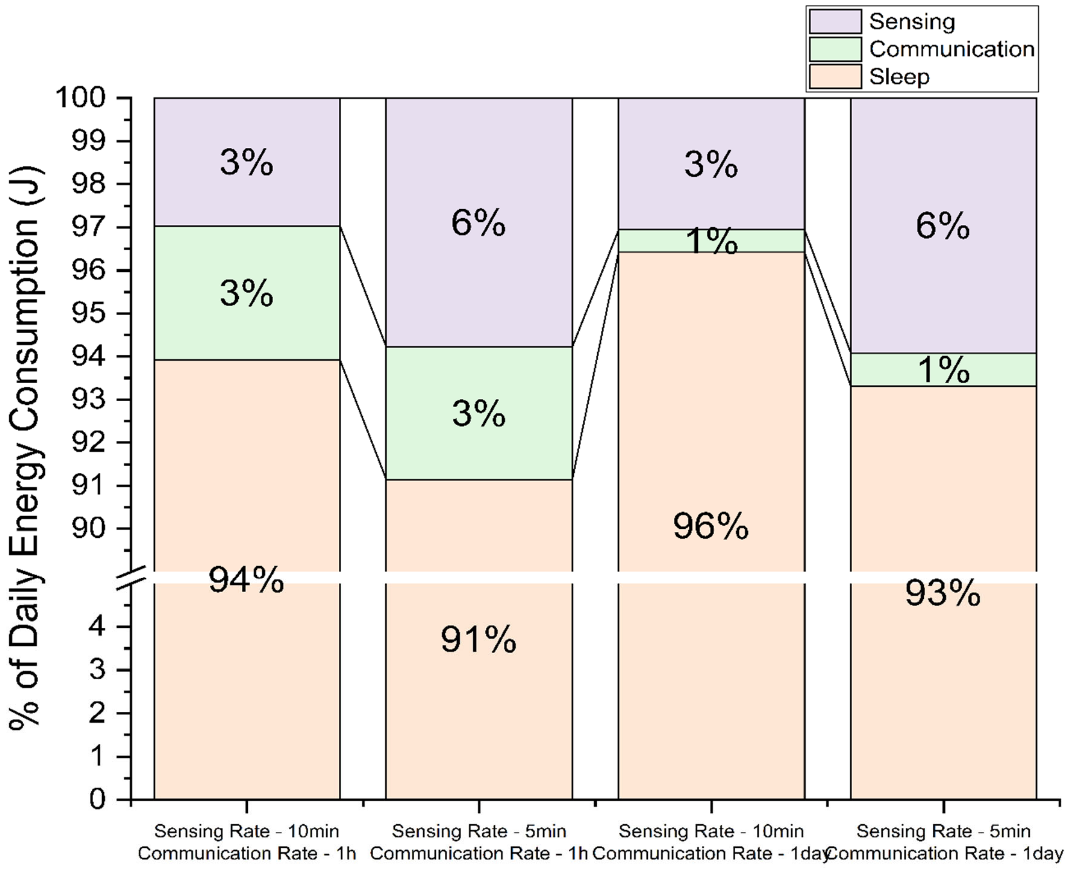


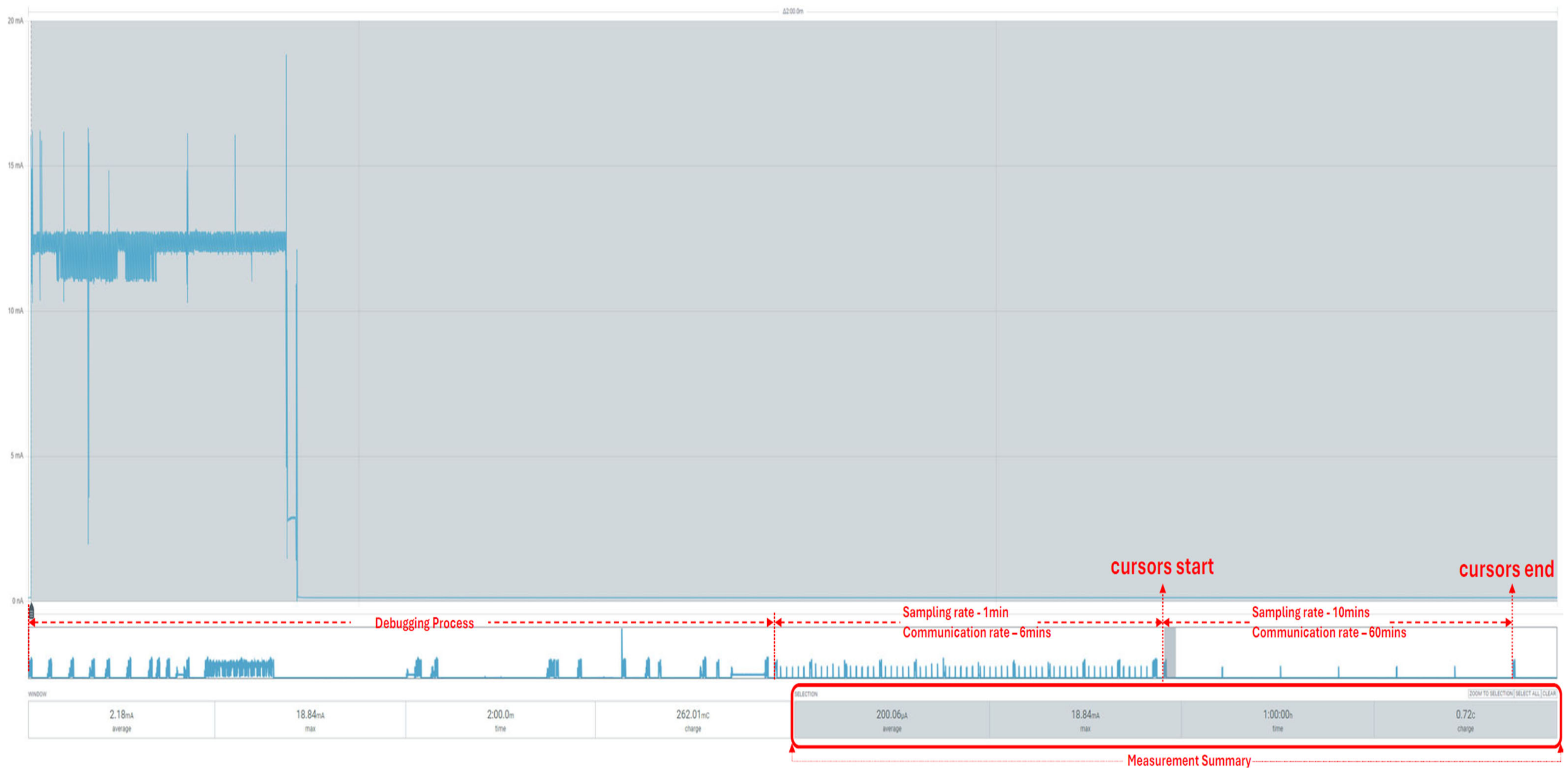
Figure 3.10. Daily energy consumption of different states (top) and energy stack (bottom) with optimised prediction.

The investigation into the star topology sleepy E-IoT system has provided us with an understanding of the relationship between communication frequency, energy consumption, and network maintenance. This sleepy network design for E-IoT system focuses on using CoAP, RPL and 6LoWPAN. Also, the system can enter the sleep state and wake-up state as scheduled to finish the planned tasks with reliable communication features. Furthermore, CoAP is applied to ensure the ability of the user to update and reconfigure the system presets. To comprehensively under-stand the impact of energy consumption with an increased number of communication windows, this work has measured the current and time duration of system activities such as read sensor, store data, and radio activities controlled by the standard IoT protocols. The observed increases in daily communication energy consumption with additional windows emphasise the importance of a balanced approach that considers network reliability and communication time. The estimation results show that optimizing the communication time overhead can reduce energy consumption further. According to Figures 3.9 and 3.10, when the system's sensing frequency remains the same, optimizing the communication window results in a reduction of system energy consumption by nearly 2% (communication once a day) and 32.5% (communication every hour). This work suggests optimizing communication windows and network activities to control the communication energy consumption, and this can further guide similar systems to take a balance between network reliability and energy consumption. In the upcoming stages, this work will be directed towards optimizing energy consumption. This will be achieved through optimizing the system in routing configuration and communication patterns, aiming to further reduce overall system power requirements.

3.3 E-IoT System Validation: Star Topology Sleepy Networks

3.3.1 System Energy Consumption

This section utilises the same experimental configuration as described in subsection 3.2.2 to validate the overall energy consumption of nodes in this sleepy network design. Current consumption at the node side was measured using the Nordic Semiconductor Power Profiler Kit II. During the 1-hour measurement, the node operated with a 10-minute sampling rate and a 60-minute communication rate, following the same scheduling configuration outlined in subsection 3.2.3. Within the one-hour test duration, the node conducted one 20-second communication window and six sensing windows. Based on the measurements, the average current consumption over the one-hour test duration was 200.06 µA (illustrated in Figure 3.11). According to the schedule design, this one-hour slot would repeat 24 times daily, the corresponding daily energy consumption is calculated as 57.04 J.



WINDOW

2.18mA	18.84mA	2:00.0m	262.01mC
average	max	time	charge

SELECTION

200.06µA	18.84mA	1:00:00h	0.72C
average	max	time	charge

Figure 3.11. 1-hour average current consumption measurement of node, with sampling rate of every 10 minutes and communication rate of every 60 minutes.

This measured value closely matched the estimated daily energy consumption of 58.9 J derived in subsection 3.2.6, with a deviation of 3.2%. Given that the daily energy consumption estimated in subsection 3.2.6 and the model applied certain simplifications based on radio activity observations, deviations between actual measurements and estimated values are inevitable. Meanwhile, the actual measurement results may also suffer from a certain degree of inaccuracy due to measurement noise and timing jitter. Overall, the validation results indicated that the activity-based energy consumption model was able to evaluate the system's energy consumption with reasonable accuracy under different data sampling and communication rate settings.

3.3.2 Latency

To evaluate the system's end-to-end latency, this work used the ping command on a Linux PC to send 20 successive requests to the node via BR. Each test run utilised identical node configuration as introduced in Section 3.2.2, with the average, minimum, and standard deviation of the round-trip delay time recorded. In the latency test, this work observed that when nodes sent network maintenance messages at the same time, ping requests could fail to deliver or be lost. Consequently, the latency test only recorded statistics with a 100% delivery rate. The delivery rate here is defined as the proportion of successfully delivered requests to the total number of requests initiated. Finally, the data from three sets of 20 ping requests were recorded and summarised in Table 3.6.

Table 3.6. Latency test statistics summary.

Ping Test No.	Average (ms)	Minimum (ms)	Maximum (ms)	Standard Deviation (ms)
1	143.06	130.505	249.47	27.163
2	150.083	128.544	308.38	46.27
3	155.286	130.394	299.372	50.015

According to Table 3.6, this work found that the average latency in the deployed system was approximately 149.5ms, with the mean minimum and maximum values were 129.8ms and 285.7ms respectively. The maximum recorded standard deviation reached 50 milliseconds, reflecting potential time overhead including data processing/encapsulation, buffer reading, and channel access delay due to CSMA/CA check. The minimum average round-trip delay observed across multiple tests approached 130ms, indicating the lower bound of end-to-end communication latency within this system. The lower bound and average of the round-trip delay would provide essential reference for further reducing the communication window and setting the packet retransmission interval in the next chapter. On the other hand, compared to the overall average latency of 149.5ms, the average latency values recorded in Table 3.6 demonstrated a maximum deviation of 4.3% and a minimum deviation of 0.4%. Consequently, the latency recorded during repeated tests demonstrated relative consistent.

In a sleepy network, data transmission naturally involves delays, as sensing data is only transmitted when the system enters a predetermined communication window. Therefore, end-to-end latency depends not only on network transmission time but is also influenced by the degree of delayed data transmission defined by the user. In practical deployments, multi-hop network topologies are typically required to cover broad or complex monitoring areas. In such scenarios, the duration of the communication window for nodes must be extended to accommodate packet forwarding across multiple hops, particularly at relay nodes. Additionally, an appropriate inter-packet interval should be configured to allow sufficient time for data reception acknowledgements and retransmissions. These factors should be considered in practical deployment to ensure the reliability of the deployed system.

3.3.3 Throughput

For evaluating the throughput of the sleepy network design, this work conducted a throughput test which the node transmitted non-confirmable CoAP PUT request continuously. It was observed that the maximum allowed CoAP payload was 115 bytes which also triggered packet fragmentation in RIOT-OS. Fragmentation introduced several challenges, such as increased protocol overhead and longer radio activity durations. These factors make the communication window optimisation more challenging. Consequently, fragmentation effects were excluded from this section and would be investigated further in the next chapter. To test throughput, a fixed inter-packet interval of 2s was applied between each CoAP request to avoid conflicts and congestions. In this experiment, each CoAP PUT message carried a 68 bytes payload (PHY Service Data Unit (PSDU) was 127 bytes).

Additionally, it was observed that packet retransmissions or delivery failures could occur when CoAP transmissions conflicted with network maintenance activities, such as RPL activities. As throughput test aims to discuss the achievable throughput during the period when the channel is available, only test results where CoAP requests were successfully delivered were used for throughput analysis. Test results affected by network maintenance activities were excluded from throughput analysis. This test was also repeated multiple times. The recorded test results are listed in Table 3.7, and the listed value represent the statistical results of six repeated tests, each comprising ten continuous CoAP transmissions.

The overall average duration for CoAP transmissions was 155.73ms with standard deviation of 7.02ms, and the resulting throughput was about 439 Bytes/second (about 3.5 kbps). As the experiment did not trigger the fragmentation mechanism, the measured throughput does not reflect additional overhead associated with increased payload sizes, fragmentation, or reassembly. Furthermore, due to the inter-packet interval which was set to 2s, it may not reflect the maximum throughput achievable in all practical deployments. The experimental results may provide a reference for single-hop throughput performance, but actual deployment scenarios may experience reduced throughput due to interference and multi-hop forwarding overhead/delays. Throughput analysis indicates that the system effectively supports the relatively low-volume and low-frequency data transmission patterns commonly required in periodic environmental monitoring applications. However, the results also demonstrate that the system is unsuitable for sustained high-throughput data transmission or the transfer of large payloads such as image or video streams.

Table 3.7. Throughput test statistics summary.

CoAP Packet No.	**Duration (ms)**		
	Average	**Overall Average**	**Standard Deviation**
1	154.39	155.73	7.02
2	150.17		
3	159.57		
4	147.08		
5	157.17		
6	160.16		
7	170.41		
8	146.96		
9	153.78		
10	157.62		

3.4 Network Exploration: From Star to Tree Topology

To improve the scalability, it is essential to develop an energy-efficient E-IoT system capable of operating in tree topology and, based on this, to further investigate how sleepy the sleepy network can become. Building upon the star topology design, it is possible to extend the scheduling arrangement to a tree topology while maintaining it with the sleepy network mechanism. Like the established schedule design for system in star topology, in this setup, multiple nodes form a hierarchical multi-hop network with an intermediate node forwarding packets from the leaf nodes to the BR. Meanwhile, as shown in the Figure 3.12, node A, B, and C will all perform sensing and communication tasks. In RIOT-OS implementations, the DODAG root of a RPL network (the border router in this case) typically initializes with a default Rank value of 256. As looking downwards in the topology, each subsequent level of nodes increases this Rank by a default increment of 256. Thus, the nodes directly below the border router (node A and node AA) have a rank of 512. Following this logic, the Rank value of the next level nodes (e.g., node B and node C) will be increased to 768. In general, nodes occupying the position at the N-th level of the tree topology are assigned a Rank value calculated as $256 + N * 256$. All nodes in the tree topology follow a synchronised sleep schedule, where the communication window occurs at the beginning of each cycle, followed by a sensing window, and then the sleep state. Like the star topology schedule, the communication interval can be configured based on the application's needs.

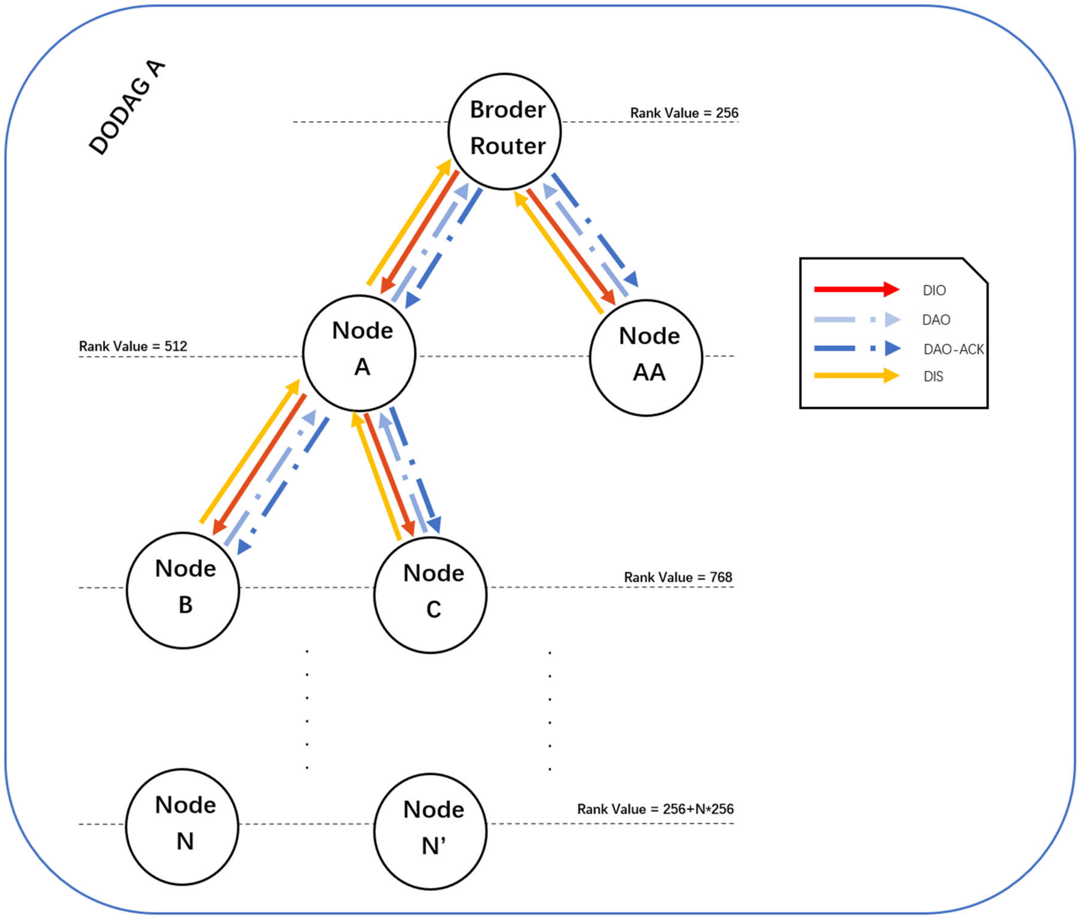


Figure 3.12. RPL node star topology-DIO & DAO exchange and DODAG.

Since the intermediate node (Node A) needs to forward packets from the child nodes, it must stay awake slightly longer than the leaf nodes (node B and C) to complete the forwarding before transitioning to the sleep state. This creates a hierarchical wake-up pattern, where nodes at higher levels of the tree (small hop count) stay awake slightly longer than nodes at lower levels (as mentioned above, nodes turn off their radios after completing their communication to save energy). At current stage, it is unclear that how long the time window intermediate node should reserve for the leaf node to send the data safely. In a short conclusion, depend on the order that leaf nodes wake-up and transfer data (simultaneously or sequentially), the communication overhead caused by the activity of maintaining associated networking and routing are various and should be further investigated. Firstly, to give a simple example, if all nodes as shown in the Figure 3.12 turn on the radio to transmit data at the same scheduled time, system needs to reserve well enough time to complete the maintenance of the network and routing before sending sensing data packet. Secondly, the system also needs algorithms to control the sequence of data transmission for each node to better avoid conflicts.

Currently, in the star topology, this work has measured the energy consumption of different network activities based on experiments, and successfully estimated the energy consumption under different communication window configurations. However, for the tree topology, due to the complexity of its structure and network maintenance mechanism, there are still several key factors (such as data forwarding, network maintenance, routing management, etc.) that cannot be clarified at this stage, which makes it difficult to directly use the above-mentioned data to estimate the energy consumption. When considering to improve the energy efficiency in multi-hop networks it is important to note that although multi-hop networks solve the problem of long distance transmission and barriers, the relay nodes in the multi-hop network must receive and forward the packets correctly [135].

Firstly, the data forwarding mechanism of tree topology brings additional energy consumption. In the star topology, all nodes send data within a scheduled communication window, while in the Tree topology, the data from the child nodes need to go through multiple hops to reach the BR. the intermediate parent nodes require additional radio activities to receive and forward the data, so they must be activated for a longer duration, and their energy consumption is also higher. The number of children for different parent nodes will result in various forwarding loads, and the current experiments have not measured the specific impact of these factors on the radio overhead. Secondly, the network maintenance mechanism of tree topology has not been completely identified. In star topology, the control information maintained by RPL is centrally managed by BR, which contains regular DIO and DAO messages. However, in tree topology, RPL maintenance involves the collaboration of parent and child nodes in different layer of the tree, which may involve more DIO and DAO exchange. In particular, when a parent node receives a DAO from a child node, whether it needs to respond further to the parent immediately, how to respond, and whether it triggers additional radio activities have not been verified through experiment.

In addition, scheduling in tree topology is more complicated, especially when determining the duration of the communication window of the intermediate parent node. In Star topology, all nodes wake up and complete data transmission within the same communication window, and the length of the communication window is easier to calculate. However, in tree topology, leaf nodes can send data and go to sleep in a shorter period of time, while parent nodes need to wait for all child nodes to finish data transmission before forwarding, which results in a longer radio on time for them. Currently, in the tree topology, so far this work has not measured the energy consumption of different levels of parent nodes, and has not clarified the additional power consumption for different numbers of child nodes. In a short conclusion, due to the complexity of data forwarding, network maintenance, and synchronous scheduling in tree topology, the next chapter will further investigate the above factors and evaluate the E-IoT System performance in a tree topology with different wake-up strategies (simultaneously and sequentially).

3.5 Summary

This chapter investigated the design and implementation of the sleepy network for E-IoT systems and related work on improving energy efficiency in resource-constrained IoT deployments through scheduled sleep mechanisms and low-power communication protocols. The sleep/wake scheduling strategy is a key component in reducing energy consumption. In this system design, nodes need to transit between sensing, communication, and sleep states to ensure that radio activity occurs during scheduled communication windows. By utilizing RTC-based wake-up scheduling, the system can efficiently synchronise node operations while reducing overheads.

To achieve efficient and expandable communication, the system employs 6LoWPAN, RPL, and CoAP as the network protocols. CoAP helps the system achieve lightweight data exchange and allows the system user to dynamically reconfigure the sensing and communication parameters through CoAP requests. RPL plays an important role in structuring the network topology. RPL control messages (e.g., DIO, DAO, and DIS) manage the formation of DODAGs to complete and maintain the routing. In the initial implementation of the sleepy network, this work has measured the average current consumption and activity duration for different radio activities. And these measurements are applied to estimate the daily energy consumption if applying sleepy network, ContikiMAC or 6TiSCH. Compared with applying ContikiMAC and 6TiSCH, it was estimated that applying sleepy network can save 61.4% and 18.6% energy, respectively.

This work first introduced a star topology as a simplified scenario for the tests. In the star topology, all nodes communicate directly with the BR during a scheduled communication window. This start network design simplifies network maintenance but has limited scalability. In 3.2, this study also measured current consumption and time duration for various network activities in star topology, including sensor data acquisition, packet transmission, and network maintenance (ICMPv6, RPL, 6LoWPAN control messages). And the experimental results confirmed that: increasing communication frequency will bring extra energy consumption, and optimizing communication window duration can significantly reduce energy overhead. However, in a tree topology, intermediate nodes need to ensure that data from leaf nodes can be forwarded correctly and maintain the associated networks and routes, all of which lead to higher energy consumption and additional communication overhead. Specifically, leaf nodes can quickly return to sleep after sending data, but the parent node must remain active until it finishes forwarding all received packets. The optimal wake-up durations for different levels of nodes are still unknown, making it difficult to predict the total energy consumption of a tree topology network.

Finally, experimental validation and evaluation in Section 3.3 demonstrated that the actual deployment energy consumption of the sleepy network design differed by only 3.2% from the computational values in Section 3.2. This indicated that the activity-based energy consumption model was able to evaluate the system’s energy consumption with reasonable accuracy under different data sampling and communication rate settings. Furthermore, in Section 3.3, this work also evaluated the system's latency and throughput, and provides reference and guidance for the communication window optimisation design in the next chapter.

To extend the coverage of the system, this work introduced the tree topology in 3.4 and will further investigate system design details in tree topology in *Chapter 4*. In a tree topology, packets will be forwarded by intermediate nodes in this kind of multi-hop network. Also, section 3.4 discussed the essential characteristics of the intermediate nodes in tree topology, such as ensuring that the intermediate nodes remain active long enough to complete data forwarding from all leaf nodes. The experiments in Section 3.2 confirmed the ability of the nodes in the system to perform the scheduled activities, such as reading sensors and storing data, turning on the radio and transmitting data on a predetermined schedule in a predetermined time slot.

The next chapter will further investigate the application of simultaneous and sequential wake-up strategies to optimize sleep scheduling and system energy consumption in tree topologies. Additional experiments will also be performed afterwards to identify the influence of changing sleep duration of nodes at different layers of the tree topology, focusing on the patterns of packet forwarding and network maintenance activities, which is aimed to improve the wake-up scheduling mechanism and optimize the communication time. Moreover, instead of maintaining a fixed communication window for all nodes, the system schedule should dynamically adjust the wake-up time of the parent node according to the number of managed child nodes. Then, the optimization of communication windows in different topologies will be addressed in the next chapter.

Chapter 4 Optimization and Energy Profiling of Low Power Sleepy Network Design for Environmental IoT System

This chapter will investigate the radio activity patterns and system schedule arrangements in a tree topology based on the experiments and measurements in Chapter 3 and finally evaluate and optimize the performance of sleepy E-IoT systems deployed in a tree topology. Specifically, this chapter will summarise the influence of different sleep durations/wake-up frequencies on the expected system radio activities, and it will also analyse the impact of tree topology of different complexity on the system.

The star topology is simple to construct and allows all nodes to communicate directly with the BR during a scheduled communication window, but this topology structure has poor scalability compares to a tree topology. To extend the scalability, the sleepy network design should incorporate a tree topology, which enables data from child nodes to be forwarded through intermediate nodes. Optimising the communication window in a sleepy network is like arranging efficient collaborating processes. An optimised communications schedule ensures that these activities are executed efficiently and in an organised manner. This section aims to organise the radio activities during the communication window and minimize the window size, while the environmental data can be delivered efficiently in a well-organized manner.

Chapter 3 indicated that optimizing the duration of the communication window can significantly reduce the energy consumption overheads. Based on these findings, this chapter focuses on the strategy to further optimize the energy consumption of the system under tree topology. In this work, some realistic and simple assumptions are made as follows. For data sampling rates, in order to maximise the impact of factors such as communication frequency on the nodes and to set reasonable simulation assumptions, the modelling and calculations covered in Section 4.6 and 4.7 will use the fastest data sampling rates (every 10 minutes) as analysed in the battery-powered long-term environmental monitoring projects in Section 2.5. In addition, this chapter investigates the influences of node wake-up sequence (simultaneously and sequentially) on the scheduling arrangement and system energy consumption, then suggests the practical optimised schedule for low-power sleepy network design. Finally, this work will investigate the influence of some key factors change in the network, such as communication rate and complexity of tree topology, on the system energy consumption with optimised schedule.

In an IPv6-based E-IoT systems using 6LoWPAN and RPL, the network is constructed and maintained through the exchanging of control messages, and these messages were introduced and illustrated in Table 2.1 and Figure 2.7. These radio activities are essential to maintain accurate neighbouring and routing table. Once all the nodes start to deploy, the BR and the individual nodes will complete its neighbour registration and routing information updates as introduced in subsection 2.3.5.2. However, when a node is in a sleep state for an extended period which exceeds the network routing information validity lifetime (defined as “RPL Lifetime”), the network topology information may become outdated or invalid when the node wakes up. When nodes wake up from extended long sleep state, the radio activity regarding to the routing information updates is different compared to the initial registration period.

4.1 Problem Definition

To begin with, this chapter investigated how nodes within a tree topology can efficiently schedule their communication while avoiding unnecessary conflicts after waking up from a continuous sleep state that exceed the default RPL lifetime (3 minutes in RIOT-OS). In this section, the experimental setup did not change the default RPL lifetime parameters, which ensured that the behaviour of the system is generic compared to the general configuration, thus keeping the generality and comparability of the results to future studies.

Therefore, in the next section, experiments were designed to investigate how nodes with different RPL rank values in a tree topology rejoin the network and complete data transmission predetermined times (longer than default RPL lifetime), either simultaneously or sequentially. Experiments focused on the communication overhead associated with the node routing information update process and the time overheads caused by the different transceiver turn-on timing. To explain efficient communication further, the idea was only using limited and constrained communication windows slot to complete essential network maintenance and data transmission.

It was noted that the assumption made in section 3.2.6 was that 2 seconds communication window can be considered. However, as previously discussed, in a tree multi-hop network, nodes waking up from long sleep must first ensure that network registrations and routing information is correctly updated before data can be delivered successfully. While optimizing the length of the communication window can reduce system energy consumption, the time required to complete network maintenance in practical deployments can be significantly longer than this initial assumption. Therefore, the previous assumption, 2 seconds communication window is evaluated and extended in this chapter to accommodate the actual network maintenance requirements.

In summary, this chapter firstly investigated the following questions:

- What network radio activities are required when the node sleep duration exceeds the default RPL lifetime?
- How to optimize communication scheduling of nodes in tree topology networks under simultaneous and sequential wake-up schemes?
- Based on the proposed optimised communication window, how can the system energy consumption in a tree topology be modelled and analysed?

The following Sections (4.2 and 4.3) introduced the experiment design and results obtained from the radio activity observations. Furthermore, the implications of these findings on the design of optimised communication schedule were also be discussed. Different from the system design introduced in Section 3.2, in the following experimental configurations, each node will actively send environmental data to the BR (DODAG root) over a multi-hop network using the CoAP PUT method. As a CoAP server, node will send "Confirmable" CoAP (CON-CoAP) PUT request through routing table to the DODAG root and wait for CoAP ACK response, which provides an acknowledgment mechanism to ensure that the data transfer is complete. The test code used for tree topology sleepy network testing has been uploaded to GitHub for public viewing, and the link was provided in Appendix B.2.

4.2 Experiment Setup

The purpose of this section is to investigate the specific network radio activity and time required for nodes in a tree topology RPL network to re-establish connectivity after sleeping for longer than the default RPL lifetime. When nodes wake up after such extended sleep, neighbour and routing information may no longer be valid, and network maintenance is required before successful data transmission. Firstly, it is necessary to expand the star topology configuration in Section 3.2 to tree topology. To effectively constrain the length of communication windows, the star topology configuration in Section 3.2 was expanded to tree topology to observe the neighbour and routing registration update mechanisms (NA, NS, RA, RS, DIS, DIO, DAO, DAO-ACK), and data forwarding (CoAP PUT and CoAP-ACK). And different from the experiment in Section 3.2, this experiment was not required to measure the current and time duration of each activity. This experiment focused on using the RIOT-OS sniffer to observe the network activities of each node in a tree topology when it wakes up. In this experiment, nodes were wake-up according to a predetermined schedule in two approaches: synchronous wake-up and sequential wake-up.

The nodes used in this experiment were consistent with the configuration outlined in Section 3.2 (previously detailed in Figure 3.3) and the transceiver was configured with channel number = 0 and channel page = 0 to obtain higher receiver sensitivity (BPSK-20kbps with -110 dBm) [188]. The primary task is to observe the fundamental radio activity required for a node to successfully rejoin the RPL network after a long period of inactivity. Therefore, nodes will be organised into a hierarchical RPL tree topology. For consistency, each node is initially programmed and enters a networking stage (duration can be customised) in which it undertakes the following tasks: 1) neighbour registration, 2) initial RTC, 3) initial RPL, 4) synchronise RTC by sending CoAP GET request, 5) set RTC alarm, 6) wait and enter the scheduled event loop synchronously. Figure 4.1 illustrates the tree topology network layout constructed for this experiment.

