TOWARDS ENHANCED RESILIENCY AND INTEROPERABILITY OF LOWPAN-BASED SMART GRID APPLICATIONS

ERHAN BATURAY ONURAL

THESIS SUBMITTED TO THE UNIVERSITY OF OTTAWA IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
Master of Applied Science, Electrical and Computer Engineering

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
FACULTY OF ENGINEERING
University of Ottawa

uOttawa

Abstract

Effective electrification is a cornerstone of modern living and its significance is expected to intensify with the ongoing urbanization and densification of residential areas. In particular, the electrification of the residential sector has emerged as a critical challenge, as recent North American construction trends show a steady shift toward high density, multi-unit developments and a decline in single family housing permits. This transition amplifies the demand for scalable, reliable and cost efficient grid infrastructure capable of supporting dense clusters of electrical loads and distributed energy resources. Achieving such transformation requires robust communication infrastructures, which form the backbone of essential services including power grid load balancing, smart meter data collection, solar energy management and Electric Vehicle (EV) charging. Ensuring the affordability, reliability and performance of these interconnected subsystems is therefore fundamental to realizing the full potential of multi-residential electrification.

Although traditional communication technologies such as Wi-Fi are mature and widely available, their deployment across relatively large or enterprise scale networks including multi-residential buildings, often proves cost prohibitive. Furthermore, Distributed Energy Resource (DER)s, smart meters and Electric Vehicle Supply Equipment (EVSE)s do not necessarily demand the high bandwidth provided by such technologies and can instead benefit from alternative paradigms that offer adequate performance at a lower cost. Narrowband communication technologies emerge as a promising alternative, presenting inherent challenges due to their lossy nature, which necessitates the implementation of efficient error correction mechanisms to maintain

reliable operation. In a continuously data driven world, selecting and tuning the appropriate error correction strategy is paramount. Given the resource constrained nature of embedded hardware, these mechanisms must also remain cost effective and computationally efficient to ensure practical deployment.

When properly configured, deployed and supported by robust error correction mechanisms, narrowband communication demonstrates substantial potential to enhance the scalability, reliability and overall practicality of electrification initiatives in multi-residential buildings. This thesis presents a holistic study on the scope of narrowband network resiliency and interoperability, addressing critical gaps in existing Low-Power Wireless Personal Area Networks (LoWPAN) implementations regarding robust connectivity and data imputation. By treating network architecture and communication aware data continuity as coupled requirements rather than independent topics, the work advances the current body of knowledge and ensures practical viability. Consequently, it offers an end-to-end evaluation framework tailored for the complex realities of multi-residential smart grid operations.

Preface

I hereby declare that I am the sole author of this thesis. I performed the majority of the design, implementation and analysis presented in this document. However, the design and implementation detailed in Section 3.3 was conducted jointly with my supervisor, Dr. Javad Fattahi. Furthermore, Dr. Fattahi provided supervision and contributed to the refinement and clarification of the concepts throughout the thesis. While the research presented in Chapter 2 and Chapter 3 has been published in the co-authored venues listed below, the material included in this thesis represents my specific contributions to those publications. The co-authors participated in other aspects of the published works not detailed in this thesis.

Part of the work described in Chapters 2 and 3 has been published in the following venues:

1. E. B. Onural, B. Jamadi, M. Bazyari, and J. Fattahi, "Real-Time AI-Driven EV Charger Load Management for Residential Electrification," in *2025 IEEE Power & Energy Society General Meeting (PESGM)*, 2025, pp. 1–5.
   DOI: 10.1109/PESGM52009.2025.11224960
2. E. B. Onural, B. Jamadi, and J. Fattahi, "Rethinking Smart Grid Mesh Network with Dual Border Routers for Asset Integration," in *2025 IEEE Electrical Power and Energy Conference (EPEC)*, Waterloo, ON, Canada, 2025, vol. 1, pp. 237–241.
   DOI: 10.1109/EPEC65543.2025.11230477

Acknowledgements

This work was supported by the Republic of Türkiye. I gratefully acknowledge this support.

I would like to express my deepest gratitude to my supervisor, Dr. Javad Fattahi, for his invaluable guidance, continuous support and patience throughout my master's studies. His technical insights and encouragement were instrumental in the completion of this research.

I am also grateful to the members of the examination committee for agreeing to serve on this committee and for their time reviewing this thesis.

I also thank my mother, father and my brother for always being by my side and for their unwavering support throughout this journey.

Finally, I would like to express my heartfelt gratitude to my wife, İdil, for her invaluable support, endless understanding and compassion.

Contents

List of Figures

List of Tables

List of Abbreviations

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks

AC Alternating Current
ACK Acknowledgment
AMI Advanced Metering Infrastructure
ANSI American National Standards Institute
API Application Programming Interface
ARIMA Auto-Regressive Integrated Moving Average

BACnet Building Automation and Control Networks
BR Border Router

CAN Controller Area Network
CoAP Constrained Application Protocol
Contiki-NG Contiki Next Generation
COSEM Companion Specification for Energy Metering
CPU Central Processing Unit
CSMA Carrier Sense Multiple Access
CSMA-CA Carrier Sense Multiple Access with Collision Avoidance
CSV Comma-Separated Values

DAO Destination Advertisement Object
DC Direct Current
DCC Data Communications Company
DER Distributed Energy Resource
DFS Dynamic Frequency Selection
DIO DODAG Information Object
DIS DODAG Information Solicitation
DLMS Device Language Message Specification
DODAG Destination Oriented Directed Acyclic Graph

EV Electric Vehicle
EVLMS Electric Vehicle Load Management System
EVSE Electric Vehicle Supply Equipment

FAN Field Area Network

FCS Fully Conditional Specification
FSK Frequency Shift Keying

GAIN Generative Adversarial Imputation Network

HAN Home Area Network
HTTP Hypertext Transfer Protocol
HVAC Heating, Ventilation, and Air Conditioning

ICMPv6 Internet Control Message Protocol version 6
IoT Internet of Things
IP Internet Protocol
IPv4 Internet Protocol Version 4
IPv6 Internet Protocol Version 6
ISM Industrial, Scientific, and Medical

JSON JavaScript Object Notation

LOADng Lightweight On-demand Ad hoc Distance-vector Next Generation
LoRa Long Range
LoWPAN Low-Power Wireless Personal Area Networks
LwM2M Lightweight Machine-to-Machine

MAC Medium Access Control
MICE Multivariate Imputation by Chained Equations
Modbus Modicon Bus
MQTT Message Queuing Telemetry Transport
MQTT-SN Message Queuing Telemetry Transport for Sensor Networks
MRHOF Minimum Rank with Hysteresis Objective Function
MSE Mean Square Error
MTU Maximum Transmission Unit

NB-IoT Narrowband Internet of Things

OCPP Open Charge Point Protocol
OF0 Objective Function Zero
OFDM Orthogonal Frequency Division Multiplexing
OQPSK Offset Quadrature Phase-Shift Keying
OS Operating System

PHY Physical Layer

PLC Power Line Communication
PRIME Powerline Related Intelligent Metering Evolution
PV Photovoltaic

QEMU Quick Emulator

RF Radio Frequency
RMSE Root Mean Square Error
RPL Routing Protocol for Low Power and Lossy Networks
RS-485 Recommended Standard 485
RTO Retransmission Timeout

SEEOF Secure and Energy Efficient Objective Function
SLIP Serial Line Internet Protocol
SRAM Static Random Access Memory
sub-GHz Sub-Gigahertz
SYN Synchronize
SYN-ACK Synchronize Acknowledgment

TCP Transmission Control Protocol
TUN Network Tunnel

UART Universal Asynchronous Receiver-Transmitter
UDP User Datagram Protocol

VLAN Virtual Local Area Network
VPN Virtual Private Network

Wi-SUN Wireless Smart Utility Network

Motivation

Electrification is a broad and multifaceted subject that intersects with many sectors. From meeting the 1.5 degree target outlined in the Paris Agreement to improving energy efficiency and supporting sustainable technologies, it is clear that many domains stand to benefit from effective electrification strategies. Its wide scope across technologies, stakeholders, and operating conditions, however, also contributes to its complexity in planning, coordination, and implementation. As systems are designed, limitations in cost, planning, or foresight often prevent the creation of solutions that are holistic in nature. This creates gaps in end to end system design, misunderstandings on roles, responsibilities, and technical requirements between parties, and fragmented approaches for deployment, monitoring, and long term operation that hinder progress of reliable and scalable electrification.

This work focused specifically on smart grid systems and distributed energy resources, with particular attention to their deployment in multi residential buildings. Within this domain and to make coordinated, building level electrification practical, certain elements such as flexible loads, variable generation, user behaviour, and device level communication appear repeatedly. It examined how the rapidly growing population of electric vehicles interacts with the power system, how photovoltaic units communicate and how their generation patterns influence building level balancing, and how heat pumps contribute to overall demand. It also studied daily consumption behaviour to understand when and how people tend to use electricity. The objective was to develop models that could coordinate these different components to achieve balanced and efficient power use, since uncoordinated operation can

increase peaks, create local constraints, and reduce overall efficiency. Through this process it became clear that reliable, predictable, and continuous measurement of consumption data is the most critical requirement for any such system, because coordination and control depend on accurate visibility of how demand and generation evolve over time.

This understanding revealed a deeper challenge. Devices from different manufacturers rely on different standards and protocols, and even at the network layer their approaches can diverge. This fragmentation leads to integration overhead, inconsistent data availability, and limited ability to scale solutions across sites, which in turn undermines coordination and reliability. A review of the literature showed that many researchers had reached similar conclusions. From this, the view emerged that electrification goals can only be met if distributed energy resources are integrated into the system in a way that reflects their dynamic nature and supports seamless operation within a cohesive communication framework, so that components can exchange data consistently and respond to changing conditions without custom, site specific reengineering.

For this reason the problem was approached from a networking perspective. By designing a communication network that allows smart grid elements to interact through a shared substrate, the work sought to extend the existing utility communication environment and address a specific but important interoperability gap, namely the ability of heterogeneous devices to communicate reliably within a common structure. Electrification is a broad challenge that cannot be resolved by a single study, yet within this focused area the aim was to advance the literature as far

as possible within the scope of this degree. To support reproducibility and follow on research a software framework was developed that encapsulates the implementations and makes them accessible for future work. The results of this research have been disseminated through relevant conferences.

This thesis contributes a communication focused approach to improving interoperability for building level electrification in multi-residential settings. By addressing the reliability and consistency of data exchange across heterogeneous devices, it supports scalable coordination of distributed energy resources without repeated site specific integration. The accompanying software framework is intended to enable reproducibility and provide a platform for future work on coordinated control, monitoring, and operations in increasingly electrified buildings.

1. Introduction

Today, electrification has become a matter of increasing importance. Nearly all the devices we use or the services we receive either operate directly on electricity or depend on systems that do [1]. Accordingly, it has become important to recognize that many future devices and services will also rely on electrical infrastructure. In order to plan with this awareness in mind, electrical systems should be designed with careful consideration of future needs.

Particularly when we think about our living spaces, we spend much of our time at home and constantly interact with various electrical systems in and around our residences. Given that a significant portion of newly approved developments in Canada are multi-unit residential buildings [2], these types of dwellings involve electrical infrastructures that must be carefully planned for both the present and the future. Even today, the integration of new technologies occurs rapidly, and in many cases, electrical infrastructures that were not designed with foresight may encounter difficulties [3]. For instance, high power consuming systems such as heat pumps and EV charging infrastructure can present integration challenges if the electrical design did not account for them in advance [4].

In this context, initiatives aimed at carbon neutrality or emission reduction often remain at the conceptual stage. While electrification offers numerous opportunities and advancements essential for societal progress, its multidisciplinary and large scale nature is not always fully appreciated. When we examine multi-unit residential settings, the first issue that stands out is infrastructural inadequacy. If panels and related feeder configurations were not designed with these requirements in mind, achieving electrification goals may become impractical even in newly constructed buildings [5]. In addition to the infras-

tructural requirements, ensuring that electrification efforts are carried out in a cost effective and rational manner is also crucial [6]. The mere feasibility of a technological solution does not necessarily imply its practical implementability or user acceptance. Therefore, developing comprehensive, well reasoned and user adoptable solutions has become an important requirement.

In recent years, the rapidly increasing number of EVs and their charging requirements have become one of the most challenging issues, particularly for multi-residential buildings. Within the broader scope of electrification, EVs introduce numerous technical and infrastructural challenges due to the fast changing nature of their adoption and the still developing state of the supporting electrical infrastructure [7].

The charging process of EVs is governed by the Open Charge Point Protocol (OCPP), which requires continuous communication with a central server [8, 9]. This approach transforms charging stations from standalone units into dependent devices that must maintain constant connectivity. Although such persistent communication may be a reasonable expectation in modern networking environments, it also necessitates that multi-residential buildings provide adequate communication infrastructure to support this requirement.

While standard Wi-Fi or Ethernet solutions are typically the first options considered, they can involve high equipment costs, particularly when covering large areas [10]. At this point, mesh networks offer an alternative, as their relay-based structure can simplify deployment [11]. Although mesh technology is not as widely used as conventional solutions, it can effectively address specific, solution oriented needs.

In particular, wireless mesh networks stand out as a communication infrastructure option for multi-residential environments because they combine the ease of wireless setup with the ability to cover larger areas using fewer devices. Wireless mesh networks are expected to integrate increasingly into many versatile areas and technologies [12]. Despite these advantages, the tradeoffs typically involve reduced bandwidth and increased latency [13]. However, given the communication requirements of EV charging systems, these limitations are often acceptable.

One of the key considerations in wireless communication is determining the frequency band over which communication will occur. Since each frequency band offers distinct physical advantages and disadvantages, it is unreasonable to define a universally optimal choice. Instead, the recommended solution should be selected based on the specific requirements of the problem at hand [14].

Given that EV charging stations communicate through OCPP and that the transmitted data primarily consists of textual information, Sub-Gigahertz (sub-GHz) communication appears to be a suitable choice. Its strong signal penetration capability and adequate data rate make it particularly well suited for this type of application [15].