During the event loop, denoted as sleepy test afterwards, nodes were designed to repeatedly wake up to send a CoAP PUT request containing assumed temperature data and timestamps, and then returned to sleep after receiving the corresponding CoAP ACK. The sleep duration was initially set to 300 s and is gradually increased over subsequent test iterations. Nodes were programmed to wake up simultaneously or sequentially. When a sequential wake-up strategy was applied, the interval between the communication slots of two nodes was intentionally set to a sufficiently large value (e.g., 5 minutes), as the minimum communication window length was not determined at the initial stage of the experiment.

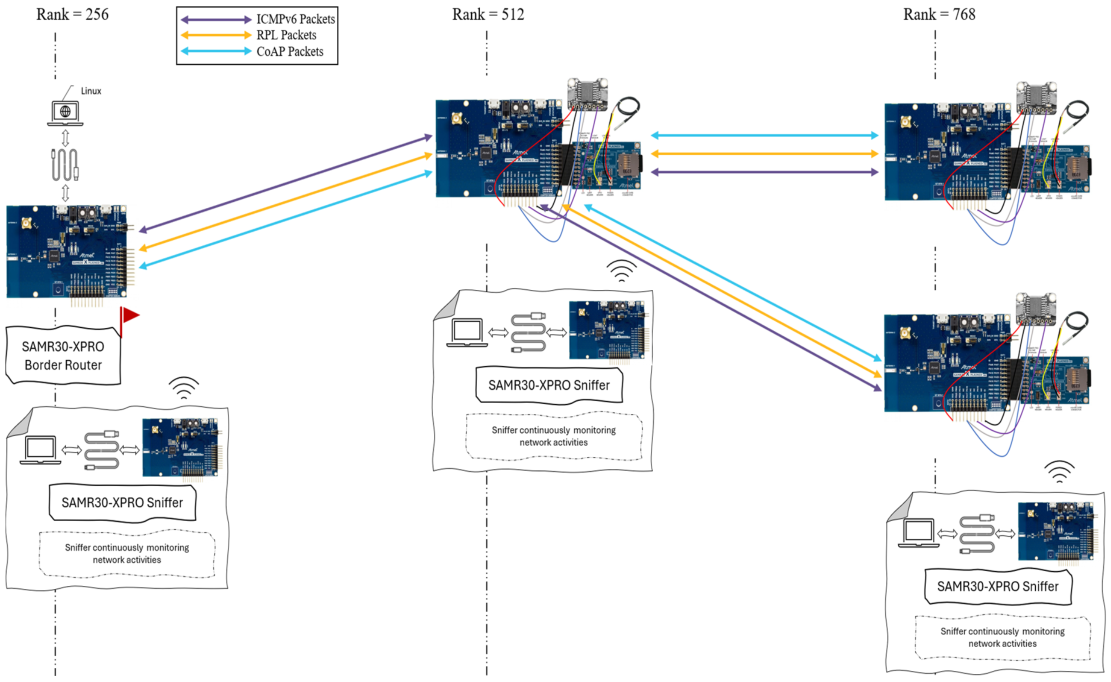

Rank = 256
ICMPv6 Packets
RPL Packets
CoAP Packets
Rank = 512
Rank = 768
Linux
SAMR30-XPRO Border Router
SAMR30-XPRO Sniffer
Sniffer continuously monitoring network activities
SAMR30-XPRO Sniffer
Sniffer continuously monitoring network activities
SAMR30-XPRO Sniffer
Sniffer continuously monitoring network activities

Figure 4.1. Tree topology network layout.

For extending the test scenarios, this sleepy test gradually introduced nodes with the same rank value. Initially, the sleepy test focused on the nodes with rank value 512, which are adjacent to the border router (rank value = 256). To begin with, this experiment deployed two neighbouring nodes with rank value = 512 and gradually increased the sleep duration of the nodes. Next, to increase the complexity of the tree topology, two neighbouring nodes with rank value = 768 and one intermediate node with rank value = 512 were deployed. Sequential wake-up approach can minimise radio collisions, while simultaneously wake-up may require longer time durations to complete the corresponding network activities under more congested traffic conditions. To avoid the unnecessary radio conflict at the beginning, each experiment applied a sequential wake-up strategy first then modified to simultaneous wake-up based on the findings form the sequential wake-up test. To clearly illustrate the experimental procedures and the incremental approach adopted in this test, Figure 4.2 provides a simplified flowchart of the test.

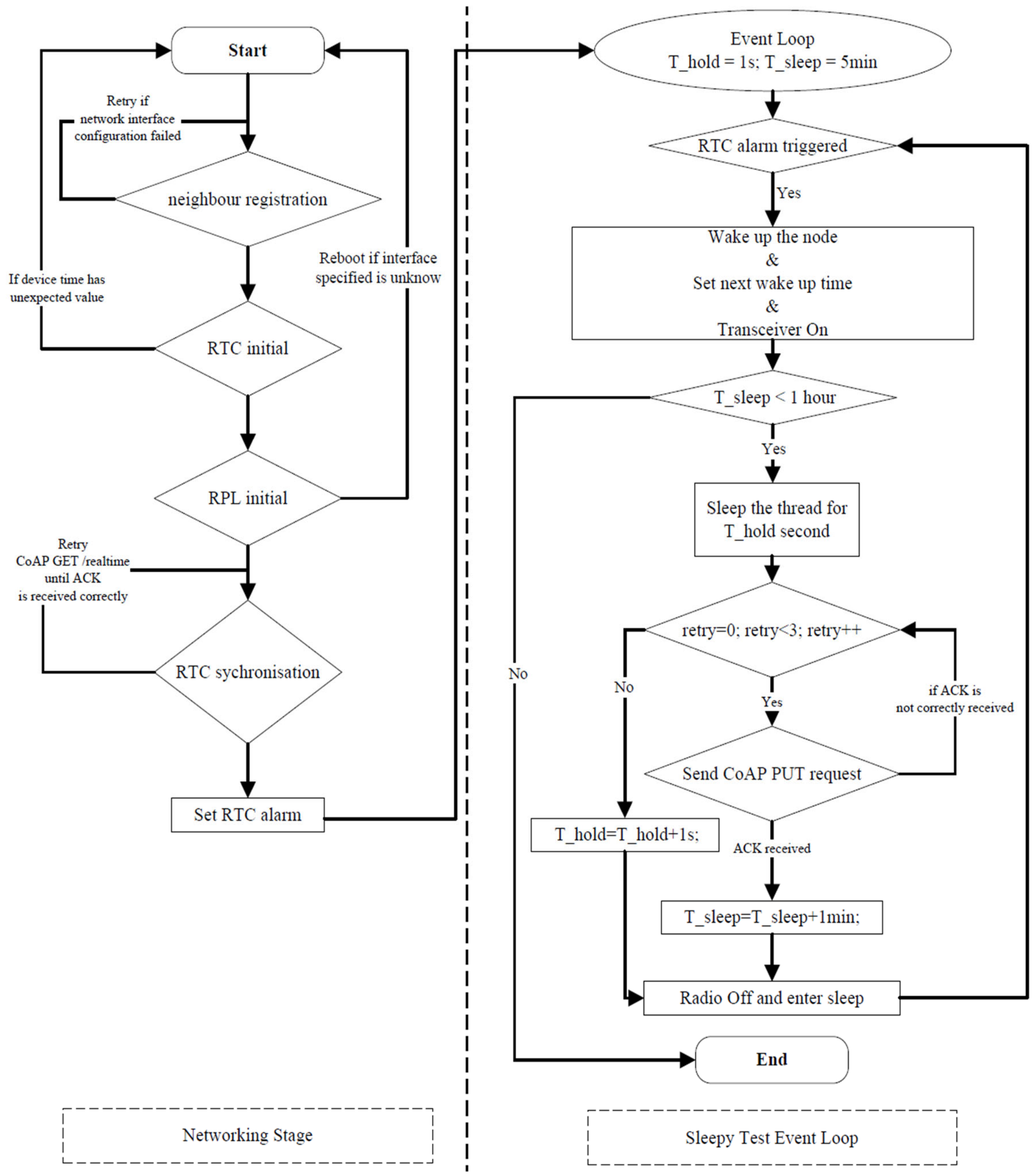

Start
Retry if network interface configuration failed
neighbour registration
Reboot if interface specified is unknow
If device time has unexpected value
RTC initial
RPL initial
Retry CoAP GET /realtime until ACK is received correctly
RTC sychronisation
Set RTC alarm
Networking Stage
Event Loop T_hold = 1s; T_sleep = 5min
RTC alarm triggered
Yes
Wake up the node & Set next wake up time & Transceiver On
T_sleep < 1 hour
Yes
No
Sleep the thread for T_hold second
retry=0; retry<3; retry++
No
Yes
if ACK is not correctly received
Send CoAP PUT request
T_hold=T_hold+1s;
ACK received
T_sleep=T_sleep+1min;
Radio Off and enter sleep
End
Sleepy Test Event Loop

Figure 4.2. Sleepy test flow chart.

During the sleepy test event loop, with an unchanging sleep duration (T_sleep), if the CoAP PUT failed to be delivered after three attempts (i.e., the ACK cannot be received correctly), the node should continuously increase the amount of waiting time before executing the CoAP PUT (T_hold). The sleep duration will be increased unless the T_hold is enough longer to ensure that CoAP PUT request can be delivered to the DODAG root. The results of the RIOT-OS sniffer were used to characterise the type, frequency and sequence of critical radio activities during network maintenance/updates. These observations, particularly the minimum duration required for neighbour registration and routing information updates, were also used to optimise communication window length.

4.3 Network Maintenance after Sleep Exceeding the Default RPL Lifetime

4.3.1 Neighbour Registration and Routing Information Updates

After updating to the RIOT-OS 2024.07 branch, the radio activities observed during network maintenance periods after waking up presented consistent patterns driven by RPL and IPv6 neighbour discovery protocols. In particular, it was observed that BR periodically multicast a single DIS message at a fixed interval of 60 seconds. And each DIS transmission triggered a burst of DIO messages as observed in the sniffer, and up to 22 DIO messages in a minute were recorded exchanging between node and BR.

During the scheduled communication window, there were no observation of RS and RA messages. An analysis of the prefix information option contained in RA messages captured during the initial network maintenance stage indicates that the Valid Lifetime field is set to values exceeding 65,400 seconds. According to the IPv6 Neighbour Discovery Protocol (NDP) defined in section 4.6.2 in RFC 4861 [100] and RIOT-OS IPv6 Neighbour Discovery documentation [202], it was identified that the "Valid Lifetime" refers to the duration for which the advertised prefix remains valid for on-link determination and address autoconfiguration. In RIOT-OS, the prefix information has been defined with infinite valid lifetime and preferred lifetime. As a result, nodes waking up from extended sleep do not need to solicit prefix information from the BR, thereby significantly reduces the number of radio activities required during neighbour registration process.

Instead, in the network, the rejoining process relied heavily on NS and NA messages to verify the availability of neighbours, and RPL control messages (e.g., DIS, DIO, DAO) to update the routing information. Figure 4.3 summarises the node behaviour in response to received RPL control messages in RIOT-OS, based on the implementation described in the RIOT-OS repository [203].

Consistent with the Trickle timer mechanism defined for RPL and discussed by Medjek et al. [101], receiving a DIS message resets the Trickle timer and triggers the transmission of DIO messages. Furthermore, sniffer observations confirmed that the BR continues to transmit periodic DIS messages even while nodes remain in a sleep state, ensuring that node can be able to update routing information once waking up. It was also observed that RIOT-OS schedules and waits for the completion of key RPL control message exchanges such as DIS and DAO.

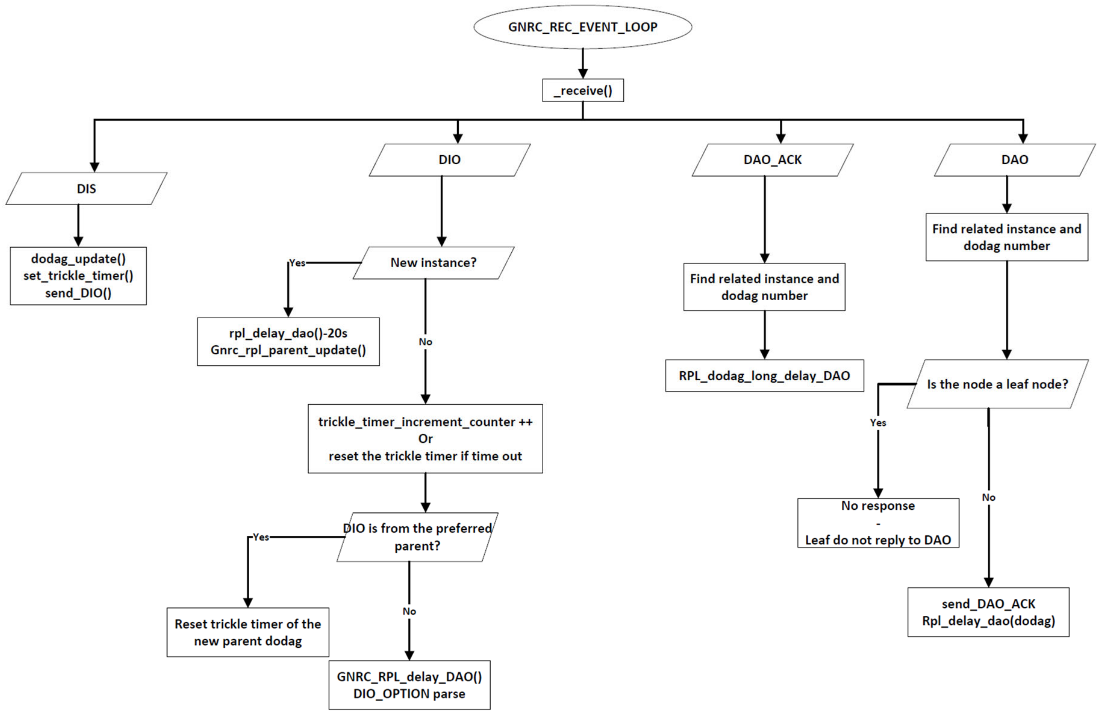


Figure 4.3. Node's behaviour in response to receiving RPL control messages in RIOT-OS.

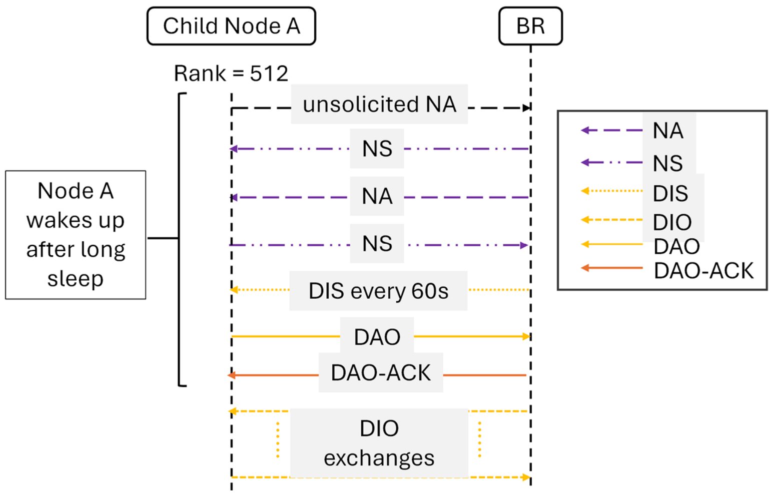


Figure 4.4. IPv6 neighbour discovery and RPL traffic after child node wake-up. Node A wakes up after long sleep while BR is always active.

In addition to RPL control traffic, IPv6 ND activities was also observed after the routing update was completed. Specifically, for nodes with rank 512 that rejoin the network after a long period of sleep, a series of NA and NS messages are exchanged between the nodes and the BR. The observed pattern (demonstrated in Figure 4.4) usually started with the node sending unsolicited NA messages to the BR, which then responded with NS messages. The BR subsequently sent NA message to the node, and finally, the node replied with another NS message. This type of bidirectional switching is used to update the neighbour registration and confirm reachability at the link layer. When two nodes with rank 512 woke up simultaneously, the number of DIO message exchanges triggered by the DIS from the BR will increase significantly. In addition, frequent NA, NS message exchanges between each node and BR were identified.

To investigate and observer the network behaviour during communication window, as per Figure 4.1 , two child nodes with a rank value of 768 (hereafter referred to as N1 and N2) and an intermediate parent node with a rank value of 512 (referred to as M1) was deployed. The simplified nodes deployment layout is shown in Figure 4.5 below. It was observed that a pattern of NA and NS message transmission existed between the remote child nodes (N1 and N2) and their parent node (M1), similar to that previously observed between nodes with a rank value of 512 and the BR.

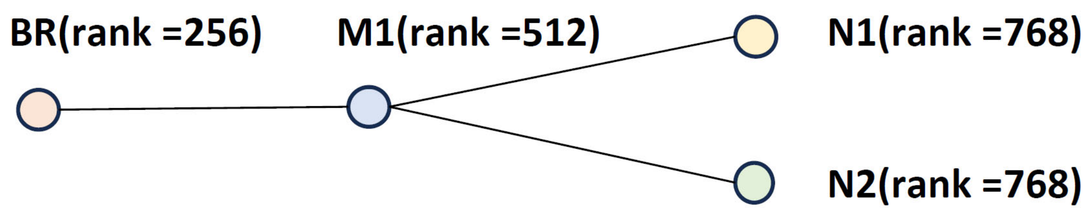


Figure 4.5. Simplified nodes deployment layout in tree topology.

When all three nodes (N1, N2, and M1) were programmed to wake up simultaneously, the radio activity observed in the network became significantly more random, although the types of messages remained consistent. This randomness is firstly characterised by the timing of NA/NS exchanges, which are no longer predictable and regular due to the simultaneous radio activity of multiple nodes. Moreover, when both N1 and N2 woke up simultaneously, each attempted to send a DAO message to its parent node. However, due to contention, the transmission of DAO messages was significantly delayed and subjected to retransmission. Furthermore, the DIS multicast from BR triggered a large volume of DIO transmissions, causing the entire channel to become congested. This congestion also delayed the transmission of other network maintenance activities.

In contrast, when all three nodes (N1, N2, and M1) were programmed to wake up sequentially, more structured and predictable radio behaviour was observed. For example, M1 completes its network maintenance and data transmission before child nodes wake-up, and child nodes are programmed to wake up in turns as well. In this case, each child node sends a single DAO message to its parent and waits for the corresponding DAO-ACK. Upon receiving each DAO, M1 forwarded a DAO message to the BR to complete the routing update. Only a limited number of NA/NS exchanges were observed under this sequential wake-up strategy, indicating that neighbour discovery activities (ND) were more predictable when child nodes were waking up sequentially. Figure 4.6 has illustrated the IPv6 ND and RPL traffic among N1 or N2, M1, and BR. However, additional NA/NS exchanges from the BR were detected when the communication window was intentionally extended to include redundancy (e.g., 1 minute). This also evident that longer radio on-hookup durations may trigger avoidable control message exchanges.

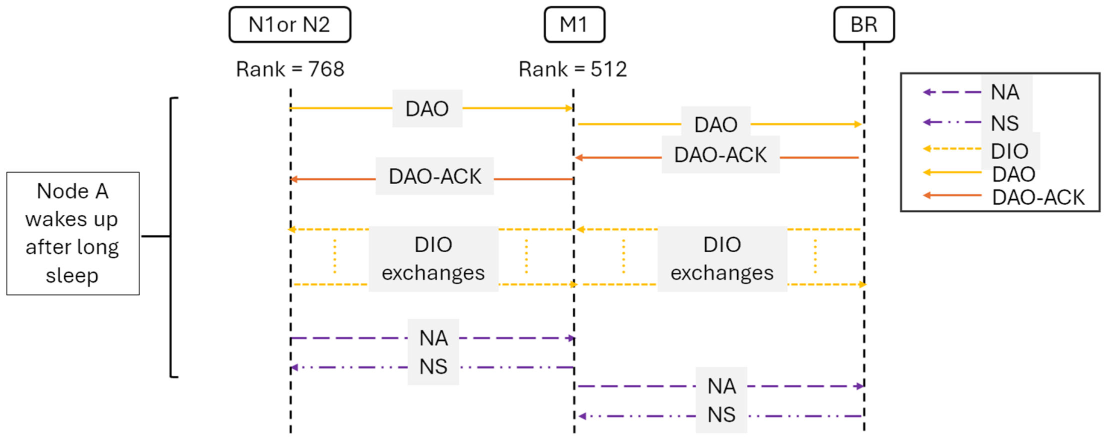


Figure 4.6. IPv6 ND and RPL traffic among N1 or N2, M1, and BR. M1 has completed its network maintenance and data transmission, and its child node (either M1 or M2) thereafter wakes up. And at any given time, only one child node of M1 may be active.

Overall, these observations demonstrated that sequential wake-up scheduling can effectively reduce the unexpected radio activities observed under simultaneous wake-up conditions. By allowing intermediate nodes (e.g., M1) to fully update routing and neighbour information before leaf nodes (e.g., N1 and N2) attempt to wake up, frequent control message exchanges are minimised.

4.3.2 The minimum time required for Network Maintenance after wake-up

As shown in the previous section, nodes waking up from long sleep must complete neighbour registration and routing information updates before reliable data transmission can take place, regardless of whether the wake-up strategy is sequential or simultaneous. These routing updates may be triggered either passively, by awaiting periodic routing update messages generated by the network, or actively by allowing nodes to initiate the rejoining process. As introduced in Section 2.3.5.2, node itself can send DIS message to request and find DODAG to join, or it can initiate DAO message to inform its reachability to parent node and DODAG root.

To minimise the duration of network maintenance after wake-up, this section first investigates whether routing information updates can be accelerated by allowing nodes to actively transmit DIS or DAO messages after wake-up, rather than passively waiting for routing maintenance traffic triggered at random or fixed intervals. Two active rejoining approaches were therefore tested for initiating RPL route updating:

1. When the node's scheduled wakeup time is reached, the node will actively send DIS messages instead of waiting for the default 60-second interval defined by the DIS timer at BR side.
2. A node actively unicasts a DAO message to the parent node after wakeup to indicate its intention to rejoin the routing topology.

Sniffer-based observations indicated that although nodes can be configured to actively transmit DIS messages to initiate rejoining, this approach often introduced additional control message exchanges leading to congestion and unpredictable radio activities. This is because a DIS initiated by the node itself will report the addition of the new node to the existing DODAG root, thereby updating the entire DODAG information. In contrast, observations indicated that the approach where nodes actively sent DAO messages to their parent nodes when waking up to initiate reconnection was more feasible. This is because DAO transmission is designed to advertise a node’s reachability within an existing RPL topology, rather than to trigger the formation of new network state. Moreover, although nodes will eventually transmit DAO messages after waking up as part of the normal RPL maintenance process, actively sending DAO messages can reduce the additional and often uncontrollable waiting time before routing updates are completed. Thus, the second approach which allow node to send DAO after waking up, is a more controllable and timely mechanism for routing reconstruction, and is therefore adopted for the subsequent experiments.

To implement the sleepy test flow chart (Figure 4.2) described in Section 4.2 and to identify the minimum time required for network maintenance after node waking up, a dedicated sleepy test function was developed. The pseudocode for the general test flow of sleepy test is shown in Table 4.1, and the function code was provided in Appendix B.3 (the complete code can be found at the link in Appendix B.2.). For both intermediate node and child node, sleepy test was performed to identify the minimum time required for neighbour-registration and routing information updates.

In this sleepy test, nodes were programmed with an initial sleep duration followed by a network maintenance stage after wake-up. To complete the network maintenance, nodes were designed to send DAO message to its parent node actively after waking up, and then waited for DAO_ACK to be received. Subsequently, each node transmitted a CoAP PUT request to test whether the network maintenance had been completed. By identifying the success or failure of CoAP data delivery, node gradually increased the T_hold duration until a safe gap is reached and applied. In this way, the minimum safe duration required for neighbour registration and routing information updates was determined.

Table 4.1. Pseudocode for the general test flow of sleepy test.

Pseudo-code for sleepy function

Input:

- T_sleep (minutes - default 3min): duration of sleep
- T_hold (seconds): duration of holding thread for neighbour-registration/routing updates
- (Optional)Wake-up sequence: for child node only, default 1 means first child to wake up, 2 means the next wakeup child node

Parameters:

- middle_gap (seconds): child node only, equals to the communication window of intermediate node

```
Turn off radio and enter low-power mode
//Default 120 iterations maximumly
while int i = 1; i < 120; i + + do
    Clear DS3231 alarm flag;
    Set next wake-up time = Current time + T_sleep;
    // for child node: wait for an extra middle_gap seconds
    Set DS3231 alarm;
    Print T_hold and T_sleep ;
    Switch system to standby mode;
    Upon wake-up:
    Turn on radio;
    Send DAO messages (retry up to 3 times if necessary);
    pause for a T_hold seconds;
    Attempt to send CoAP PUT; retry until CoAP ACK is received or max retries (default
    3) reached;
    Record and print success/failure counts;
    if success counts != 0 then
        T_sleep += 1min;
    end
    else
        T_hold ++;
    end
    Turn off radio;
end
After loop, turn on radio and finish execution;
```

To support wake-up scheduling with accurate timing in sleepy network design, this test developed a timing mechanism, and its pseudocode is shown in Table 4.2. This algorithm calculates the wake-up time for each node based on its sleep duration, wake-up order and RPL level. In addition, a fixed intermediate gap is introduced to separate the wake-up times of nodes with different ranks, which helps to avoid conflicts.

Table 4.2. Pseudocode for wake-up timer calculation.

Compute Target Wake-Up Time

Input:
- Current_time (seconds)
- T_sleep (minutes)
- Sequence (an integer indicating the node's wake-up order, and sequence ≥ 1)
- Window_size (communication window size in seconds)
- middle_gap (seconds): communication window size for intermediate node
- N (rank number)

Output: WakeupTime (in seconds; or as a time structure)

```
begin
    sleep_in_seconds ← T sleep × 60;
    //Convert sleep duration from minutes to seconds
    base_time ← Current_time + sleep in seconds;
    //Compute the wake-up time without additional offsets
    Wakeup_time ← base_time + (middle_gap ∗ (N/256-2)) + (sequence − 1) * Window
    size;
    return Wakeup time;
```

Using this test procedure, the required minimum network maintenance time was identified for different wake-up strategies (whether sequential or simultaneous wake-up) within a tree topology. Firstly, in a simplified tree topology involved a single child node (N1) and its parent (M1), M1 was kept active with its radio switched on and N1 were sending DAO actively to update the routing information with 10 mins fixed sleep duration. Typically, in a 2-day test, N1 was able to transmit a CoAP PUT request within approximately 1 s after actively sending a DAO message, and the longest time that CoAP PUT cost was 3s (with 2 CoAP PUT attempts: one failed one delivered). Over the entire test duration, a data delivery rate of 100% was achieved, with a measured transmission success rate of 83.7%. This result demonstrates that active DAO transmission enables effective routing updates without requiring nodes to wait passively for network-triggered maintenance process.

When the network was extended to include two leaf nodes (N1 and N2) at the same rank level (rank = 768), same as previously illustrated in Figure 4.5, after M1 has finished communicating, both N1 and N2 can update the routing information by sending DAOs actively. In the case where N1 and N2 sequentially entered their communication windows after M1 completed its transmission, each node required approximately 3 seconds (including the active DAO sending) to complete neighbour registration and routing information updates. In contrast, when N1 and N2 were set to wake up and communicate simultaneously, the communication delay between the two nodes was further increased, with each node requiring about 5 seconds to complete the same network maintenance process. In addition, it was observed that if two child nodes simultaneously attempted to transmit CoAP PUT requests, the resulting wireless communication congestion would lead to a significant number of retransmissions, thereby affecting the system's responsiveness and energy efficiency. Therefore, it is necessary to schedule N1 and N2 to stagger the timing of sending CoAP PUT request if nodes are scheduled to wake up simultaneously.

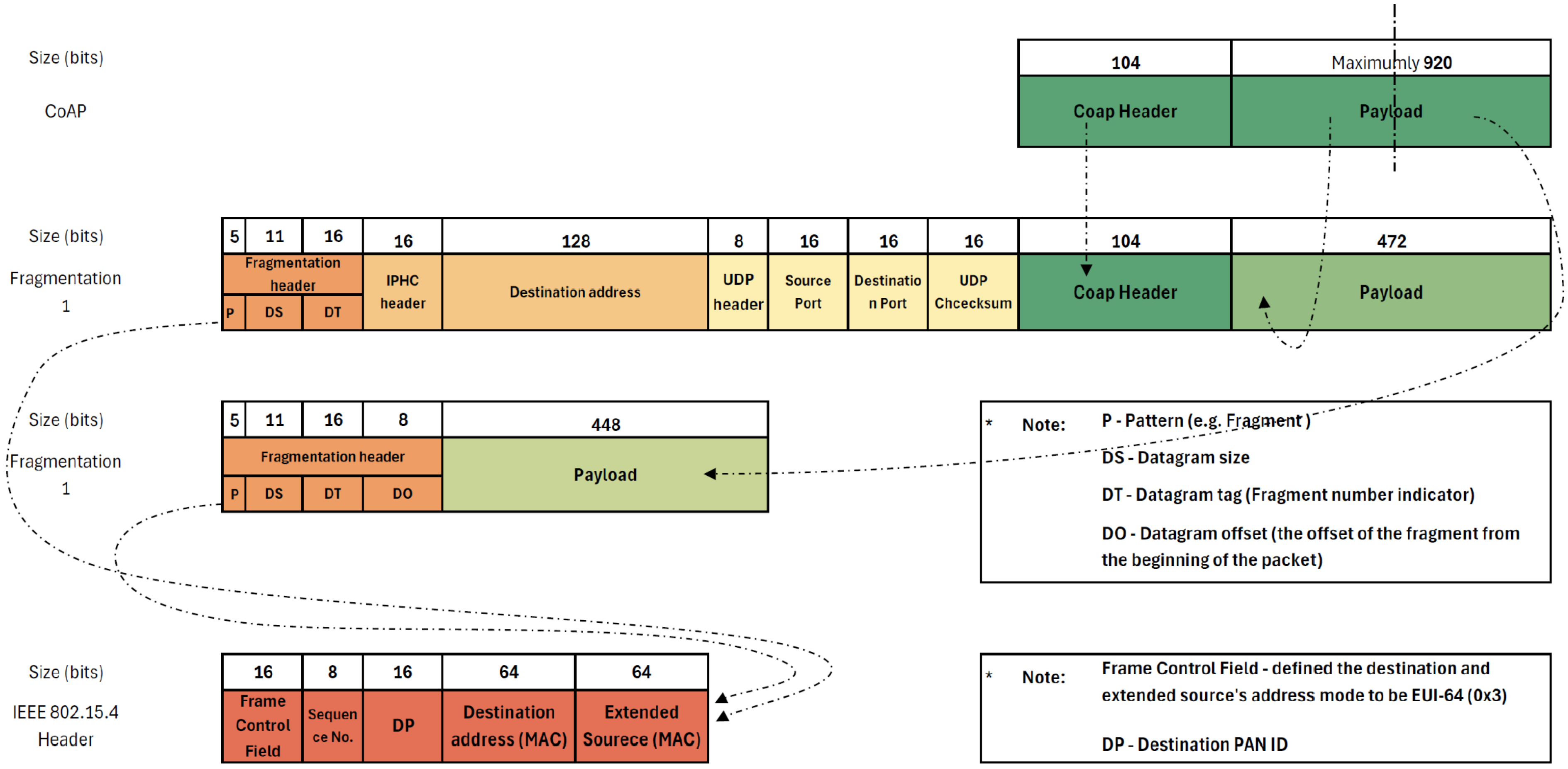
4.4 Payload Fragmentation and Retransmission of CoAP

As shown in the previous section, when multiple nodes in a tree topology wake up and attempt to communicate simultaneously, increased contention can lead to repeated neighbour discovery exchanges and retransmissions. Under stable network conditions, retransmissions are typically infrequent. When multiple nodes contend for the channel within a limited communication window, retransmission may occur and result in unnecessary energy consumption, data transmission delays, and data loss. In addition to minimising the time required for network maintenance, the duration of data transmission using CoAP requests must also be carefully constrained. This involves accounting for the impact of payload fragmentation mechanisms on the minimum duration required for data transmission, as well as the algorithm-controlled transmission gap between CoAP retries. Therefore, the subsequent two subsections, 4.4.1 and 4.4.2 investigates the impact of CoAP payload fragmentation and retransmission behaviour on the minimum required data transmission window length.