In wireless communication, signal losses can increase significantly [16], making it essential to incorporate mechanisms that minimize their impact. In systems designed with the assumption that data from sensors or similar sources will be received within a specific time interval, the absence of data during that interval can directly affect system stability. This issue becomes particularly critical in systems that make dynamic, data driven decisions

[17]. An example of such a system can be seen in the dynamic power balancing of charging stations connected to electrical panels within multi-residential buildings. Data imputation is an established technique when applied properly to mitigate data loss for these type of systems [18]. In this context, since each system has distinct operational requirements, different data imputation algorithms may yield varying results [19]. Therefore, analyzing and selecting the most appropriate imputation algorithm for the problem at hand is a crucial consideration in overall system design.

This thesis examines the feasibility of achieving electrification objectives in multi-residential buildings. It highlights that electrification efforts inherently depend on reliable and scalable communication infrastructures and therefore places particular emphasis on identifying and analyzing the associated networking requirements. The thesis is organized into clearly defined sections to improve clarity and readability and to facilitate a structured presentation of the proposed analysis and findings. As an introduction, Chapter 1 describes the topics covered and includes subsections that contain literature reviews corresponding to the upcoming chapters. The thesis provides a systematic way to design wireless mesh networks and further evaluates their potential in terms of installation complexity and performance in Chapter 2. The effects of potential data losses in wireless networks on such systems are discussed and various data imputation algorithms are examined in Chapter 3. To assess how these wireless mesh networks might perform dynamically in real-world scenarios, a test environment is presented along with detailed descriptions of its setup and operation in Chapter 4. An overall outcome and closure is given in Chapter 5.

1.1 Background and Literature Review

The rapid growth of building electrification and DERs has increased the dependence on reliable smart grid operation along with coupled communication needs. To this end, the review first establishes the communication infrastructure that enables building-scale connectivity, then considers how missing telemetry is handled when that infrastructure is imperfect, and finally discuss the toolchain used to study these mechanisms under controlled conditions.

Table 1.1: Comparison of wireless frequency bands

Property	**sub-GHz Narrowband**	**2.4 GHz ISM Band / 5–6 GHz Broadband**
Typical frequency range	• 470–510 MHz (APAC) • 780 MHz (CN) • 868 MHz (EU) • 902–928 MHz (NA)	• 2.4–2.48 GHz (ISM) • 5.15–5.85 GHz (UNII) • 5.9–7.1 GHz (Wi-Fi 6E/7)
Propagation characteristics	• Excellent wall penetration • Long range connectivity • High reliability in dense buildings	• Medium (2.4 GHz) to poor (5–6 GHz) penetration • Shorter indoor range • Performs well only with line-of-sight
Typical data rate	• 10–300 kbps • Narrowband optimized	• 100–1000+ Mbps (Wi-Fi 5/6/7) • Wideband operation
Energy consumption	• Very low power • Suitable for battery devices	• Moderate (2.4 GHz) • High (5–6 GHz) • Not suitable for constrained devices
Key limitations	• Limited bandwidth • Low data rate	• Weak penetration at 5–6 GHz • High interference in 2.4 GHz • Higher power consumption

1.1.1 Narrowband mesh networks for smart grid applications

Wireless mesh networks are widely used in smart grid systems due to their advantages, and these benefits extend to other application domains [20, 21]. One of the most critical factors in wireless network design is the choice of frequency band, since each band provides different advantages and limitations based on its physical characteristics [22]. In the context of Internet of Things (IoT) and smart grid communication, bands can be grouped into three main categories: sub-GHz narrowband frequencies, the 2.4 GHz Industrial, Scientific, and Medical (ISM) band, and the 5–6 GHz broadband range [23, 24]. Table 1.1 presents a comparison of these three bands in terms of their usage contexts and characteristic properties. To motivate the emphasis on narrowband mesh in later sections, the discussion first contrasts these bands in terms of capacity, coverage, and suitability for multi hop operation.

The 5–6 GHz bands provide wide channel capacity in wireless communication and are known as the frequency ranges used by Wi-Fi 5, Wi-Fi 6, and Wi-Fi 7. These bands support channel widths starting from 20 MHz and extending to 40, 80, 160, and even 320 MHz, which enables high data rates, low latency, and increased capacity under heavy traffic [23, 25, 26]. As a result, they form the foundation of Wi-Fi access solutions in environments that demand high bandwidth such as apartments, offices, and campus networks [27].

Although they offer substantial throughput, the 5–6 GHz bands also come with important limitations due to their physical characteristics. Because the wavelength is shorter

at these frequencies, penetration through walls, concrete, and metal is significantly weaker compared to 2.4 GHz [28]. The higher absorption and reflection lead to reduced indoor coverage and require a larger number of access points to serve the same area [29]. In addition, certain outdoor segments require Dynamic Frequency Selection (DFS), meaning channels can be restricted or changed due to radar detection rules [30].

In the smart grid context, the 5–6 GHz bands are generally not considered suitable for endpoint devices or low power sensors. These Physical Layer (PHY)s consume more energy, provide shorter range, show weak penetration, and are less capable of maintaining stable multi hop connectivity in mesh topologies [22]. For this reason, smart grid devices such as smart meters, inverters and EVSEs do not use 5 GHz mesh directly [20]. Instead, these bands are used mostly on the gateway side for backhaul connectivity or for high capacity indoor access. In summary, the 5–6 GHz bands offer strong advantages in throughput and capacity but are not appropriate for narrowband IoT or smart grid applications that require low power operation, long range, and robust penetration [23]. This behavior is not exclusive to the smart grid domain. It is also observed in other wireless systems, since the inherent nature of high frequency microwave propagation causes penetration to decrease as wavelength decreases [31].

Between the broadband 5–6 GHz range and sub-GHz narrowband options, the 2.4 GHz ISM band is often viewed as a compromise: it benefits from wide ecosystem support, but it remains more constrained by interference and indoor attenuation than sub-GHz links. The 2.4 GHz ISM band is one of the most widely used frequency ranges in both the smart grid literature and the broader IoT ecosystem [20, 21, 32]. Since IEEE 802.11 (Wi-Fi),

IEEE 802.15.4 (Zigbee, Thread), Bluetooth Low Energy, and several proprietary Radio Frequency (RF) protocols operate in this band, the ecosystem and chipset diversity are extensive [33]. This broad availability lowers hardware cost and improves accessibility.

From a physical perspective, the 2.4 GHz band offers weaker penetration compared to sub-GHz frequencies [28]. Concrete, brick, and metal surfaces attenuate the signal more quickly and realistic indoor coverage is typically within 10 to 30 meters depending on the building structure [34]. One of the main drawbacks of 2.4 GHz is interference caused by heavy usage. The interference in such wireless networks must be managed effectively. Frequency selection, channel sharing and power allocation play an essential role in this process [35–37]. Wi-Fi routers, Bluetooth devices, microwave ovens, wireless headphones, and many other consumer electronics operate in this band, meaning hundreds of devices may share the same channel in a single environment. This leads to collisions, latency, and retransmissions, which are especially problematic for mesh networks [38]. When IEEE 802.11 and IEEE 802.15.4 coexist in the same space, it is well documented that 802.15.4-based systems such as Zigbee, Thread, and RPL over 802.15.4 experience significant performance degradation under Wi-Fi traffic [33, 39].

In the smart grid context, the 2.4 GHz band is mainly suitable for short range Home Area Network (HAN) connections within apartments, small sensor networks, and low volume telemetry traffic. However, due to penetration limitations and interference related reliability issues, it is not as commonly used as sub-GHz mesh profiles such as 802.15.4g and Wireless Smart Utility Network (Wi-SUN) Field Area Network (FAN) [22]. In addition, the energy consumption of 2.4 GHz modules is generally higher than that of sub-GHz

narrowband PHYs, which makes them less desirable for battery powered meters or inverter telemetry applications [40].

These limitations motivate the use of sub-GHz narrowband mesh in dense built environments, where link budget, wall penetration, and multi hop stability tend to dominate over raw throughput. The sub-GHz band encompasses all unlicensed and regionally permitted frequencies below 1 GHz and is critically important for IoT and smart grid applications that require long range, strong penetration, and low power consumption [20, 41, 42]. Common examples include 868 MHz in Europe, 902–928 MHz in North America, 780 MHz in China, and 470–510 MHz in parts of the Asia-Pacific region. These frequencies form the primary operating range for technologies such as IEEE 802.15.4g, Wi-SUN FAN, Long Range (LoRa), Sigfox, and various narrowband RF protocols [43–48].

A defining advantage of the sub-GHz band is its long wavelength, which enables significantly better propagation through walls, concrete, metal surfaces, and structural obstacles. This physical property allows reliable connectivity even in challenging environments such as multi story residential buildings, underground parking structures, large campuses, and distribution infrastructure beneath electrical transformers [49, 50]. The literature shows that sub-GHz signals can provide three to six times longer range compared to 2.4 GHz at equivalent transmit power. This extended range directly improves overall network reliability in multi hop mesh topologies [51, 52].

Because the band is narrow, its achievable data rates are relatively low. However, most smart grid communication consists of small telemetry packets, state information, and

control messages [53, 54]. As a result, the lower throughput of sub-GHz technologies is sufficient for smart meters, inverters, EVSE units, and other distribution equipment. The narrowband structure also enables far more energy efficient communication, which is a critical advantage for battery powered devices and long lived field equipment [55].

In the smart grid domain, sub-GHz has effectively become the standard physical layer for mesh based Advanced Metering Infrastructure (AMI) networks [56]. Internet Protocol Version 6 (IPv6)-based solutions such as Wi-SUN FAN combined with IEEE 802.15.4g PHY can support large scale mesh deployments [57, 58]. sub-GHz is also one of the most suitable physical layers for Routing Protocol for Low Power and Lossy Networks (RPL)-based multi hop routing, since its longer and more stable links allow the formation of deeper Destination Oriented Directed Acyclic Graph (DODAG) structures [59].

Overall, each band has unique advantages and use cases in this domain. The appropriate technology should be selected based on application specific requirements. Different frequency bands support different network scales and use cases. Having established why sub-GHz links are preferred for these deployments, the next step is to compare the main sub-GHz protocol families that are commonly considered for smart grid telemetry and control.

sub-GHz protocols have been used across a wide range of applications, and the literature shows that different physical layers and protocol families evolved to satisfy distinct requirements. Over time, this has produced a diverse ecosystem of communication options with varying capabilities in terms of range, robustness, and data rate.

One of the most widely adopted standards in this space is IEEE 802.15.4g, which supports Frequency Shift Keying (FSK), Orthogonal Frequency Division Multiplexing (OFDM), and Offset Quadrature Phase-Shift Keying (OQPSK) modulation [60]. The narrowband design is well suited for smart meter telemetry because it provides low power consumption and resilience in noisy environments. Post 2020 research highlights the additional stability provided by OFDM-based modes in dense urban deployments [61]. Today, this standard forms the foundation of the Wi-SUN ecosystem and plays a central role in the standardization of sub-GHz mesh communication [62].

LoRa represents another important physical layer that uses chirp spread spectrum to achieve long range communication even at very low signal levels. The artificially expanded chirp signals allow LoRa to reach kilometer scale coverage and make it attractive for low data rate sensor networks [63]. Studies show that LoRa performs well in smart city deployments especially in environments with significant multipath effects. The tradeoff is its low data rate, which limits its suitability for applications requiring frequent control or bidirectional communication [64].

Sigfox uses an ultra narrowband approach to maximize noise immunity by restricting channel width. This technique provides long range and extremely low power operation for applications that tolerate very low data throughput. The literature notes that while Sigfox is effective for large area sensor deployments, it is inadequate for energy systems that require responsive two way control [65].

Wireless M-Bus is another sub-GHz protocol widely deployed in Europe. It uses nar-

rowband FSK together with periodic transmission intervals and is common in water and gas metering. Its support for mesh topologies is limited, however, so it has become less prominent in new IoT-driven smart grid architectures [66].

A more recent addition to this space is Wi-Fi HaLow, known as IEEE 802.11ah, which is the first Wi-Fi standard to operate in sub-GHz frequencies. HaLow is designed to provide wide coverage and low power consumption below 1 GHz. It extends Wi-Fi into the long range domain with links that can reach several hundred meters to over one kilometer [67]. Recent studies show that HaLow can handle high device density and thus has potential in smart city and industrial IoT applications. By combining long range and scalable data rates, HaLow occupies a distinctive position among sub-GHz IoT protocols [68].

One of the most mature and widely used sub-GHz solutions is the Wi-SUN FAN architecture. Wi-SUN integrates the IEEE 802.15.4g physical layer with IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN), IPv6, and RPL to provide a full multi hop mesh routing system [62]. The literature consistently reports robust performance under interference and high interoperability in multi vendor deployments [56, 69, 70]. As a result, Wi-SUN has become the de facto standard for large scale smart meter projects.

Finally, many utilities historically used proprietary narrowband FSK mesh systems that contributed significantly to early smart metering deployments. The literature shows that these systems were effective within their targeted domains but created interoperability barriers due to their closed design [71, 72]. Post 2020, these systems have been largely replaced by Internet Protocol (IP)-based solutions.

Overall, the contemporary literature demonstrates that sub-GHz protocols provide long range, low power, and stable connectivity well suited for smart grid communication. IEEE 802.15.4g and Wi-SUN are dominant in mesh based telemetry, whereas LoRa and Sigfox target low data rate sensor networks. Wireless M-Bus retains importance in regional metering, and Wi-Fi HaLow is emerging as a new long range IP layer.

This diversity indicates that sub-GHz communication will continue to play a critical role in energy systems, DER integration, and large scale IoT networks. In practice, the major protocol families occupy different points in the design space, trading off range, data rate, power consumption and deployment model. However, selecting a PHY and Medium Access Control (MAC) is only part of the story: achieving interoperability and end-to-end reachability in multi hop deployments typically requires an IP adaptation layer and a consistent routing framework.

6LoWPAN-based IP communication is one of the central mechanisms that enables low power, multi hop wireless infrastructures in smart grid applications to integrate with the broader IP environment. LoWPAN networks built on IEEE 802.15.4 operate under strict memory, processing, and frame size constraints, yet smart meters, field controllers, and DERs are increasingly expected to communicate directly through IPv6. The literature therefore treats 6LoWPAN not merely as an additional protocol but as a structural element that provides flexibility, scalability, and interoperability across smart grid communication systems [73, 74].