4.4.1 Payload Size Analysis and 6LoWPAN Fragmentation

To investigate the impact of application-layer payload size on data transmission time, sniffer-based observations were performed on CoAP PUT messages generated by the RIOT-OS CoAP implementation. The objective of this analysis is to identify the application layer payload constrains controlled by RIOT-OS, thereby determining the PSDU size of fragmented packets. Based on the analysis of captured packets, it was observed that the maximum supported CoAP payload size is 115 bytes. This limitation is determined by the fixed CoAP Protocol Data Unit (PDU) buffer size defined in RIOT-OS, which is 128 bytes in total [204]. This buffer includes a 13-byte CoAP header, leaving at most 115 bytes available for application-layer payload.

Sniffer observations further indicated that CoAP payload of 115 bytes triggers 6LoWPAN fragmentation. Specifically, the captured packets show that a single CoAP message with a 115-byte payload is fragmented into two packets**,** with PSDU sizes of 122 bytes and 82 bytes, respectively (see Appendix C.1 for the corresponding Wireshark capture). Figure 4.7 illustrates the IEEE 802.15.4 frame encapsulation process following 6LoWPAN fragmentation. As illustrated, the first fragment begins with a 32 bits fragmentation header followed by a compressed IPv6 Header Compression (IPHC) header, UDP header (compressed), CoAP header, and partial payload. While the second fragment contains the remaining payload bytes and the corresponding fragmentation header. Each fragment is then passed independently to the IEEE 802.15.4 MAC layer, where it is encapsulated with the appropriate MAC header before transmission.



* Note: P - Pattern (e.g. Fragment)

DS - Datagram size

DT - Datagram tag (Fragment number indicator)

DO - Datagram offset (the offset of the fragment from the beginning of the packet)

* Note: Frame Control Field - defined the destination and extended source's address mode to be EUI-64 (0x3)

DP - Destination PAN ID

Figure 4.7. Encapsulation and Fragmentation Process from CoAP to IEEE 802.15.4 Data Frame

4.4.2 Reliable CoAP Transmission Design under Payload and Timing Constraints

Table 4.3. Pseudocode for CoAP PUT request and retransmission mechanism.

```
CoAP Message Transmission and Retransmission Mechanism

message_ack_flag ← 0;

In Function_CoAP_Resp_Handler(response);
begin
    if response indicates timeout or error then
        message_ack_flag ← 0;
    else if response is valid (ACK received) then
        message_ack_flag ← 1;

In Retransmission Mechanism of CoAP PUT
Begin
    retries ← 0;
    while (message ack flag == 0 and retries < MAX RETRIES) do
        message_ack_flag ← 0;
        Send CoAP Message;
        Start timer for period T;
        wait ← 0;
        while (message_ack_flag == 0 and wait < 3) do
            Hold the thread for 0.2*3 s ;
            wait ← wait +1;
        if (message_ack_flag == 1) then
            return success;
        else
            retries ← retries +1;
    if (message_ack_flag == 1) then
        return success;
    else
        return failure;
```

Subsection 4.4.1 has identified the maximum supported CoAP payload size in RIOT-OS is 115 bytes. Based on this constraint, up to six groups of 18-byte reading can be included in one single CoAP PUT request in the system of this study. Table 4.3 introduces the pseudocode of CoAP PUT request and retransmission method in this work. A global flag variable, $message\_ack\_flag$ is used to indicate whether a CON-CoAP message has been acknowledged by the routing parent. After sending the message, the system enters a retransmission loop until an acknowledgement is received or the maximum number of retries is reached (default maximum retries is limited to 3 times).

According to the single-hop delay test results in Table 3.6 of subsection 3.3.2, the average maximum value was approximately 300ms. Accordingly, under the experimental arrangement shown in Figure 4.1, it can be conservatively estimated that the round-trip delay experienced by child nodes N1 and N2 in the tree topology could reach 600ms. Thus, a 600ms interval was selected for the CoAP retransmission mechanism. Subsequently, it was observed that when a single node transmits a CoAP PUT request using the retransmission mechanism described above, a data transmission interval of at least 2 seconds was required to ensure reliable transmission without triggering unnecessary retries. In the topology shown in Figure 4.5, when the three nodes (MA, N1 and N2) were activated sequentially, each node was able to complete the CoAP PUT request (including 6 groups of reading) within its allocated 2-second data transmission window without conflicts.

However, when both N1 and N2 were allowed to wake up simultaneously (allowing N1 and N2 to initiate CoAP requests at the same time), packet collisions and duplications were observed, as well as unpredictable NA/NS radio activities. When N1 and N2 nodes are programmed to start CoAP transmissions sequentially (with a 2-second interval), CoAP packet duplication was effectively avoided, although unpredictable NA and NS radio activity were still observed.

4.5 Communication Window Design under Different Wake-Up Strategies

To evaluate and optimise the communication schedule in different wake-up strategies in a multi-node sleepy network, an experiment was deployed as described in Section 4.2 (see Appendix C.2 for the corresponding Wireshark captures). Two wake-up strategies were tested: simultaneous and sequential communication. As described previously, in the simultaneous approach, nodes N1 and N2 enter their communication stages concurrently, waking up immediately after node M1 completes its data transmission. Initially, to avoid unnecessary conflicts, N1 and N2 were enter the communication window sequentially. A summary of selected settings and key findings about this sleepy network test is listed below:

- Usually, CoAP PUT Requests (108 bytes) were completed within 1 second and were executed at the very beginning of the scheduled data transmission slot for CoAP.
- Confirmable (CON-CoAP) PUT request were enabled. Therefore, each received CoAP request was replied to with a corresponding CoAP ACK packet.
- Based on experimental observations, it was found that the 2-second time slot duration for CON-CoAP PUT request was the sufficient and reliable to complete the data transmission, including the activation of the retransmission mechanism (up to three retries by default).
- Multicast DIS messages initiated periodically (every 60 seconds) by BR were found to be essential for maintaining RPL information for all the nodes. When DIS is transmitted at the beginning of an intermediate node's communication window, it was more likely that this information will be received reliably.

Note that the time reading of DS3231 RTC supported in RIOT-OS is second-level precision and epoch time was used for time synchronisation. Figure 4.8 illustrates sequential wake-up communication window design in a 4 nodes tree topology scenario. The timeline shows both transmit (Tx) and receive (Rx) events across all nodes, with protocol activity color-coded by type (DAO/multicast-DIO/multicast-DIS, NS/NA, CoAP/CoAP-ACK, IEEE 802.15.4 ACK). Nodes M1, N1, and N2 are programmed to enter communication stage sequentially, as described in the task scheduling strategy at the top left of the figure. The optimised scheme can be described as three phases.

- ***Phase 1***: Intermediate node M1 wakes up first, actively sending DAO to DODAG root (BR) and waits for DIS initiated by BR, followed by the exchange of DIO messages between these two nodes. This networking stage of M1 will maintain for 6 seconds. Following that, during the data delivery slot, M1 is scheduled to send a CON-CoAP PUT request within a 2 second time slot. In total so far, M1 maintains the radio on state for 8 seconds, and then retains the thread and radio on state for an additional 10 seconds (5 seconds per child node for N1 and N2).
- ***Phase 2***: N1 wakes up 8 seconds after M1 wakes up, then turning on the radio at the scheduled time to actively send DAO message and wait for DIO exchanges within the first 3 seconds. Following that, M1 is designed to enter a 2 second slot for data delivery through CON-CoAP PUT request. The total length of communication window for child node N1 is 5 seconds.
- ***Phase 3***: N2 wakes up 13 seconds after M1 wakes up, then turning on the transceiver to repeat the same process as described in ***Phase 2*** for another 5 seconds.

At the very beginning of the communication window (8 seconds for M1 and 5 seconds for N1/N2), the nodes all actively send DAO messages. Consistent with the previous discussion, when nodes are activated sequentially, the number of retransmissions and unexpected neighbour registration activity can be minimised if DIS is transmitted at the beginning of an intermediate node's communication window. This communication window design under the sequential wake-up strategy has been tested in the lab for 3 days with the communication interval of 60 minutes. As the success rate was identified to be 95.5% with data delivery rate of 100%. After 3 days, the test results indicated that the success rate was 81% with data delivery rate of 100%.

The experiments on the communication window optimisation under the simultaneous wake-up strategy initially followed the experience in which N1 or N2 took 3 seconds to complete the DAO exchange. Although the message queue is automatically managed by RIOT when two nodes start sending DAO messages at the same time, the need to exchange NA/NS between nodes was observed to be increased. This result argues that waking up N1 and N2 simultaneously (and actively initiating a DAO) may elevate the priority of updating or confirming the neighbour state in the NDP, and therefore trigger the associated NS/NA activities. Therefore, compared to the sequential wake-up strategy, it is tested that the working design is to increase the communication time reserved for N1 and N2 to 5 seconds before sending a CoAP request. Similarly, NA/NS message exchange activities between N1/N2 and M1 were observed during the reserved CoAP activity slot. As a result, the duration of the data transmission time slot was tested to increase to 5s. However, although it was observed in the early stage of the test that the message transmission was sent with a high success rate according to the expected pattern (illustrated in Figure 4.9), the exchange of NA/NS messages between M1 and N1/N2 and BR disrupted the original CoAP transmission schedule, resulting in delayed CoAP transmission and multiple retransmissions. The optimised scheme in Figure 4.9 can be described as two phases.

Phase 1: Intermediate node M1 wakes up first, actively sending DAO, and wait for DIS and DIOs exchanges between itself and BR. Similar to sequential wake up optimisation, the networking stage of M1 will last 6 seconds and data delivery slot will schedule as 2 seconds, in total of 8 seconds. Then intermediate node M1 retains the thread and radio on state for an additional 10 seconds.

Phase 2: N1 and N2 wake up together to actively send DAO messages to parent node. The networking stage is reserved for 5 seconds for both child nodes, and the data delivery slot is reserved for 5 seconds. Compared to sequential wake up optimisation schemes, the redundancy in this longer communication window for child node is reserved for possible neighbour discovery activities.

To decrease the CoAP retransmission, the next test tried increasing the time slot reserved for DAO messages and CoAP sending to 10s (extending the communication window to 20s in total), and the delivery rate was found to be less than 70% after 24h of testing (60 minutes communication rate) as the CoAP retransmission is unavoidable. Therefore, even if the duration of the reserved CoAP transmission window for each node is increased by 400% compared to the setting in the simultaneous wake-up strategy, there will still be uncontrollable CoAP retransmissions. Also, compare with sequential wake-up schedule in Figure 4.8, the amount of radio activities and the length of communication window are both increased which will result in a higher system energy consumption. Therefore, the energy consumption evaluations in the next section will be based on the communication window optimization design under the sequential wake-up strategy.

In both Figure 4.8 and Figure 4.9, each CoAP PUT request shown in the diagrams represents a payload of 108 bytes. As described in Section 4.4, each request is fragmented into two packets with PSDU sizes of 122 bytes and 82 bytes respectively. For each fragment received, a link-layer ACK (IEEE 802.15.4 ACK) is issued by the receiver. Thus, the complete CoAP transmission process involves two fragments and corresponding MAC ACKs. However, to improve clarity and readability, only one CoAP request event is depicted in the diagrams. This simplification avoids clutter while still representing the full CoAP transaction. It is understood that the complete CoAP transmission process involves two fragments and their corresponding ACKs in actual operation.

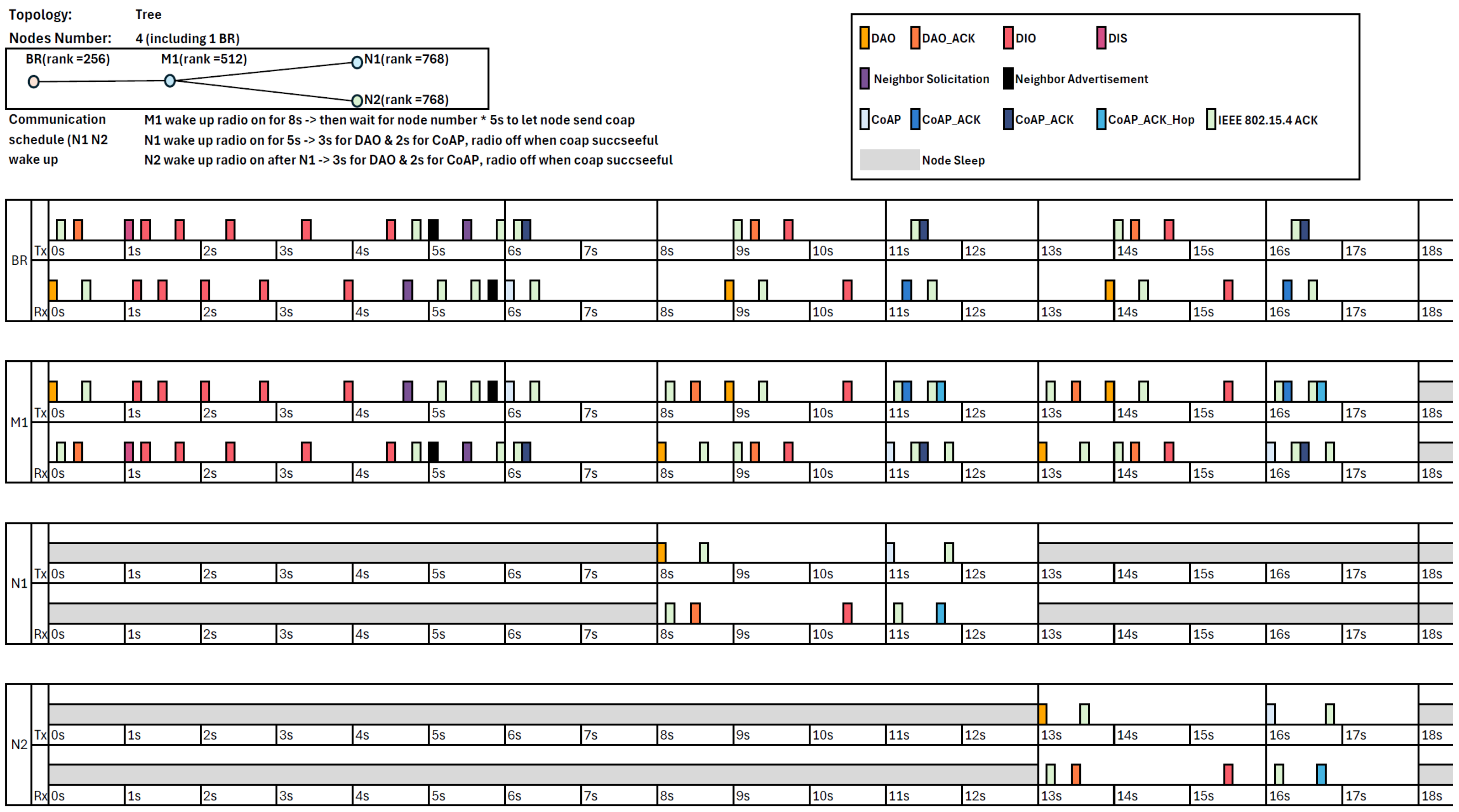
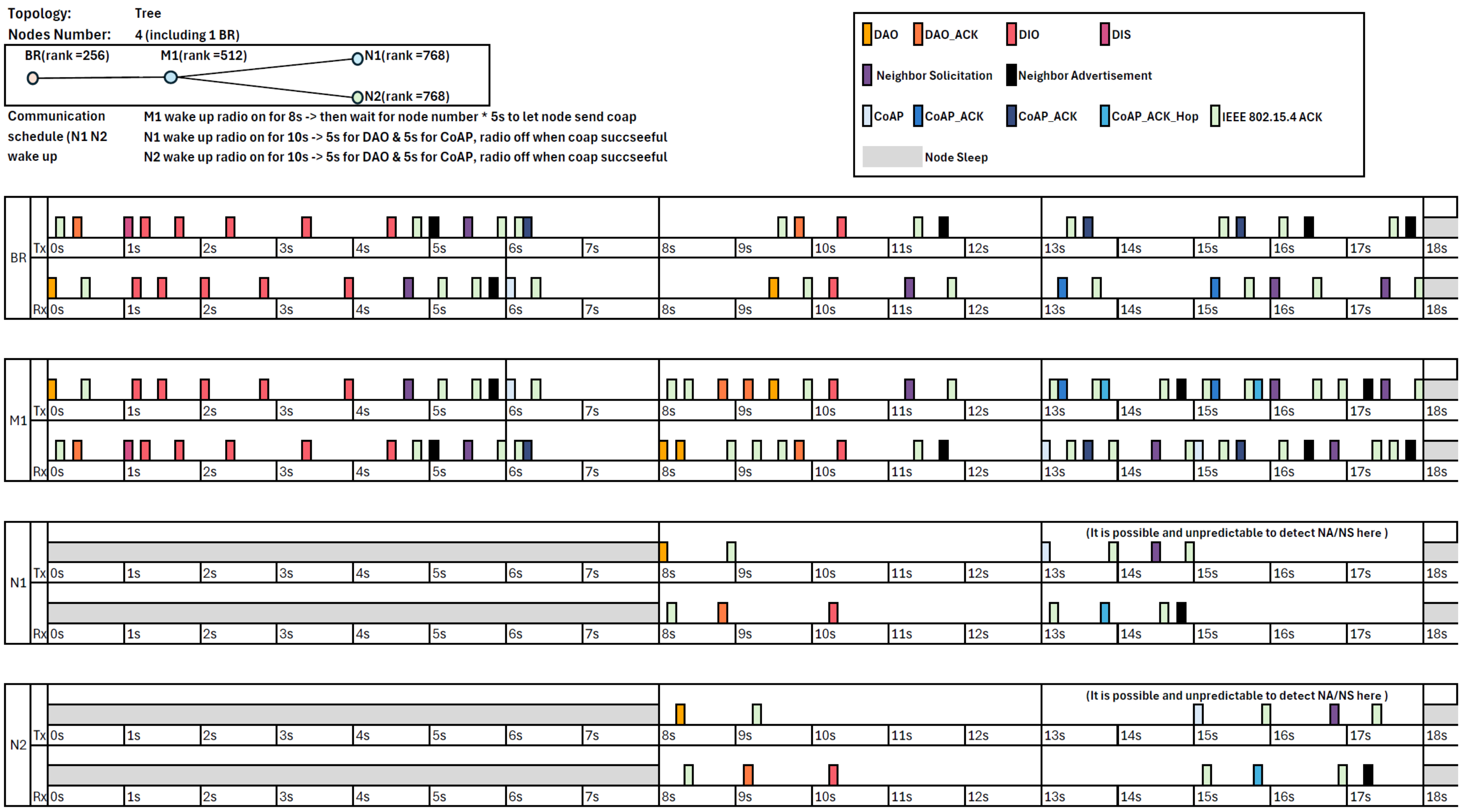

Topology: Tree
Nodes Number: 4 (including 1 BR)
BR(rank =256)
M1(rank =512)
N1(rank =768)
N2(rank =768)
Communication schedule (N1 N2 wake up
M1 wake up radio on for 8s -> then wait for node number * 5s to let node send coap
N1 wake up radio on for 5s -> 3s for DAO & 2s for CoAP, radio off when coap succseeful
N2 wake up radio on after N1 -> 3s for DAO & 2s for CoAP, radio off when coap succseeful
DAO
DAO_ACK
DIO
DIS
Neighbor Solicitation
Neighbor Advertisement
CoAP
CoAP_ACK
CoAP_ACK
CoAP_ACK_Hop
IEEE 802.15.4 ACK
Node Sleep
BR
M1
N1
N2
Tx
Rx
0s
1s
2s
3s
4s
5s
6s
7s
8s
9s
10s
11s
12s
13s
14s
15s
16s
17s
18s

Figure 4.8. Optimisation of the communication window under sequential wake-up strategy. M1 wakes up for 18s (6s for M1's network updates, 2s for sending M1's data using CoAP, and then the rest 10s reserved for two child nodes); N1 and N2 both wake up for 5s sequentially (3s for network updates and 2s for sending CoAP packet).



Communication schedule (N1 N2 wake up

M1 wake up radio on for 8s -> then wait for node number * 5s to let node send coap

N1 wake up radio on for 10s -> 5s for DAO & 5s for CoAP, radio off when coap succseeful

N2 wake up radio on for 10s -> 5s for DAO & 5s for CoAP, radio off when coap succseeful

Figure 4.9. Optimisation of the communication window under simultaneous wake-up strategy. M1 wakes up for 18s (6s for M1's network updates, 2s for sending M1's data, and then the rest 10s reserved for two child nodes); N1 and N2 both wake up for 10s simultaneously (5s for network updates and 2s for sending CoAP packet).

4.6 Energy Consumption Modelling for Tree Topology Sleepy Networks

The previous section has explained the design of communication window scheduling based on different wake-up strategies. As discussed above, sequential wake-up strategy can potentially avoid radio contention and enable the system to achieve more stable timing behaviour. Therefore, in the case of applying the sequential wake-up strategy, this section will attempt to estimate the energy consumption of the nodes at each level of the topology and the daily energy consumption of the system, includes ND, RPL control messages and CON-CoAP-based data transmission.

4.6.1 Timing and Energy Consumption Estimation of Network Activities

Although subsection 3.2.4 provides measurements of average current and timing for different system activities, not all radio activities have been individually measured compared to those covered in Section 4.3. Although it is possible to obtain more accurate energy consumption data through direct current measurements for each radio activity on individual node with different position in the tree topology, such an approach can be replaced by estimation for system-level energy consumption evaluation. While measurement-based analyses can provide more accurate estimates of daily energy consumption, the estimation methodology used next in this work is sufficient to obtain reliable results within an acceptable inaccuracy. In addition, optimisation results for the communication window established in Section 4.5, is mainly based on protocol behaviour and timing requirements rather than the exact time consumption of the different activities. Therefore, neither the direct measurement of radio activity nor the estimation through analysis affects the scheduling logic, as well as the optimisation results in the previous section.

In terms of RPL radio activities, subsection 3.2.4 does not cover the measurement of DIS message exchange. Consequently, the subsequent discussion will refer to the PPDU timing for BPSK 20 kbps in the SAM R30 Xpro board datasheet [188]. Referring to the section 14.3 of SAM R30 Xpro datasheet [188], the maximum PSDU (127 bytes) timing under BPSK (20 kbps) is 50.8 ms which results in a rate of 0.4 ms/byte. The typically DIS message without frame check sequence (FCS) captured by sniffer is 29 bytes, 64 bytes for DAO message, 32 bytes for DAO-ACK message, and 79 bytes for DIO message. According to the current and time duration measurements mentioned in subsection 3.2.4, the average time duration for DIO message is 93 ms, 139.5 ms for DAO and DAO-ACK messages together. Generally, RPL control messages such as DIO, DAO, DAO-ACK, and DIS are generated and processed at the network layer, and they are encapsulated by the 6LoWPAN adaptation layer and passed on to the MAC and PHY layers next. It can therefore be assumed that, prior to transmission at the PHY level, protocol processing and queuing overheads vary only slightly between different network layer message types. To calculate the PPDU timing, the time cost for 5 bytes synchronisation header (SHR) and 1-byte PHY header (PHR), as well as the frame check sequence (FCS), should be considered. Referring to the datasheet, the PPDU timing can be calculated using the equation 4.1 below where FCS_length is 2 bytes:

$$\begin{aligned} T_{PPDU} &= T_{PHR} + T_{SHR} + PSDU_{bytes} * 0.4\, ms/byte \\ &= 2.0\, ms + 0.4\, ms + \left(Packet_{bytes} + FCS_{bytes}\right) * 0.4\, ms/byte \end{aligned} \quad 4.1$$

Based on the PPDU timing calculations for BPSK at 20 kbps, the physical-layer transmission durations for typical RPL control messages are estimated to be 34.8 ms for a DIO message (79 bytes), 28.8 ms for a DAO message (64 bytes), and 16.0 ms for a DAO-ACK message (32 bytes). And the time overhead except PPDU timing for DIO message is 58.2ms, and 94.7ms for DAO and DAO-ACK together. Therefore, the average time overhead except PPDU timing across RPL control messages is approximately 50.97 ms. For the DIS message (29 bytes), which was not directly measured in the previous experiments, the total time duration is estimated to be 65.77 ms based on above discussion. This average non-PPDU time overhead is 50.97 ms including MCU processing/encapsulation, buffer reading, 6LoWPAN compression and fragmentation, MAC-layer framing, and channel access delay due to CSMA/CA check. However, given the current capabilities of embedded processing technologies, the onboard frame processing time is typically very short, often shorter than the minimum interframe spacing (IFS) of 12 symbol periods (0.6 ms) specified by the IEEE 802.15.4 standard [76]. The extra time overhead can be caused by MAC layer processing and channel assessment, as well as the onboard processing time cost. RIOT-OS has applied the unslotted CSMA/CA. Thus, referring to IEEE 802.15.4-2011, the default maximum number of backoffs is defined as 4 in CSMA/CA and the default BE value is ranging from 3 to 5. According to Equation 2.1, with 20 kbps BPSK modulation, a single delay caused by each backoff may reach 7 ms, 15 ms, 31 ms, and 31 ms respectively (duration of symbol period is 50 µs [205]). The accumulated delay caused by backoffs can reach to 84 ms and Anwar et al.'s study [206] has indicated that the latency will increase exponentially if the CCA fails, and the caused delay is random and unpredictable.

The IEEE 802.15.4 MAC acknowledgement (MAC-ACK) exchanges are captured after Neighbour Discovery (ND) activities, DAO/DAO-ACK transmissions and forwarding, as well as CoAP message transmissions. Figure 4.10 illustrates the MAC-ACK frame structure. According to sniffer observations, the MAC-ACK frame contains a 3-byte MAC header (MHR) [62], including a 2-byte Frame Control field and a 1-byte Sequence Number.

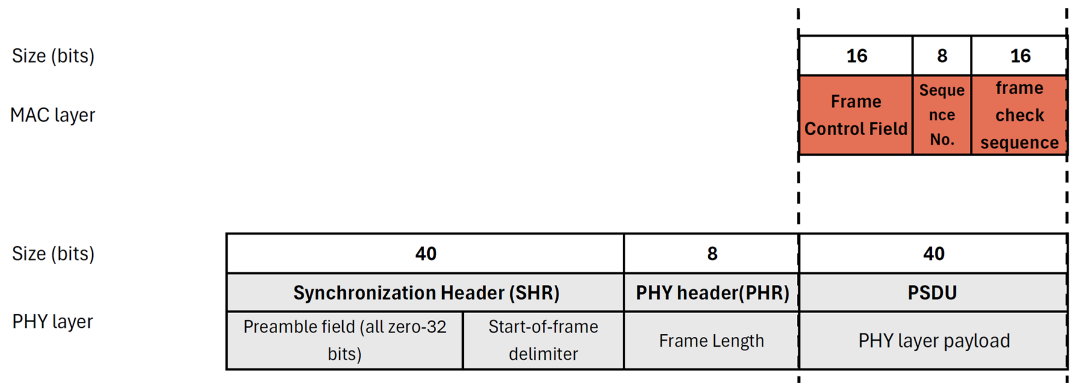


Figure 4.10. IEEE 802.15.4 MAC-ACK frame structure (adapted from [188] and [62]).

Under BPSK modulation, the maximum MAC timeout for receiving an IEEE 802.15.4 ACK frame is theoretically 264 symbols (20 kbps), which corresponds to a duration of 13.2 ms [188]. In actual sniffer-based measurements, the observed interval between data transmission and ACK reception is also reasonably distributed between 20 ms and 30 ms. Due to the short duration and limited energy impact of MAC-ACK, this study does not use existing measurement tools to directly measure the MAC-ACK duration and current consumption. In addition, in RIOT-OS, the transmission and reception of MAC-ACKs are programmed in the radio driver and managed internally. The time duration of a MAC-ACK frame can be estimated to be approximately 4.4 ms using equation 4.1 above and the current consumption will be estimated same as the communication state average current. This estimation is further supported by Anwar et al.'s study [206], which showed a transmission time of 0.608 ms for a 4-byte MAC payload at 2.4 GHz/250 kbps which corresponds to about 7.3 ms under 20 kbps. Therefore, considering the different lengths of PHY preamble, FCS and 0 payload in MAC-ACK, the 4.4 ms estimate used in this study is a reasonable approximation.

Also, according to the SAM R30 Xpro datasheet, the average current in receiving mode is 9.4 mA, which is approximately 51.65% of the average current in transmitting mode (18.2 mA) [188]. Moreover, for ND, RPL or CoAP messages where the current consumption at the receiving end is not measured in section 3.2.4, the average receive current will be estimated as 51.65% of the corresponding transmission current. To evaluate and estimate the daily energy consumption for each node, the RX current reading of 9.4 mA will be used for the full duration of the estimated DIS reception (65.77 ms), which includes not only the PHY-layer processing but also the processing and decoding activities in the upper protocol layers.

For each message type, the transmission or reception duration is also related to the actual packet size. However, the actual packet size, such as RPL and CoAP messages, varies depending on the length of the options and payloads to be carried. Although packet transmission durations vary in size, the average current consumption values previously measured in 4.4.2 for the are still reasonable to simulate the node's energy consumption because the measured current averages are based on a long sampling time of the radio activity. According to the radio observation results in section 4.3, as an example, a DAO message transmitted from node N1 or N2 to intermediate node M1 typically contains about 62 bytes, including a 32-byte information contains in RPL option field; subsequent DAO messages forwarded from M1 to the BR carry additional RPL options including all the child's transit information, and packet size is increased due to the two additional RPL options. For each additional child node, an additional 26 bytes of information will be added to the subsequent DAO message carrying additional RPL options, containing the RPL target information (the IPv6 address of the child node) and the RPL transport information (path control frame). Similarly, for the CoAP message, CoAP PUT request forwarded from N1/N2 to M1 was 9 bytes smaller than the packets that M1 forwarded to BR due to the extra 8 bytes source address and 1 byte hop limit frame.

Table 4.4. Time durations and descriptions of estimated radio activities.

	Message	Descriptions	PSDU size without FCS (Bytes)	Time Duration (ms)
Measured	DIO	DODAG Information Object messages	79	93
	DAO	DODAG Advertisement Object	64	139.5
	DAO-ACK	DAO Acknowledgement	32	
	CoAP response	A CoAP response contains 18 bytes data	78	130
Estimated	DIS	DODAG Information Solicitation messages	29	65.77
	DAO-H	The DAO messages send by the parent node to upper node after receiving DAO from its child. Assuming the number of child nodes is N_child.	64+26*N_child	94.7*0.5+28.8+0.4*26*N_child =76.15+10.4* N_child
	IEEE 802.15.4 ACK	The MAC acknowledgment frame used to confirm the reception of packet	3	4.4
	CoAP	CoAP message initiated by source node (108 bytes data)	122 and 75 bytes in two fragments	182.8
	CoAP-H	CoAP message forwarded by intermediate nodes (108 bytes data)	123 and 83 bytes in two fragments	186.4
	CoAP-ACK	The acknowledgement frame used to confirm the reception of CoAP request	56	121.2
	CoAP-ACK-H	The CoAP acknowledgement frame forwarded by intermediate nodes with extra 8 bytes destination address frame	64	124.4

To initially evaluate the system daily energy consumption, section 3.2.6 estimated daily energy consumption without considering the actual CoAP header and the extra PPDU time cost by the extra payload. In section 3.2.4, the CoAP GET request time duration is measured based on 18 bytes data payload (totally 80 bytes PSDU), including temperature data and timestamps. For 115 bytes payload size, two fragments will be generated and format 201 bytes PSDU including 2 FCS frames. Thus, the extra time overhead can be calculated as 52.4 ms including a 40 symbols (2 ms) minimum longer IFS (LIFS) [76]. Additionally, the CoAP PUT request forwarded from M1 to BR would cost another extra 3.6 ms due to the PSDU increasing for 9 bytes. Similarly, for CON-CoAP request, the CoAP-ACK is normally 56 bytes, and the CoAP-ACK that BR send to intermediate node for hopping to the destination will have extra 8 bytes destination address frame. As discussed before, the onboard processing time including the application and network layer could be lower than 0.6 ms, hence the additional processing time due to the increased payload length will be ignored. Next, based on the findings in subsection 3.2.4, the time duration for different radio activities is summarised and listed in Table 4.4 according to the observation in section 4.3 including the estimated results discussed above.