The adaptation layer lies at the core of the 6LoWPAN architecture. It converts IPv6

packets into forms that can be carried over IEEE 802.15.4 frames and applies header compression or fragmentation when required. In the reverse direction, it reassembles incoming fragments and exposes them to the IPv6 stack as standard packets. Through this mechanism, LoWPAN nodes can function as IP endpoints, which allows smart grid applications to reach meters, inverters, or charging stations directly through IPv6 addresses [74, 75].

Header compression is one of the most significant features of 6LoWPAN because the default 40 byte IPv6 header poses a substantial overhead within constrained frame sizes. When User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) headers are added, the overhead grows even further and can dominate the payload of small application messages. 6LoWPAN compression reduces IPv6 and UDP headers to only a few bytes by exploiting the fact that most address information can be inferred locally and that many fields contain fixed or repeated values [76, 77]. Research shows that this reduction increases effective throughput in smart meter networks, where periodic and small telemetry packets are dominant. It also decreases energy consumption by lowering the number of radio transmissions. Reduced header size further helps to limit collisions and retransmissions [78–80].

Fragmentation and reassembly are necessary to support the minimum IPv6 packet size requirement of 1280 bytes. Since IEEE 802.15.4 frames can carry only limited payloads, a large IPv6 packet must be divided into several LoWPAN fragments, each with its own identifier and offset. The receiver collects these fragments and reconstructs the original packet. The literature notes that fragmentation has a dual effect. It enables richer application messages to be transported, yet increases the probability of loss in multi hop

networks and elevates retransmission cost [81–83]. As a result, smart grid applications are commonly designed with small payloads to avoid or minimize fragmentation [84, 85].

Mesh-under and route-over represent two forwarding models for multi hop communication within 6LoWPAN networks. In mesh-under, forwarding decisions are made inside the adaptation layer. The IPv6 stack treats the LoWPAN as a single link and all hops are hidden from the IP layer. In route-over, each node operates as a full IPv6 router and forwarding takes place at the IP layer. The literature highlights that route-over aligns naturally with IP-based routing protocols such as RPL and provides better interoperability in multi vendor environments [82, 86]. Mesh-under has been used in some proprietary systems for link level optimization, but in smart grid contexts the dominant trend is toward route-over and open IP-based standards [84, 87, 88].

Standard IPv6 neighbor discovery is unsuitable for LoWPAN environments due to its broadcast oriented messages and large headers. These characteristics lead to unnecessary transmissions, increased energy consumption, and substantial radio congestion. 6LoWPAN introduces neighbor discovery optimizations that restrict broadcast usage and maintain neighbor information in more efficient ways. Research shows that reducing neighbor discovery traffic is critical not only for energy efficiency but also for DODAG stability. Excessive neighbor discovery messages increase collision rates in noisy sub-GHz environments and trigger frequent reconvergence events in RPL networks [89, 90].

Together, these mechanisms have a direct impact on resilience and interoperability in smart grid deployments. The adaptation layer, header compression, fragmentation behav-

ior, forwarding model, and neighbor discovery optimizations allow meters, field controllers, and DERs to be integrated into a scalable IP-based mesh network.

Smart electrical meters are central components of modern smart grids. Advanced Metering Infrastructure systems typically use RF mesh or power line communication to connect meters [91, 92]. Meter communication standards differ by region. In North America, RF mesh solutions based on IPv6 and RPL or proprietary protocols are common. In parts of Europe, systems often rely on Power Line Communication (PLC) technologies such as Powerline Related Intelligent Metering Evolution (PRIME) or G3-PLC or on wireless M-Bus [66, 72].

Upper layer protocols also vary. American National Standards Institute (ANSI) C12.22 and Device Language Message Specification (DLMS)/Companion Specification for Energy Metering (COSEM) are widely used to structure metering data [93, 94]. Interoperability has long been a challenge because many deployments locked utilities into closed ecosystems tied to specific vendors. Standard organizations such as IEEE and IEC have worked toward common data models and interfaces including IEEE 1701 and 1702 and IEC 62056. These efforts have helped, yet practical deployments remain diverse [95, 96].

Since 2020, the trend has shifted toward IP-based communication for meters. Solutions that use 6LoWPAN and IPv6 together with common data schemas allow devices from different manufacturers to operate on the same mesh network. This direction can be seen in the Wi-SUN profile used by many vendors and in regulatory requirements that mandate open standards in AMI systems [56]. In North America, similar initiatives have

gained momentum. The United States has adopted open communication requirements through standards such as IEEE 1547.1, which drive utilities toward more uniform and IP compatible interfaces, while several large scale AMI deployments now use RF mesh with IPv6-based routing [97]. In Canada, provincial utilities have pursued comparable strategies, particularly in regions where large AMI rollouts rely on multi vendor meter fleets that must interoperate on the same network. France uses an open PLC protocol in its Linky program, while the United Kingdom employs a Zigbee based home network with a standardized Data Communications Company (DCC) backend [98–100]. Although these technologies differ, all aim to increase interoperability across diverse metering infrastructures.

DERs include elements such as photovoltaic systems, battery storage systems, heat pumps, and EVs, and many different devices that act as flexible load or storage [101]. In contrast to smart meters, DER devices have developed in much more fragmented markets. Many DER devices rely on communication interfaces defined by individual manufacturers or industry groups. This has led to a complex set of protocols [102].

Solar inverters often use Modicon Bus (Modbus) over Recommended Standard 485 (RS-485) or TCP/IP, yet data registers are vendor specific. The SunSpec Alliance introduced a standardized data model over Modbus, but adoption has been uneven [103]. Heat pumps and Heating, Ventilation, and Air Conditioning (HVAC) units may use Building Automation and Control Networks (BACnet) or proprietary home automation protocols [104]. EV charging stations communicate with backend systems via OCPP [8, 9]. OCPP requires continuous communication; therefore, connectivity and integration with the network are critical. As of 2025, there is no unified interface that covers all DER types. Each class of

DER generally relies on its own set of standards, which limits seamless integration [105].

Regulators and standard organizations have increased their efforts to improve DER interoperability after 2020. IEEE 1547-2018 and its 2020 update require DER devices to support standardized communication functions [97]. This requirement has encouraged adoption of IEEE 2030.5, also known as Smart Energy Profile 2.0 [106]. Despite these advances, many DER units installed in earlier years do not support these newer interfaces. Additional approaches are therefore required to integrate legacy DER devices into modern networks. Recent studies have examined IoT-based gateways that translate inverter data from Modbus or Controller Area Network (CAN) into IP packets so that DER telemetry can travel over the network used by smart meters [107]. Other work demonstrates that EV charging coordination can be improved when chargers communicate with building energy controllers over local mesh networks or Wi-Fi [108].

These studies point to a broader structural challenge in today's smart grid deployments. Although modern AMI 2.0 frameworks emphasize the need for fully bidirectional communication between field devices and utilities, this requirement is not yet achieved in a consistent or unified manner. In many cases, DER devices may possess some form of outbound connectivity, yet their communication is routed through vendor specific cloud platforms or third party applications. As a result, utilities often receive delayed or filtered information rather than direct real-time telemetry. Integration therefore depends on a chain of intermediate services rather than a communication layer that is designed end to end with interoperability in mind. Figure 1.1 highlights the existing parallel communication paths during DER interaction and pinpoints the bottleneck for interoperability.

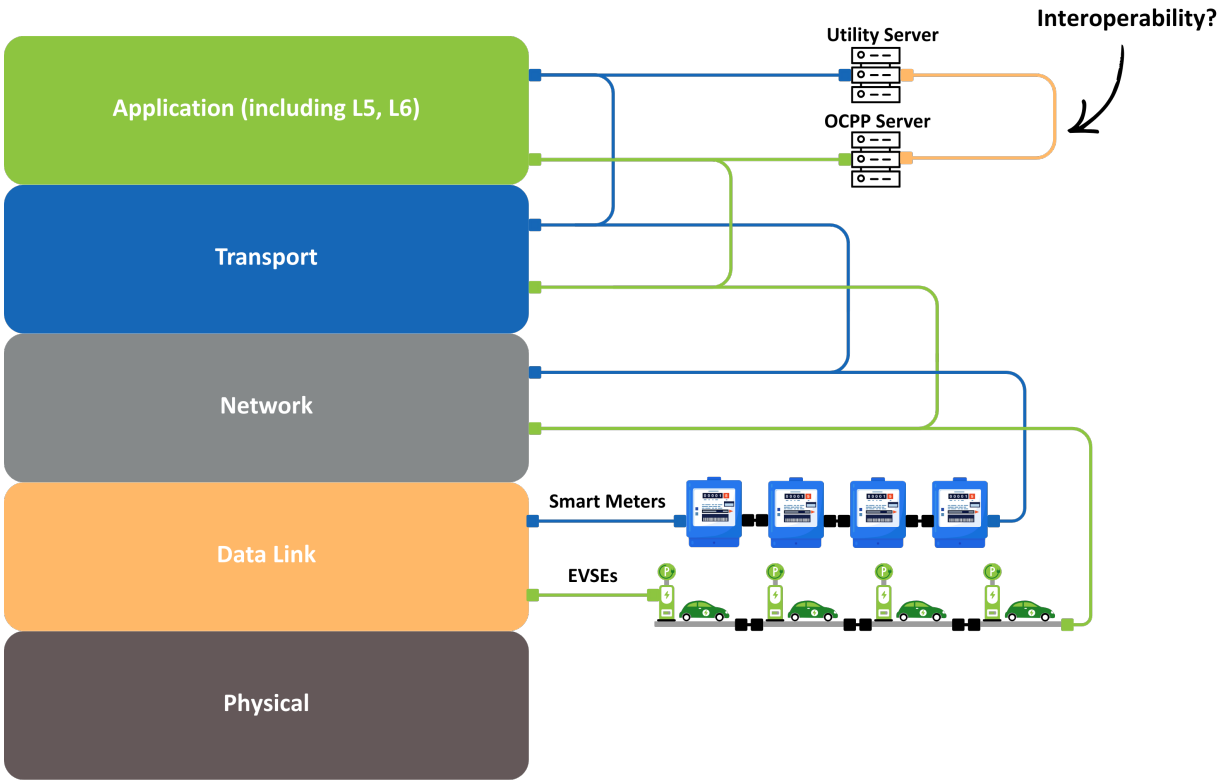

Figure 1.1: A hypothetical communication path for smart meters and EVSEs from the networking perspective.

This fragmented situation also makes coordination across mixed DER types difficult to achieve. Photovoltaic (PV) inverters, EV chargers, and storage units often operate within their own communication ecosystems, which results in parallel data paths that never converge at the network layer [109]. In many deployments, utilities receive information from these devices only through vendor specific or third party cloud platforms rather than through a direct and unified communication channel [110]. Such indirect pathways introduce additional latency because data must travel through external servers before reaching

the utility. They also raise security concerns, since sensitive operational data is exposed to external cloud infrastructures outside the utility or building domain. Moreover, each translation step and intermediate relay point increases the likelihood of data loss, stale information, or inconsistent updates, creating reliability issues across the system [111].

Even when attempting to integrate DERs, the process is typically realized through protocol translation gateways or ad hoc backend interfaces instead of a common networking infrastructure. The literature reports several prototype systems that bridge DERs using IoT gateways or local controllers, yet these solutions remain system specific and do not resolve the fundamental absence of a unified communication fabric [112, 113]. As a result, the need for a cohesive networking approach that can accommodate diverse DER interfaces and present them as part of a single, reliable, and secure operational domain remains a prominent gap in current smart grid architectures [114].

In this context, a narrowband IPv6 mesh that can serve as a shared communication layer for diverse DER devices becomes an attractive candidate. Rather than replacing existing device interfaces, it can provide a consistent transport path within the building or distribution environment, allowing utilities or energy management systems to access field level data through a single mesh based network. When designed to work as a bridge, this type of mesh can extend coverage and improve resilience, which is valuable in multi-residential buildings and other dense installations. Such an approach offers a practical step toward the type of integrated and bidirectional communication envisioned in AMI 2.0 without requiring immediate replacement of heterogeneous DER equipment. To analyze how such IP-based mesh networks behave under realistic constraints and failures, it is

common to rely on open research platforms that implement the relevant protocol stack end to end.

Contiki Next Generation (Contiki-NG) is an open source operating system designed for IoT devices operating over low power and lossy networks and has become one of the most widely used research platforms for modeling sub-GHz mesh networks. Its lightweight kernel, event driven execution model, and energy aware scheduler allow embedded devices to be simulated under realistic conditions. Because the full protocol stack from the physical layer to the application layer is openly implemented, the platform enables both the evaluation of existing standards and the design of new algorithms. This capability is especially valuable in studies related to smart meter telemetry, distributed energy resource monitoring, and building sensor networks [115].

Contiki-NG ships with an IPv6 6LoWPAN stack and standardized application protocols including Constrained Application Protocol (CoAP), Lightweight Machine-to-Machine (LwM2M) and Message Queuing Telemetry Transport (MQTT), enabling end-to-end evaluation over RPL mesh networks [116]. CoAP runs over UDP and leverages header compression and fragmentation for request-response telemetry in constrained deployments, while MQTT/Message Queuing Telemetry Transport for Sensor Networks (MQTT-SN) use publish-subscribe to aggregate data at edge brokers. These choices trade off reliability, latency and overhead, and surveys generally report CoAP/LwM2M within constrained IP networks and MQTT for wide area telemetry [117–119].

A central component of Contiki-NG is its implementation of the RPL routing protocol.

RPL is an IPv6 routing standard for low power networks and bases its routing decisions on a performance metric defined by the routing objective function. This metric may reflect link quality, packet delivery rate, or energy cost. At the core of RPL is the routing graph called the DODAG. Each node has a position in the DODAG which is expressed by a rank value derived from the chosen objective function rather than from physical distance. For example, in an energy based objective function each link is assigned a cost and the rank increases cumulatively along the selected path [116].

A node's rank does not necessarily reflect physical distance but instead represents an abstract cost derived from the routing objective. Let $R(\cdot)$ denote the RPL rank function and let $O(\cdot, \cdot)$ denote the incremental objective cost, where $R(n)$ is the rank value assigned to node n and $O(p, n)$ is the cost increment associated with selecting parent p for node n. When a node n selects a parent p, it assigns itself a rank value that increases cumulatively along the path toward the root. This relationship is expressed by

$$R(n) = R(p) + O(p, n) \tag{1.1}$$

This equation shows that each node inherits the rank of its preferred parent and augments it by a cost that reflects the quality or expense of the link between the two devices. The incremental nature of the rank ensures that the DODAG maintains an acyclic structure and that routing decisions converge toward the root through monotonic progression.