4.6.2 Energy Consumption Model for Sequential Wake-Up Strategy

The deployed nodes will firstly enter the networking stage to synchronise RTC timestamp with BR and form the network. After the initial synchronisation, the node goes to sleep until the start of the beginning of next hour. From then on, all the nodes will enter the sleepy network schedule as introduced in Figure 3.6 and aligned to start precisely at the beginning of that hour. During the event loop, all the child nodes will be active sequentially to do the communication as shown in Figure 4.8. For all nodes, the time of entering the communication window is aligned, but there is a certain interval between the switching on of the transceiver and the start of the communication window for each node according to the requirements of sequential wake-up. For calculating the daily energy consumption of the nodes, the energy consumption of the nodes at each level of the topology is modelled next for the sensing stage, the communication stage and the sleep stage. Note that the board is working at 3.3V regulated by two on-board regulators. To distinguish between the various ranks of nodes, the $\mathrm{RPL\_Rank}$ values of the nodes will be used in the modelling to calculate the energy consumption of the nodes in the topology as shown in Figure 4.5. Additionally, the data sampling rate is every 10 minutes, and transmission rate is every 30 minutes according to the most frequent rate from the discussed battery-powered long-term environmental monitoring projects in Section 2.7. Parameters used in this subsection are listed and explained as below:

- N_s –the total number that node periodically reads the sensor based on the data sampling rate in minute.

- N_c – the total number that node periodically enters the communication window based on the communication rate in minute.
- R_s – the data sampling rate in minute.
- R_c –the communication rate in minute.
- E_sense- energy consumption for one entire sensing window.
- E_sense'- energy consumption for the sensing window following the communication window.
- $N_{sequence}$- sequence number to identify the wake-up sequence for nodes with same RPL rank value.
- R- the RPL rank value of the node.
- N_child- the number of child nodes associated to this parent node.
- E_{other}- representing the sum of energy consumption for waking up the node and turn on & off the transceiver as required according to the Figure 3.6.

For the sensing stage, node will wake up from sleep and enter the sensing state to read the sensor, store the data, and enter sleep state again once finished. For the subsequent sensing window next to the communication window, node will only read and store the data and enter to sleep state. The energy consumption can be calculated using equation 4.2 where E_{switch} is the average energy consumption node switch between sleep and sensing state. According to the Section 3.2, E_{sense} is 8.275 mJ and E_sense' is 8.036 mJ.

$$E_{sense} = E_{switch} * 2 + E_{read} + E_{store}$$

$$and\ E_{sense}' = E_{switch} + E_{read} + E_{store} \qquad 4.2$$

During the communication window, node will wake up and turn on transceiver to execute the networking activities until CoAP PUT request is successful. In a tree topology, M1, N1 and N2 will all enter the communication window aligned. However, N1/N2 will stay sleep until the previous node's scheduled communication is complete compared to M1 as explained in Section 4.5. $N\_sequence$ is defined here to distinguish the wake-up sequence that nodes with same rank value. For the intermediate node, M1, the energy consumption during the communication window involves the networking consumption $E_{networking}^{M1}$ and CoAP PUT request transmission consumption E_{CoAP}^{M1}. During the networking period, there are RPL message and ND message exchanging that can be expected. Following that, M1 still need to keep waking up and help N1 and N2 to forward the DAO, DAO-ACK and CoAP messages. For each child node, the energy consumption of M1 acts as an intermediate node is defined as $E_{hopping}^{M1}$. Regarding to the $E_{networking}^{M1}$, the typical number of multicast DIO messages transmitted and received by M1 are 6 and 4 respectively. Also, there will be mostly 2 times ND message (1 NS and 1 NA) exchange observed between BR and M1. Equation 4.3 has defined below to detail the energy consumption of communication window for intermediate node M1.

$$E_{com}^{M1} = E_{networking}^{M1} + E_{CoAP}^{M1} + E_{hopping}^{M1} * N_{child} + E_{other} \qquad 4.3$$

Based on equation 4.3, the energy consumption during the first 6 seconds networking stage after M1 waking up, $E_{networking}^{M1}$ is about 235.42 mJ. E_{CoAP}^{M1} is 83.93 mJ, $E_{hopping}^{M1}$ for 1 child node is 209.53 mJ and E_{other} is 2.76 mJ. Assuming there are 2 child nodes same as shown in Figure 4.5, then E_{com}^{M1} is about 741.17 mJ.

For each node with rank value of 768, it will wake up and turn on the radio after an offset period of sleep which is used to wait other node to finish the scheduled communication tasks. Taking the previous tree topology as an example, the $N_{sequence}$ can be 1 or 2 for different lowest layer node. If the N2 has the $N_{sequence}$ of 2, then before waking up, it will sleep for another 13 seconds once entering scheduled communication window starting time point. The energy consumption for N1 or N2 during communication window, E_{com}^{child}, can be detailed using equation 4.4.

$$E_{com}^{child} = E_{sleep_{offset}}^{child} + E_{networking}^{child} + E_{CoAP}^{child} + E_{other}$$
$$where\ E_{sleep_{offset}}^{child} = 3.3 * I_{sleep} * [(R \bmod 256 - 2) * 8s + (N_{sequence} - 1) * 5s] \qquad 4.4$$

M1, N1, N2's daily energy consumption can be formulated as follows. However, the E_{sense}^{node} including the CoAP data transmission energy consumption will be affected by the data sampling rate. As mentioned earlier, the energy stacks for each state will be illustrated next using data sampling rate of every 10 minutes and transmission rate of every 30 minutes. Specifically, if data sampling rate is 3 times faster than communication rate. There will be only 3 groups of data to be transmitted in one single communication window, results in a 54 bytes data payload. Then the estimated time duration for CoAP and CoAP-H messages are respectively 182.8 ms and 186.4 ms without considering fragmentation. Suppose the CoAP communication is ideal and no retransmission is required, the communication time should be 1 second.

$$E_{daily}^{node} = (E_{com}^{node} + E_{sense}^{node}{}') * N_c + E_{sense}^{node} * (N_s - N_c) + E_{sleep}^{node} \qquad 5.5$$

4.7 Model-Based Evaluation of Energy Consumption in a Tree Topology

4.7.1 Energy Consumption under the Optimised Communication Window design

Sleepy network daily energy consumption and energy stack for same data sampling rate (every 10 minutes) and two communication rate (every 60 minutes and 30 minutes) is illustrated in Figure 4.11. Similarly, compared with energy consumption of nodes in star topology in Section 3.2.6, the daily communication energy consumption rises as the communication rate becomes more frequent. If the communication rate is every 60 minutes and data sampling rate is every 10 minutes, node daily energy consumption is 56.37J for intermediate node and 42.67J for child node (47.23J on average for 3 nodes).

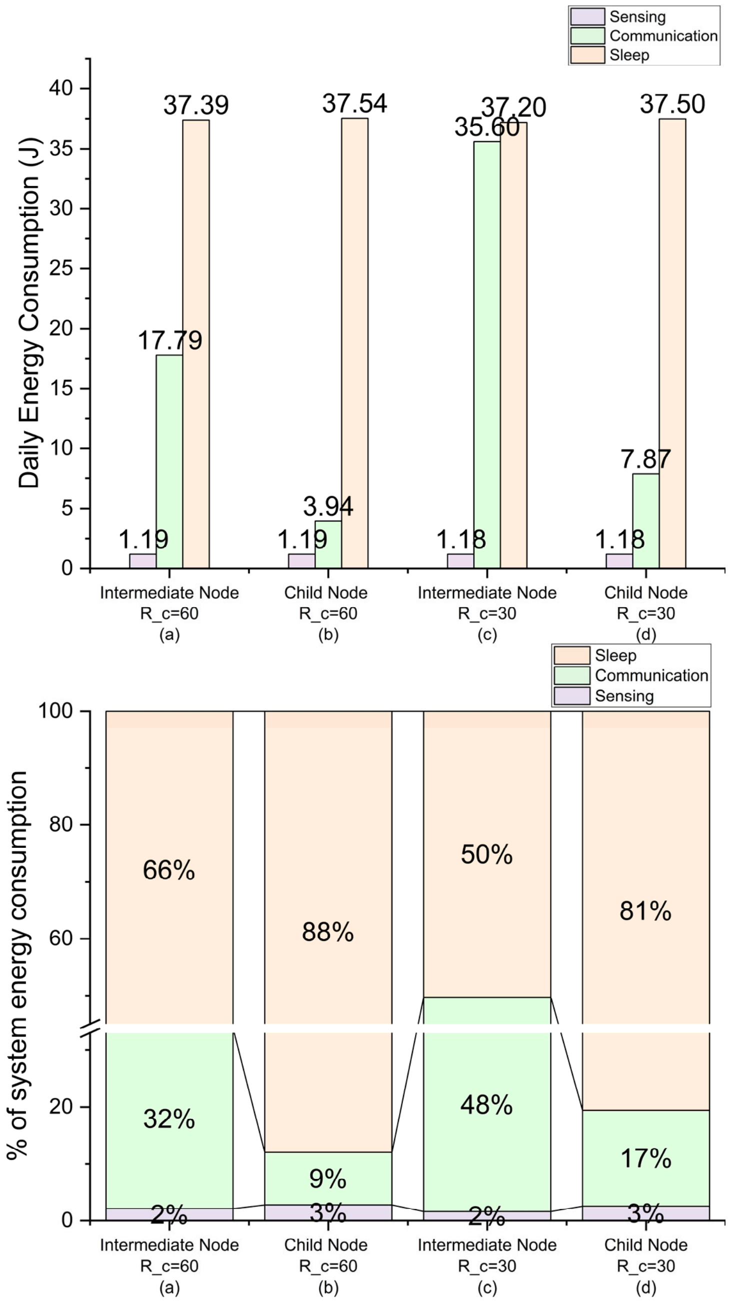


Figure 4.11. Sleepy network daily energy consumption (top) and energy stack (bottom) comparation. For both panels, from left to right, results are based on a fixed sampling rate of 10 minutes, with communication rate of 60 minutes ((a) intermediate node, (b) child node) and 30 minutes ((c) intermediate node, (d) child node).

Assuming the ContikiMAC or 6TiSCH is applied in this system, a daily energy consumption estimation (including sensing energy consumption) can be done as following. If ContikiMAC is applied, assuming the CoAP data transmission keep the same and duty cycle is still 1.58%, the daily energy consumption is 105.46J on average for 3 nodes. Alternatively, if 6TiSCH is applied, the estimated daily energy consumption is 58.07J on average for 3 nodes. However, the influence of sleep energy consumption on the daily energy consumption of the system is obvious, as the energy consumption for sleep is about 37J per day. In the estimation, sleep energy consumption for ContikiMAC and 6TiSCH occupying approximately 35% and 57% of the total energy consumption respectively. Thus, the sleep energy consumption will affect the comparison results proportionally when comparing the sleepy mechanism with applying ContikiMAC or 6TiSCH. By subtracting the sleep energy consumption, the average daily energy consumption of the systems with optimised sleepy network design, applying 6TiSCH or ContikiMAC consumes 9.7J, 21.3J and 69.2J, respectively. Therefore, compared with same system applying 6TiSCH and ContikiMAC, sleepy network with optimised communication window design can save an average of 54.5% and 86.0% respectively.

However, it is noticed that the energy consumption for node N1/N2 (child node) is not doubled or more while the number of communication window and data payload are doubled (communication rate of every hour - 92.17% of RIOT' CoAP payload limitation). It has indicated that the higher utilisation rates of CoAP packet payloads can instead improve efficiency, and the 6LoWPAN fragmentation mechanism can help with energy utilization efficiency. Figure 4.12 shows the daily energy consumption for R_s is every 10 minutes and R_c ranging from every 10 to 60 minutes with step size of 10 minutes. Also, it has illustrated the percentage of correlated communication energy drop compared with when R_c equals to every 10 minutes. The daily communication energy consumption decreases exponentially as R_c becomes slower and gets closer to 6 times of R_s. In other words, as the amount of sensed data (e.g., 18 bytes) in a packet sent per communication gets closer to the system limit (i.e., 115 Bytes) the energy efficiency increases. This is consistent with the general theory, hence only a simple quantitative illustration is made here (see equation 4.6). For this study, this result has suggested that maximum continuous data groups (defined as G_m) – number of data groups can be attached in one data transmission should be as close as possible to 115 bytes limitation. Thus, as an example, if 18 bytes of data would be written in a single sensor reading, then R_c should be at least 6 times slower than R_s. If the R_s is every 10 minutes, then the most frequent R_c that can ensure the energy efficiency is every 60 minutes, which leading to 6 groups of data in one transmission. Also, R_c should be increased in integer multiples of G_m as formalized below. Note that an increase of 1 in the N number results in an increase of 1s in the time required for an extra CoAP transmission (2 fragments) to complete.

$$\frac{R_c}{R_s} = N * G_m, where\ N\ can\ be\ 1, 2, 3, \ldots \qquad 4.6$$

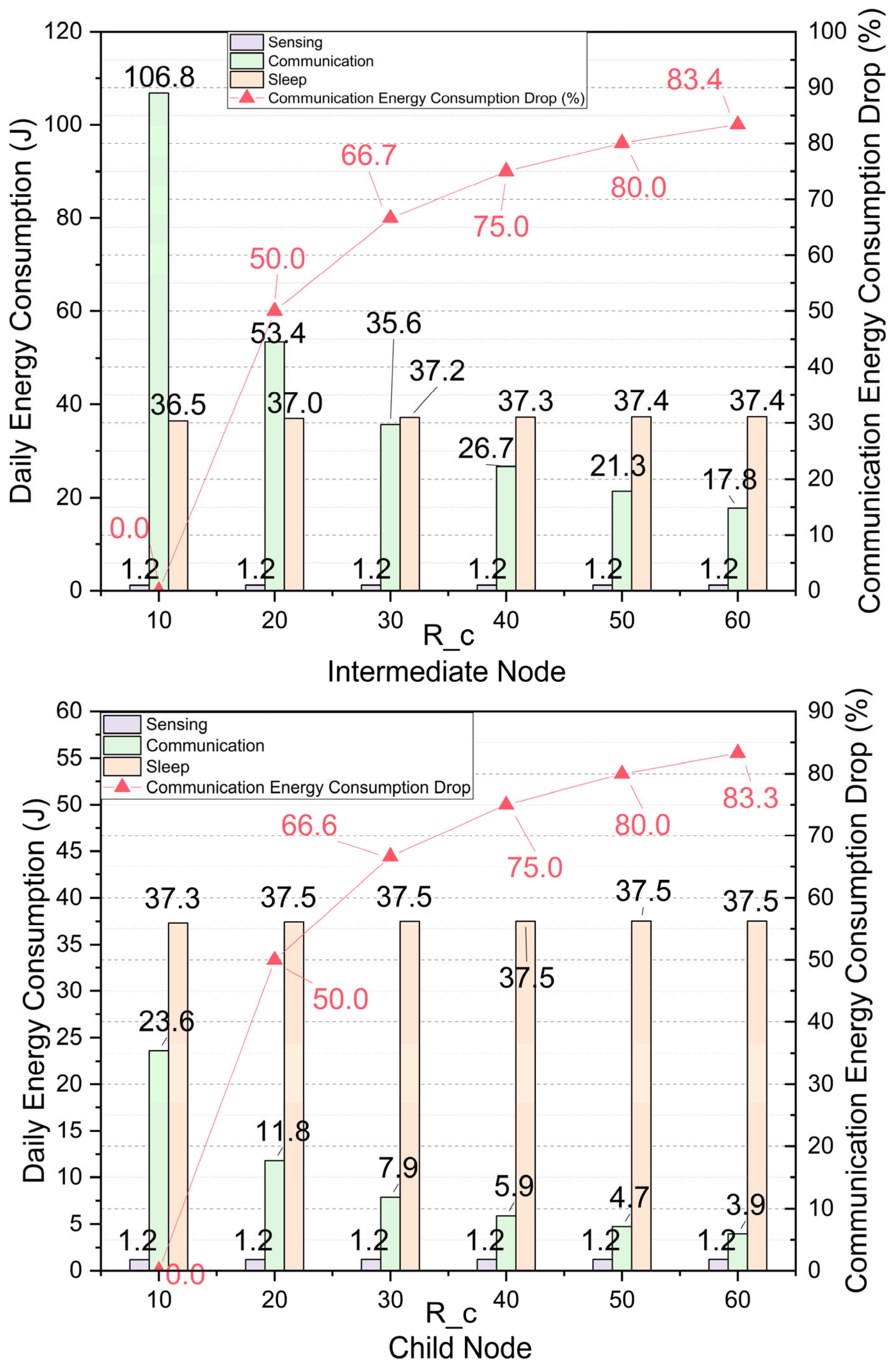


Figure 4.12. Daily energy consumption of an intermediate node (top) and a child node (bottom) under different communication rates. As the communication frequency decreases, communication energy consumption decreases in a non-linear manner. Red triangles indicate the percentage reduction in communication energy consumption relative to $R_c = 10\text{min}$ case.

Additionally, the communication energy consumption for intermediate node M1 has much higher that child node, which is potentially because of two reasons: 1) the longer communication window slot length and 2) more packets has been received and transmitted compared with single child node.

If 2600mAh 18650 Lithium-Ion rechargeable battery is used for all nodes, the estimated system lifetime is 488 days and 645 days for intermediate node and child node respectively (R_s is every 10 minutes and R_c is every 60 minutes).

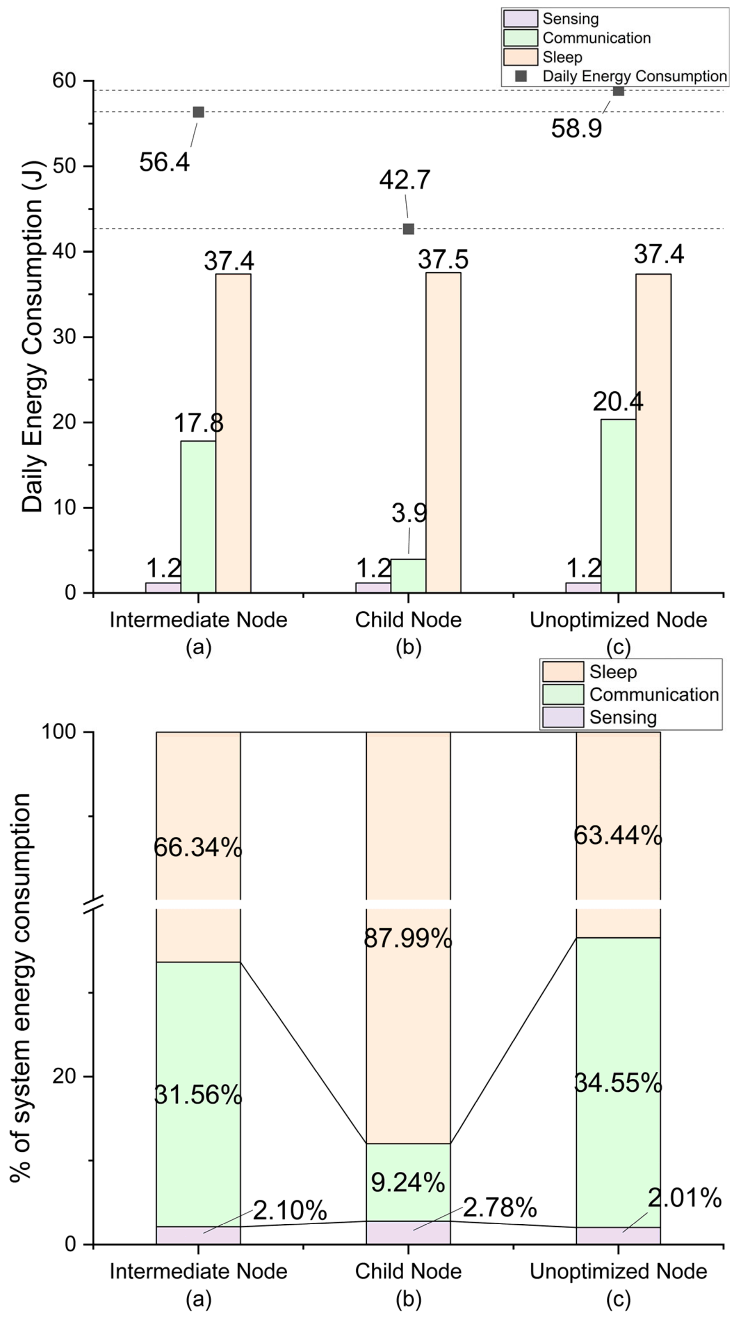


Figure 4.13. Sleepy network daily energy comparation (top) and energy stack (bottom). For both panels, from left to right, results are based on a fixed sampling rate of $R_s = 10$ minutes and communication rate of $R_c = 60$ minutes. (a) and (b) refer to intermediate node and child node respectively using the optimised sequential wake-up scheme, while (c) illustrates the node using unoptimised communication window according to Figure 3.9.

Figure 4.13 shows the energy consumption comparation between node with optimised and un optimised communication window, assuming sensing every is every 10 minutes and communication rate is every 60 minutes. Comparing the daily energy consumption of the above nodes using the optimised communication window with the unoptimised scenario in section 3.2.6, although the energy consumption of the intermediate nodes is not significantly reduced, the energy consumption of the child nodes is dropped very significantly. Generally, the energy consumption of the intermediate nodes is reduced by about 4.2% while that of the child nodes is reduced by 27.5% compared to the unoptimised scenario. Note that the communication window size for unoptimised node is 20 seconds which is 10% more than the 18 seconds window length applied in intermediate node. In a tree topology, radio packets are transmitted and received more frequently. Moreover, intermediate nodes must stay awake during the wake-up periods of their child nodes and handle approximately twice as many radio activities as child nodes. Consequently, the observed 4.2% reduction in energy consumption is still reasonable. On the other hand, the average energy consumption of the optimised three nodes is reduced by about 19.7% compared to the unoptimised scenario. Thus, the optimised communication schedule in tree topology has averagely save more than 20% energy compared with classic scheduled sleepy network in general. By applying the optimised communication schedule, sleepy network can be more “sleepy” while saving more energy, all under a relatively reliable communication condition - tested by a nearly 81% success rate and a 100% data delivery rate.

The above figure (4.13) has shown the energy stack for the optimised daily communication window of the nodes in the case of $R_c = 60$ and $R_s = 10$. However, according to equation 4.6, the change in the daily energy consumption of the nodes with increasing N is not yet specified. This also means that the change in the daily energy consumption of the nodes caused with the decreasing number of daily communication windows (when N increases) are still to be discussed. To evaluate that, Figure 4.14 has illustrated the variation of daily energy consumption of different nodes for different values of N.

In Figure 4.14, assuming that the data sampling rate is still every 10 minutes, thus, N is ranging from 1 to 24 and the number of daily communications is equal to $ROUNDDOWN\left(\frac{24}{N}\right)$. As The data sampling rate is fixed, then the amount of data to send every day is same for all nodes. When N is equals to any number of integers $\{1,2,3,4,5,7,9,13\}$, the number of communication windows in 24 hours change by exactly one compared to the adjacent values of N N within the range of 1 to 24. For example, when N is equal to 5, the number of communication window changes from 6 (N=4) to 4. Additionally, it is worth noting that when N equals to other number (ranging from 1 to 24) other than the set of integers $\{1,2,3,4,5,7,9,13\}$, a noticeable increase in energy consumption is observed compared to the previous N number. This increase is due to the transmission of additional G_m data groups during the communication. The reason for discussing this is as follows, if one observes only these few data points, it is possible to fit the influence of the communication window reduction on the daily energy consumption of the node. As a result, in this work, the average increase in daily energy consumption in joules (y) per node, caused by an additional G_m data group, is fitted as a function of the number of daily communication windows (x) - detailed in following equation 4.7 and 4.8 respectively. Also, the fitted curves describing the daily energy consumption (in joules) of intermediate and child nodes as a function of the number of daily communication windows are presented in equation 4.9 and equation 4.10, respectively. Note that the N values of the used data for fitted curve in equation 4.9 and 4.10 belong to the set {1, 2, 3, 4, 6, 8, 12, 24}.

$$\text{intermediate node:} \quad y = 0.02778 + 0.26843x \;\; where\ error \;\propto [-0.12\%, 1.104\%] \tag{4.7}$$

$$\text{child node:} \quad y = 0.00256 + 0.1027x \;\; where\ error \;\propto [-0.18\%, 0.213\%] \tag{4.8}$$

$$y = 41.60459 + 0.61855x - 0.000102132x^2 \; where\ error < \mp 0.0043\% \tag{4.9}$$

$$y = 39.86963 + 0.1216x - 0.0000337012x^2 where\ error < \mp 0.07\% \tag{4.10}$$

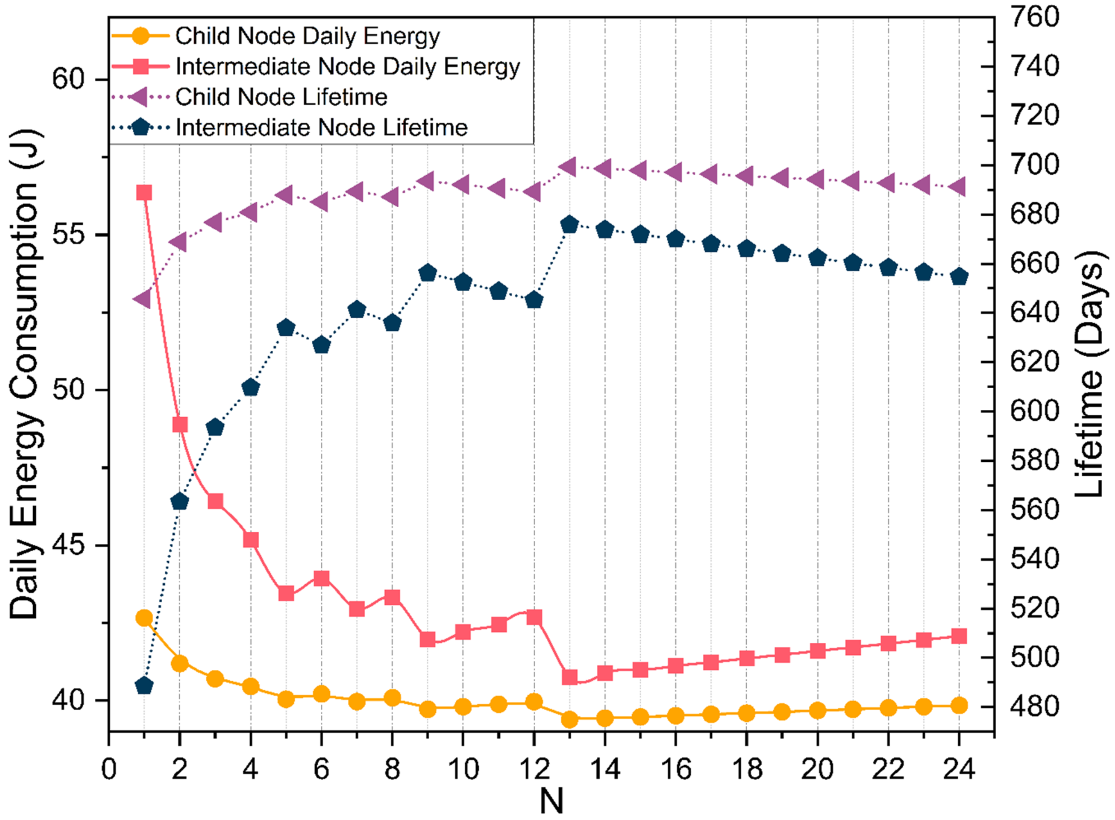


Figure 4.14. Daily energy consumption and lifetime of different nodes at different N values. Results are based on a fixed sampling rate of R_s = 10 minutes, and N controls the number of daily communication windows as described in Function 4.6.

4.7.2 Energy Consumption Impact of Transmission Success Rate

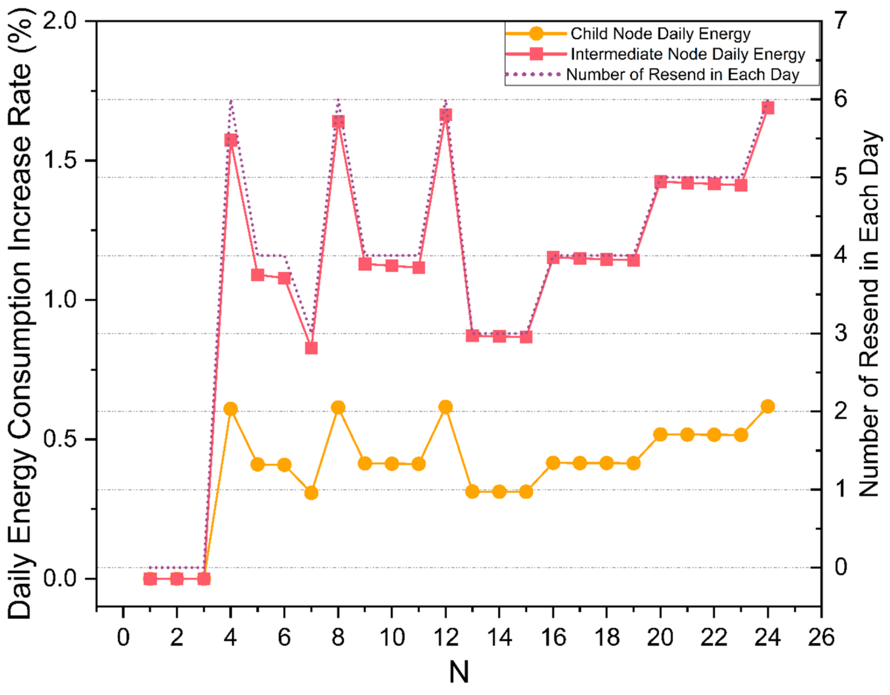


Figure 4.15. The influence of daily number of retransmissions on the percentage of energy consumption increases (assuming success rate is 81%). The left Y-axis shows the percentage increase in daily energy consumption for intermediate nodes and sub-nodes without retransmission, while the right Y-axis shows the number of daily retransmissions for different N. Simulation result shows that an extra retransmission for every four CoAP packets, with this assumption based on an 81% transmission success rate. Parameter N controls the number of daily communication windows as described in Function 4.6.

In the 3-day test previously discussed, the average transmission success rate was measured as 81%, calculated as the ratio of successfully received packets to the total number of transmission attempts. Repeated radio activity results in additional energy consumption, and although the increase in energy consumption per retransmission is relatively small as a percentage of daily energy consumption, quantifying this impact may provide a methodology and baseline for future work. To quantify and evaluate the energy impact of this retransmission overhead, the number of transmission attempts per packet was calculated based on the inverse of the success rate, i.e. an average of 1.23 transmission attempts were required. Therefore, in below simulation, it has defined that an additional retransmission is required for every 4 CoAP packets with 108 bytes data. Figure 4.15 shows the influence of daily number of retransmissions on the percentage of energy consumption increases as the N number changing from 1 to 24. It has identified that the increase rate is strongly correlated to the number of retransmissions, and the retransmission energy for this simulation used is 0.04109 J for child node and 0.11853 J for intermediate node. It also can be observed that another influence that affect the energy increases caused by retransmission is the number of communication window. Also, the smoothed portion of the curve tends to decrease because the impact of retransmission declines slightly, as the length of the communication window increases then leading to the increase in daily energy consumption. In short, with an 81% success rate, the impact of retransmission on the energy consumption of the system is less than 0.24 J (child node) and 0.71 J (intermediate node).

4.7.3 Energy Consumption Impact of Increased Routing Depth

The discussion in the previous two subsections (4.7.1 and 4.7.2) was based on the RPL tree topology with three layers. However, it has not yet analysed the energy consumption of nodes within the network following an increase in routing levels, nor the resulting energy consumption changes brought about by newly added child nodes. In general, when a new node with rank 1024 joins the network, the forwarding tasks of the upper-layer nodes increase, thereby raising energy consumption. This subsection discussed how adding two nodes with a rank value of 1024 affects the energy consumption of nodes at different positions within the topology.