The function $O(p, n)$ captures the output of the objective function mechanism, which defines how link properties influence parent selection. This mechanism does not impose

a specific metric but instead provides a general mechanism that allows RPL to optimize for different network goals such as reliability, delay, or energy consumption. For each candidate parent, the objective function mechanism evaluates the cost of the parent–child relationship according to

$$O(p, n) = w \cdot m(p, n) \tag{1.2}$$

where $m(p, n)$ denotes the chosen link metric measured on the connection between p and n and w is a scaling factor that adjusts the contribution of this metric. Typical metrics include the expected transmission count (ETX), packet delivery rate, or per link energy cost. Minimum Rank with Hysteresis Objective Function (MRHOF) and Objective Function Zero (OF0) are widely used objective functions due to their simple nature [120]. These metrics can be configured according to the needs of the deployment since RPL does not mandate a single metric. By combining the inherited rank with the local cost evaluation, the objective function enables each node to identify the parent that minimizes its cumulative path cost while preserving the global consistency of the DODAG.

To illustrate how rank accumulates, the root is assigned $R(\text{root}) = 0$. A node n considers two candidate parents p_1 and p_2 with $R(p_1) = 256$ and $R(p_2) = 384$. The chosen link metric is ETX and the scaling factor is set to $w = 128$. The measured values are $m(p_1, n) = 1.2$ and $m(p_2, n) = 1.0$.

Using (1.2), the incremental objective costs are calculated as

$$O(p_1, n) = 128 \cdot 1.2 = 153.6 \qquad O(p_2, n) = 128 \cdot 1.0 = 128$$

Using (1.1), the resulting candidate ranks for n become

$$R_{p_1}(n) = 256 + 153.6 = 409.6 \qquad R_{p_2}(n) = 384 + 128 = 512$$

Since the cumulative rank through p_1 is smaller, the node selects p_1 as its preferred parent even though the direct link to p_2 has a slightly lower ETX. In both cases, $R(n)$ remains strictly larger than the selected parent rank, which preserves the monotonic progression toward the root.

Together, these expressions capture how RPL structures the network and guides routing. Rank places nodes relative to the root, while the objective function maps link metrics to routing cost. This design supports efficient route construction across diverse requirements in low power and lossy networks [121].

RPL relies on three control messages. DODAG Information Object (DIO) messages disseminate routing information, including rank and DODAG parameters, to help nodes select parents. DODAG Information Solicitation (DIS) messages let nodes request fresh DIO updates to join the topology faster. Destination Advertisement Object (DAO) messages establish downward routes by advertising node reachability to the border router [122]. Table 1.2 summarizes these message roles.

In Contiki-NG, RPL is implemented as a modular subsystem [116]. This implementation closely matches the routing core used in Wi-SUN networks. Wi-SUN FAN relies on the same DODAG-based multi hop routing model and the behavior of its routing logic can be simulated with high fidelity in Contiki-NG [62]. Although Wi-SUN includes additional

Table 1.2: RPL control message types, functions and directions

Message Type	Purpose	Information Carried	Direction
DIO (Information Object)	• Announces DODAG configuration • Supports join and maintenance • Supports upward route formation	• DODAG ID and version • Objective function parameters • Rank information • Trickle timer settings	• Broadcast from root downward • Propagates through the DODAG
DAO (Advertisement Object)	• Establishes downward routes • Advertises child reachability • Enables source routing when supported	• Target prefixes • Transit information • Route lifetime	• Sent upward from children to parents • Reaches the root for downward routing
DIS (Solicitation)	• Solicits DIOs from neighbors • Discovers nearby DODAGs • Useful for network bootstrap	• Optional solicitation options • Flags requesting specific DIO information	• Sent locally to neighbors • Triggers inbound DIO from receivers

security profiles, timing strategies, and channel hopping features, the fundamental routing behavior is dictated by RPL, making Contiki-NG a suitable tool for analysing Wi-SUN style networks. As a result, it is frequently used to study parent selection algorithms, rank stability, route degradation, and the effect of DIO propagation intervals on network performance [69, 123, 124].

Contiki-NG allows researchers to modify the routing core and design new objective functions. This is especially helpful for studies that target energy aware parent selection or novel routing configurations. Various topologies and custom optimization functions are proposed in the smart grid domain [116, 125]. Such setups can be modelled in Contiki-NG, and performance indicators such as stability, delay distributions, and the likelihood of routing loops can be examined. Many studies in the energy domain, including DER integration, reliability of command and control flows, route stabilization, and coverage

optimization, rely on Contiki-NG due to these strengths. This naturally leads to the question of how such networks should be evaluated when disruptions occur, since mesh deployments in the field must maintain acceptable performance under failures rather than only under nominal conditions.

Resilience has received increased attention in recent evaluations of smart grid networks, but it is often conflated with reliability and availability. Reliability reflects the probability that devices or links will operate without failure for a given period. Availability describes the proportion of time a service remains operational. Resilience, however, refers to the extent to which a network maintains service quality in the presence of failures and how quickly and predictably it can recover once disruption occurs [126, 127]. For this reason, the literature notes that assessing node or link failure rates alone is insufficient for smart grid LoWPANs. Instead, researchers emphasize resilience oriented approaches that characterize network behavior under unexpected conditions [128, 129].

Within this framework, the explicit definition of distinct failure models becomes essential. Classical node failures, link outages, and temporary interference events are inherited from traditional wireless sensor network literature, yet smart grid environments introduce additional failure scenarios. The loss of upstream gateways that connect the RPL border router to utility backends and the temporary unavailability of cloud based management services form separate dimensions of resilience [130–132]. Several studies model these events as backend loss or partial regional partitioning and examine how alternative backup strategies inside the LoWPAN can mitigate their impact. An increasing body of work argues that these failure modes should be analysed not in isolation but jointly [129, 133, 134].

A variety of performance metrics are employed to evaluate resilience quantitatively. Table 1.3 lists some of the frequently used metrics. Packet delivery ratio, end-to-end delay, reconvergence time, route stability, and the percentage of the network that remains reachable under specific failures are among the most common [134, 135]. Packet delivery ratio indicates how much data reaches its destination despite disruption and interference, while delay evaluates how quickly commands or measurements can be delivered in time sensitive applications [136]. In RPL-based networks, reconvergence time measures how long the DODAG requires to reach a new steady state once the root or a critical intermediate node is lost. Route stability reflects how often and how significantly parent selections change. Coverage ratio expresses what fraction of the network retains connectivity in a given failure scenario [137]. The literature consistently highlights that these metrics must be interpreted together. For example, a high delivery ratio accompanied by unstable routes may introduce further failures later [138, 139].

The relationship between these resilience metrics and the behavior of RPL and 6LoWPAN has been demonstrated extensively. RPL provides local and global repair mechanisms that reorganize damaged regions of the network. Local repair confines reconfiguration to the affected subtree, thereby limiting control traffic. Global repair reconstructs the entire DODAG when the root is lost or when large scale disruptions occur. On the 6LoWPAN side, fragmentation behavior and neighbor discovery optimizations influence the number of retransmissions on failing links, the likelihood of collisions, and the resulting reconvergence duration [81, 140]. The data imputation strategies examined in this thesis also align with this perspective. By reconstructing missing or delayed telemetry through learning based

Table 1.3: Resilience metrics in RPL-based smart grid networks

Metric	Primary meaning	Resilience interpretation
Packet delivery ratio (PDR)	• Fraction of packets that reach their destination • Captures impact of losses, interference and congestion	• High PDR indicates the network maintains service • A high PDR with unstable routes may still hide future disruptions
End-to-end delay	• Time from packet generation to reception at the destination • Reflects queuing, retransmissions and path length	• Critical for time sensitive control and protection traffic • Delay growth under failures reveals degraded service quality
Reconvergence time	• Time required for the DODAG to reach a new steady state • Typically measured after root or key parent failures	• Short reconvergence implies fast recovery from disruptions • Long reconvergence exposes the system to extended service gaps
Route stability	• Frequency and magnitude of parent changes in the DODAG • Captures oscillations and flapping in parent selection	• Stable routes reduce control overhead and packet reordering • Excessive churn may trigger further failures despite good PDR
Coverage ratio / Reachability	• Fraction of nodes that remain connected to the root • Often evaluated under specific failure scenarios	• Indicates the rate of operational availability • Helps compare alternative backup or multi root strategies

methods, the logical resilience of the system improves. Even when packets are lost at the physical layer, the upper layers can still operate on consistent and timely information [141–145].

Smart grid LoWPANs differ significantly from classical sensor networks due to their diverse traffic classes and quality-of-service requirements. Meter reading messages are periodic and tolerant to moderate delays, whereas voltage monitoring or emergency status signals require lower delay and higher delivery assurance. Commands sent to DERs form an even more sensitive class because they must be delivered reliably and within a specific time window [20, 73, 146]. The literature therefore discusses the use of different RPL objective functions for distinct traffic classes, queue management, and prioritization mechanisms. In some cases, multi path or backup route strategies could be utilized to protect critical traffic. The limited data rate of sub-GHz links further requires careful selection of packet size, reporting period, and retransmission parameters to satisfy these requirements [120, 147, 148].

Another important dimension involves interoperability and data models. Smart grid environments contain meters, DERs, protection equipment, and building automation devices that evolved under different standards and often cannot communicate directly. Standards such as IEC 61850 object models, DLMS/COSEM structures, and IEEE 2030.5 have attempted to harmonise this diversity by defining devices in terms of structured data and service interfaces rather than raw bit fields [149, 150]. The literature includes extensive discussion of applying such data models on top of IP and RPL-based LoWPANs, designing gateways that translate protocols at border routers, and measuring interoperability levels

in multi vendor deployments [151–153]. From this perspective, resilience encompasses not only the successful transmission of packets but also the preservation of meaningful and processable information across heterogeneous systems.

1.1.2 Data imputation in narrowband networks

In narrowband LoWPAN deployments, telemetry packets may be lost or delayed, which appears as gaps in the collected measurements and can affect monitoring and control functions. Therefore, in addition to studying the root causes of discontinuity under narrowband constraints and mitigating them, it is necessary to examine how missingness arises and forms structured patterns, and which imputation methods can reconstruct time series without increasing network cost. In narrowband networks, data continuity has deemed itself as a central challenge for the correctness of smart grid monitoring and control applications. Smart meters, DERs and other field devices typically send periodic measurement packets over sub-GHz RF based narrowband networks [20]. The limited data rate of these networks, their multi hop topologies, as well as interference and transient disconnections lead to gaps in the measurement streams that are collected in the form of time series. Maintaining a consistent information state is crucial even in cases where some of the measurements do not reach the control center or arrive with significant delay [92, 154]. Figure 1.2 illustrates such data continuity challenge, where periodic measurements from field devices experience packet losses and delayed arrivals due to narrowband network constraints.

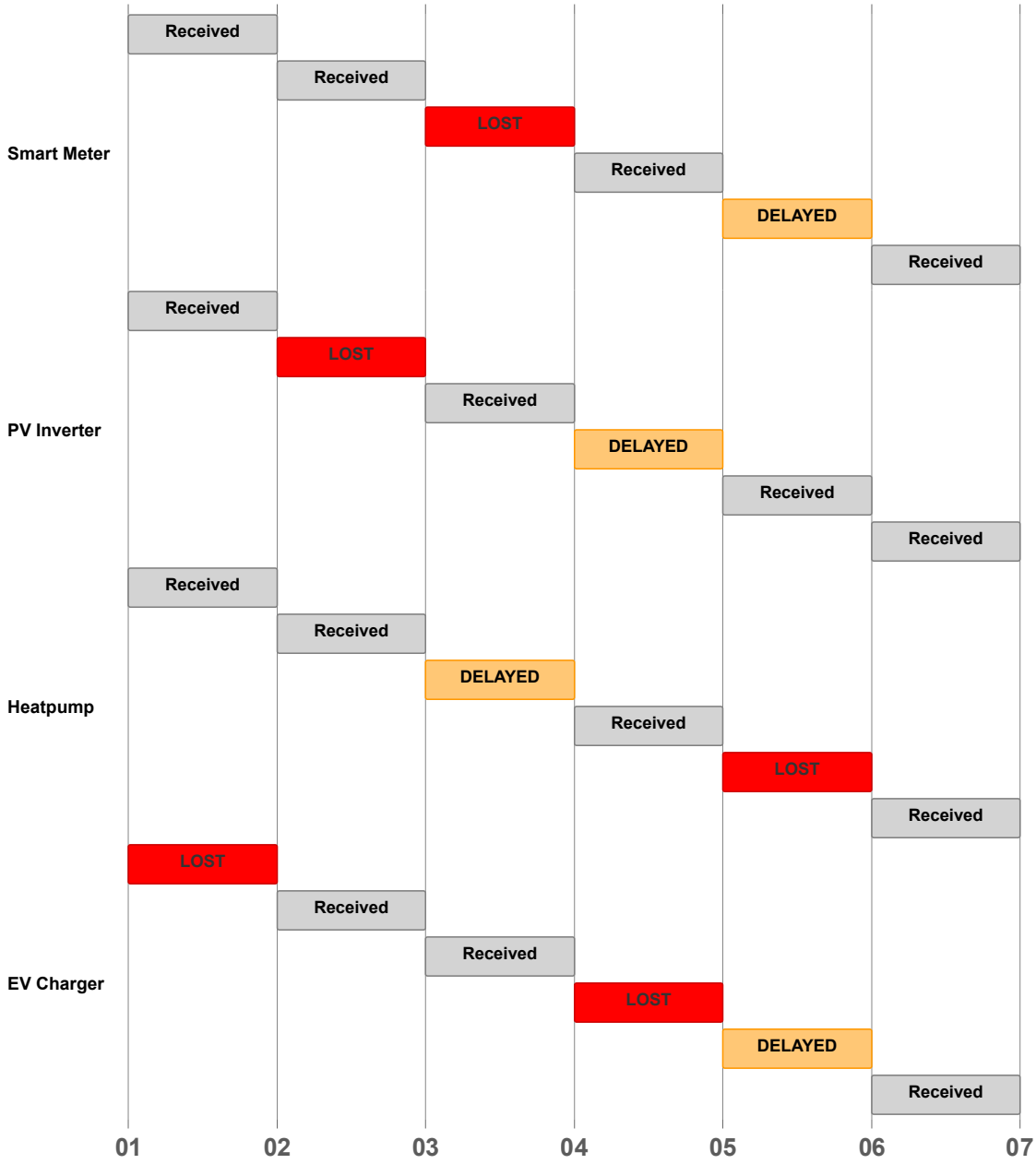

Figure 1.2: Periodic measurements from field devices experience packet losses (with red color) and delayed arrivals (with orange color) due to narrowband network constraints.

The essential feature that distinguishes the narrowband network environment from other wired or broadband wireless solutions is that aggressive retransmission for lost packets is not always feasible. Repeated transmissions decrease the already limited data rate and introduce a non negligible energy cost. For this reason, in smart grid LoWPANs, only a limited number of retransmission attempts are performed instead of continuous intensive retransmission [75, 155, 156]. This mandatory choice at the network side places classical monitoring and prediction algorithms, which rely on the assumption of complete data across all layers, in a difficult position. Consequently, recent literature has focused increasingly on data imputation methods that are designed specifically for narrowband networks [145, 157, 158].

The concept of data imputation aims to reconstruct missing or corrupted measurements using statistical or learning based models. Classical approaches rely on forward filling, linear interpolation, or simple time series models that incorporate seasonality [159, 160]. However, many studies have shown that in smart grid applications, the profiles of load and generation exhibit strong temporal dependencies, abrupt changes, and complex patterns resulting from climate conditions and user behavior [161, 162]. Therefore, more advanced approaches are employed, such as Auto-Regressive Integrated Moving Average (ARIMA), state dependent prediction, Kalman filtering, and hidden state space models. More recently, methods based on neural networks for time series completion, autoencoder based representation learning, and Generative Adversarial Imputation Network (GAIN) based synthesis have been proposed [163–165]. The common objective of these studies is to reconstruct missing measurements at the upper layer with high accuracy without in-

creasing the transmission cost in narrowband networks. Table 1.4 lists some of the data imputation approaches commonly employed.

Table 1.4: Common data imputation approaches

Approach	Methods	Characteristics
Classical	• Forward and backward filling [160] • Linear interpolation [160] • Seasonal averaging [160]	• Low complexity, real time capable • No training required • Limited accuracy for irregular patterns
Statistical and Time Series	• ARIMA [163, 166] • Kalman filtering [163, 166] • Hidden state space models [163]	• Captures temporal dependencies • Moderate computational cost • Requires parameter tuning
Deep Learning	• RNN and LSTM networks [163] • Autoencoder representations [163] • GAN based synthesis [163, 165]	• High accuracy for complex patterns • Learns climate and behavior effects • Requires large training data

Data continuity challenge shows itself in various aspects due to the variety of elements employed in the smart grid domain. The challenge often is not limited to the time series of a single meter. Many studies exploit the joint statistical structure of multiple meters or DERs that are connected in time and space [154, 167]. For example, consumption profiles of apartments fed by the same transformer exhibit certain correlations due to shared climate conditions and a common time based tariff structure [168, 169]. In the literature, multivariate time series imputation approaches use these correlations to estimate missing values both from neighboring time steps and from the data of neighboring nodes. This multidimensional perspective can provide higher accuracy and better generalization compared with simple interpolation methods that rely on a single series [170, 171].

Within the smart grid context, an additional importance of the data continuity challenge emerges from its relationship with resilience. In the traditional view, resilience is mostly measured through network metrics such as packet delivery ratio, delay, and recon-

vergence time. Recent studies emphasize that if losses at the physical layer are compensated at the upper layer through data imputation, the logical resilience of the system can be improved [172, 173]. In other words, the absence of some measurements at the physical layer may be acceptable if they can be filled in a timely and consistent manner with a suitable imputation method. Under these conditions, energy management algorithms, demand response mechanisms, and fault detection processes can maintain an acceptable quality of service [174]. This perspective leads to a new definition of resilience that jointly considers the network and data layers.

The literature has started to examine data imputation methods not only as tools for filling missing values but also as elements of a broader interoperability and modeling framework. Meters and DERs from different manufacturers may operate with different sampling periods and measurement resolutions. On a narrowband network, transforming these heterogeneous data streams into a single common time base and data structure often involves the estimation of certain missing values. In this way, data imputation becomes a fundamental mechanism that both reduces data gaps caused by the physical limitations of the network and merges heterogeneous measurement streams [175]. As a result, it provides a continuous and consistent flow of information for upper layer applications.

Missingness in narrowband smart grid data is typically structured rather than independent, and its causes span physical propagation, MAC/routing behavior, and end-to-end system factors. As commonly occurred, missing data arises first from physical layer and propagation conditions. sub-GHz signals are affected by attenuation and shadowing in indoor environments. Multi path fading can also cause consecutive losses in long mesh

chains. Interference from other unlicensed systems produces additional loss that varies with time and environment. Many studies therefore model missing data as a process that has temporal and spatial structure rather than as independent events [157, 176].

A second group of causes originates from MAC and network layer behavior. Contention based medium access, limited retransmissions, and buffer constraints at intermediate nodes lead to drops during congestion [177, 178]. When 6LoWPAN fragmentation is used, the loss of a single fragment prevents reassembly and increases the effective missing data rate. RPL networks also experience temporary route instability during parent changes or repair procedures, creating short periods of grouped losses [121, 179].

System level factors also contribute. Gateway nodes may fail to record packets during maintenance or temporary outages even when they are received on the radio side. Backend inconsistencies and time synchronization errors can further produce artificial gaps. This view highlights that missing data reflects failures along the entire end to end path rather than only at the wireless link [180–182]. A hypothetical distribution over these different causes for missing data is illustrated in Figure 1.3.

The literature also identifies some characteristic patterns for the data losses. Burst losses are common because interference often affects specific links or regions for short periods. Loss rates may also vary with daily load profiles in residential or industrial settings. Spatial correlation appears when meters in similar radio conditions, such as basements or transformer zones, experience losses at the same times. Some of these losses are inherently due to the changing levels of activity in daily life [183–185]. These patterns

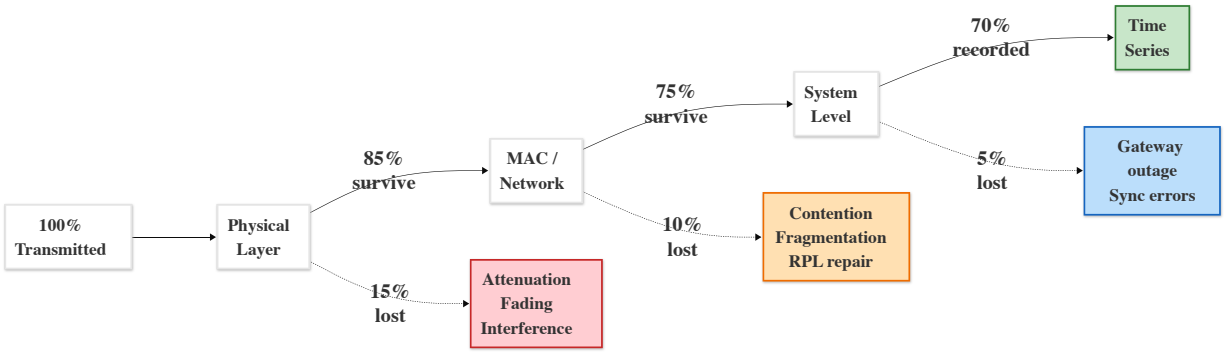

Figure 1.3: A hypothetical data loss distribution over the different causes for missing data. Note that the ratios are subject to change depending on the environment and other factors.

indicate that imputation methods should consider both temporal structure and correlations among neighboring nodes.

Missing data patterns differ by traffic class as well. Periodic meter reports usually show regular loss behavior while event based packets are sparse and more critical for diagnostics. Table 1.5 lists the temporal and spatial patterns of missing data and Table 1.6 summarizes the traffic class characteristics in smart grid networks. The measurement streams from DERs have stronger temporal variability and require more advanced estimation methods [154, 186, 187].

Table 1.5: Temporal and spatial patterns of missing data

Pattern	**Characteristic**	**Design Consideration**
Burst losses	Consecutive drops from interference	Multi step gap handling
Daily variation	Loss rates vary with load cycles	Time of day awareness
Spatial correlation	Similar RF conditions fail together	Cross node correlation

Given these missingness mechanisms, imputation methods are typically grouped into sta-

Table 1.6: Traffic class characteristics in smart grid networks

Traffic Class	**Characteristic**	**Design Consideration**
Periodic reports	Regular interval and high volume	Seasonality modeling
Event packets	Sparse but critical for diagnostics	Anomaly aware methods
DER streams	Periodic but volatile	Weather and behavior models

tistical baselines, model based estimators, and learning based approaches that trade computational cost for improved reconstruction under long or structured gaps. The simplest data imputation methods rely on traditional statistical techniques. Mean, median, or mode replacement, constant value filling, and hot deck selection are common examples. These approaches are fast and easy to apply, but the literature notes that they often distort correlation structure, reduce variance, or weaken temporal relationships in time dependent data [159, 188].

More advanced statistical and model based approaches estimate missing values by exploiting relationships in the data. Regression imputation predicts each missing variable from the others through a fitted model. Expectation maximization alternates between parameter estimation and missing value estimation within a joint probability model. Multiple imputation creates several plausible complete data sets and combines the results of downstream analysis. A widely used method in this class is Multivariate Imputation by Chained Equations (MICE), which builds conditional models for each variable with missing entries and iterates them until a stable distribution is reached [189, 190].

Classical machine learning methods use similarity structure or latent data patterns. K nearest neighbor imputation replaces a missing value with the local average of the most

similar records and has been shown to outperform simple row averages in early studies [159]. Later work introduced matrix factorization, kernel models, and clustering based techniques to better capture high dimensional structure [157]. Tree based approaches such as MissForest are also popular because they handle nonlinear relations and mixed data types [189].

Recent work increasingly focuses on deep learning based imputation. Time series models use recurrent networks, gated units, and attention based structures to capture temporal dependencies directly [191]. Other studies apply autoencoder or variational autoencoder models to learn a latent representation and reconstruct missing segments [192]. GAIN and diffusion based imputers generate multiple plausible completions and can model complex distributions [193]. Survey papers show that no single deep method performs best in all conditions and that performance depends strongly on data type, missingness level, and missingness mechanism [189, 194].

Overall, the data imputation methods are often discussed into three categories. The first consists of simple statistical and interpolation techniques. The second contains advanced statistical and model based approaches, including multiple imputation and distance driven methods. The third covers machine learning and hybrid models such as random forest, support vector methods, matrix factorization, and neural network based methods. Tables 1.7 and 1.8 gives a list of advantages and limitations among these different groups and the associated common methods.

In narrowband smart grid settings, imputation is commonly treated as an enabling step for

Table 1.7: Data imputation methods and their advantages

Category	**Method**	**Advantage**
Simple Statistical	• Mean, median, and mode [159, 188] • Constant value filling [159, 188] • Forward and backward filling [159, 188] • Linear interpolation [159, 188] • Hot deck imputation [159, 188]	• Fast to compute • Trivial to apply • Preserves recent values • Produces smooth transitions • Uses real observed values
Advanced Statistical	• Regression imputation [189, 190] • Expectation Maximization [189, 190] • Multiple Imputation and MICE [189, 190] • State space and Kalman [163]	• Models variable relationships • Handles latent structure • Quantifies imputation uncertainty • Captures temporal dynamics
Machine Learning	• K Nearest Neighbors [159, 194] • MissForest [157, 159, 194] • Matrix factorization [157, 191, 194] • RNN, LSTM, and Attention [157, 191] • Autoencoder and VAE [191, 192] • GAN and Diffusion [157, 191, 193]	• Nonparametric approach • Handles nonlinear patterns • Captures latent factors • Models temporal dependencies • Learns compact representations • Generates realistic samples

Table 1.8: Data imputation methods and their limitations

Category	**Method**	**Limitation**
Simple Statistical	• Mean, median, and mode [159, 188] • Constant value filling [159, 188] • Forward and backward filling [159, 188] • Linear interpolation [159, 188] • Hot deck imputation [159, 188]	• Distorts variance and correlation • Ignores data structure • Propagates stale data • Fails on abrupt changes • Requires similar donors
Advanced Statistical	• Regression imputation [189, 190] • Expectation Maximization [189, 190] • Multiple Imputation and MICE [189, 190] • State space and Kalman [163]	• Assumes linearity • Slow convergence • Computationally intensive • Requires parameter tuning
Machine Learning	• K Nearest Neighbors [159, 194] • MissForest [157, 159, 194] • Matrix factorization [157, 191, 194] • RNN, LSTM, and Attention [157, 191] • Autoencoder and VAE [191, 192] • GAN and Diffusion [157, 191, 193]	• Distance metric sensitive • Slow on large datasets • Assumes low rank structure • Needs large training data • Black box behavior • Training instability

downstream analytics and control, and recent work evaluates methods under missingness regimes that reflect multi hop wireless constraints. Data imputation studies in narrowband smart grid networks aim to compensate for the loss of data continuity caused by low rate and multi hop wireless communication. Technologies such as sub-GHz RF mesh, LoRa, and Narrowband Internet of Things (NB-IoT) frequently experience packet loss, extended outages, and burst gaps. For this reason, the literature treats imputation as a necessary preprocessing step before load forecasting, state estimation, anomaly detection, and demand response [154, 195]. Survey papers consistently describe missing and corrupted measurements as the weakest link in the smart grid data chain and emphasize that imputation must match the characteristics of narrowband networks [196, 197].

Early work focused on statistical and model based approaches. Consumption time series from smart meters were completed using mean, median, or linear interpolation, followed by regression based and multiple imputation methods. The practical objective was to reconstruct the load on the distribution system and obtain acceptable accuracy despite missing samples. Multivariate imputation frameworks built separate models for different consumer groups and time periods and maintained the general load shape even with high missingness. These methods provided a stable baseline but their performance dropped sharply under long consecutive gaps, which are common in narrowband networks [154, 196, 198].