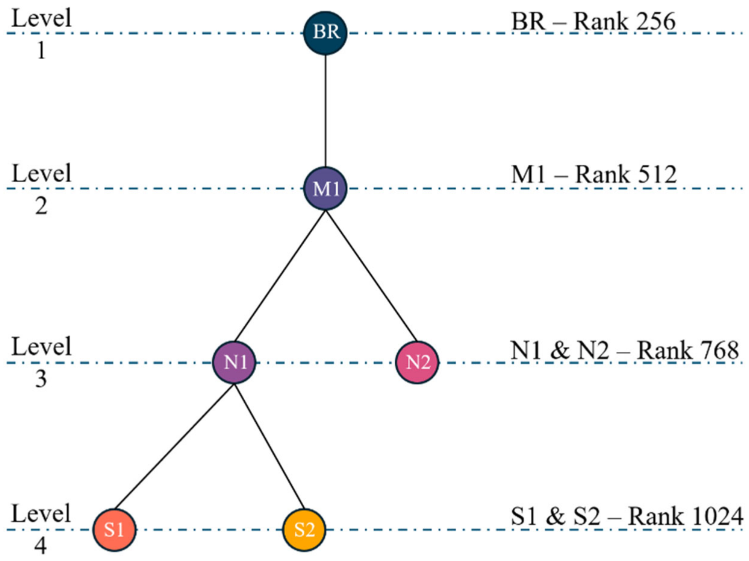


Figure 4.16. Extended tree topology layout.

From the previously used topology layout to the extended one shown in Figure 4.16, the tree topology has been extended to 6 nodes including the addition of 2 new child nodes (S1, S2 node with rank value = 1024) belonging to N1. As discussed in 5.3 the DAO sent from M1 to BR is 64 bytes, at this stage during the simulation it is assumed that the DAOs are not fragmented and therefore the length of a single DAO packet is limited to the maximum PSDU (127 bytes). It is observed that each child node causes an increase of 26 bytes in the DAO sent upwards by the parent node, so the number of new children added to the parent in the simulation is limited to 2. As a result, the S1 and S2 will extend the time cost on DAO messaging for extra 3 seconds in the following simulation. In this subsection, this work will simulate the energy consumption at all levels of nodes (except BR) based on the previous modelling and calculations. By analysing the nodes‘ energy consumption, this section will quantify the impact of the addition of the lowest level nodes (S1 and S2) on all the other levels' nodes, such as the percentage increase in energy consumption and the approximate fitted curve.

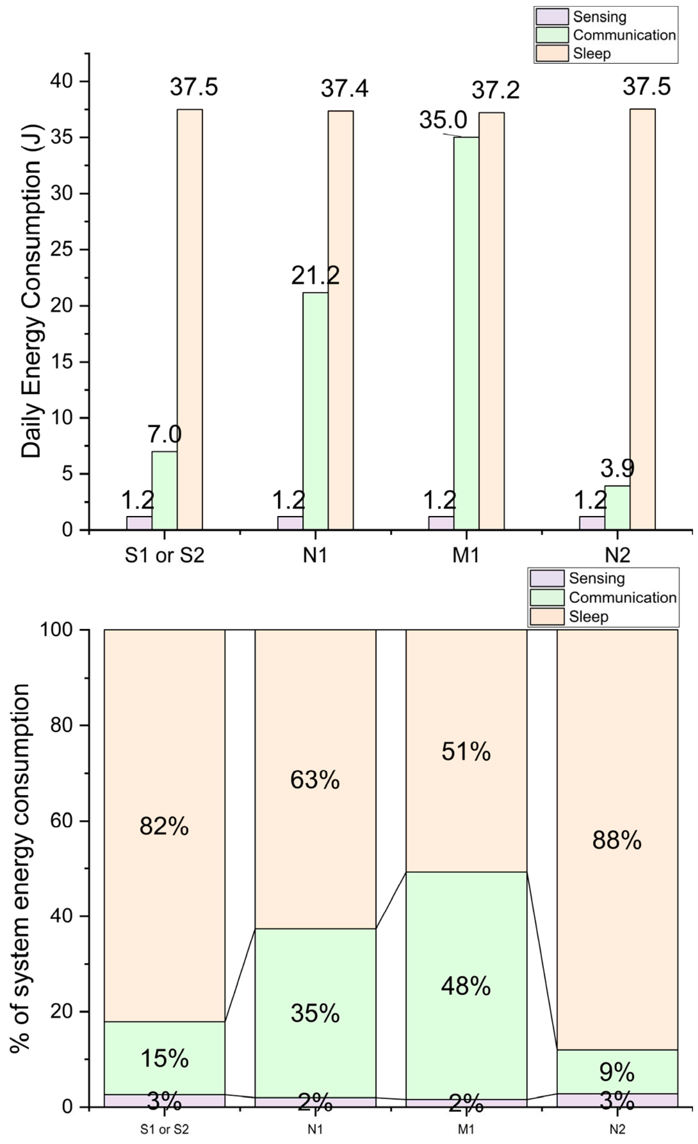


Figure 4.17. The daily energy consumption (top) and energy stack (bottom) of nodes in the extended tree topology with a data sampling rate of $R_s = 10$ minutes and a communication rate of $R_c = 60$ minutes.

Figure 4.17 shows the energy consumption stack for nodes in extended tree topology ($R_s = 10, R_c = 60$). Compared with Figure 4.13, the energy consumption of M1 has increased with 17.0J daily (30.1%), and N1 node has increased with 17.1J daily (40.0%). Also, the daily energy consumption of S1 and S1 are 45.7 J which is 3.0J increases compared with N2. If 2600mAh 18650 Lithium-Ion rechargeable battery is used, the respective lifetimes for M1, N1 and S1/S2 are 375, 461 and 603 days. Compared to N2, S1/S2 has a maximum energy consumption increases of 1.5J when N=1, and has a minimum impact of 0.04313 J when N=24. The same extend of the energy consumption increase due to the same level of communication redundancy caused by S1 and S2 to the parent node. Additionally, both N1 and M1 consumed more energy compared to when S1/S2 were not added, but N1 experienced a higher proportional increase in energy consumption. This is because of the communication tasks undertaken by forwarding are more energy expensive than before. Similarly, the daily energy consumption changes for nodes as the number of daily communication windows decreases (as N increases) are discussed next. Figure 4.18 illustrates the variation of daily energy consumption of different nodes for different values of N. Like Figure 4.14, due to the transmission of additional G_m data groups, when N equals to other number (ranging from 1 to 24) other than the set of integers {1,2,3,4,5,7,9,13}, a noticeable increase in energy consumption is observed compared to the previous N number. However, with the addition of S1 and S2, the rate of increase in energy consumption caused by their addition to the other nodes is more important to investigate next. To maintain the generality of the modelling, the following simulation results will assume a retransmission rate of 0.

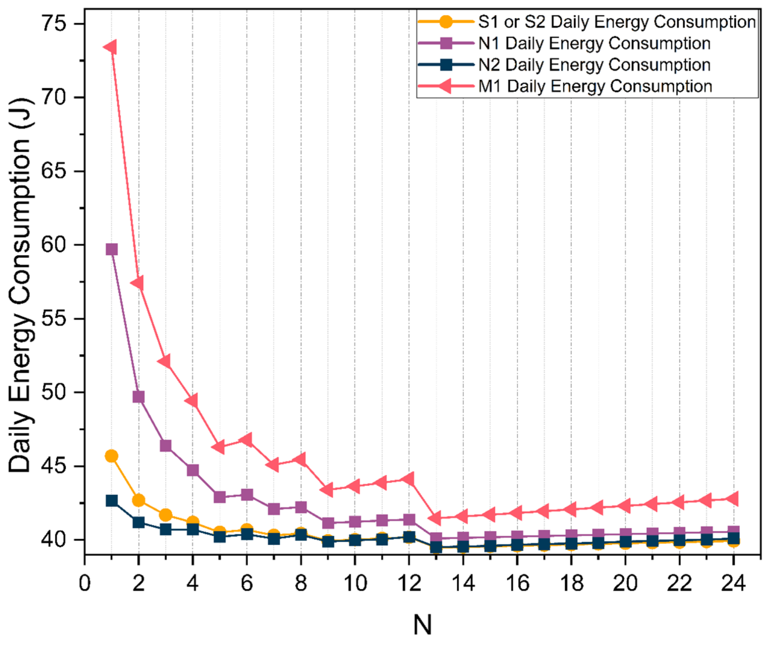


Figure 4.18. Daily energy consumption of different nodes in extended tree topology for different N values, and N controls the number of daily communication windows as described in Function 4.6.

The increasing rate of daily energy consumption of N1 and M1 for each joined sub-node ($R = 1024$) is shown in figure 4.19. Same as section 4.7.1, the ratio of energy consumption increase caused by the newly added nodes to other parent nodes for different values of N will be modelled next. In Figure 4.19, since the consistency between the energy consumption increase in (a) and (b) is high, the influence will be modelled based on the average value of the two. By using the N values of in the set {1, 2, 3, 4, 6, 8, 12, 24}, the fitted curve has been modelled as detailed in equation 4.11. In the extended tree topology, by applying equation 4.9, 4.10 and 4.11, it is possible to simulate the change in energy consumption when applying additional child nodes on different nodes.

$$y = 8.52774 * x^{-1}, error\ rate < 0.00003\% \qquad 4.11$$

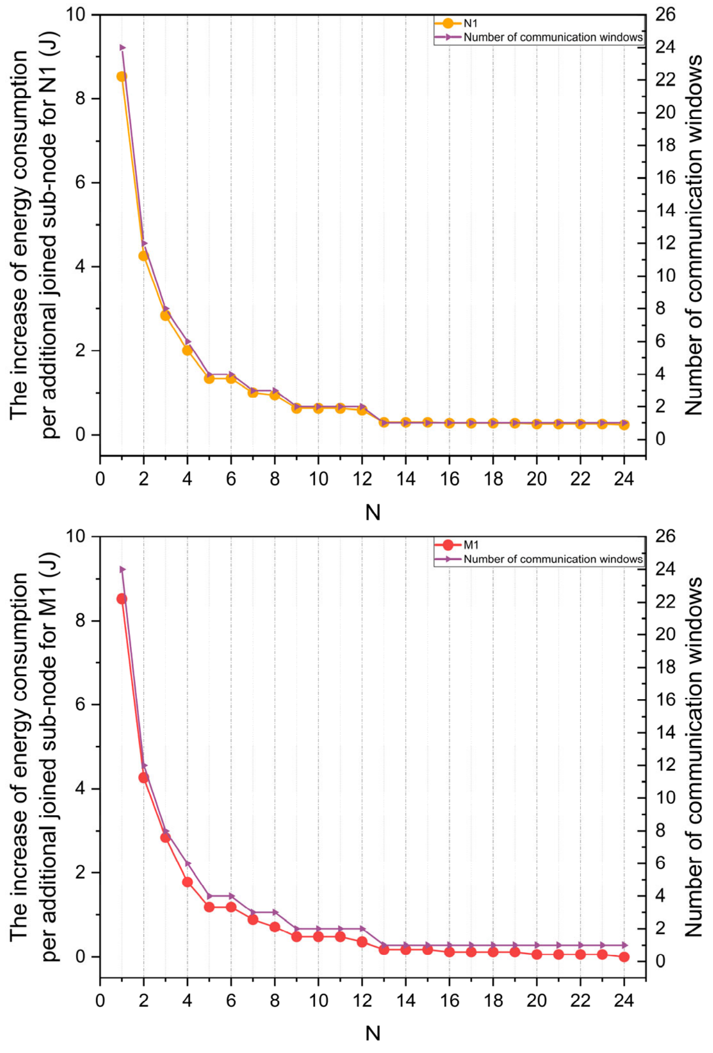


Figure 4.19. Increase in daily energy consumption of N1 (top) and M1 (bottom) per additional joined sub-node ($R = 1024$) in the extended tree topology. The left Y-axis shows the increase of energy consumption of different node per additional joined sub-node. And the right Y-axis shows the number of communication window which is relative to N numbers.

Equation 4.11 was obtained by fitting the two highly similar purple curves shown in Figure 4.19. Equation 4.11 may be used to estimate the additional energy consumption cost incurred by the newly added node with a rank value of 1024 for the upstream nodes. The significance of this estimation approach is that, as routing depth continues to increase, the same formula can be used to approximate the energy consumption variations of other nodes within the topology of a sleepy network design, under the condition of a fixed data sampling rate R_s of 10 minutes. Moreover, for upstream nodes, their data forwarding load increases proportionally with the number of lowest-level sub-nodes. Consequently, in the design of a sleepy network under a tree topology, the overall network lifetime is primarily constrained by the energy consumption of the uppermost node (such as M1).

Additionally, a method can be provided to estimate the maximum number of nodes the entire network can accommodate, subject to certain reasonable assumptions. Note that the above assumption, compared to the explanation of optimised communication window for sequential wake-up in Section 4.5, the S1 and S2 will wake up then spent extra 3 second on RPL routing information updates when the tree topology extended. To make an estimation of maximum number of nodes that can be supported in the extended tree topology, this work will maintain the time requirement of CoAP transmission within 2s (same as Figure 4.8). Also, it is assumed that the round-trip time cost per hop is 150ms, which is the average latency recorded in the ping test results of Section 3.3.2. As tested in Section 4.5, CoAP PUT requests (108 bytes) were completed within 1 second and were executed at the very beginning of the scheduled data transmission slot for CoAP. Therefore, in the estimation of maximum number of nodes in the tree topology, it was assumed that the data transmission time was still 2s with approximately 1s. Using this 1s for the transmission delay of added nodes, the rank value of the most remote node is 1792 (i.e. the seventh layer).

Also, to consider and include the extra delay for the network maintenance activities, it was assumed that the time duration required for neighbour registration and routing updates was extended with an extra 3s every time when rank value is increased with 256 starting from 512. On the other hand, as child nodes are continually added, the communication window length of the upper-level node with rank 512 will gradually increase, thereby approaching the start time of the next sensing window defined by the sampling rate. To make this estimation, the sampling rate was set with every 10 minutes to limit the increasing of communication window. Therefore, the maximum number of nodes can be deployed in the extended tree topology would theoretically be 54 (including BR). Figure 4.20 is shown a simplified layout of tree topology with 54 nodes.

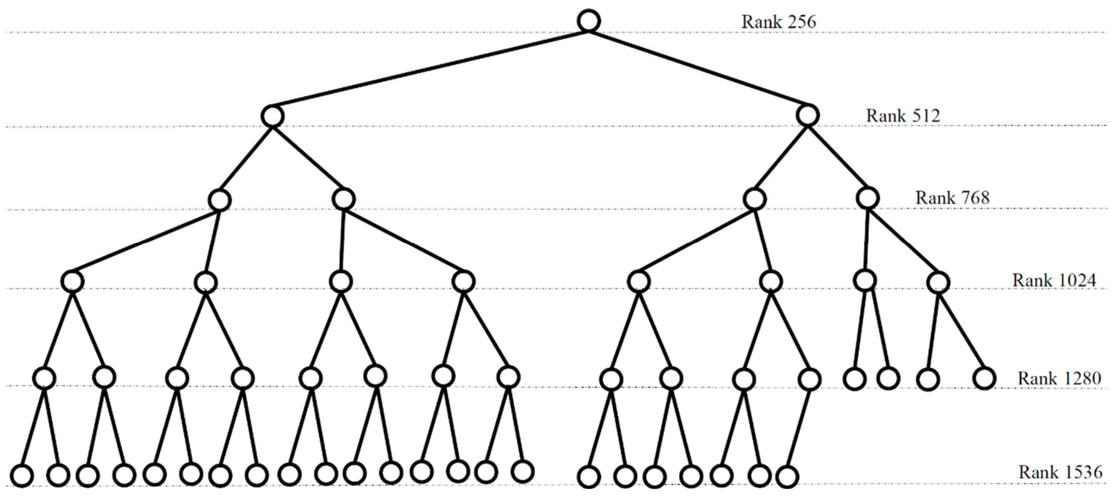


Figure 4.20. The tree topology layout with maximum number (54, including BR) of nodes.

In this design, the communication window length has been specifically configured to address for potential retransmissions, enabling multiple hop transmissions to be completed within a single wake cycle. In practice, data transmission is typically accomplished within approximately one second which is less than the allocated two-second data transmission duration.

For the extended topology as shown in Figure 4.20, the rank value for most remote node is 1536. Assuming the round-trip time cost per hop is 150ms, the average delay for the most remote node can reach 750ms. Furthermore, each data packet contains 108 bytes of data, corresponding to a throughput of 1.15 kbps. This estimation is based on the following assumptions that nodes will not miss their allocated communication windows and will remain fully active throughout their wake-up periods. Consequently, packets can be forwarded immediately without waiting for subsequent communication windows. However, missing a communication window will delay packets that failed to be sent until the next scheduled window, resulting in increased end-to-end delay. However, nodes that miss the communication window will not affect subsequent data transmission by other nodes, and it is still ensured that unsend data will be transmitted in next communication window.

4.8 Summary

The objectives of this chapter were to optimize the communication window schedule for the low power sleepy network and evaluate the influence of some important factors' change on system energy consumption. To achieve that, Section 4.3 investigated the radio activity and time required for nodes to reconstruct network when sleep duration exceeds a predetermined RPL Lifetime (3 minutes) in a tree topology network. The star topology explored in Chapter 3 was simple to build and allows all nodes to communicate point-to-point during a predefined communication window, but this topology is less scalable compared to the tree topology. Therefore, in Section 4.3, this work concentrated on the tree topology network, and investigated the network reconnection and data forwarding process following the wake-up of nodes from the sleep state. The investigated network reconnection and data forwarding process involved multiple network maintenance messages such as ND and routing information updating (e.g., NA, NS, DIS, DIO, DAO, DAO-ACK, and CoAP messages). Experimental results showed that, unlike a star topology, a tree topology involves more complex control message exchanges for node network maintenance. Especially when multiple nodes wake up and try to connect to the network simultaneously, duplicated message exchanges and radio congestion are likely to occur which results in increased latency and energy consumption. When all nodes woke up at the same time, the duplicated message exchanges and radio congestion were observed, and the network maintenance process usually took 33% more time than the sequential wake-up approach. In contrast, the sequential wake-up approach is effective in reducing the likelihood of network congestion, but it introduces additional complexity in execution. It was also observed that nodes actively sending DAO messages to update routing information is more certain and reliable than passively waiting for network response. Therefore, it is recommended that nodes adopt a sequential wake-up strategy and send DAO messages actively at the beginning of communication window to ensure the reliability and energy efficiency of the network reconnection process.

Next, based on Chapter 3, to investigate how to optimise the communication window scheduling in tree topology networks, Section 4.4 examined the maximum size of the CoAP payload and the implementation of the 6LoWPAN fragmentation mechanism in RIOT-OS. It was found that the maximum CoAP payload supported in the RIOT-OS system is 115 bytes and the effect of the 6LoWPAN fragmentation mechanism on CoAP requests is clarified. To avoid additional energy consumption and to improve the system's predictability, this section proposed a reliable design scheme for CoAP transmission and discussed the retransmission mechanism design for this work. Experimental results demonstrated that transmitting a maximum payload of 115 bytes requires a communication spacing of 2 seconds. The required time interval for network reconstruction and data transmission following the wake-up of a node under different wake-up strategies has been clearly investigated previously. Therefore, in Section 4.5, this work proposed an optimised scheme for the scheduling of nodes' communication window under different wake-up strategies (sequential and simultaneous). In addition, this section compared the effects of sequential and simultaneous wake-up strategies on the communication stability of the system. The experimental results suggested that the optimised communication window scheduling with sequential wake-up strategy in tree topology can effectively improve the system communication efficiency.

To examine the consequences of changes in key factors in the network on the energy consumption of the system, such as the communication rate, the increase in routing depth and the number of packets that need to be sent for a single communication, Section 4.6 modelled the energy consumption of the system with the optimised scheduling solution. To ensure the feasibility of the modelling, a fixed data sampling rate (10 minutes) was used. After simulating and estimating the current and duration of the nodes during the activities such as ND, RPL control messages and CoAP data transmission, this work modelled the energy consumption of each node at different positions (RPL rank value) in the topology. It was also found that the payload size utilisation will affect the energy efficiency of the system, therefore it was recommended that the data payload for a single sending activity should be as close as possible to the system limit. For this design, with communication and data sampling rates of 60 minutes and 10 minutes, respectively, the node daily energy consumption is 56.37J for intermediate node (about 488 days lifetime using 2600mAh 18650 battery), 42.67J for child node (about 645 days lifetime), and 47.23J on average for 3 nodes (about 583 days lifetime). Excluding the influence of sleep energy consumption, comparing systems with 6TiSCH or ContikiMAC applied, the sleepy network with optimised communication schedule could save an average of 54.5% and 86.0% respectively. Finally, Section 4.7 evaluated the energy consumption of tree topology networks based on the established model in Section 4.6, investigated the influence of different communication rates, packet retransmissions and increase in routing depth on the energy consumption of individual node of the system. Section 4.7 also concluded the changes in the energy consumption of typical nodes by fitting the simulation results.

Chapter 5 Conclusion and Future Work

In contrast to ESN technology, E-IoT uses IP-based protocols, thus allowing sensor networks to connect directly to the Internet and gaining from standardisation. In addition to increased flexibility, E-IoT supports bidirectional access between the user and the sensor network. The objective of this work is to propose a low-power sleepy network design for IPv6-based environmental IoT systems, focusing on how to minimise the communication window duration while maintaining communication reliability. To achieve this goal, this thesis has made the following three main contributions.

- **Research question 1:** *Can a schedule-based sleepy network reduce overall energy consumption compared to duty cycling MAC approaches?*
  - **Design and implementation of a low-power sleepy network system:** In conjunction with the discussion of protocol selections in Chapter 3, this work identifies protocol options including 6LoWPAN, RPL and CoAP. An environmental IoT system was developed using standard protocols (IEEE 802.15.4 MAC, 6LoWPAN, RPL and CoAP) and hardware platform. This chapter has also estimated the daily energy consumption of system applying sleepy network schedule (38.48J) and MAC protocols such as 6TiSCH (47.25J) and ContikiMAC (99.54J) based on the current consumption measurements. In addition, system activities such as sensing, radio transmission, neighbour discovery and routing control were measured. These measurements provide the basis for calculating daily energy consumption and optimising communication scheduling. It was discussed that, nodes have the potential to complete the necessary radio activities in 2 seconds resulting in a possibility of saving more energy through optimising the communication schedule.
- **Research question 2:** *How can energy consumption be reduced by optimising communication windows within a tree topology? and what is the resulting energy consumption model?*
  - **Optimization of Communication Windows in a Tree Topology:** This work investigates the minimum time required for network maintenance, routing information updates and sensing data transmission after a node wakes up. In addition, this work proposes an optimisation scheme for communication window scheduling under different wake-up strategies (sequential and simultaneous) in a tree topology. Assuming the ContikiMAC or 6TiSCH is applied in this system, for communication rate of every hour and data sampling rate of every 10 minutes, the daily energy consumption of sleepy network design with optimised communication schedule can save an average of 54.5% compared with applying 6TiSCH, and 86.0% compared with ContikiMAC.
  - **System Energy Consumption Modelling and Profiling in Tree Topology:** Based on the above measurements of radio activities in a star topology and the variations in a tree

topology, this work predicts the energy consumption of different radio activities in complex topologies. Based on the prediction results and modelling, Section 4.7 summarises the daily energy consumption of the system with an optimised communication scheduling scheme. Afterwards, the influence of communication rate changes, packet retransmissions and increase in routing depth on the daily energy consumption of the system is also analysed. Finally, this work proposes a method to predict the energy consumption changes of typical nodes by fitting the influence mode of energy consumption changes.

This research has established assumptions and scope limitations before reaching any conclusions, which consequently leads to certain limitations in the study. Firstly, in Section 3.2, this work provides detailed measurements of node activities within a star topology. Although the energy consumption of star topology nodes is estimated based on observed numbers and patterns of network activities, the energy consumption of each activity has been calibrated using measured values. However, due to uncertainties such as measurement noise and timing jitter, the recorded current and time duration inevitably contain errors. The estimated node energy consumption remains approximately 3.2% higher than the actual measured consumption. Although the error is relatively small, it demonstrates that the measurement and modelling methods adopted inevitably introduce uncertainty to some extent. While the energy verification within a star topology demonstrates the proposed method is feasible, measurement errors may accumulate or become more apparent when the topology is extended to a more complex tree topology.

Compared to systems in actual deployment, current work is still being progressed under certain assumptions. The objective of the current work is to minimise energy consumption while ensuring reliable communication. However, when deployed in the real world, some extreme scenarios may challenge the reliability of current designs and results, such as errors in node time synchronisation or unexpected node disconnections. On the one hand, when a node experiences time synchronisation errors, it may miss scheduled communication windows or initiate communication at unexpected times. In such circumstances, data intended to be transmitted may conflict with other nodes' transmission activities or even result in data loss. To address this challenge, this work requires designing appropriate periodic synchronisation windows in future based on deployment requirements. According to a predefined schedule, the user can easily send the real-time epoch to every node within the synchronisation window to achieve time synchronisation. Additionally, when data delivery errors at a node accumulate to a certain threshold, the node shall be forced into sleep mode and subsequently restarted within a subsequent synchronisation window through specially designed mechanism to re-establish time synchronisation.

On the other hand, when a node becomes faulty or unreachable due to energy depletion or other factors, the planned topology will be serious affected. Under this situation, data from such nodes cannot be delivered according to the originally scheduled timetable, and the transmission at other node may also be affected due to changes in the topology. Therefore, it is also necessary for this work to detect the reachability of each node in the original plan during the initial phase of the periodic synchronisation window. Therefore, it is also necessary for this work to detect the reachability of each node in the original plan during the initial phase of the periodic synchronisation window opening. Furthermore, before the resynchronisation begins, an algorithm should be designed to adaptively modify each node's schedule in response to topology changes.

5.1 Contributions

This work focuses on sleepy networks rather than MAC-based solutions as it has the potential to reduce radio switch-on frequencies and optimise overall energy consumption. Through a review of key technologies and protocols related to ESN and E-IoT development and applications, Chapter 2 provides an overview of the technologies and protocols at each layer of the network stack. To identify the requirements and properties for ESNs mentioned in first research question, Chapter 2 also analysed and categorised over 20 ESN projects that have been implemented. Moreover, inspired by the work of Borges et al. [135], this chapter also analysed the existing ESN projects by determining the characteristic parameters. The characterisation parameters are monitoring target, sensor type, operating system (OS), sampling rate, communication rate, power management, data delivery model, network protocol, and those parameters are applied here to characterise the requirements and properties of ESN system which has also been detailed in Table 2.6. The relevant cases were classified into five categories, and the diversity design of each project was characterised according to eight parameters. By comparing the characteristics of different technologies, it was summarised that the protocols should be considered in the system are IEEE 802.15.4, 6LoWPAN, RPL and CoAP.

This work designed a sleepy network system on RIOT-OS. Through the application of high precision RTC, DS3231, this work introduced a schedule-based sleep/wake-up scheduling mechanism for the above system in both star and tree topologies. By analysing the different control strategies in the sleep/wake-up mechanism in WSN, including topology control and duty cycling schemes, this work also clarified the characteristics of the different strategies. Moreover, in subsection 3.2.5, the energy consumption in the cases of 6TiSCH, ContikiMAC and sleepy network scheduling were also estimated and compared. In the estimation described in subsection 3.2.5, the daily energy consumption of the sleepy network approach could save 61.4% energy compared to ContikiMAC and 18.6% compared to 6TiSCH under the same system design. In addition, this work experimentally measured and recorded the average current and duration of each type of network activity in the system, such as neighbour discovery and routing maintenance (RPL related radios). By utilizing the measured power consumption and the deployed sleepy system schedule, Chapter 3 has also concluded the system daily energy consumption and lifetime, as well as the effect that data sampling rate/communication rate can bring to the system. For communication once a day and data sampling every 10 minutes, the daily energy consumption is about 39.8 J, and estimated lifetime is approximately 700 days (estimated using 2600mAh 18650 Lithium-Ion rechargeable battery). Based on this, the chapter also presents a reliable assumption for reducing the length of the communication window, i.e., the shortest time required for communication is expected based on Equation 3.1. The system energy consumption is predicted by combining the characteristics of various types of network activities with actual measurements. The prediction results show that the optimised communication window can bring about a maximum reduction in energy consumption of more than 30% (communication rate of every hour). So far, this work has discussed the architecture of the sleepy network and analysed the real energy consumption. Through comparison, it is found that the schedule-based wake/sleep mechanism is more energy efficient than the duty cycling MAC mechanisms (6TiSCH and ContikiMAC) in situations where frequent data sending is not required (e.g., communication every hour).

Nodes do not generate any radio activity while sleeping, so it is necessary for the node to complete some radio activities (e.g. neighbour registration, routing information maintenance, etc.) before it sending data upwards [190]. In the more complex tree topology, to investigate the research question 2, this work further investigated the influence of different sleep durations/wake-up frequencies on the expected system radio activities, especially on neighbour discovery and routing update activities. Through observing and analysing the radio activity patterns of nodes waking up after a sleep duration greater than RPL lifetime (3 mins), this work proposed and tested the optimised scheme for the scheduling of nodes' communication window with sequential/simultaneous wake-up strategy in section 4.5. Note that the system is operated based on epoch time, and the RTC time reading supported by RIOT-OS has a step size of 1 second.

Firstly, in the tree topology system shown in Figure. 4.5, the optimised communication scheduling under the sequential wake-up strategy is shown in Figure 4.8. Briefly, the minimum communication time required by an intermediate node after waking up is approximately 6s for network maintenance, 2s for completing its scheduled CoAP transmission, and additional 5s reserved for communication with each child node. For child nodes operating under the optimised communication schedule with a sequential wake-up strategy, the minimum time required is approximately 3s to complete network maintenance after waking up and 2s to complete the scheduled CoAP communication. When nodes operating under the simultaneous wake-up strategy as illustrated in Figure 4.9, the time requirements for intermediate nodes remain consistent with the sequential wake-up strategy. However, for two child nodes, the minimum time required for each node to wake up and complete network maintenance and scheduled CoAP communication is approximately 5s and 5s, respectively. Further test revealed that the system performance under the synchronised wake-up strategy is inadequate compared to the sequential wake-up strategy. It was found that the system under the simultaneous wake-up strategy, although the scheduling scheme was functional, achieved a delivery rate of less than 70% in a 24-hour continuous test (even though the communication window of the child nodes was increased to 20s). In contrast, the system applying sequential wake-up achieved 100% delivery rate and 95.5% success rate during test. In short, this work investigated and proposed the sleepy network design with optimised communication window schedule, and identified that system applying sequential wake-up strategy has better reliability compared with simultaneously wake-up.

Following on that, based on the actual measurements of radio activity in Chapter 3 and the more complicated radio activity patterns in the tree topology, section 4.6 modelled the energy consumption of the system using the optimised scheme for communication window with sequential/simultaneous wake-up strategy. On average, the energy consumption of the optimised three nodes was reduced by about 19.7% compared to the unoptimised scenario ($R_s = 10\ mins, R_c = 60\ mins$). In subsection 4.7.1, This work also compared the optimised sleepy network design with system assuming ContikiMAC or 6TiSCH is applied. Excluding the influence of sleep energy consumption, the daily energy consumption of the sleepy network with optimised communication schedule saved 86.0% compared to ContikiMAC and 54.5% compared to 6TiSCH. It is obviously that the optimised sleepy network design is more energy efficient than the system if applying ContikiMAC or 6TiSCH. 6TiSCH is a synchronisation scheme with high reliability and real-time capabilities. Compared to the sleepy networks, the advantage of using ContikiMAC and 6TiSCH is that user will be able to get data more frequently or even in real-time which can provide better flexibility. Nodes will remain synchronised for the running time after schedule is established if applying 6TiSCH, and ContikiMAC is an asynchronous protocol. As opposed to the above-mentioned application of the MAC protocol, sleepy network also needs to consider resynchronise the RTC timer to compensate the time drift. However, the problem is that network maintenance will be continuous which costs more energy, and sleepy network can potentially reduce the energy cost of network maintenance during the time period when nodes are not required to send data.

Subsequently, during the modelling process, this work also summarised the possible time-consumption variations of the various types of radio activities depending on the node's position in the tree topology. Based on the results of the modelling, Section 4.7 also explored the impact of changes in several system indexes on the energy consumption of individual nodes of the system (including communication rates, packet retransmission and increase in routing depth), and modelled the energy consumption variations of nodes in different position by fitting the simulation results. In section 4.7, it found that the ratio between the data sampling rate and the communication rate should be aligned with the system's payload limits (detailed in equation 4.6). Also, with a fixed data sampling rate (every ten minutes), this section also estimated the relationship between the daily energy consumption of the intermediate node/sub-node and the number of daily communication windows by applying curve fitting method. Under the same topology and with a success rate of 81%, the retransmission has an impact of less than 0.24 J and 0.71 J on the daily energy consumption of the child and intermediate node, respectively. Finally, when the topology is changed (check Figure 4.16), the original intermediate node and child node's energy consumption increases by 30.1% and 40.0%, respectively.