Later studies shifted toward time series oriented approaches that account for the temporal and seasonal structure of load profiles. Autoregressive models, state space estimators, and multivariate regression frameworks improved performance under both random and

bursty missingness [166, 199]. Some work extracted features representing daily or weekly load patterns and filled missing segments using averages from similar profile classes. These pattern based methods outperformed simple interpolation when long outages occurred, since they explicitly modelled profile structure [200, 201].

More recent research integrates machine learning and deep learning. Denoising autoencoders learn latent representations of daily profiles and reconstruct missing values from this representation [202]. Diffusion based generative models that take historical context and load shape as input report lower error for both random and burst gaps than classical and machine learning baselines [203]. Benchmark studies comparing statistical, machine learning, and deep time series models on the same meter data show that complex deep models offer clear advantages for long gaps, although they involve a trade off between accuracy and computational cost [166, 194]. Some studies move beyond offline pre processing and integrate real-time data imputation directly into digital twins and control loops, particularly in low inertia microgrid and nanogrid settings where measurement loss immediately impacts stability and synchronization [18, 19].

Work that explicitly incorporates narrowband properties has expanded mainly in the context of NB-IoT and low power wide area deployments. These studies report prolonged outages due to connectivity constraints and characterize missing data in terms of connection period, reporting frequency, and outage patterns [204, 205]. Related work proposes two stage frameworks that first perform data cleaning and anomaly detection and subsequently complete household level time series using reference curves derived from aggregated meter data. While not narrowband specific, such frameworks are well suited to embedding struc-

tured outage patterns commonly observed in narrowband communication systems [196, 206].

A separate line of research involves working on meter and consumption data as signals on graphs. Each consumer becomes a node and edges represent physical or similarity relations. Graph signal imputation uses local message passing to fill missing values while denoising the series [207, 208]. These methods exploit both temporal and spatial correlations and can reconstruct values even when a single node experiences extended loss, using information from neighboring meters and network level constraints.

Taken together, the literature demonstrates substantial progress in imputing missing smart meter data, evolving from simple statistical baselines to advanced deep learning and graph based formulations that exploit temporal structure, spatial correlations, and network level information. These approaches have significantly improved reconstruction accuracy, particularly under long consecutive gaps that challenge classical methods. Figure 1.4 presents a timeline of influential studies on data imputation in this domain since 2020.

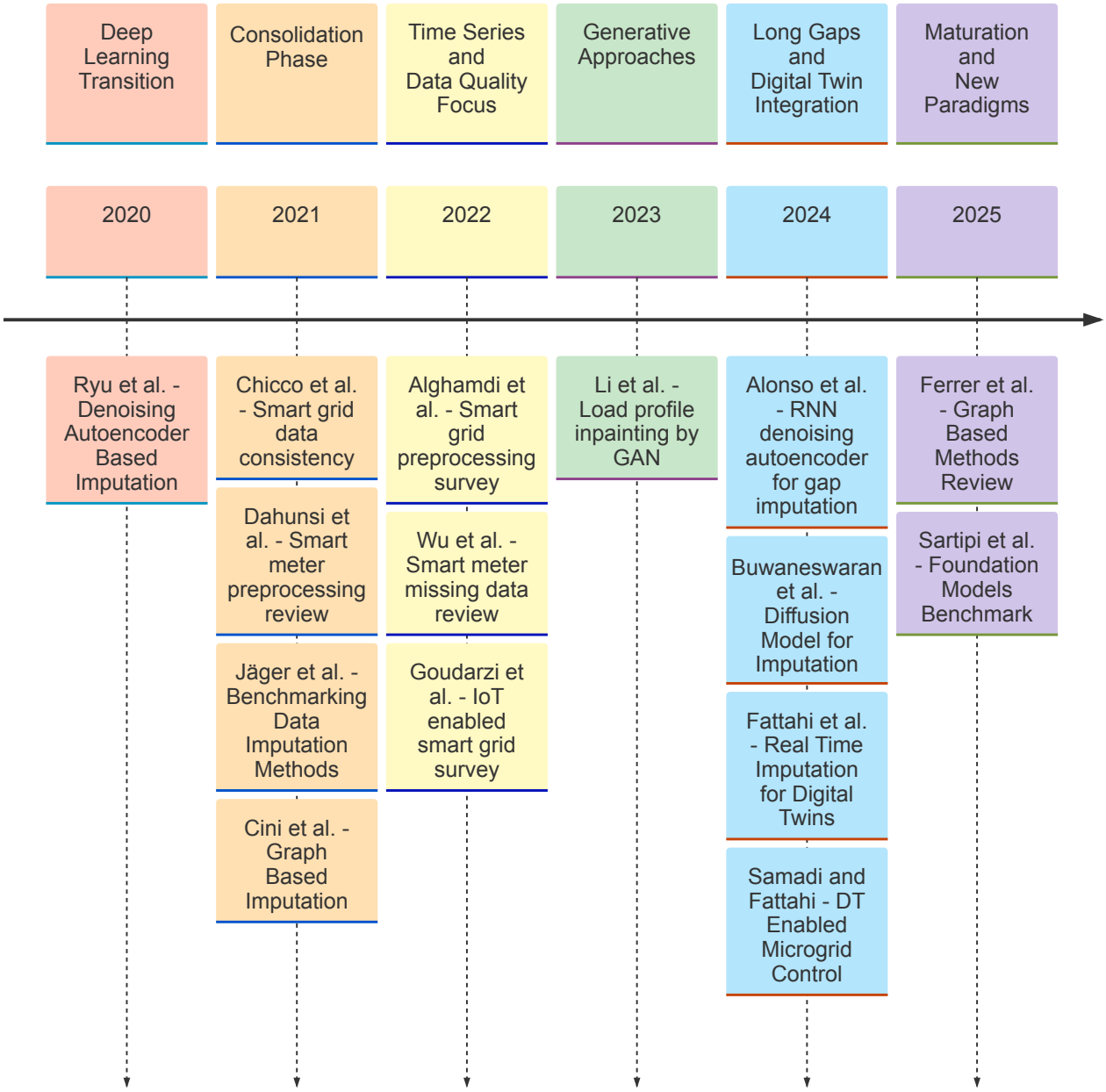

Figure 1.4: Timeline of representative studies on data imputation for smart grid measurements under missing and corrupted data conditions (2020–2025).

Nevertheless, most existing imputation techniques are developed and evaluated under assumptions that do not fully reflect the operational constraints of narrowband smart grid networks such as limited bandwidth, multi hop communication, time varying connectivity, and heterogeneous outage patterns across meters. As a result, there remains a gap between state-of-the-art imputation performance and the robustness required for deployment in large scale narrowband smart grid systems, motivating further research into imputation methods that are explicitly communication aware, scalable, and resilient to realistic narrowband network conditions.

1.1.3 LoWPAN modeling tool

LoWPAN protocol evaluation and resilience analysis often require controlled experimentation at scale, which is difficult to achieve in field deployments. Cooja is part of the Contiki-NG ecosystem and is a network simulation framework designed for wireless nodes with constrained resources.

The defining feature of Cooja is its multilayer simulation model. The core design allows multiple node types to coexist in the same scenario. Local nodes compile and execute the application on the host machine, hardware emulated nodes simulate real motes and offer a simplified execution model [209]. This makes it possible to study detailed single node behavior and large topologies with hundreds of nodes within the same tool. Literature on Cooja emphasize that this multilayer structure enables the exploration of a wide range of protocol combinations for IoT and wireless sensor network research [210].

Another major strength of Cooja is its ability to execute the native Contiki-NG networking stack. Studies show that a full IP based stack comprising IPv6, the 6LoWPAN adaptation layer, the RPL routing protocol, and the CoAP application layer can be evaluated end to end on Cooja [116, 210]. A large body of RPL research measures setup time, packet delivery ratio, delay, and control overhead through systematic Cooja experiments. Early work on RPL performance relies on Cooja to investigate network formation dynamics, the influence of objective functions, and the role of topology density [211, 212]. More recent studies compare RPL objective functions under mobility, varying node density, and different reporting intervals through Cooja based evaluation [213, 214].

At the configuration level, Cooja provides a reproducible structure for defining topology, node behavior, and experimental events. Cooja fundamentally executes as a Java application running on a host computer and serves as the simulation environment for Contiki and Contiki-NG. Figure 1.5 illustrates the Cooja process within the host computer from the Operating System (OS) perspective. Cooja's execution model relies on three primary components: the scenario file, the node implementations, and the makefile structure. The makefile defines the build rules, the selected border router, the example applications, and the general compilation options. It ensures that the same code base is compiled for every node in the simulation. The node implementations capture the behavioral logic of each simulated device, including the network stack, the RPL mechanisms, the application layer processes, and the thread structure. At this layer, researchers can modify the RPL algorithm, introduce a new objective function, or develop additional protocols at the application layer. The timing of interactions among nodes, their positions, the radio and channel

models, and event based disturbances such as node failure, temporary link loss, or sudden traffic increase are specified in the scenario file. This file acts as a central configuration that defines the topology, the initial conditions, the triggered events, and the firmware associated with each node type. A summary of these three components is presented in Table 1.9. Through this modular architecture, Cooja supports both detailed inspection of individual node behavior and large scale experiments with hundreds of devices under complex mesh networking conditions.

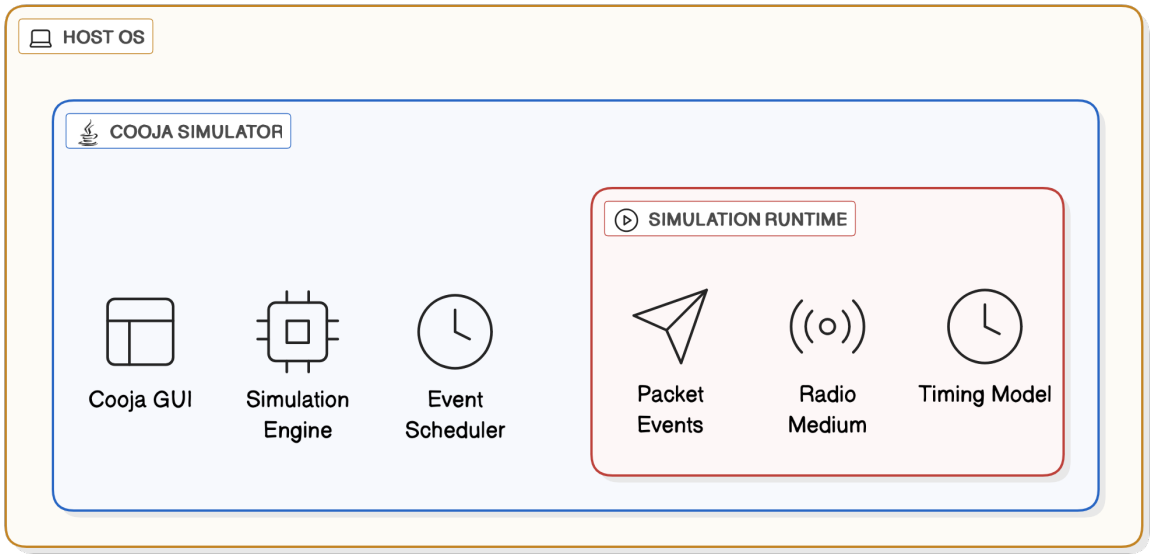

Figure 1.5: Cooja simulator architecture showing the layered structure from host operating system to simulation runtime components.

Cooja supports verification and validation for narrowband smart grid deployments, and is often used to evaluate multi hop routing metrics and interoperability in 6LoWPAN/RPL AMI and smart meter networks. For example, the evaluation of the Lightweight

Table 1.9: Core components of Cooja simulation configuration

Component	Element	Purpose and Scope
Makefile	• Build rules • Border router selection • Example applications • Compilation options	• Compilation targets and dependencies • Network coordinator specification • Application entry points • Consistent code base across nodes
Node Implementations	• Network stack • RPL routing logic • Objective functions • Application processes • Thread structure	• uIP and 6LoWPAN implementation • Topology formation and maintenance • Path selection customization • User defined behavior • Concurrent task management
Scenario File (.csc XML)	• Node positions and topology • Radio and channel models • Initial conditions • Event triggers • Firmware mapping	• Spatial network layout • Propagation and interference • Simulation starting state • Node failure and link loss • Code to node type association

On-demand Ad hoc Distance-vector Next Generation (LOADng) protocol for bidirectional AMI traffic uses Contiki OS and Cooja to compare delay, packet delivery ratio, and control overhead with RPL [215]. Similarly, the energy aware Secure and Energy Efficient Objective Function (SEEOF) objective function for battery powered smart meters is integrated into Contiki RPL and tested in Cooja where its impact on energy usage, network lifetime, and delivery ratio is demonstrated for narrowband metering environments [216]. Smart grid oriented studies also combine Cooja with physical testbeds to evaluate packet delivery ratio, delay, and scalability under realistic conditions [136].

Security and resilience research in AMI and smart grid networks frequently relies on Cooja as well. Machine learning based detection studies for RPL rank attacks often use Contiki-NG and Cooja to emulate RPL IoT topologies, generate benign and attack traffic,

and train anomaly detectors from network derived features [217]. Cooja is also used to assess authentication and neighbor discovery protection for smart meters, reporting impacts on packet delivery ratio, delay, and availability [218, 219].

1.1.4 Research gaps and thesis position

Table 1.10 outlines the observed limitations mentioned above along with the corresponding thesis position and the chapter scope. A persistent limitation in the studies in the literature, is the absence of a unified LoWPAN-based communication fabric that can support both metering and DER connectivity within buildings. To reiterate, the current literature often treats meters and DERs as separate communication islands. While LoWPAN and RPL based mesh solutions are widely studied for smart meters, PV inverters, battery systems, and EV chargers typically connect through different interfaces or dedicated IP networks. As a result, a unified LoWPAN based communication fabric rarely emerges even within a single building and device level interoperability relies heavily on gateways and protocol translation layers. The unified narrowband IP fabric targeted in this thesis addresses this gap by exploring a combined approach that has received limited attention in existing work.

This separation is compounded by limited understanding of resiliency in dense multi-residential deployments, where heterogeneous RF zones and complex building layouts challenge assumptions used in controlled evaluations. Studies of LoWPAN mesh networks usually focus on regular topologies or controlled test scenarios. Dense multi residential buildings add complexity through multi level structures, basements, parking areas, and rooftop regions with distinct propagation conditions. When multiple border routers, het-

Table 1.10: Summary of research gaps and thesis position/contributions

Observed limitation	**Thesis position and contribution**	**Scope**
Lack of a unified LoWPAN-based communication fabric for meters and DER devices in buildings	Targets a building scale narrowband IP fabric that can provide a common transport layer across heterogeneous devices, reducing dependence on ad hoc gateways and protocol specific islands.	Chapter 2
Incomplete understanding of resiliency in dense multi-residential deployments	Characterizes resiliency under building realistic constraints (multi level structure, heterogeneous RF zones, and time varying conditions) to establish practical strengths and limitations of LoWPAN mesh operation.	Chapters 2 and 4
Single points of failure and limited redundancy at border routers and gateways	Evaluates architectural and operational approaches to reduce gateway criticality and improve continuity under border router failures, with attention to multi-residential feasibility and measurable resiliency outcomes.	Chapters 2 and 4
Limited treatment of LoWPAN specific telemetry loss and data imputation inside control/coordination loops	Treats imputation as a communication aware mechanism and evaluates it beyond offline accuracy, focusing on maintaining usable information state under burst loss and delay patterns observed in narrowband meshes.	Chapter 3
Absence of an end-to-end evaluation methodology combining LoWPAN networking, DER/EV use cases, and data imputation	Integrates network behavior, building level DER/EV scenarios, and data reconstruction into a unified evaluation workflow to quantify interactions and support design relevant conclusions.	Chapters 2 to 4

erogeneous RF zones, and time varying load patterns coexist, the resiliency profile of the mesh is not well characterized. This gap limits the ability to assess the practical strengths and limitations of narrowband mesh networks in realistic building scenarios.

Related to this, border routers and gateways remain common single points of failure, and redundancy designs that are practical for multi-residential settings are not yet treated in a systematic manner. Current narrowband meter and DER networks often rely on a single border router or building gateway as a critical point of connectivity. Software faults, hardware failures, or overload at this node can disconnect the entire mesh from external systems. Although multi border router concepts have started to appear in the literature, systematic designs for redundancy in multi residential settings and their impact on resiliency metrics remain limited. This shortcoming slows the development of practical design guidelines and standardizable redundant architectures.

At the same time, telemetry loss and data imputation are mostly studied as offline analytics problems, while their role in maintaining LoWPAN-based control and coordination behavior under realistic loss patterns remains less developed. Work on smart grid analytics typically studies missing data and imputation in the context of load forecasting or state estimation. In contrast, the effect of LoWPAN specific telemetry loss patterns on control and coordination loops remains less explored. The stability of control algorithms under long burst losses and their sensitivity to errors, and the combined effect of losses, have not been modelled in detail. This gap suggests that imputation must be evaluated not only for statistical accuracy but also for its role in maintaining the behavior of LoWPAN based feedback and coordination mechanisms.

These limitations reflect a broader methodological gap: networking performance, building level DER/EV scenarios, and data reconstruction techniques are typically evaluated separately rather than within a single end-to-end framework. Existing studies tend to separate networking performance, device level scenarios, and data processing methods. Some evaluate LoWPAN and RPL in isolation, others analyse DER and EV load coordination, while another body of work compares imputation techniques on smart meter datasets. Very few studies combine all dimensions in a unified framework that captures narrowband mesh behavior, building level DER and EV use cases, and the influence of imputation as combined. Without such an integrated methodology, it becomes difficult to understand how network design, data reconstruction, and control strategies interact, limiting the availability of quantitative evidence for realistic design decisions.

In response, the thesis position is to treat narrowband LoWPAN connectivity, communication aware data continuity, and building realistic evaluation as coupled requirements rather than independent topics. This thesis focuses on LoWPAN based narrowband smart grid communication in multi residential buildings and aims to address the identified research gaps by developing solutions that operate across smart meters, DERs, and EVs. The work approaches the problem in an integrated manner that considers the network architecture, the underlying communication technologies, and the patterns and imputations of telemetry loss together, with the goal of enabling interoperable, feasible, and resilient smart grid operations in multi residential settings.

2. Narrowband Mesh Networks in Smart Grid Applications

2.1 Overview

Narrowband mesh networking has become a practical candidate for connecting meters and building scale DERs in multi-residential and similar environments, where coverage, penetration, and interoperability constraints are dominant. In such settings, the primary communication elements include smart meters, photovoltaic systems, associated inverters, heat pumps, and electric vehicle charging stations. As these DERs continue to diversify in the coming years, ensuring interoperability is expected to become increasingly important. Existing DER technologies do not provide a fully unified interface, and many systems rely on distinct communication standards.

To support interoperable operation across these devices, this chapter develops and evaluates a communication network through which heterogeneous DERs can interface. Sub-GHz mesh networks are evaluated as a suitable approach, as they extend system coverage and enable integration with traditional Wi-Fi or Ethernet networks by acting as a bridging layer. This bridging role is implemented through extensions to Contiki-NG that enable IPv6 RPL networks to operate as bridging mechanisms. A core requirement for achieving this functionality is support for multiple border routers. Accordingly, modifications are introduced to both the TCP/IP stack and the border router implementation to realize the

desired bridging structure. Figure 2.1 illustrates the intended network topology, which incorporates the proposed mesh implementation as a bridge.

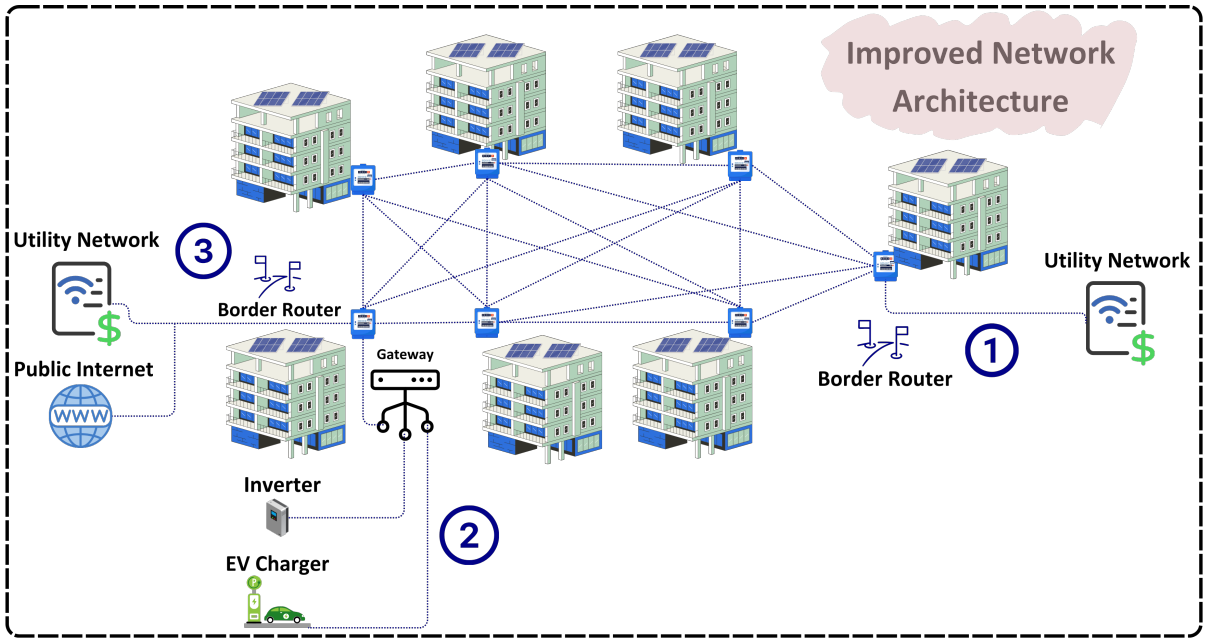

Figure 2.1: Intended design of the mesh network topology, including DERs and smart meters. The figure details (1) the point of utility network access, (2) the introduced native gateway inside the 2nd Border Router (BR) that integrates DERs seamlessly (3) the redundant public internet path used by the 2nd BR which improves network resiliency. Adapted from the author's prior publication [220]. © 2025 IEEE.

Using this bridge architecture, operational scenarios can be evaluated and the resulting implementation insights can be summarized. Network stack modifications and the corresponding Contiki-NG findings can be reported. Different approaches can be examined to determine their suitability under varying conditions, and the requirements necessary to enable a functional bridge can be identified. The discussion can proceed by (i) describing the bridge mechanisms and supporting developments, and (ii) defining the system model

and the conditions that can influence performance. The mechanisms underlying the bridge structure and the associated developments are discussed in Section 2.2. The conditions affecting mesh network performance, including topology characteristics, error rates, and packet sizes, are defined as part of the system model in Section 2.3.

2.2 Bridge Functionality by Multiple Border Routers

RPL-based LoWPAN meshes are typically operated with a single border router that acts as the DODAG root, which limits ingress/egress redundancy and constrains the use of the mesh as a building scale bridge between distinct IP networks. To mitigate this limitation, a secondary node is enabled to provide the same border router forwarding while remaining non-root, and reachability is advertised so that nodes can associate external routes with the appropriate egress point.

2.2.1 Forming a custom mesh network on contiki-ng

Because the Contiki-NG operating system includes both the operating system itself and the TCP/IP stack, it allows the execution of a fully customized TCP/IP implementation on a device while directly observing its runtime behavior. In addition, to support rapid testing across a large number of devices, the Cooja simulation environment is used. This environment allows the custom Contiki-NG implementations to be executed within Cooja and experimental results to be collected in a controlled and scalable manner.

In this use case, where the mesh network described by RPL in Contiki-NG is employed

as a bridge between two distinct networks, the system must support entering and exiting the mesh through two separate nodes. This design provides two entry points into the mesh, enabling data to enter at one node and exit from another, effectively relaying information across extended distances such as between floors or buildings. A primary challenge is that, within the Contiki-NG network stack, border router functionality can only be assigned to a single node. This behavior is reasonable in certain respects, as the border router is responsible not only for ingress and egress but also for overall network management. In RPL networks, the topology is maintained as a DODAG, with the border router acting as the root of this structure. The management responsibilities of the border router further depend on whether the network operates in storing or non-storing mode. These responsibilities can be summarized in Table 2.1.

In the intended design, a second node is required to exhibit border router like behavior. However, since a DODAG can have only a single root, the proposed approach requires enabling this second border router to provide ingress and egress capabilities without assuming the full root role. At the same time, retaining selected management functions of the border router on this secondary node is considered beneficial, particularly for network recovery in scenarios where the primary border router becomes unreachable. The resulting network structure can be represented as illustrated in Figure 2.2.

To use the RPL mesh network as a bridge, both the upstream and downstream border routers must be utilized properly. Understanding how the border router manages ingress and egress requires examining how it uses its interfaces. A border router relies on two interfaces for this operation. One interface connects to the internal mesh network and

Table 2.1: Responsibilities of the RPL Border Router in Storing and Non-Storing Modes

Responsibility	**Storing Mode**	**Non-Storing Mode**
Acts as the RPL DODAG root	Yes	Yes
Maintains downward routing tables	Stores next hop information for descendant nodes.	No routing state is stored in intermediate nodes.
Downward route construction	Distributed across routers. Each router stores routing state for its sub tree.	Centralized at the border router. It computes full downward paths.
Source routing header usage	Not required. Routing uses distributed tables.	Required. The border router inserts full source routing headers.
Forwarding upward traffic	Yes	Yes
Forwarding downward traffic	Uses stored next hop entries in routing tables.	Uses the source routing header to guide packets.
Knowledge of leaf node paths	Distributed among routers.	Fully centralized at the border router.

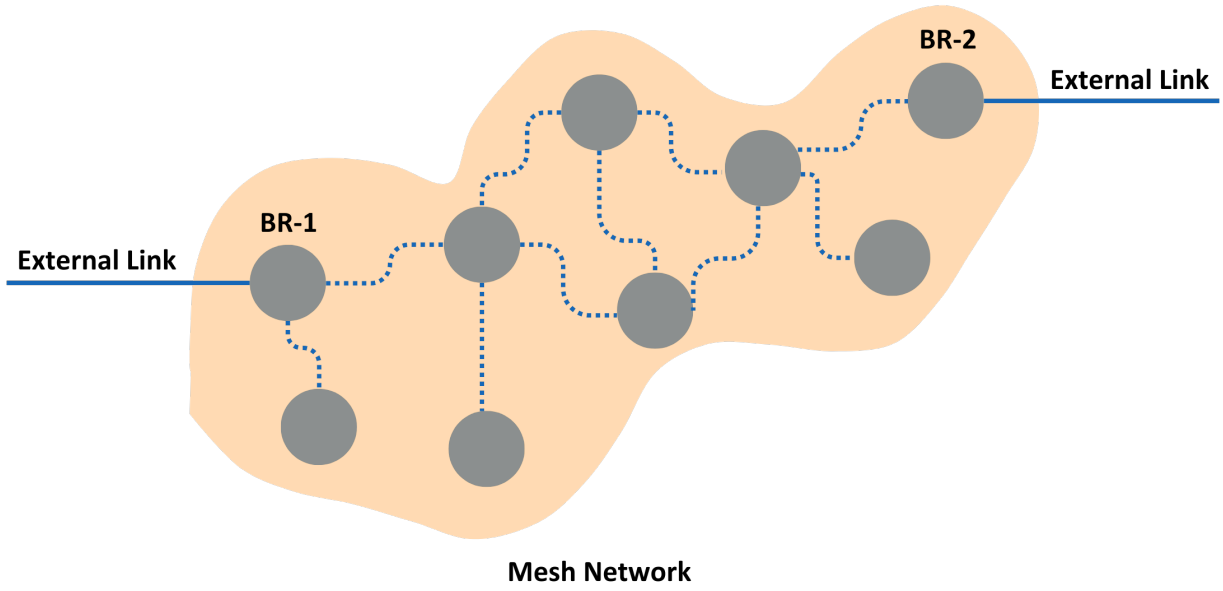

Figure 2.2: Hypothetical double-BR RPL Mesh Network.

the other connects to the external network. The border router receives packets from the internal network that must reach the external network and forwards them according to the routing path. Likewise, it forwards packets arriving from the external network into the mesh.