5.2 Future Work

As a summary of the work done in the doctoral research, there are still some aspects of this work that need to be further improved. For the areas where further work can be carried out, at the system level, one important direction is to further increase the minimum step size of the optimised communication window scheme. Currently, the optimised communication window schedule has a minimum changing step size of seconds due the limitation of RTC time reading in RIOT-OS. It is potential to consider and design a more precise optimisation scheme in millisecond in the future. Therefore, it is necessary to further optimise the design of the communication window schedule. For child nodes, an algorithm can be designed to recognise the reception of both DAO-ACK and CoAP-ACK, which can further reduce the unnecessary radio idle time. For intermediate nodes, alternative timer control can be considered to more accurately reserve the data transmission intervals required by the child nodes. Optimisation schemes of communication windows with higher step size will further reduce the system energy consumption. Meanwhile, it is important to note that the data rate will decrease when the carrier frequency becomes lower which requires a longer transmission duration. Therefore, if the step size of the optimisation scheme can be further improved, future work can propose a more flexible scheme that can be widely applied to other frequency bands.

Future work could also introduce other energy-saving approaches, such as data reduction mechanisms. Concise Binary Object Representation (CBOR) [207] is a binary serialization format based on JavaScript Object Notation (JSON) which allows encoding of structured data to reduce the amount of data transferred and system energy consumption. In addition, Dias et al. has categorised and summarised the predicational-based data reduction method in WSN which can be applied in both nodes and cluster heads [19]. The above data de-redundancy methods can reduce the system's energy consumption, but they also introduce new complexities, especially in terms of computation.

Therefore, an important future research direction is to consider analysing the impact that the introduction of such methods may have on similar systems in the thesis.

In addition, the selection of sensor type may also have an impact on the design of the communication schedule. This system design has not been implemented, so in the future, it is necessary to plan a field test to deploy the optimised sleepy network design. In real-world deployments, the selection of sensors still needs to be determined by combining the users' requirements with the specific characteristics of the deployment environment. Chapter 3 found that nodes can be equipped with a single sensor or multiple sensors at the same time, and the size of a single data sample can be a dozen bytes or more (images). In real deployment, the equipment of various sensor should affect the optimised communication rate selection and daily energy consumption, which can be used to valid the accuracy of the system modelling and energy profiling results in this thesis. Also, comparing the predictions with the observed system lifetime after long-term deployment is a potential topic. Chapter 5 of this thesis provided a methodology to evaluate the daily energy consumption of the corresponding system, as well as a percentage estimation model that determines the impact of different factors (e.g. data sampling/communication rate, retransmission rate and topology changes) on the energy consumption of a typical node. In the future, predicting the energy consumption of nodes at different locations in the topology based on possible deployments, and predicting the effect of the real single-sampled data size on the energy consumption of the system using the analytical process and formulas in Chapter 5 is also a valuable research topic.

Outside the system level, current systems demonstrate that E-IoT can provide bi-directional communication capabilities and good interoperability. In the future, the system could also allow the user to dynamically adjust the parameters of the schedule, such as the data sampling and communication rate, according to actual user requirements. The current design is there to push data upwards along the tree topology, in the future, it will also be necessary to consider introducing a sensible design to accommodate possible downwards data flows. Depending on the current design, future works could also consider extending the communication window or reserving specific time slots to receive messages (such as changing data sampling rate and resynchronise RTC timer) from users without redeploying nodes. Users can easily maintain synchronisation by iterating through all nodes and sending the real-time clock to them using the CoAP PUT.

Appendix A Publications

Publications

A.1 Testing Low Power IP-Based Sensor Networks in a Forest Environment

Martinez, K.; Lu, J.; Kendall, S.; Hart, J. K.

In previous research we showed that a fully IP (standards compliant) sensor network could be used over a large mountainous study area (mountainsensing.org). One of the advantages of these low power 6LoWPAN networks is their automatic mesh networking, which not only adapts to faults but allows systems to communicate over longer distances than high power radio. Another advantage is the web-like CoAP protocol we used which allows data to be gathered using simple Python code. Advances in protocol standardisation and the increased availability of sub-GHz radios being included in ICs for IoT and other applications means there is a greater choice of hardware.

This paper describes our experiments in a forest environment, where there are signal issues due to trees and foliage but also radio shadows due to the terrain. They are thus representative of many earth science sensing scenarios. We used a low cost system-on-chip microcontroller which includes a low power 868MHz radio (Microchip SAMR 30) and the open source RIOT operating system (www.riot-os.org) to provide the network functionality. The studies were carried out in The New Forest in the UK.

A.2 Understanding Energy Consumption in Real-World E-IoT systems with CoAP, RPL, and 6LoWPAN

Jiawei Lu[0009-0003-4326-6722], Kirk Martinez[0000-0003-3859-5700] and Alex Weddell[0000-0002-6763-5460]

University of Southampton, University Road, Southampton, UK
Jiawei.Lu@soton.ac.uk, {km,asw}@ecs.soton.ac.uk

Abstract. This paper investigates the energy consumption of Environmental Internet of Things (E-IoT) systems using CoAP, RPL, and 6LoWPAN. We focus on Sleepy Networks, a method to save energy in battery-powered nodes where nodes spend most of their time sleeping and have regular communications windows. Our experiments used the Microchip SAM R30 Xplained Pro platform and measured the current and time duration of system activities. This included reading sensors, storing data, and the radio activities controlled by the network protocols. After measuring the power consumption of the various states in detail, we explored the effect of network maintenance on scheduling and energy consumption. The results indicate that reducing sleep energy consumption is fundamentally important. While considering the user requirements and the volume of produced data, the results show that optimizing the communication window can reduce energy consumption further. Therefore, we suggest to further reduce the energy consumption of the system, it is necessary to take a balance between network reliability and energy consumption.

Keywords: E-IoT, Energy Consumption, RPL, CoAP, 6LoWPAN.

1 Introduction

Wireless sensor networks (WSN) can be used to monitor remote/hazardous environments such as unpopulated or geohazard-prone areas where the resources are limited, and system maintenance is challenging. Martinez et al. [1] proposed the notion of Environmental Sensor Networks (ESNs) which are WSNs specifically designed to carry out continuous sensing of the natural environment. The development of low-power protocols standardised for sensor networks and IP-based standards [2], can enable full internet connectivity and enhance the interoperability of ESNs, enabling them to become more open and heterogeneous networks. The notion of Environmental IoT (E-IoT) [3] emerges to allow bidirectional access between users and sensor nodes which emphasises device connectivity across vendors and integrates with the internet. The system design should consider the collaboration of different protocols and technologies to withstand various real-world environmental challenges. Power consumption is one of the most critical elements that will decide the system's lifetime and reliability [3]. Thus, Low-Power protocols and efficient power management of the system are critical when developing remote E-IoT systems.

From the application layer to the physical layer, various low-power protocols can be applied to the low-power network stack, such as the Constrained Application Protocol (CoAP), Low-power Wireless Personal Area Networks based on IPv6 (6LoWPAN), and the IPv6 Routing Protocol for Low-Power and Lossy-Networks (RPL). We concentrate on an IP-based E-IoT system so non-IP-based protocols such as Long Range Wide Area Network (LoraWAN) will not be considered here. 6LoWPAN was designed for the requirements of low-power networks and brings uniformity to IoT networks. CoAP [4] is an application protocol designed for these resource-limited systems [4]. CoAP has low complexity as it uses UDP, not connection-based TCP protocol. While UDP lacks built-in mechanisms for error recovery or retransmission of lost packets CoAP provides these features. To enhance packet reliability, a CoAP transaction can optionally wait for an acknowledgement (ACK) from the receiver and retransmit the packet after a timeout. The inclusion of a 2-byte Message ID in the CoAP header allows the sender to identify ACKs to avoid duplicate transmissions. RPL can effectively extend the range of the network through packet forwarding. Previous studies have focused on improving the efficiency, applicability, and operation of 6LoWPAN, RPL and CoAP by simulation [5], and have concluded that limited power is the main challenge of WSNs based on these protocols [6]. We are interested here in the real-world energy efficiency of E-IoT systems using these low-power protocols.

Improving system energy efficiency to extend system lifetime is a common challenge in designing a real-world E-IoT system. It is critical for remote and hazardous environment monitoring systems as it fundamentally decides the feasibility and maintenance issues of deployment. IEEE 802.15.4 MAC is the de facto underlying MAC specification for devices to run in low-power mode to save power [7]. The mechanism of IEEE 802.15.4 MAC has defined to avoid collisions is Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA). MAC protocols such as ContikiMAC [8] which build on top of IEEE 802.15.4 are designed to optimise the power consumption further. ContikiMAC uses a Radio Duty Cycle (RDC) and CSMA [8]. It applies a duty cycle mechanism (8 Hz by default), waking up nodes for transmission after two successive Clear Channel Assessments (CCAs) are identified [8]. The Mountain Sensing [9] project used ContikiMAC as it saves power while making the network appear to be always alive. The Sleepy network implemented in this study schedules the nodes to intermittently communicate and run in a low-power sleep state after completing planned tasks. Nodes are designed to wake up periodically to execute required tasks such as sensing and data transmission on a regular schedule. We focus on a sleepy network rather than a MAC-based solution, as it has the potential to reduce unnecessary transitions and potentially optimise overall energy consumption.

In previous research, we explored the practical implementation of energy-efficient E-IoT systems in some real-world scenarios. The Mountain Sensing project [9] implemented an E-IoT system using ContikiMAC which proved that 868 MHz 6LoWPAN with Contiki can support the ESNs more efficiently compared with 2.4 GHz-based ESNs. It also showed that the multi-hop 6LoWPAN network could cover 3.5 km over a mountainous environment. A system based on a low-cost system-on-chip (ATSAMR30) was tested in a forest environment, which provides an in-forest range of typically 150-200m depending on the environment [10]. This SoC has an 868 MHz low-power radio (13mW) and ran the RIOT operation system (RIOT-OS) [11].

In this experiment, our objective is to analyse the energy consumption of a sleepy E-IoT system using CoAP, RPL, and 6LoWPAN based on RIOT-OS. This aims to explore the impact of different design choices on the system's lifetime. The system features can be split into three aspects: (1) user-configurable system, (2) energy-efficient sleep schedule, and (3) adaptive energy-efficient RPL configuration. This E-IoT system should be able to allow the user to configure and change the features of the system, such as communication and sensing frequency, depending on their requirements. Meanwhile, the system is adaptive to dynamic changes which means that the system should be able to intelligently modify the configurations such as schedule and routing parameters, to adapt to the configuration change. When designing such an efficient sleepy network, the fundamental task is to ensure the system is energy efficient while providing the users with data when they require it. With efficient collaboration between system software and hardware, all the components should be able to reliably work with the sleepy network schedule. The IoT network stack requires a certain level of system activity for maintenance, which could potentially clash with our use of minimal communications windows. The analysis of the system's current consumption for different activities during sleep, sensing, and communication state will be introduced in Section 3.

This paper is an experimental investigation into the energy consumption of the sleepy E-IoT network. By measuring and analysing current consumption under various scenarios, we not only contribute experimental data to the field but also reveal potential areas for optimization. This research is particularly relevant in the context of advancing the understanding of low-power IoT device behaviour, offering valuable benchmarks for future studies. Moreover, our approach to system design, including the utilization of low-power protocols and sleep scheduling, demonstrates practical experience in improving energy efficiencies.

2 System Architecture and Experiment Setup

2.1 Hardware and Software Configuration

In this experiment, Microchip SAM R30 Xplained Pro (SAMR30 Xpro) [12] was selected as this hardware platform. It has an ultra-low power ATSAMR30G18A

SoC which supports sub-GHz (868 MHz band) radio (AT86RF212B integrated transceiver). This MCU only has 256 kB flash, which is used for code, so an I/O1 Xplained Pro SD card extension board was used to provide sensor data storage/buffering. A DS18B20 Digital Temperature Sensor was used with its default 12-bit resolution supported by RIOT-OS (the conversion time is constrained by the driver leading to typically 750ms sensing time). The embedded crystal of SAMR30 Xpro provides the functional RTC with an accuracy of ±20 ppm. In our experiment, the accuracy of RTC can affect the synchronisation and schedule timekeeping. An accurate RTC, DS3231, was used to ensure the accuracy of synchronisation of wake-up and scheduling (±2 ppm from 0°C to +40°C, and (±3.5 ppm from -40°C to +85°C). By applying the DS3231, assuming that the working temperature is between 0°C to +40°C, the maximum time drift per week is about 1.2s. Compared with using the SAMR30 Xpro's built-in oscillator (about 10% as accurate as the DS3231from 0°C to +40°C), using DS3231 can enhance the synchronisation reliability, as the DS3231 provides enhanced accuracy across a wide temperature. Wireshark and a RIOT-OS Sniffer [13] were used to capture the wireless packet exchanges.

2.2 Network protocol stack

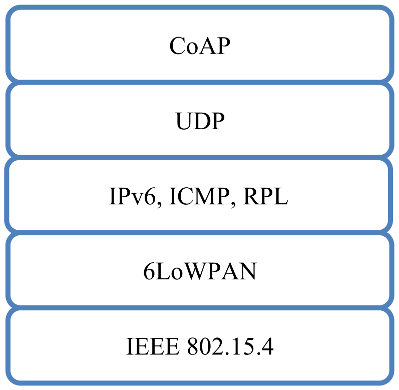


Fig. 1. Network protocol stack.

6LoWPAN, RPL, and CoAP are used in the system. These low-power protocols allow devices to form a star/tree topology, reduce energy consumption, and ensure robust communications. The RPL standard [14] extends the range of the network and supports scalability, making it well-suited for constrained environment deployments. CoAP provides a simple RESTful request/response interaction model and supports features to make it possible for devices to reply to data requests as well as respond to configuration changes. This IoT network protocol stack is shown in Fig. 1.

RIOT-OS default network stack, Generic (GNRC) network stack, fully supports IPv6 with 6LoWPAN and implements RPL [15]. The Destination-Oriented Directed Acyclic Graph (DODAG) is the fundamental structure of RPL, the parent node and child node have different Rank values within a single DODAG. RPL allows the node to communicate with surrounding nodes by exchanging ICMPv6 control messages: DODAG Advertisement Object (DAO) DODAG Information Objective (DIO) DODAG Information Solicitation (DIS) are used to select a root and join a DODAG [16]. Four RPL instances are: *(a)* Single DODAG with one root; *(b)* Multiple DODAG; *(c)* Signal DODAG that uses a virtual root on the backbone network and *(d)* a combination of *(a) (b)* and *(c)* dependent on the scenario [17]. Fig. 2 below shows an example of an RPL star topology DODAG. Our experiment currently only deploys nodes in a star topology with one border router (BR) as the parent.

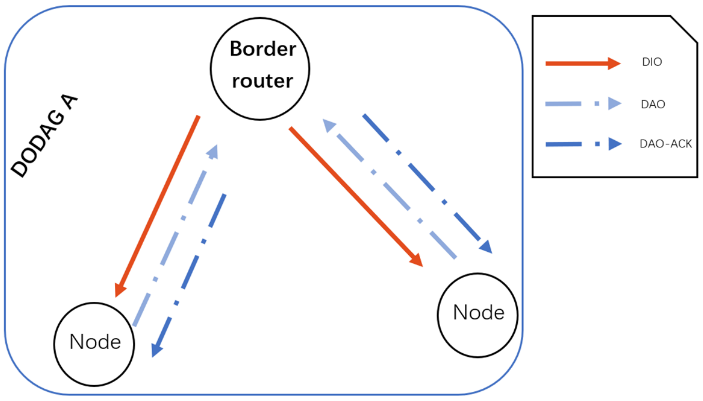


Fig. 2. RPL node star topology-DIO & DAO exchange and DODAG.

The exchange of the RPL control message is periodic and controlled by the RPL "Trickle Timer" which has four parameters:

(1) I_{min}: the minimal interval of DIO ($I_{min} = 3$ in this experiment which equals 8ms).

(2) I_{max}: the maximum time interval between DIOs ($I_{max} = 20$ in this experiment, approximately 2.33 h).

(3) K: the constant determines doubling the interval between transmissions ($K = 10$ in this experiment).

(4) DIO timer: Interval for DIO sending is 0, Imin, ..., Imax.

2.3 Experiment Setup

In our experiment shown in Fig. 3, GNRC network stack, 6LoWPAN, RPL, and CoAP are all provided by RIOT-OS. RTC3231, Atmel I/O1 Xplained Pro, and a DS18 sensor were connected to the SAM R30 Xplained Pro board. Radio packets were captured by a RIOT-OS sniffer to help correlate activity timings with current consumption changes, simultaneously measured using a Nordic Semiconductor Power Profiler Kit II and Agilent 34411A multimeters. The initial phase involved investigating the current consumption of the fundamental networking activities controlled by GNRC network stack. Subsequently, the experiment enabled RPL to explore the characteristics and current consumption of the various RPL control messages.

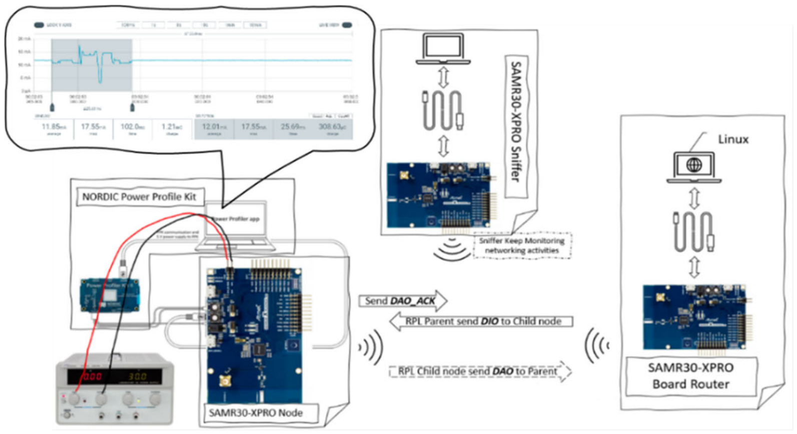


Fig. 3. Example of system current measurement experiment.

Two scenarios were considered: (1) a predetermined sensing rate with one daily communication window, and (2) a predetermined sensing rate with regular communication windows distributed daily. The Border Router (BR) was powered by an always-on Linux PC. Thus, the measurement focused on the energy consumption analysis of the individual nodes. In this experiment, the Linux PC sent CoAP GET requests to the nodes to retrieve their data. These are routed through the BR which converts IPv6 packets to and from 6loWPAN. Several activities were measured in terms of the current and time duration. Table 1 provides an overview of the expected activities for the sensor node.

Energy consumption was measured across three states: Sleep, Sensing & Communication. Specifically, during the sensing and data storage state, the node takes readings from the DS18B20 and stores the data on the SD card. During the communication state, the focus of network activities is ICMPv6 control messages (Router/Neighbour Solicitation/Advertisement), 6LoWPAN packets, DIO, and DAO/DAO_ACK managed by the GNRC network and RPL routing protocol. The aim is to identify the current consumption, time duration, and hence energy consumption of these network activities.

Table 1. Node's activities overview.

State	Activity	Description
Sensing	Set system to sensing state	Node wake-up from sleep mode
	Read sensor and store data	DS18 temperature data read and stored on the IO1 Xplained Pro SD card
	Sensing state	Node stays in IDLE mode and radio is off
	Set system to sleep state	Node enters sleep
Communication	Turn off the radio	Set transceiver to OFF
	6LoWPAN activity and Router Advertisement	Node sends 6LoWPAN and Router Advertisement packets to BR
	Neighbour Solicitation	Node sends Neighbour Solicitation to determine Neighbours
	Neighbour Advertisement	In response of receiving Neighbour Solicitation
	DIO	DIO control message activities of the node
	DAO-Tx and DAO_ACK-Rx	Node sends DAO message to BR and receive DAO_ACK from BR
	CoAP Response	Receive CoAP request and respond
	Communication	Node stays in IDLE mode with radio on
	Set system to sleep state	Node turns radio off and enters sleep state

3 Results Analysis and Discussion

3.1 Sleepy Network SoC Activity Current Measurement

Network activities for maintaining GNRC network and RPL network are managed by the border router. The periodicity of these activities which construct GNRC and RPL network was observed every 60s. Fig. 4 is a sleepy network system SoC current measurement of 60s communication activities. It has been tested that periodic sleep and wake-up schedules can work well with the 60s communication window, and the SoC current consumption of the sleep state is sensible low, which also proves that the fundamental system can support an available sleepy E-IoT system. To begin with, a simple sleep and wake-up schedule was used with a periodic 1-minute communication and sleep state to capture the measurements. In summary, (1) the SoC sleep state current drops to a low level of 2.56 μA, (2) the average current of the RPL control message is over 11 mA, (3) the average current of the board in the IDLE state reaches 11.8 mA. Activities such as Router/Neighbour Solicitation/Acknowledgement DIO, DAO, DAO-ACK and response to CoAP requests were then identified. Table 2 below summarises the average SoC current for different activities.

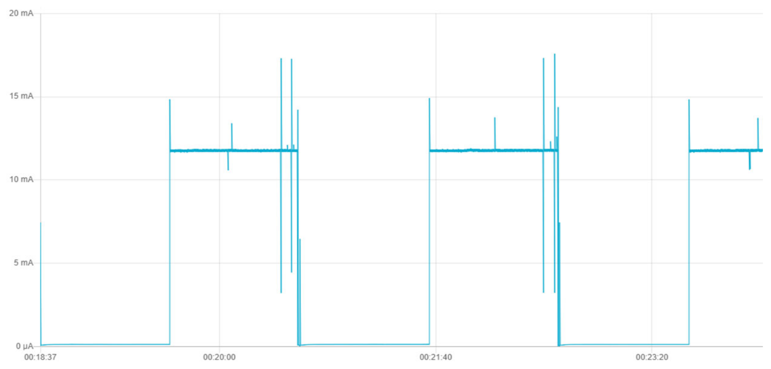


Fig. 4. SoC current measurement at 3.3V of all the activities of the sleepy network.

Theoretically, more network maintenance and control messages can be expected with the increment of communication windows which increases the system's energy consumption. To comprehensively understand the impact of energy consumption with an increased number of communication windows, we measured the current and time duration of the sleep and wake-up activities in a practical system.

Table 2. Average SoC current consumption measurements at 3.3V for different activities.

Activity Information	Average Current
DIO	11.2 mA
DAO-Tx and DAO_ACK-Rx	12.5 mA
Neighbour Solicitation	11.4 mA
Neighbour Advertisement	11.4 mA
CoAP response	12.1 mA
Sleep State – Standby Mode	2.56 µA

3.2 Sleep E-IoT System Schedule Design

A general sleep E-IoT system schedule example is shown in Fig. 5. As illustrated, devices are scheduled to switch between the sensing state and sleep state. Also, the communication window will appear at the beginning of one period, followed by a sensing window, if the communication frequency is once a day. If the required communication frequency is more than once per day, the communication window will be evenly distributed in one day, followed by a sensing state. For example, if the communication rate configured is 4 times daily, the communication window will occur at the beginning of every 6 hours. Similarly, the communication window will open at the beginning of every hour if the communication is configured to happen every hour, then turn off the radio to go to the sensing state subsequently. As Fig. 5 shows above, the system will start to sleep after all the activities are finished. This design can avoid unnecessary energy consumption by switching the device on from the sleep state. System configurations such as the sensing rate and communication rate are changeable by sending a CoAP PUT to the nodes from the Linux PC. To investigate the energy consumption, all activities' current consumption and time duration were measured and listed in Table 3. These results can be used to calculate the system's daily energy consumption. We note that the sleep state of this development board has an increased sleep current consumption compared with the findings illustrated in Table 2 due to the peripherals such as the SD card extension board.

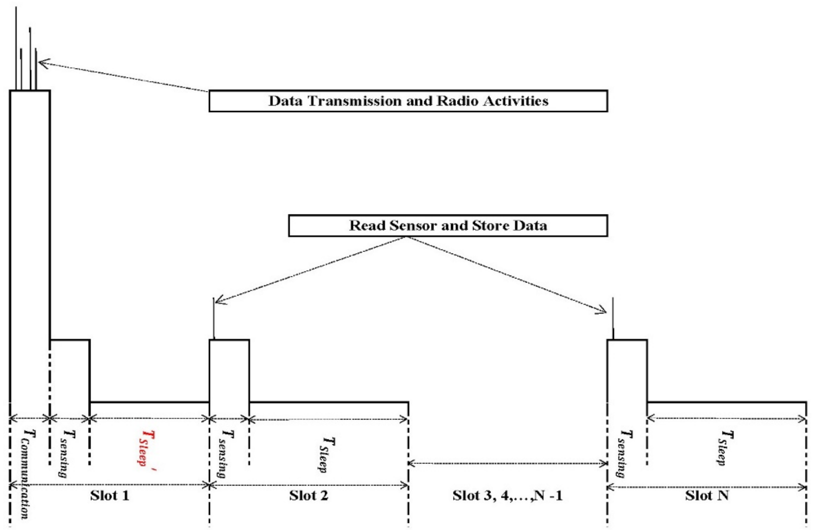


Fig. 5. Sleepy E-IoT system schedule overview

The nodes are synchronised by sending a CoAP PUT from the Linux PC to set the RTC of each node. This initial configuration allows users to accurately set the RTC timer for all nodes. Once the RTCs are configured, all nodes seamlessly enter the synchronised schedule at the commencement of the next hour. This synchronisation strategy supported by DS3231, compared with using the board’s built-in oscillator, improves the synchronisation accuracy of every node to operate within the schedule. This synchronisation method ensures that the entire system functions operate cohesively and reduces potential risks in timing and activity. Currently, all the nodes are synchronised to wake up sequentially, where they take turns to communicate. Thus, in this experiment, the energy consumption caused by overhearing is minimised in the laboratory environment. In future work, synchronisation process optimizations will be explored to further facilitate the initial setup, fundamentally supporting the overall efficiency of the system.

Table 3. Average current and time duration of different sleepy E-IoT system activities

Activity Information	Time Duration (s)	Average Current (mA)
6LoWPAN and Router_Advertisement	0.188	14.915
Neighbour_Solicitation	0.144	13.848
Neighbour_Advertisement	0.142	13.788
DIO	0.093	12.524
DAO-Tx & DAO_ACK-Rx	0.1395	14.702
CoAP Response	0.13	14.832
Set system to sensing state	0.049	1.476
Set system to sleep state	0.048	1.475
Turn off the radio	0.048	7.679
Set system to communication state	0.095	6.523
Read sensor and store data	0.798	2.963
Communication state	NA	12.85
Sensing state	NA	2.85
Sleep state– Standby Mode	NA	0.132

3.3 Overall System Energy Consumption

The time for the BR to periodically maintain the network was observed to be 60s including all the GNRC and RPL activities that cyclically construct the network. To begin with, the communication window defaults to 60s to allow all the network communication setups. Fig. 6 shows the daily energy stack of different schedule examples. This result indicates that daily communication energy consumption has increased as the number of communication windows increases. Significantly, the dominant factor in energy use is the sleep state so any deployable system should minimise this. As a result, in the case of one 60s communication window per day with a sensing rate of every 10 minutes, the daily energy consumption is 38.6J and the estimated system lifetime is 712 days (estimated using 2600 mAh 18650 Lithium-Ion rechargeable battery). Obviously, in a star topology, that 60s is longer than the necessary time overhead as the communication window can be shut after all the data is transmitted.

A potential solution to further optimise the system energy consumption is to reduce the unnecessary communication window length. The following assumption is made to focus on evaluating the energy consumption related to data transmission through CoAP. After the initial network construction phase, subsequent communication windows can be considered to exclude network maintenance activities, such as ICMPv6, RPL, and 6LoWPAN control messages. Each sensing event will create 18 Bytes of data, and each CoAP packet can maximumly contain 97 Bytes of payload [18]. Thus, 5 groups of readings can be transmitted with one CoAP packet. Assuming N sensor readings will be taken before each communication window, the minimum time required for one communication window can be summarised in Function (1). T_{IDLE} includes the time cost to wake up the node and turn the radio on. T_{CoAP} is the time response cost for a single CoAP request. $T_{Rdio\_STANDBY}$ is the time cost for turning off radio after completing data transmission. Adding up the minimum communication activity and sensing activity (sensing every 5/10 minutes and communication every hour) implies a 2s communications window could safely be used.

$$T_{communication} = T_{IDLE} + ROUNDUP(N * \frac{1}{5}) * T_{CoAP} + T_{Radio\_STANDBY} \quad (1)$$

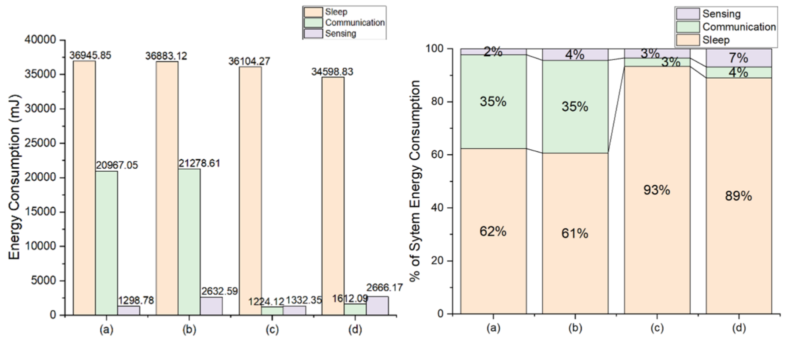


Fig. 6. Daily energy consumption of different states and energy stack (a and b coms every hour, c d coms daily) all 60s coms window. For both panels, from left to right, they are: (a) sensing every 10 minutes, 60s communication every hour; (b) sensing every 5 minutes, 60s communication every hour; (c) sensing every 10 minutes, 60s communication every day and (d) sensing every 5 minutes, 60s communication every day.

We can make similar assumptions for different communication frequencies. Referring to Fig. 5, if the communication is scheduled once per day, the value of $ROUNDUP(T_{communication} + T_{sensing})$ is equal to 5s (sensing every 10 minutes) and 9s (sensing every 5 minutes). Estimated results illustrated in Fig. 7 show that total energy consumption increased by 6.4% compared to (a) and (c), and a nearly 9.9% increase compared with (b) and (d). Theoretically, optimizing the system energy consumption by minimizing the time overhead of network maintenance such as RPL control message, is potentially valuable. In future work, it is potentially worth reducing the communication time overhead by focusing on minimizing unnecessary RPL control messages while ensuring basic RPL reconstruction requirements.

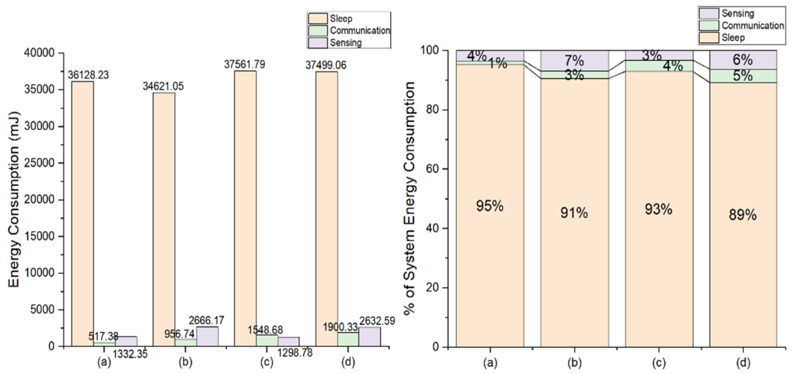


Fig. 7. Sleepy network daily energy stack with optimised prediction. For both panels, from left to right, they are: *(a)* sensing every 10 minutes, communication every day and *(b)* sensing every 5 minutes, communication every day; *(c)* sensing every 10 minutes, communication every hour and *(d)* sensing every 5 minutes, communication every hour.

4 Conclusion

Our investigation into the star topology sleepy E-IoT system has provided our understanding of the relationship between communication frequency, energy consumption, and network maintenance. This Sleepy E-IoT system design focuses on using CoAP, RPL and 6LoWPAN. Also, the system can enter the sleep state and wake-up state as scheduled to finish the planned tasks with reliable communication features. Furthermore, CoAP is applied to ensure the ability of the user to update and reconfigure the system presets. To comprehensively understand the impact of energy consumption with an increased number of communication windows, we have measured the current and time duration of system activities such as read sensor, store data, and radio activities controlled by the standard IoT protocols. The observed increases in daily communication energy consumption with additional windows emphasise the importance of a balanced approach that considers network reliability and communication time. The estimation result shows that optimizing the communication time overhead can reduce energy consumption further. According to Figures 6 and 7, when the system's sensing frequency remains the same, optimizing the communication window results in a reduction of system energy consumption by more than 1.6% (communication once a day) and 30.86% (communication every hour). We suggest optimizing communication windows and network activities to control the communication energy consumption, and this can further guide similar systems to take a balance between network reliability and energy consumption. In the upcoming stages, our attention will be directed towards optimizing energy consumption. This will be achieved through optimizing the system in routing configuration and communication patterns, aiming to further reduce overall system power requirements.

References

1. Martinez, K., Hart, J.K., Ong, R.: Environmental sensor networks. Computer 37, 50-56 (2004)

2. Nikoukar, A., Raza, S., Poole, A., Güneş, M., Dezfouli, B.: Low-Power Wireless for the Internet of Things: Standards and Applications. IEEE Access 6, 67893-67926 (2018)

3. Hart, J.K., Martinez, K.: Toward an environmental Internet of Things. Earth and Space Science 2, 194-200 (2015)

4. Shelby, Z., Hartke, K., Bormann, C., Frank, B.: RFC 7252: The constrained application protocol (CoAP). Internet Engineering Task Force (2014)

5. V, M., Giri, A.: Analysing the performance of 6LoWPAN- CoAP and RPL-CoAP on LoRaWAN in Constrained Environment. In: 2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), pp. 1-6. (Year)

6. Ashrif, F.F., Sundararajan, E.A., Ahmad, R., Hasan, M.K., Yadegaridehkordi, E.: Survey on the authentication and key agreement of 6LoWPAN: Open issues and future direction. Journal of Network and Computer Applications 221, 103759 (2024)

7. Khanafer, M., Guennoun, M., Mouftah, H.T.: A Survey of Beacon-Enabled IEEE 802.15.4 MAC Protocols in Wireless Sensor Networks. IEEE Communications Surveys & Tutorials 16, 856-876 (2014)

8. Dunkels, A.: The contikimac radio duty cycling protocol. Swedish Institute of Computer Science (2011)

9. Bragg, G.M., Martinez, K., Basford, P.J., Hart, J.K.: 868MHz 6LoWPAN with ContikiMAC for an Internet of Things environmental sensor network. In: 2016 SAI Computing Conference (SAI), pp. 1273-1277. (Year)

10. Martinez, K., Lu, J., Kendall, S., Hart, J.K.: Testing Low Power IP-Based Sensor Networks an a Forest Environment. In: AGU Fall Meeting Abstracts, pp. IN025-003. (Year)

11. Baccelli, E., Gündoğan, C., Hahm, O., Kietzmann, P., Lenders, M.S., Petersen, H., Schleiser, K., Schmidt, T.C., Wählisch, M.: RIOT: An open source operating system for low-end embedded devices in the IoT. IEEE Internet of Things Journal 5, 4428-4440 (2018)

12.Microchip: "SAM R30 IEEE 802.15.4 Sub-GHz System-in-Package Datasheet". In: Technology (ed.). datasheet (2017)

13.RIOT: RIOT Sniffer Application, https://github.com/RIOT-OS/applications/blob/master/sniffer/tools/README.md

14.Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., Alexander, R.: RPL: IPv6 routing protocol for low-power and lossy networks. In: RFC 6550. (Year)

15.Lenders, M.: Analysis and comparison of embedded network stacks. Master's thesis, Freie Universitat Berlin (2016)

16.Chen, Y., Chanet, J.P., Hou, K.M.: RPL Routing Protocol a case study: Precision agriculture. In: First China-France Workshop on Future Computing Technology (CF-WoFUCT 2012), pp. 6 p., (Year)

17.Winter, T., Thubert, P., Brandt, A., Hui, J.W., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J.-P., Alexander, R.K.: RFC 6550: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. Internet Requests for Comments 1-157 (2012)

18.Jeong, Y., Son, S., Lee, S., Lee, B.: A Total Crop-Diagnosis Platform Based on Deep Learning Models in a Natural Nutrient Environment. Applied Sciences 8, 1992 (2018)

Appendix B Experiment Code

1. All of the code developed for this work is available on the GitHub repository at https://github.com/Jiawei-Lu/Sleepy.

B.1 Code for 3.3. Design and Implementation of Sleepy Networks for E-IoT is available at:

https://github.com/Jiawei-Lu/Sleepy/tree/master/examples/star_topology

B.2 Code for Chapter 4. Optimization and Energy Profiling of Low Power Sleepy Network Design for Environmental IoT System are available at:

1. https://github.com/Jiawei-Lu/Sleepy/tree/master/examples/202407gnrc_networking

2. https://github.com/Jiawei-Lu/Sleepy/tree/master/examples/202407gnrc_network-ing_2nodes

3. Development logs are available at: https://github.com/Jiawei-Lu/Sleepy/blob/master/RE-ADME.md

B.3 Sleepy test code

Sleepy Function:

```
#define MAIN_QUEUE_SIZE     (8)
static msg_t _main_msg_queue[MAIN_QUEUE_SIZE];

int ds3231_print_time(struct tm testtime)
{
    int re;
    char dstr[ISOSTR_LEN];
    re = ds3231_get_time(&_dev, &testtime);
    if (re != 0) {
        puts("error: unable to read time");
        return 1;
    }

    size_t pos = strftime(dstr, ISOSTR_LEN, "%Y-%m-%dT%H:%M:%S", &testtime);
    dstr[pos] = '\0';
    printf("The current time is: %s\n", dstr);

    return 0;
}

struct tm _riot_bday = {
    .tm_sec = 42,
    .tm_min = 10,
    .tm_hour = 15,
    .tm_wday = 3,
    .tm_mday = 22,
    .tm_mon = 8,
    .tm_year = 123
};

void radio_off(gnrc_netif_t *netif){
    netopt_state_t state = NETOPT_STATE_SLEEP;
    while ((netif = gnrc_netif_iter(netif))) {
            /* retry if busy */
            while (gnrc_netapi_set(netif->pid, NETOPT_STATE, 0,
                &state, sizeof(state)) == -EBUSY) {}
    }
}
/*
        case NETOPT_STATE_STANDBY:
            at86rf2xx_set_state(dev, AT86RF2XX_STATE_TRX_OFF);
            break;
        case NETOPT_STATE_SLEEP:
            at86rf2xx_set_state(dev, AT86RF2XX_STATE_SLEEP);
            break;
        case NETOPT_STATE_IDLE:
            at86rf2xx_set_state(dev, AT86RF2XX_PHY_STATE_RX);
            break;
        case NETOPT_STATE_TX:
*/

void radio_on(gnrc_netif_t *netif){
    netopt_state_t state = NETOPT_STATE_IDLE;
    while ((netif = gnrc_netif_iter(netif))) {
            /* retry if busy */
            while (gnrc_netapi_set(netif->pid, NETOPT_STATE, 0,
                &state, sizeof(state)) == -EBUSY) {}
    }
}

int _send_dao(void){

    _dao_check = 0;
    _dao_count = 0.00;
    _dao_attampt = 0;
    while (_dao_check == 0 && _dao_attampt < 3){

    for (uint8_t i = 0; i < GNRC_RPL_INSTANCES_NUMOF; ++i) {
        if (gnrc_rpl_instances[i].state == 0) {
            continue;
        }

        dodag = &gnrc_rpl_instances[i].dodag;

        gnrc_rpl_send_DAO(dodag->instance, NULL, dodag->default_lifetime);
    }

        puts("send DAO\n");
        int _retries_wait = 0;
        while (_dao_check == 0 && _retries_wait<3){
        // while (_dao_check == 0){
            ztimer_sleep(ZTIMER_MSEC, 0.2* MS_PER_SEC);
            printf("%d\n", _retries_wait);
            _retries_wait++;
        }
        _dao_count += _retries_wait*0.2;
        _dao_attampt++;
    }
    printf("%f", _dao_count);
    return _dao_count;
}

static int _cmd_dao_send(int argc, char **argv){
    if (argc > 1 || strcmp(argv[0], "dao") != 0) {

        return 1;
    }
    _dao_check = 0;
    while (_dao_check == 0){

    for (uint8_t i = 0; i < GNRC_RPL_INSTANCES_NUMOF; ++i) {
        if (gnrc_rpl_instances[i].state == 0) {
            continue;
        }
        dodag = &gnrc_rpl_instances[i].dodag;

        /*
        After run gnrc_rpl_send_DAO(), when DAO_ACK is received
        --> dodag->dao_ack_received
        --> _dao_check == 1;
        */
        gnrc_rpl_send_DAO(dodag->instance, NULL, dodag->default_lifetime);
    }
        puts("send DAO\n");

        int _retries_wait = 0;
        while (_dao_check == 0 && _retries_wait<2){
            ztimer_sleep(ZTIMER_MSEC, 0.2* MS_PER_SEC);
            printf("%d\n", _retries_wait);
            _retries_wait++;
        }

    }
    return 0;
}

static int sleepy(int argc, char **argv){
    if (argc < 2 || argc > 4) {
       return _print_usage(argv);
    }
    radio_off(radio_netif);
    for (int _i=1;_i<120;_i++){
        struct tm _time;
        int _res = ds3231_clear_alarm_1_flag(&_dev);
        if (_res != 0) {
            puts("error: unable to clear alarm flag");
            return 1;
        }
        _res = ds3231_get_time(&_dev, &_time);
        if (_res != 0) {
            puts("error: unable to read time");
            return 1;
        }
        time_t _start_time, _test_time;
        _start_time = mktime(&_time);
        int time = atoi(argv[1]);
        if (argc > 2){
            sequence = atoi(argv[2]);
        }
        if (argc == 4){
            _node_com_window = atoi(argv[3]);
        }

        _test_time = (_start_time/60+time)*60;
        double diff = difftime(_test_time, _start_time);

        _time.tm_sec += ((int)diff + middle_gap + (sequence -1) *
(int)_node_com_window);// * ONE_S;
        mktime(&_time);
        printf("watting %d seconds to start sleepy test\n", (int)((int)diff +
middle_gap + (sequence -1) * _node_com_window));
        printf("radio off \n");
        _res = ds3231_set_alarm_1(&_dev, &_time, DS3231_AL1_TRIG_H_M_S);
        if (_res != 0) {
            puts("error: unable to program alarm");
            return 1;
        }
        pm_set(SAML21_PM_MODE_STANDBY);

        puts("waking up and ready to do coap \n");
        message_ack_flag =0;
        radio_on(radio_netif);

        /*-------------------------------------------202407 GCoAP (argc=5)--
---------------------------------------------*/
        int _argc2 = 5;
        char *_argv2[] = {"coap", "put", "-c",
"coap://[2001:630:d0:1000::d683]:5683/data", "test data"};  //mini-linux-pc--
remote


        message_ack_flag = 0;
        retries = 0;

        /*-----------------------------Send DAO to rejoin-------------------
-------------*/
        _send_dao();



        // /*-----------------------------Send DAO to rejoin----------------
----------------*/
        // _gnrc_rpl_send_dis();

        ztimer_sleep(ZTIMER_MSEC, (_node_com_window - 4 - _dao_count)*
MS_PER_SEC);
        printf("dao_count: %f\n", (_dao_count));
        message_ack_flag = 0;
        while (message_ack_flag != 1 && retries < 15){
            _coap_result = gcoap_cli_cmd(_argc2,_argv2);
            if (_coap_result == 0) {
                printf("Command executed successfully, and Reyries: %d\n",
retries);
                int wait = 0;
                while(message_ack_flag == 0 && wait < 3){
                    puts("waitting for the message sent flag\n");
                    ztimer_sleep(ZTIMER_MSEC, 0.2* MS_PER_SEC); //DO NOT use
NS_PER_MS as it crushes program
                    printf("waitting time is %d\n", wait);
                    wait++;
                }
            }else {
                printf("Command execution failed\n");

            }
            retries++;
            count_total_try++;
        }
        count_successful++;
        printf("successful : %d, total : %d\n", count_successful, count_to-
tal_try);

        radio_off(radio_netif);
    }
    radio_on(radio_netif);
    return 1;
}

static int _print_usage(char **argv)
{
    printf("usage: %s <sleep duration in minutes> <wake up sequence No.> [op-
tional communication window size]\n", argv[0]);
    printf("       %s sleepy 10 2 5\n", argv[0]);
    printf("       %s wake node up in the begining of 10 minutes after (se-
quence number No.2,with communication window sieze of 5)\n", argv[0]);
    return 1;
}

#if defined(MODULE_PERIPH_GPIO_IRQ) && defined(BTN0_PIN)
static void btn_cb(void *ctx)
{
    (void) ctx;
    puts("BTN0 pressed.");
}
#endif /* MODULE_PERIPH_GPIO_IRQ */


static const shell_command_t shell_commands[] = {
    { "coap", "CoAP example", gcoap_cli_cmd },
    { "sleepy", "make system sleepy and wake up n minutes after", sleepy },
    { "dao", "Send DAO", _cmd_dao_send },
}
```

Appendix C Data

1. CoAP maximum payload test Wireshark capture details:

https://github.com/Jiawei-Lu/Sleepy/blob/master/COAP_wireshark_details.pdf

2. Communication window design under different wake-up strategies - Wireshark capture details:

https://github.com/Jiawei-Lu/Sleepy/tree/master/Communication%20window%20design%20under%20different%20wake-up%20strategies

3. Average current and time duration of different sleepy E-IoT system activities:

https://github.com/Jiawei-Lu/Sleepy/blob/master/Average%20current%20and%20time%20duration%20of%20different%20sleepy%20E-IoT%20system%20activities.pdf

References

[1] K. Ashton. "That ‘internet of things’ thing." https://www.rfidjournal.com/expert-views/that-internet-of-things-thing/73881/ (accessed 11th Dec, 2025).

[2] P. Suresh, J. V. Daniel, V. Parthasarathy, and R. Aswathy, "A state of the art review on the internet of things (iot) history, technology and fields of deployment," presented at the 2014 International conference on science engineering and management research (ICSEMR), 2014, doi: 10.1109/ICSEMR.2014.7043637.

[3] K. Martinez, J. K. Hart, and R. Ong, "Environmental sensor networks," *Computer,* vol. 37, no. 8, pp. 50-56, 2004, doi: 10.1109/MC.2004.91.

[4] J. K. Hart and K. Martinez, "Environmental sensor networks: A revolution in the earth system science?," *Earth-Science Reviews,* vol. 78, no. 3, pp. 177-191, 2006/10/01/ 2006, doi: 10.1016/j.earscirev.2006.05.001.

[5] J. K. Hart and K. Martinez, "Toward an environmental internet of things," *Earth and Space Science,* vol. 2, no. 5, pp. 194-200, 2015/05/01 2015, doi: 10.1002/2014EA000044.

[6] M. H. Alsharif, S. Kim, and N. Kuruoğlu, "Energy harvesting techniques for wireless sensor networks/radio-frequency identification: A review," *Symmetry,* vol. 11, no. 7, p. 865, 2019, doi: 10.3390/sym11070865.

[7] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Networks,* vol. 7, no. 3, pp. 537-568, 2009/05/01/ 2009, doi: 10.1016/j.adhoc.2008.06.003.

[8] O. Vermesan *et al.*, "Internet of things strategic research roadmap," in *Internet of things-global technological and societal trends from smart environments and spaces to green ict*: River Publishers, 2022, pp. 9-52.

[9] S. Lin *et al.*, "Atpc: Adaptive transmission power control for wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN),* vol. 12, no. 1, pp. 1-31, 2016, doi: 10.1145/2746342.

[10] X. Chu and H. Sethu, "Cooperative topology control with adaptation for improved lifetime in wireless sensor networks," *Ad Hoc Networks,* vol. 30, pp. 99-114, 2015/07/01/ 2015, doi: 10.1016/j.adhoc.2015.03.007.

[11] F. M. Costa and H. Ochiai, "A comparison of modulations for energy optimization in wireless sensor network links," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, 2010: IEEE, pp. 1-5, doi: 10.1109/GLOCOM.2010.5683412.

[12] M. Moghaddam *et al.*, "A wireless soil moisture smart sensor web using physics-based optimal control: Concept and initial demonstrations," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 3, no. 4, pp. 522-535, 2010, doi: 10.1109/JSTARS.2010.2052918.

[13] X. Li *et al.*, "A multi-interface ice and snow remote monitoring platform in the polar region," *IEEE Sensors Journal,* vol. 14, no. 11, pp. 3738-3744, 2014, doi: 10.1109/JSEN.2014.2309654.

[14] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Communications Surveys & Tutorials,* vol. 13, no. 3, pp. 443-461, 2011, doi: 10.1109/SURV.2011.060710.00094.

[15] D. Petrovic, R. C. Shah, K. Ramchandran, and J. Rabaey, "Data funneling: Routing with aggregation and compression for wireless sensor networks," presented at the IEEE International Workshop on Sensor Network Protocols and Applications, 2023/06/11, 2003, doi: 10.1109/SNPA.2003.1203366.

[16] T. Sheltami, M. Musaddiq, and E. Shakshuki, "Data compression techniques in wireless sensor networks," *Future Generation Computer Systems,* vol. 64, pp. 151-162, 2016/11/01/ 2016, doi: 10.1016/j.future.2016.01.015.

[17] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," presented at the International Conference on Information Technology: Coding and Computing (ITCC'05), 4-6 April 2005, 2005, doi: 10.1109/itcc.2005.43.

[18] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: A survey," *IEEE Wireless Communications,* vol. 14, no. 2, pp. 70-87, 2007, doi: 10.1109/MWC.2007.358967.

[19] G. M. Dias, B. Bellalta, and S. Oechsner, "A survey about prediction-based data reduction in wireless sensor networks," *ACM Comput. Surv.,* vol. 49, no. 3, p. Article 58, 2016, doi: 10.1145/2996356.

[20] M. Chen and M. L. Fowler, "Data compression trade-offs in sensor networks," presented at the Mathematics of Data/Image Coding, Compression, and Encryption VII, with Applications, 2004, doi: 10.1117/12.562187.

[21] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proceedings of the 7th symposium on Operating systems design and implementation*, 2006, pp. 381-396. [Online]. Available: https://www.usenix.org/legacy/event/osdi06/tech/full_papers/werner-allen/werner-allen_html/.

[22] A. Awang and M. H. Suhaimi, "Rimbamon©: A forest monitoring system using wireless sensor networks," in *2007 International Conference on Intelligent and Advanced Systems*, 25-28 Nov. 2007 2007, pp. 1101-1106, doi: 10.1109/ICIAS.2007.4658555.

[23] R. C. Carrano, D. Passos, L. C. S. Magalhaes, and C. V. N. Albuquerque, "Survey and taxonomy of duty cycling mechanisms in wireless sensor networks," *IEEE Communications Surveys & Tutorials,* vol. 16, no. 1, pp. 181-194, 2014, doi: 10.1109/SURV.2013.052213.00116.

[24] A. Dunkels, "The contikimac radio duty cycling protocol," 2011.

[25] T. Winter *et al.*, "Rfc 6550: Rpl: Ipv6 routing protocol for low-power and lossy networks," *Request for Comments,* pp. 1-157, 2012.

[26] H. Kharrufa, H. A. A. Al-Kashoash, and A. H. Kemp, "Rpl-based routing protocols in iot applications: A review," *IEEE Sensors Journal,* vol. 19, no. 15, pp. 5952-5967, 2019, doi: 10.1109/JSEN.2019.2910881.

[27] N. M. Shakya, M. Mani, and N. Crespi, "Seeof: Smart energy efficient objective function: Adapting rpl objective function to enable an ipv6 meshed topology solution for battery operated smart meters," in *2017 Global Internet of Things Summit (GIoTS)*, 2017: IEEE, pp. 1-6, doi: 10.1109/GIOTS.2017.8016252.

[28] B. Mohamed and F. Mohamed, "Qos routing rpl for low power and lossy networks," *International Journal of Distributed Sensor Networks,* vol. 11, no. 11, p. 971545, 2015, doi: 10.1155/2015/971545.

[29] P. O. Kamgueu, E. Nataf, T. D. Ndié, and O. Festor, "Energy-based routing metric for rpl," Inria, 2013. [Online]. Available: https://inria.hal.science/hal-00779519v1.

[30] N. Khelifi, S. Oteafy, H. Hassanein, and H. Youssef, "Proactive maintenance in rpl for 6lowpan," in *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2015: IEEE, pp. 993-999, doi: 10.1109/IWCMC.2015.7289218.

[31] M. Barcelo, A. Correa, J. L. Vicario, and A. Morell, "Cooperative interaction among multiple rpl instances in wireless sensor networks," *Computer Communications,* vol. 81, pp. 61-71, 2016, doi: 10.1016/j.comcom.2015.12.008.

[32] K. Doddapaneni, E. Ever, O. Gemikonakli, I. Malavolta, L. Mostarda, and H. Muccini, "Path loss effect on energy consumption in a wsn," in *2012 UKSim 14th International Conference on Computer Modelling and Simulation*, 28-30 March 2012 2012, pp. 569-574, doi: 10.1109/UKSim.2012.87.

[33] S. Madakam, V. Lake, V. Lake, and V. Lake, "Internet of things (iot): A literature review," *Journal of Computer and Communications,* vol. 3, no. 05, p. 10, 2015, doi: 10.4236/jcc.2015.35021.

[34] A. Ramaprasad, A. Sánchez-Ortiz, and T. Syn, "A unified definition of a smart city," in *International Conference on Electronic Government*, 04 Augest 2017: Springer, pp. 13-24, doi: 10.1007/978-3-319-64677-0_2.

[35] A. Khare, G. Merlino, F. Longo, A. Puliafito, and O. P. Vyas, "Design of a trustless smart city system: The #smartme experiment," *Internet of Things,* vol. 10, p. 100126, 2020/06/01/ 2020, doi: 10.1016/j.iot.2019.100126.

[36] D. Bruneo, S. Distefano, F. Longo, and G. Merlino, "An iot testbed for the software defined city vision: The #smartme project," in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, 18-20 May 2016 2016, pp. 1-6, doi: 10.1109/SMARTCOMP.2016.7501678.

[37] D. Bruneo *et al.*, "An iot service ecosystem for smart cities: The #smartme project," *Internet of Things,* vol. 5, pp. 12-33, 2019/03/01/ 2019, doi: 10.1016/j.iot.2018.11.004.

[38] A. Bassi *et al.*, *Enabling things to talk: Designing iot solutions with theiot architectural reference model*. Springer Berlin, Heidelberg, 2013.

[39] T. Kramp, R. Kranenburg, and S. Lange, "Introduction to the internet of things," 2013, pp. 1-10, doi: 10.1007/978-3-642-40403-0_1.

[40] J. V. V. Sobral, J. J. P. C. Rodrigues, R. A. L. Rabêlo, J. Al-Muhtadi, and V. Korotaev, "Routing protocols for low power and lossy networks in internet of things applications," *Sensors,* vol. 19, no. 9, p. 2144, 2019, doi: 10.3390/s19092144.

[41] K. Gulati, R. S. Kumar Boddu, D. Kapila, S. L. Bangare, N. Chandnani, and G. Saravanan, "A review paper on wireless sensor network techniques in internet of things (iot)," *Materials Today: Proceedings,* vol. 51, pp. 161-165, 2022/01/01/ 2022, doi: 10.1016/j.matpr.2021.05.067.

[42] C. Buratti, A. Conti, D. Dardari, and R. Verdone, "An overview on wireless sensor networks technology and evolution," *Sensors,* vol. 9, no. 9, pp. 6869-6896, 2009, doi: 10.3390/s90906869.

[43] E. Egea-López, J. Vales-Alonso, A. Martinez-Sala, P. Pavon-Marino, and J. Garcia-Haro, "Simulation tools for wireless sensor networks," presented at the Summer Simulation Multiconference - SPECTS 2005, 2005. [Online]. Available:

https://www.researchgate.net/profile/Pablo-Pavon-Marino/publication/255642447_Simulation_Tools_for_Wireless_Sensor_Networks/links/556f07ed08aeab77722828d9/Simulation-Tools-for-Wireless-Sensor-Networks.pdf.

[44] H. N. Pham, D. Pediaditakis, and A. Boulis, "From simulation to real deployments in wsn and back," in *2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 18-21 June 2007 2007, pp. 1-6, doi: 10.1109/WOWMOM.2007.4351800.

[45] A. Nikoukar, S. Raza, A. Poole, M. Güneş, and B. Dezfouli, "Low-power wireless for the internet of things: Standards and applications," *IEEE Access,* vol. 6, pp. 67893-67926, 2018, doi: 10.1109/ACCESS.2018.2879189.

[46] J. K. Hart and K. Martinez, "Sensor networks and geohazards," in *Reference module in earth systems and environmental sciences*: Elsevier, 2020, doi: 10.1016/B978-0-12-818234-5.00037-7.

[47] V. P. Al Gore, "The digital earth: Understanding our planet in the 21st century," 1999.

[48] P. Stacey and D. Berry, "Towards a digital earth: Using archetypes to enable knowledge interoperability within geo-observational sensor systems design," *Earth Science Informatics,* vol. 11, no. 2, pp. 307-323, 2018, doi: 10.1007/s12145-018-0340-z.

[49] F. Hiromichi, P. Limlahapun, and K. Takaharu, "Real time monitoring for imja glacial lake in himalaya — global warming front monitoring system," in *2008 SICE Annual Conference*, 20-22 Aug. 2008 2008, pp. 2578-2581, doi: 10.1109/SICE.2008.4655100.

[50] Wirawan, S. Rachman, I. Pratomo, and N. Mita, "Design of low cost wireless sensor networks-based environmental monitoring system for developing country," in *2008 14th Asia-Pacific Conference on Communications*, 14-16 Oct. 2008 2008, pp. 1-5.

[51] B. K. Maharrey, A. S. Lim, and S. Gao, "Interconnection between ip networks and wireless sensor networks," *International Journal of Distributed Sensor Networks,* vol. 8, no. 12, p. 567687, 2012, doi: 10.1155/2012/567687.

[52] A. Sofwan, Sumardi, M. Ridho, A. Goni, and Najib, "Wireless sensor network design for land-slide warning system in iot architecture," in *2017 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, 18-19 Oct. 2017 2017, pp. 280-283, doi: 10.1109/ICITACEE.2017.8257718.

[53] A. Fabre *et al.*, "Deploying a 6lowpan, coap, low power, wireless sensor network: Poster abstract," 2016, doi: 10.1145/2994551.2996707.

[54] G. M. Bragg, K. Martinez, P. J. Basford, and J. K. Hart, "868mhz 6lowpan with contikimac for an internet of things environmental sensor network," in *2016 SAI Computing Conference (SAI)*, 13-15 July 2016 2016, pp. 1273-1277, doi: 10.1109/SAI.2016.7556143.

[55] K. Martinez, R. Ong, and J. Hart, "Glacsweb: A sensor network for hostile environments," in *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.*, 2004: IEEE, pp. 81-87, doi: 10.1109/sahcn.2004.1381905.

[56] K. Martinez, P. Padhy, A. Riddoch, R. Ong, and J. Hart, "Glacial environment monitoring using sensor networks," in *Proceedings of the Workshop on Real-World Wireless Sensor Networks (REALWSN'05), Stockholm, Sweden*, 2005: Citeseer, pp. 20-21.

[57] T. Ward, K. Martinez, and T. Chown, "Simulated analysis of connectivity issues for sleeping sensor nodes in the internet of things," presented at the Proceedings of the 11th ACM

symposium on Performance evaluation of wireless ad hoc, sensor, & ubiquitous networks, Montreal, QC, Canada, 2014, doi: 10.1145/2653481.2653490.

[58] W. M. El-Sayed, H. M. El-Bakry, and S. M. El-Sayed, "Integrated data reduction model in wireless sensor networks," *Applied Computing and Informatics,* vol. 19, no. 1/2, pp. 41-63, 2023, doi: 10.1016/j.aci.2019.03.003.

[59] Z. Alliance. "Zigbee specification." https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf.

[60] F. Kuntke, V. Romanenko, S. Linsner, E. Steinbrink, and C. Reuter, "Lorawan security issues and mitigation options by the example of agricultural iot scenarios," *Transactions on Emerging Telecommunications Technologies,* vol. 33, 05/01 2022, doi: 10.1002/ett.4452.

[61] P. Thubert, "An architecture for ipv6 over the time-slotted channel hopping mode of ieee 802.15. 4 (6tisch)," *Request for Comments,* May 2021, doi: 10.17487/RFC9030.

[62] S. Farahani, "Chapter 1 - zigbee basics," in *Zigbee wireless networks and transceivers*, S. Farahani Ed. Burlington: Newnes, 2008, pp. 1-24, doi: 10.1016/B978-0-7506-8393-7.00001-7.

[63] A. Rayes and S. Salam, "Iot protocol stack: A layered view," in *Internet of things from hype to reality: The road to digitization*, A. Rayes and S. Salam Eds. Cham: Springer International Publishing, 2022, ch. 5, pp. 97-152, doi: 10.1007/978-3-030-90158-5_5.

[64] "Ieee standard for information technology-- local and metropolitan area networks-- specific requirements-- part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (wpans)," *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003),* pp. 1-320, 2006, doi: 10.1109/ieeestd.2006.232110.

[65] R. Gorrepotu, N. S. Korivi, K. Chandu, and S. Deb, "Sub-1ghz miniature wireless sensor node for iot applications," *Internet of Things,* vol. 1-2, pp. 27-39, 2018/09/01/ 2018, doi: 10.1016/j.iot.2018.08.002.

[66] L. Klingbeil and T. Wark, "A wireless sensor network for real-time indoor localisation and motion monitoring," in *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, 22-24 April 2008 2008, pp. 39-50, doi: 10.1109/IPSN.2008.15.

[67] S. Kim *et al.*, "Evaluation of a 433 mhz band body sensor network for biomedical applications," *Sensors*, vol. 13, no. 1*,* pp. 898-917doi: 10.3390/s130100898.

[68] H. Li *et al.*, "An omnibearing remote control night-light based on 315 mhz wireless radio frequency," *Journal of Sensor Technology and Application,* vol. 3, pp. 1-8, 2015, doi: 10.12677/jsta.2015.31001.

[69] K. Martinez, J. K. Hart, P. J. Basford, G. M. Bragg, T. Ward, and D. S. Young, "A geophone wireless sensor network for investigating glacier stick-slip motion," *Computers & Geosciences,* vol. 105, pp. 103-112, 2017/08/01/ 2017, doi: 10.1016/j.cageo.2017.05.005.

[70] D. Gao, S. Zhang, and F. Zhang, "Has-mac: A hybrid asynchronous and synchronous communication system for energy-harvesting wireless sensor networks," *Wireless Personal Communications,* vol. 119, 07/01 2021, doi: 10.1007/s11277-021-08304-7.

[71] J. Kim, J. On, S. Kim, and J. Lee, "Performance evaluation of synchronous and asynchronous mac protocols for wireless sensor networks," in *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*, 25-31 Aug. 2008 2008, pp. 500-506, doi: 10.1109/SENSORCOMM.2008.80.

[72] M. Khanafer, M. Guennoun, and H. T. Mouftah, "A survey of beacon-enabled ieee 802.15.4 mac protocols in wireless sensor networks," *IEEE Communications Surveys & Tutorials,* vol. 16, no. 2, pp. 856-876, 2014, doi: 10.1109/SURV.2013.112613.00094.

[73] "Ieee standard for low-rate wireless networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011),* pp. 1-709, 2016, doi: 10.1109/ieeestd.2016.7460875.

[74] M. Oliver and A. Escudero, "Study of different csma/ca ieee 802.11-based implementations," *EUNICE contribution,* 1999.

[75] D. De Guglielmo, S. Brienza, and G. Anastasi, "Ieee 802.15.4e: A survey," *Computer Communications,* vol. 88, pp. 1-24, 2016/08/15/ 2016, doi: 10.1016/j.comcom.2016.05.004.

[76] "Ieee standard for local and metropolitan area networks--part 15.4: Low-rate wireless personal area networks (lr-wpans)," *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006),* pp. 1-314, 2011, doi: 10.1109/ieeestd.2011.6012487.

[77] S. Farahani, "Chapter 3 - zigbee and ieee 802.15.4 protocol layers," in *Zigbee wireless networks and transceivers*, S. Farahani Ed. Burlington: Newnes, 2008, ch. 3, pp. 33-135, doi: 10.1016/B978-0-7506-8393-7.00003-0.

[78] J. D. R. Nepomuceno and N. M. C. Tiglao, "Performance evaluation of 6tisch for resilient data transport in wireless sensor networks," in *2017 International Conference on Information Networking (ICOIN)*, 11-13 Jan. 2017 2017, pp. 552-557, doi: 10.1109/ICOIN.2017.7899546.

[79] S. B. Yaala and R. Bouallegue, "On mac layer protocols towards internet of things: From ieee802.15.4 to ieee802.15.4e," in *2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 22-24 Sept. 2016 2016, pp. 1-5, doi: 10.1109/SOFTCOM.2016.7772165.

[80] X. Vilajosana, K. Pister, and T. Watteyne, "Rfc 8180: Minimal ipv6 over the tsch mode of ieee 802.15. 4e (6tisch) configuration," *Request for Comments,* 2017.

[81] X. Vilajosana, T. Watteyne, T. Chang, M. Vučinić, S. Duquennoy, and P. Thubert, "Ietf 6tisch: A tutorial," *IEEE Communications Surveys & Tutorials,* vol. 22, no. 1, pp. 595-615, 2019, doi: 10.1109/COMST.2019.2939407.

[82] D. Hauweele, R.-A. Koutsiamanis, B. Quoitin, and G. Z. Papadopoulos, "Thorough performance evaluation & analysis of the 6tisch minimal scheduling function (msf)," *Journal of Signal Processing Systems,* 2021/06/02 2021, doi: 10.1007/s11265-021-01668-w.

[83] T. Watteyne *et al.*, "Industrial wireless ip-based cyber –physical systems," *Proceedings of the IEEE,* vol. 104, no. 5, pp. 1025-1038, 2016, doi: 10.1109/JPROC.2015.2509186.

[84] M. Michel and B. Quoitin, "Technical report: Contikimac performance analysis," *arXiv preprint arXiv:1404.3589,* 2014, doi: 10.48550/arXiv.1404.3589.

[85] M. Amirinasab Nasab, S. Shamshirband, A. T. Chronopoulos, A. Mosavi, and N. Nabipour, "Energy-efficient method for wireless sensor networks low-power radio operation in internet of things," *Electronics,* vol. 9, no. 2, p. 320, 2020, doi: 10.3390/electronics9020320.

[86] J. Hui and P. Thubert, "Rfc 6282:Compression format for ipv6 datagrams over ieee 802.15. 4-based networks," *Request for Comments,* 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6282.

[87] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Rfc 4944:Transmission of ipv6 packets over ieee 802.15. 4 networks," *Request for Comments,* vol. 4944, p. 130, 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4944.txt.

[88] N. Kushalnagar, G. Montenegro, and C. Schumacher, "Rfc 4919: Ipv6 over low-power wireless personal area networks (6lowpans): Overview, assumptions, problem statement, and goals," *Request for Comments,* 2007. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4919.

[89] H. A. A. Al-Kashoash and A. H. and Kemp, "Comparison of 6lowpan and lpwan for the internet of things," *Australian Journal of Electrical and Electronics Engineering,* vol. 13, no. 4, pp. 268-274, 2016/10/01 2016, doi: 10.1080/1448837x.2017.1409920.

[90] I. A. ISMAILI, A. Azyat, N. Raissouni, N. B. Achhab, A. Chahboun, and M. Lahraoua, "Comparative study of zigbee and 6lowpan protocols," in *Third International Conference on Computing and Wireless Communication Systems, ICCWCS 2019*, 2019: European Alliance for Innovation (EAI), doi: 10.4108/eai.24-4-2019.2284215.

[91] H. A. Al-Kashoash and A. H. Kemp, "Comparison of 6lowpan and lpwan for the internet of things," *Australian Journal of Electrical and Electronics Engineering,* vol. 13, no. 4, pp. 268-274, 2016, doi: 10.1080/1448837X.2017.1409920.

[92] C. Gomez, A. Minaburo, L. Toutain, D. Barthel, and J. C. Zuniga, "Ipv6 over lpwans: Connecting low power wide area networks to the internet (of things)," *IEEE Wireless Communications,* vol. 27, no. 1, pp. 206-213, 2020, doi: 10.1109/MWC.001.1900215.

[93] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, and J.-C. Zúñiga, "Rfc 8724: Schc: Generic framework for static context header compression and fragmentation," *Request for Comments,* 01 April 2020, doi: 10.17487/RFC8724.

[94] Y. Chen, J. P. Chanet, and K. M. Hou, "Rpl routing protocol a case study: Precision agriculture," in *First China-France Workshop on Future Computing Technology (CF-WoFUCT 2012)*, Harbin, China, 2012-02-16 2012, 2012, p. 6 p. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00681319.

[95] P. Thubert, "Rfc 6552: Objective function zero for the routing protocol for low-power and lossy networks (rpl)," *Request for Comments,* 2012. [Online]. Available: https://datatracker.ietf.org/doc/rfc6552/.

[96] A. Brandt *et al.*, "Rpl: Ipv6 routing protocol for low-power and lossy networks," *Internet Requests for Comments,* 2012.

[97] O. Gnawali and P. Levis, "Rfc: 6719the minimum rank with hysteresis objective function," *Request for Comments,* p. 13, 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6719.html?theme=2019.

[98] S. Chakrabarti, E. Nordmark, and C. Bormann, "Rfc 6775: Neighbor discovery optimization for ipv6 over low-power wireless personal area networks (6lowpans)," *Request for Comments,* 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6775.html.

[99] A. Musaddiq, R. Ali, J.-G. Choi, B.-S. Kim, and S. W. Kim, "Collision observation-based optimization of low-power and lossy iot network using reinforcement learning," *Computers, Materials & Continua,* vol. 67, pp. 799-814, 11/23 2020, doi: 10.32604/cmc.2021.014751.

[100] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Rfc 4861: Neighbor discovery for ip version 6 (ipv6)," *Request for Comments,* 2007, doi: 10.17487/RFC4861.

[101] F. Medjek, D. Tandjaoui, N. Djedjig, and I. Romdhani, "Multicast dis attack mitigation in rpl-based iot-llns," *Journal of Information Security and Applications,* vol. 61, p. 102939, 2021/09/01/ 2021, doi: 10.1016/j.jisa.2021.102939.

[102] Z. Shelby, K. Hartke, and C. Bormann, "Rfc 7252:The constrained application protocol (coap)," *Request for Comments,* 2014. [Online]. Available: http://www.rfc-editor.org/info/rfc7252.

[103] M. V and A. Giri, "Analysing the performance of 6lowpan- coap and rpl-coap on lorawan in constrained environment," in *2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, 2-4 Nov. 2023 2023, pp. 1-6, doi: 10.1109/CSITSS60515.2023.10334082.

[104] F. F. Ashrif, E. A. Sundararajan, R. Ahmad, M. K. Hasan, and E. Yadegaridehkordi, "Survey on the authentication and key agreement of 6lowpan: Open issues and future direction," *Journal of Network and Computer Applications,* vol. 221, p. 103759, 2024/01/01/ 2024, doi: 10.1016/j.jnca.2023.103759.

[105] C. Bormann, S. Lemay, H. Tschofenig, K. Hartke, B. Silverajan, and B. Raymor, "Rfc 8323: Coap (constrained application protocol) over tcp, tls, and websockets," *Request for Comments,* 2018, doi: 10.17487/RFC8323.

[106] P. Saint-Andre, "Rfc 6120:Extensible messaging and presence protocol (xmpp): Core," *Request for Comments,* 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6120.

[107] F. Baker and D. Meyer, "Rfc 6272: Internet protocols for the smart grid," *Request for Comments,* 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6272.

[108] P. Gupta and M. I. O. P, "A survey of application layer protocols for internet of things," in *2021 International Conference on Communication information and Computing Technology (ICCICT)*, 25-27 June 2021 2021, pp. 1-6, doi: 10.1109/ICCICT50803.2021.9510140.

[109] M. O. A. Enany, H. M. Harb, and G. Attiya, "A comparative analysis of mqtt and iot application protocols," in *2021 International Conference on Electronic Engineering (ICEEM)*, 3-4 July 2021 2021, pp. 1-6, doi: 10.1109/ICEEM52022.2021.9480384.

[110] M. Bansal and Priya, "Performance comparison of mqtt and coap protocols in different simulation environments," in *Inventive Communication and Computational Technologies*, Singapore, G. Ranganathan, J. Chen, and Á. Rocha, Eds., 25 September 200: Springer Singapore, pp. 549-560, doi: 10.1007/978-981-15-7345-3_47.

[111] F. Palmese, E. Longo, A. E. Redondi, and M. Cesana, "Coap vs. Mqtt-sn: Comparison and performance evaluation in publish-subscribe environments," in *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, 09 November 2021: IEEE, pp. 153-158, doi: 10.1109/WF-IoT51360.2021.9595725.

[112] P. Levis *et al.*, "Tinyos: An operating system for sensor networks," in *Ambient intelligence*: Springer, Berlin, Heidelberg, 2005, pp. 115-148, doi: 10.1007/3-540-27139-2_7.

[113] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *29th annual IEEE international conference on local computer networks*, 2004: IEEE, pp. 455-462, doi: 10.1109/LCN.2004.38.

[114] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He, "The liteos operating system: Towards unix-like abstractions for wireless sensor networks," in *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, 22-24 April 2008 2008, pp. 233-244, doi: 10.1109/IPSN.2008.54.

[115] Zephyr-project. "Zephyr project documentation." https://docs.zephyrproject.org/latest/index.html (accessed 15th Jan, 2025).

[116] E. Baccelli *et al.*, "Riot: An open source operating system for low-end embedded devices in the iot," *IEEE Internet of Things Journal,* vol. 5, no. 6, pp. 4428-4440, 2018, doi: 10.1109/jiot.2018.2815038.

[117] E. Baccelli, O. Hahm, M. Wählisch, M. Günes, and T. Schmidt, "Riot: One os to rule them all in the iot," 2012. [Online]. Available: https://inria.hal.science/hal-00768685v1/document.

[118] T. Reusing, "Comparison of operating systems tinyos and contiki," *Sens. Nodes-Oper. Netw. Appl.(SN),* vol. 7, p. 7, 2012, doi: 10.2313/NET-2012-08-2_02.

[119] A. Milinković, S. Milinković, and L. Lazić. "Choosing the right rtos for iot platform." https://infoteh.etf.ues.rs.ba/zbornik/2015/radovi/RSS-2/RSS-2-2.pdf (accessed 15th Jan, 2025).

[120] A. Antony and S. Sarika, "A review on iot operating systems," *Int. J. Comput. Appl,* vol. 176, pp. 33-40, 2020, doi: 10.5120/ijca2020920245.

[121] A. Hicham, A. Sabri, A. Jeghal, and H. Tairi, "A comparative study between operating systems (os) for the internet of things (iot)," *Transactions on Machine Learning and Artificial Intelligence,* vol. 5, no. 4, 2017, doi: 10.14738/tmlai.54.3192.

[122] "Riot os - an operating system for the iot: Comparison of current operating systems." RIOT-OS. http://www.riot-os.org/#nutshell (accessed 1st Spetember, 2020).

[123] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, "Operating systems for low-end devices in the internet of things: A survey," *IEEE Internet of Things Journal,* vol. 3, no. 5, pp. 720-734, 2016, doi: 10.1109/JIOT.2015.2505901.

[124] M. Lenders *et al.*, "Connecting the world of embedded mobiles: The riot approach to ubiquitous networking for the internet of things," *arXiv* 2018, doi: 10.48550/arXiv.1801.02833.

[125] Y. B. Zikria, S. W. Kim, O. Hahm, M. K. Afzal, and M. Y. Aalsalem, "Internet of things (iot) operating systems management: Opportunities, challenges, and solution," *Sensors,* vol. 19, no. 8, p. 1793, 2019, doi: 10.3390/s19081793.

[126] M. Qutqut, A. Al-Sakran, F. Almasalha, and H. Hassanein, "Comprehensive survey of the iot open source oss," *IET Wireless Sensor Systems,* vol. 8, 10/02 2018, doi: 10.1049/iet-wss.2018.5033.

[127] Contiki-NG. "Contiki-ng documentation." https://docs.contiki-ng.org/en/develop/ (accessed 15th Jan, 2025).

[128] Stanford-University. "Tinyos: Platform hardware." http://tinyos.stanford.edu/tinyos-wiki/index.php/Platform_Hardware (accessed 15th Jan, 2025).

[129] RIOT-OS. "Supported boards." RIOT-OS. https://www.riot-os.org/boards.html (accessed 15th Jan, 2025).

[130] Zephyr-project. "Supported boards and shields." https://docs.zephyrproject.org/latest/boards/index.html#supported-boards-and-shields (accessed 15th Jan, 2025).

[131] K. Klues *et al.*, "Tosthreads: Thread-safe and non-invasive preemption in tinyos," presented at the Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, Berkeley, California, 2009, doi: 10.1145/1644038.1644052.

[132] J. Dudak, G. Gaspar, S. Sedivy, and R. Budjac, "Utilization of rtos solutions in iot modules based on risc microcontrollers," in *Computer Science On-line Conference*, 2022: Springer, pp. 80-93, doi: 10.1007/978-3-031-09073-8_8.

[133] T. Watteyne *et al.*, "Openwsn: A standards ‐ based low ‐ power wireless development environment," *Transactions on Emerging Telecommunications Technologies,* vol. 23, no. 5, pp. 480-493, 2012, doi: 10.1002/ett.2558Digital Object Identifier (DOI).

[134] T. Claeys, F.-X. Molina, M. Vučinić, T. Watteyne, and E. Baccelli, "Riot and openwsn 6tisch: Happy together," in *2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN)*, 2020: IEEE, pp. 1-6, doi: 10.23919/PEMWN50727.2020.9293070.

[135] L. M. Borges, F. J. Velez, and A. S. Lebres, "Survey on the characterization and classification of wireless sensor network applications," *IEEE Communications Surveys & Tutorials,* vol. 16, no. 4, pp. 1860-1890, 2014, doi: 10.1109/COMST.2014.2320073.

[136] M. El-Aaasser and M. Ashour, "Energy aware classification for wireless sensor networks routing," in *2013 15th International Conference on Advanced Communications Technology (ICACT)*, 27-30 Jan. 2013 2013, pp. 66-71.

[137] B. Rashid and M. H. Rehmani, "Applications of wireless sensor networks for urban areas: A survey," *Journal of Network and Computer Applications,* vol. 60, pp. 192-219, 2016/01/01/ 2016, doi: 10.1016/j.jnca.2015.09.008.

[138] M. B. Marinov, I. Topalov, E. Gieva, and G. Nikolov, "Air quality monitoring in urban environments," in *2016 39th International Spring Seminar on Electronics Technology (ISSE)*, 2016: IEEE, pp. 443-448.

[139] L. Jen-Hao *et al.*, "Developed urban air quality monitoring system based on wireless sensor networks," in *2011 Fifth International Conference on Sensing Technology*, 28 Nov.-1 Dec. 2011 2011, pp. 549-554, doi: 10.1109/ICSensT.2011.6137040.

[140] M. Ferreira, R. Fernandes, H. Conceição, W. Viriyasitavat, and O. K. Tonguz, "Self-organized traffic control," presented at the Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking, Chicago, Illinois, USA, 2010, doi: 10.1145/1860058.1860077. [Online]. Available: https://doi.org/10.1145/1860058.1860077.

[141] V. Hartkopf and V. Loftness, "The robert l. Preger intelligent workplace a transformative living laboratory at carnegie mellon university," in *Facilities @ management*, 2024, pp. 465-485, doi: https://doi.org/10.1002/9781394213313.ch56.

[142] M. Magno, T. Polonelli, L. Benini, and E. Popovici, "A low cost, highly scalable wireless sensor network solution to achieve smart led light control for green buildings," *IEEE Sensors Journal,* vol. 15, no. 5, pp. 2963-2973, 2015, doi: 10.1109/JSEN.2014.2383996.

[143] J. Botero-Valencia, L. Castano-Londono, D. Marquez-Viloria, and M. Rico-Garcia, "Data reduction in a low-cost environmental monitoring system based on lora for wsn," *IEEE Internet of Things Journal,* vol. 6, no. 2, pp. 3024-3030, 2018, doi: 10.1109/JIOT.2018.2878528.

[144] C. M. University. "Intelligent workplace." https://www.cmu.edu/homepage/innovation/2007/spring/intelligent-workplace.shtml (accessed 10th Spet., 2024).

[145] K. Lam and V. Srivastava, "Living in the intelligent workplace structuring and managing building operation information," 10/11 2005. [Online]. Available: https://www.researchgate.net/publication/26901602_Living_in_the_Intelligent_Workplace_Structuring_and_Managing_Building_Operation_Information.

[146] V. Hartkopf *et al.*, "The robert l. Preger intelligent workplacetm: The living laboratory at carnegie mellon university," in *Creating the productive workplace*: Routledge, 2017, pp. 192-

221. [Online]. Available: https://www.taylorfrancis.com/chapters/edit/10.4324/9781315658834-13/robert-preger-intelligent-workplacetm-volker-hartkopf-vivian-loftness-azizan-aziz-khee-poh-lam-steve-lee-erica-cochran-bertrand-lasternas.

[147] D. J. A. Rustia, C. E. Lin, J.-Y. Chung, Y.-J. Zhuang, J.-C. Hsu, and T.-T. Lin, "Application of an image and environmental sensor network for automated greenhouse insect pest monitoring," *Journal of Asia-Pacific Entomology,* vol. 23, no. 1, pp. 17-28, 2020/04/01/ 2020, doi: 10.1016/j.aspen.2019.11.006.

[148] A. D. Kevin and P. J. Shannon, "Sensor web: A new instrument concept," in *Proc.SPIE*, 2001, vol. 4284, pp. 1-9, doi: 10.1117/12.426856.

[149] G. Tolle *et al.*, "A macroscope in the redwoods," presented at the Proceedings of the 3rd international conference on Embedded networked sensor systems, San Diego, California, USA, 2005, doi: 10.1145/1098918.1098925.

[150] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003, pp. 491-502, doi: 10.1145/872757.872817.

[151] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 14-27, doi: 10.1145/958491.958494.

[152] L. Selavo *et al.*, "Luster: Wireless sensor network for environmental research," presented at the Proceedings of the 5th international conference on Embedded networked sensor systems, Sydney, Australia, 2007, doi: 10.1145/1322263.1322274.

[153] M. Vlachos, C. Papathanasiou, V. Tsiakos, G. Tsimiklis, and A. Amditis, "Low-cost sensing solutions for harsh environments: Exploring wireless and satellite solutions in desert environmental monitoring," in *IGARSS 2024 - 2024 IEEE International Geoscience and Remote Sensing Symposium*, 7-12 July 2024 2024, pp. 7525-7528, doi: 10.1109/IGARSS53475.2024.10642661.

[154] N. A. A. Ali and N. A. A. Latiff, "Environmental monitoring system based on lora technology in island," in *2019 IEEE International Conference on Signals and Systems (ICSigSys)*, 16-18 July 2019 2019, pp. 160-166, doi: 10.1109/ICSIGSYS.2019.8811066.

[155] R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer, "Field testing a wireless sensor network for reactive environmental monitoring [soil moisture measurement]," in *Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004.*, 14-17 Dec. 2004 2004, pp. 7-12, doi: 10.1109/ISSNIP.2004.1417429.

[156] M. Bromage, K. Obraczka, and D. Potts, "Sea-labs: A wireless sensor network for sustained monitoring of coral reefs," in *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, Berlin, Heidelberg, I. F. Akyildiz, R. Sivakumar, E. Ekici, J. C. d. Oliveira, and J. McNair, Eds., 2007: Springer Berlin Heidelberg, pp. 1132-1135, doi: 10.1007/978-3-540-72606-7_97.

[157] K. Fleming *et al.*, "The self-organizing seismic early warning information network (sosewin)," *Seismological Research Letters,* vol. 80, no. 5, pp. 755-771, 2009, doi: 10.1785/gssrl.80.5.755.

[158] M. Picozzi *et al.*, "Applications of a low-cost, wireless, self-organising system (sosewin) to earthquake early warning and structural health monitoring," 2014, pp. 263-288, doi: 10.1007/978-3-642-12233-0_14.

[159] R. Butler *et al.*, "The global seismographic network surpasses its design goal," *Eos, Transactions American Geophysical Union,* vol. 85, no. 23, pp. 225-229, 2004, doi: 10.1029/2004EO230001.

[160] J. Berger, P. Davis, and G. Ekström, "Ambient earth noise: A survey of the global seismographic network," *Journal of Geophysical Research: Solid Earth,* vol. 109, no. B11, 2004/11/01 2004, doi: 10.1029/2004JB003408.

[161] N. O. a. A. Administration. "Dart® (deep-ocean assessment and reporting of tsunamis)." NOAA Center for Tsunami Research. https://nctr.pmel.noaa.gov/Dart/ (accessed 1st Sept., 2024).

[162] C. Meinig, S. E. Stalin, A. I. Nakamura, and H. B. Milburn, "Real-time deep-ocean tsunami measuring, monitoring, and reporting system: The noaa dart ii description and disclosure," *NOAA, Pacific Marine Environmental Laboratory (PMEL),* pp. 1-15, 2005. [Online]. Available: https://www.ndbc.noaa.gov/dart/dart_ii_description_6_4_05.pdf.

[163] X. Li, X. Cheng, P. Gong, and K. Yan, "Design and implementation of a wireless sensor network-based remote water-level monitoring system," *Sensors,* vol. 11, no. 2, pp. 1706-1720, 2011, doi: 10.3390/s110201706.

[164] B. Son and Y.-s. Her, "A design and implementation of forest-fires surveillance system based on wireless sensor networks for south korea mountains," *IJCSNS Int. J. Comput. Science and Network Security,* vol. 6, 11/30 2005. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f4d47b28584d7ecf1a9010dfd7d02c59f0bf66ba.

[165] X. li, X. Cheng, P. Gong, and K. Yan, "Design and implementation of a wireless sensor network-based remote water-level monitoring system," *Sensors (Basel, Switzerland),* vol. 11, pp. 1706-20, 12/01 2011, doi: 10.3390/s110201706.

[166] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Application-specific sensor network for environmental monitoring," *ACM Trans. Sen. Netw.,* vol. 6, no. 2, p. Article 17, 2010, doi: 10.1145/1689239.1689247.

[167] S. Anandakrishnan, S. G. Bilén, J. V. Urbina, R. G. Bock, P. G. Burkett, and J. P. Portelli, "The geopebble system: Design and implementation of a wireless sensor network of gps-enabled seismic sensors for the study of glaciers and ice sheets," *Geosciences,* vol. 12, no. 1, p. 17, 2022, doi: 10.3390/s110201706.

[168] M. J. Murphy *et al.*, "Experiences building and deploying wireless sensor nodes for the arctic tundra," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 10-13 May 2021 2021, pp. 376-385, doi: 10.1109/CCGrid51090.2021.00047.

[169] G. Schaefer, F. Ingelrest, and M. Vetterli, "Potentials of opportunistic routing in energy-constrained wireless sensor networks," in *European conference on wireless sensor networks*, 2009: Springer, Berlin, Heidelberg, pp. 118-133, doi: 10.1007/978-3-642-00224-3_8.

[170] S. Misra, M. Pavan Kumar, and M. S. Obaidat, "Connectivity preserving localized coverage algorithm for area monitoring using wireless sensor networks," *Computer Communications,* vol. 34, no. 12, pp. 1484-1496, 2011/08/02/ 2011, doi: 10.1016/j.comcom.2010.03.002.

[171] A. Rowe, R. Mangharam, and R. Rajkumar, "Rt-link: A global time-synchronized link protocol for sensor networks," *Ad Hoc Networks,* vol. 6, no. 8, pp. 1201-1220, 2008, doi: 10.1016/j.adhoc.2007.11.008.

[172] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 181-192, doi: 10.1145/958491.958513.

[173] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks," presented at the 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings., 2004, doi: 0.1109/IPDPS.2004.1303264.

[174] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings. Twenty-first annual joint conference of the IEEE computer and communications societies*, 2002, vol. 3: IEEE, pp. 1567-1576, doi: 10.1109/INFCOM.2002.1019408.

[175] T. Van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 171-180, doi: 10.1145/958491.958512.

[176] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd annual Hawaii international conference on system sciences*, 2000: IEEE, p. 10 pp. vol. 2, doi: 10.1109/HICSS.2000.926982.

[177] V. Paruchuri, S. Basavaraju, A. Durresi, R. Kannan, and S. S. Iyengar, "Random asynchronous wakeup protocol for sensor networks," in *First International Conference on Broadband Networks*, 2004: IEEE, pp. 710-717, doi: 10.1109/BROADNETS.2004.71.

[178] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space," *IEEE transactions on mobile computing,* vol. 1, no. 1, pp. 70-80, 2002, doi: 10.1109/TMC.2002.1011060.

[179] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th international conference on Embedded networked sensor systems*, 2006, pp. 307-320, doi: 10.1145/1182807.1182838.

[180] P. Guo, T. Jiang, Q. Zhang, and K. Zhang, "Sleep scheduling for critical event monitoring in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems,* vol. 23, no. 2, pp. 345-352, 2011, doi: 10.1109/TPDS.2011.165.

[181] F. Alfayez, M. Hammoudeh, and A. Abuarqoub, "A survey on mac protocols for duty-cycled wireless sensor networks," *Procedia Computer Science,* vol. 73, pp. 482-489, 2015/01/01/ 2015, doi: 10.1016/j.procs.2015.12.034.

[182] D. Ye and M. Zhang, "A self-adaptive sleep/wake-up scheduling approach for wireless sensor networks," *IEEE Transactions on Cybernetics,* vol. 48, no. 3, pp. 979-992, 2018, doi: 10.1109/TCYB.2017.2669996.

[183] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II*, 4-6 April 2005 2005, vol. 2, pp. 8-13 Vol. 2, doi: 10.1109/ITCC.2005.43.

[184] M. Knyva, D. Gailius, G. Balčiūnas, D. Pratašius, P. Kuzas, and A. Kukanauskaitė, "Iot sensor network for wild-animal detection near roads," *Sensors*, vol. 23, no. 21, p. 8929doi: 10.3390/s23218929.

[185] J. Brusey, J. Kemp, E. Gaura, R. Wilkins, and M. Allen, "Energy profiling in practical sensor networks: Identifying hidden consumers," *IEEE Sensors Journal,* vol. 16, no. 15, pp. 6072-6080, 2016, doi: 10.1109/JSEN.2016.2570420.

[186] A. A. A. Shabaneh, A. M. Ali, C. K. Ng, N. K. Noordin, A. Sali, and M. H. Yaacob, "Review of energy conservation using duty cycling schemes for ieee 802.15.4 wireless sensor network (wsn)," *Wireless Personal Communications,* vol. 77, no. 1, pp. 589-604, 2014/07/01 2014, doi: 10.1007/s11277-013-1524-y.

[187] R.-C. Wang, R.-S. Chang, and H.-C. Chao, "Internetworking between zigbee/802.15. 4 and ipv6/802.3 network," *SIGCOMM Data Communication Festival,* 2007, doi: 10.48550/arXiv.1002.1146.

[188] Microchip. *"Sam r30 ieee 802.15.4 sub-ghz system-in-package datasheet"*, datasheet. [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/70005302A.pdf.

[189] J. Lu, K. Martinez, and A. Weddell, "Understanding energy consumption in real-world e-iot systems with coap, rpl, and 6lowpan," in *Proceedings of IEMTRONICS 2024*, P. G. Bradford, S. A. Gadsden, S. K. Koul, and K. P. Ghatak, Eds., 2025 2025: Springer Nature Singapore, pp. 183-195, doi: 10.1007/978-981-97-4780-1_14.

[190] P. Musilek, P. Krömer, and T. Bartoň, "Review of nature-inspired methods for wake-up scheduling in wireless sensor networks," *Swarm and Evolutionary Computation,* vol. 25, pp. 100-118, 2015/12/01/ 2015, doi: 10.1016/j.swevo.2015.07.007.

[191] Microchip. *Samr30 xplained pro user's guide*, datasheet. [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/UserGuides/50002612A.pdf.

[192] Microchip-Technology. *I/o1 xplained pro user's guide*, datasheet. [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/UserGuides/Atmel-42078-IO1-Xplained-Pro_User-Guide.pdf.

[193] Digikey. *Ds3231 real time clock (rtc) clock timing qwiic, stemma qt platform evaluation expansion board*, Digikey. [Online]. Available: https://www.digikey.co.uk/en/products/detail/adafruit-industries-llc/5188/15189155.

[194] T. Instruments. "Cc1120 high-performance rf transceiver for narrowband systems." https://www.ti.com/lit/ds/symlink/cc1120.pdf?ts=1742531132626 (accessed 20th Mar, 2025).

[195] RIOT. "Openwsn network stack." RIOT-OS. https://doc.riot-os.org/group__pkg__openwsn.html#details (accessed 14th Sept., 2021).

[196] X. Vilajosana, K. Pister, and T. Watteyne, "Rfc 8180: Minimal ipv6 over the tsch mode of ieee 802.15. 4e (6tisch) configuration," *Request for Comments,* no. RFC8180, 2017, doi: 10.17487/RFC8180.

[197] T. Chang, M. Vučinić, X. Vilajosana, D. Dujovne, and T. Watteyne, "6tisch minimal scheduling function: Performance evaluation," *Internet Technology Letters,* vol. 3, 05/01 2020, doi: 10.1002/itl2.170.

[198] Microchip-Technology-Inc. "5.2.6 sleep." Microchip Technology Inc. https://onlinedocs.microchip.com/oxy/GUID-F3278C5F-01B5-4C1A-827C-63C822AD64C5-en-US-1/GUID-3A0873A3-84AB-4D73-8B4C-1229F8B21F6A.html (accessed 1st April, 2025).

[199] Y. Zhang, H. Huang, Q. Huang, and Y. Han, "6tisch iiot network: A review," *Computer Networks,* vol. 254, p. 110759, 2024/12/01/ 2024, doi: 10.1016/j.comnet.2024.110759.

[200] A. Y. Barnawi, G. A. Mohsen, and E. Q. Shahra, "Performance analysis of rpl protocol for data gathering applications in wireless sensor networks," *Procedia Computer Science,* vol. 151, pp. 185-193, 2019/01/01/ 2019, doi: 10.1016/j.procs.2019.04.028.

[201] Y. Jeong, S. Son, S. Lee, and B. Lee, "A total crop-diagnosis platform based on deep learning models in a natural nutrient environment," *Applied Sciences,* vol. 8, p. 1992, 10/19 2018, doi: 10.3390/app8101992.

[202] RIOT-OS. "Ipv6 neighbor discovery ndp.H file reference." RIOT-OS. https://api.riot-os.org/ndp_8h.html (accessed 1st April, 2025).

[203] R.-O. Contributors. "Gnrc_rpl.C – rpl implementation in gnrc stack." RIOT-OS. https://github.com/RIOT-OS/RIOT/blob/master/sys/net/gnrc/routing/rpl/gnrc_rpl.c (accessed 1st April, 2025).

[204] RIOT-OS. "Gcoap.H file reference." RIOT-OS. https://doc.riot-os.org/gcoap_8h.html (accessed 10th Jan, 2026).

[205] Microchip-Technology-Inc. "7.1.4 symbol period." Microchip Technology Inc. https://onlinedocs.microchip.com/oxy/GUID-68B476F7-3358-44C0-AA34-927F53677287-en-US-3/GUID-434751B1-2BAA-4DD7-8382-871E2F00DF9A.html (accessed 1st April, 2025).

[206] M. Anwar, Y. Xia, and Y. Zhan, "Tdma-based ieee 802.15.4 for low-latency deterministic control applications," *IEEE Transactions on Industrial Informatics,* vol. 12, no. 1, pp. 338-347, 2016, doi: 10.1109/TII.2015.2508719.

[207] C. Bormann and P. Hoffman, "Rfc 7049: Concise binary object representation (cbor)," *Request for Comments,* 2013, doi: 10.17487/rfc7049.