The path through which data is delivered from the internal network to the border router differs between the non-storing and storing modes due to the way routing information is maintained. In non-storing mode, any node that encounters an unknown destination assumes that it must be reachable through the border router and therefore forwards the packet to it. If the border router finds a next hop in its routing table it forwards the packet. Conversely, if it does not find a valid route from the routing table, it drops it.

This behavior is standard in the RPL implementation for non-storing mode and does not require modification.

In storing mode, each node maintains its own routing information. A node first checks whether it has a route to the destination. If it does not, it forwards the packet to its parent. Through this stepwise propagation the packet can eventually reach the border router, although unlike in non-storing mode it does not reach it directly.

These differences also affect forwarding efficiency under intra mesh communication. One important detail is that while the non-storing mode simplifies operation it can also lead to unnecessary forwarding in some cases. Because every packet must always be delivered through the border router even when shorter paths exist, the non-storing mode can become inefficient. In contrast, the storing mode enables each node to maintain routing information which allows packets to be forwarded through optimal paths. The figures 2.3 and 2.4 illustrate an example of such inefficient transmission in comparison. In each case, node A sends a packet to node B but the paths are different due to storing and non-storing mode routing mechanisms.

When a second border router is introduced, some routes become reachable only through this additional router, which means that forwarding every external route to the primary border router is no longer appropriate. Nodes in the network must know in detail which external routes are reachable through which border router. This issue can be addressed through the use of DAO messages. RPL already uses DAO messages during network formation to disseminate topology information to the nodes. Figure 2.5 illustrates how

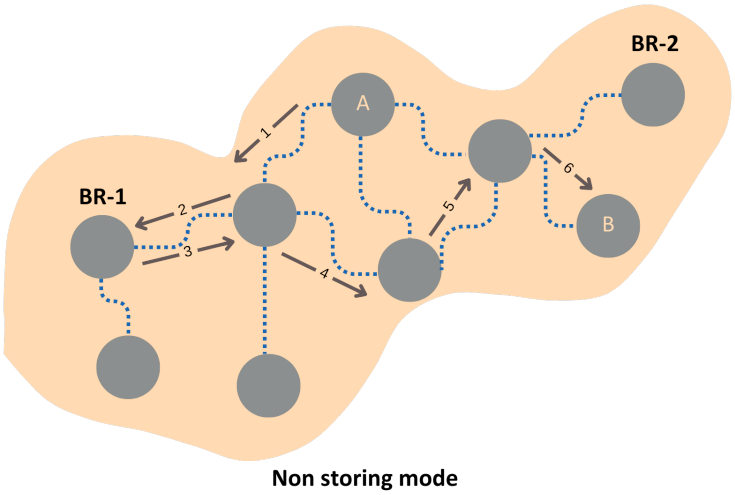

Figure 2.3: Hypothetical routing between nodes A and B in non storing mode. BR-1 is the root node.

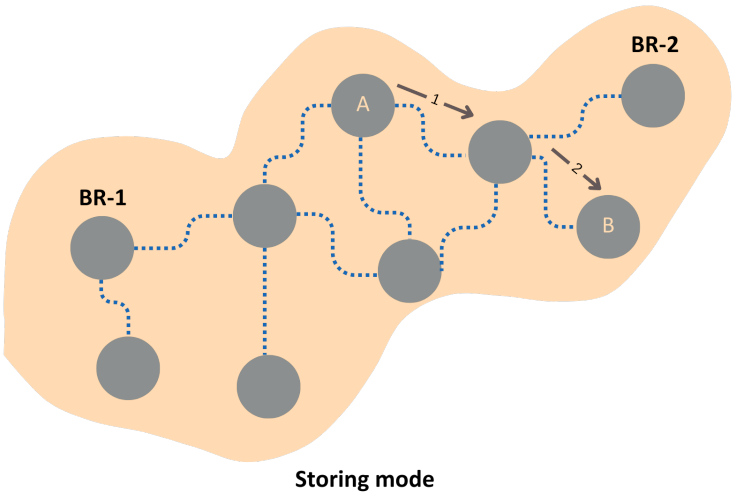

Figure 2.4: Hypothetical routing between nodes A and B in storing mode. BR-1 is the root node.

these DAO messages can be advertised by the second BR, so that other nodes route specific external prefixes to this BR rather than to the default root.

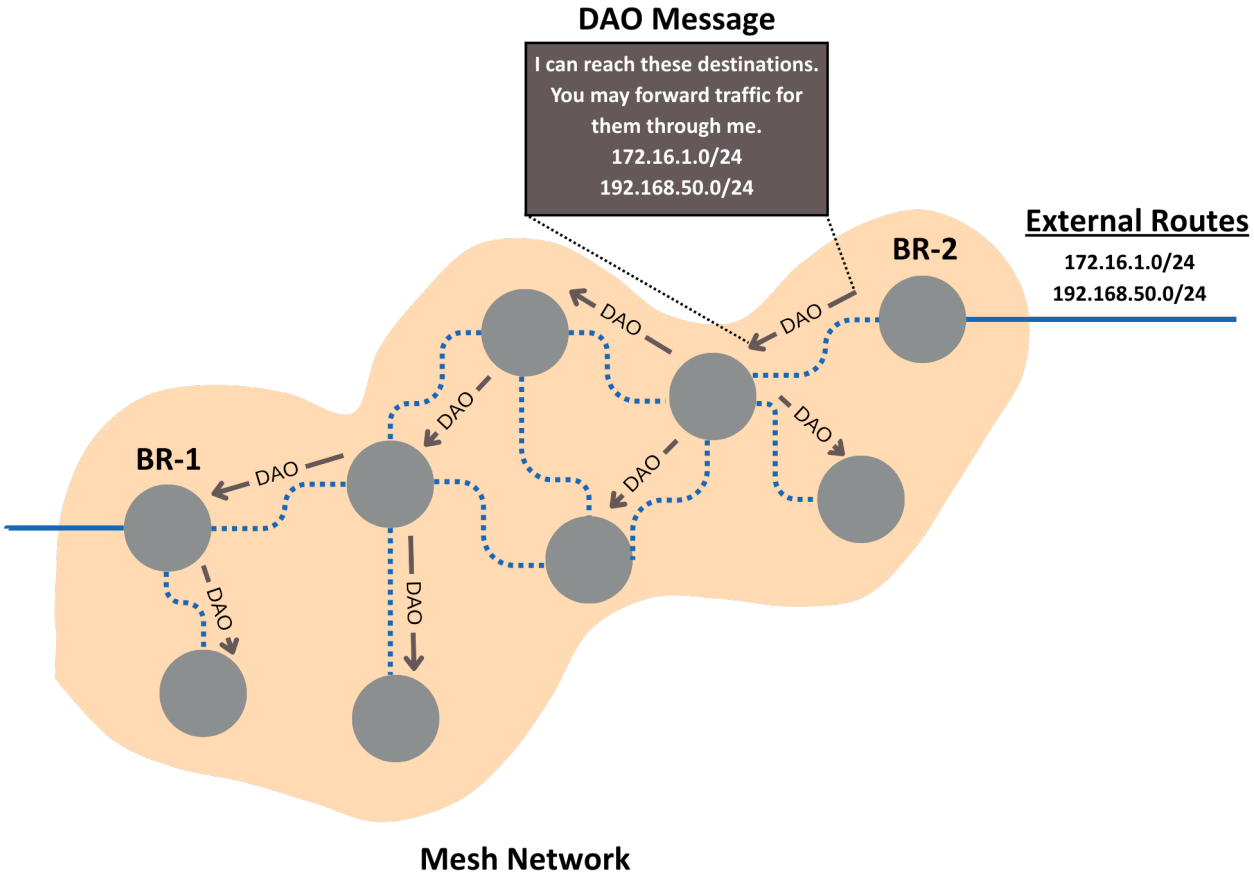

Figure 2.5: Hypothetical flow of DAO message advertisement.

To realize the desired bridge architecture, storing mode is employed together with custom DAO messages for the secondary border router. This approach enables two ingress and egress points within the network and ensures that each external route is correctly associated with the appropriate border router. Achieving this functionality requires several modifications to the implementation. In particular, the secondary border router must incorporate border router capabilities while remaining configured as a non root node, as

the RPL specification permits only a single DODAG root at any given time. Based on the system requirements, the secondary border router is intended to provide external forwarding functionality without assuming network management responsibilities unless explicitly required.

Since different nodes in the network serve distinct purposes, they are grouped into three categories: the root node, simple nodes and bridge nodes. The roles and functions of each group are summarized in Table 2.2.

Table 2.2: Node types and their functions in the custom mesh implementation

Node Type	**Role in Network**	**Key Functions**
Root Node	DODAG Root and Primary Border Router	DODAG root initialization. DIO and DAO ACK handling. External interface via `tun10`. Primary ingress and egress.
Simple Node	Regular RPL Node	DODAG child participation. Storing mode route notifications. Intermediate hop forwarding.
Bridge Node	Secondary Border Router	Non root child operation. DAO based host reachability. External interface via `tun11`. Secondary ingress and egress.

The roles of the relevant network elements within the topology are shown in Figure 2.6. Each node role indicated in Figure 2.6 corresponds to its own implementation file, which is compiled separately and executed in parallel depending on the scenario. Each implementation contains the Contiki-NG operating system, the corresponding RPL TCP/IP

stack and its own configuration. When these implementations are executed in parallel for all nodes the complete network behavior including joining procedures, topology updates, heartbeat exchanges and other communication events can be observed and analyzed systematically. Figure 2.7 illustrates the layers of components that constitute the described implementation.

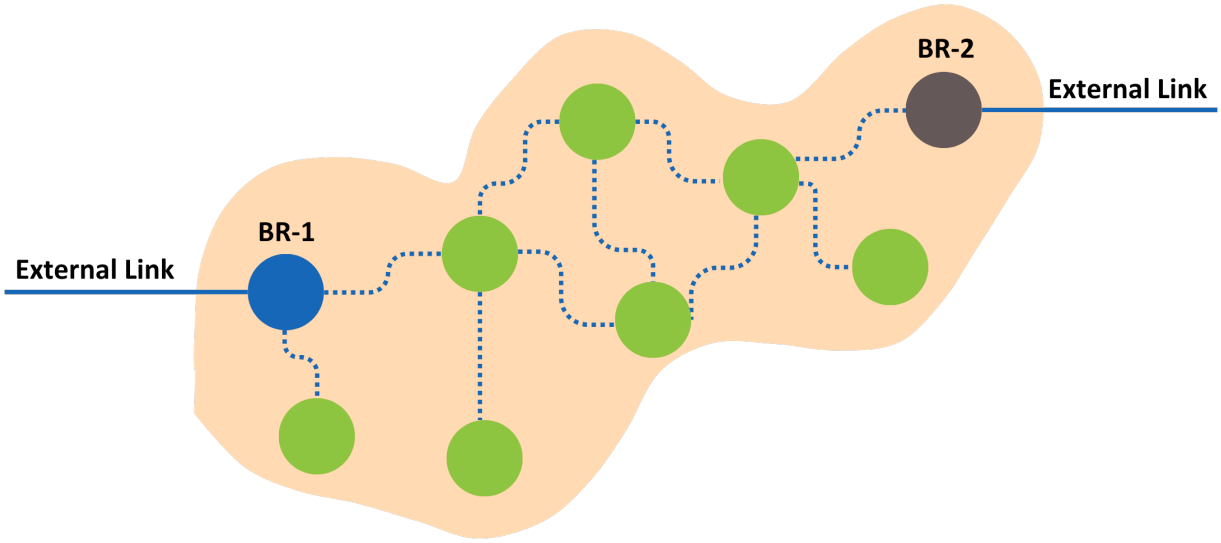

Figure 2.6: Different roles within the RPL mesh network. Blue denotes the root node(primary BR), brown denotes the bridge node(secondary BR) and the green denotes the simple nodes.

The components shown in Figure 2.7 highlight the layers where modifications were applied, particularly within the RPL TCP/IP stack and the custom implementation layer. On the Contiki-NG OS, dedicated tasks were created and designed to operate in a concurrent manner to process incoming network events and produce the corresponding responses. The RPL TCP/IP layer was extended with a custom bridge capability while the custom implementation layer defined the roles of each node and configured them to join the network

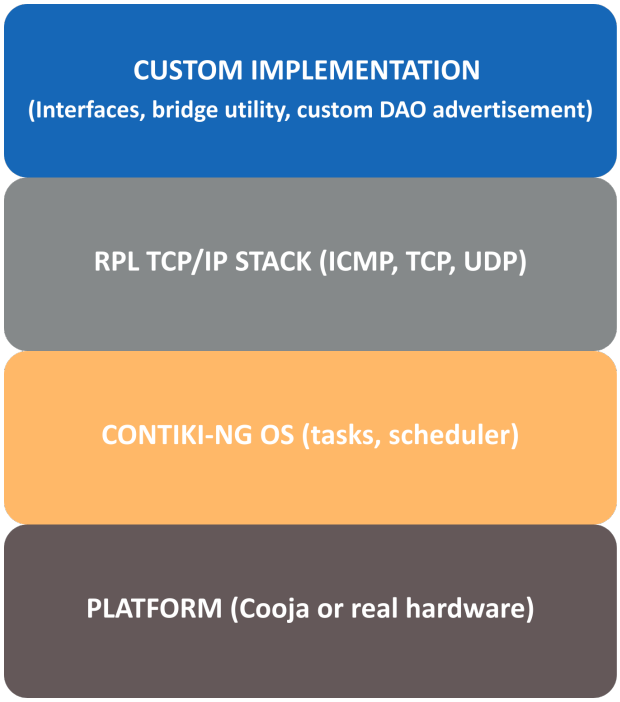

Figure 2.7: Different components of the implementation

according to the required hierarchy.

Each border router requires a custom Serial Line Internet Protocol (SLIP) implementation to interface with its external network. This implementation logically resides between the Platform and the Contiki-NG OS layers. SLIP operates by transporting encapsulated IP packets over a Universal Asynchronous Receiver-Transmitter (UART) interface, therefore incoming traffic must be received through the SLIP layer and delivered to the RPL TCP/IP stack for processing. Conversely, packets generated by the RPL stack must be encapsulated and forwarded through the SLIP interface toward the external network. The Figure illustrates how each border router forwards traffic between the two networks

through its dedicated interfaces.

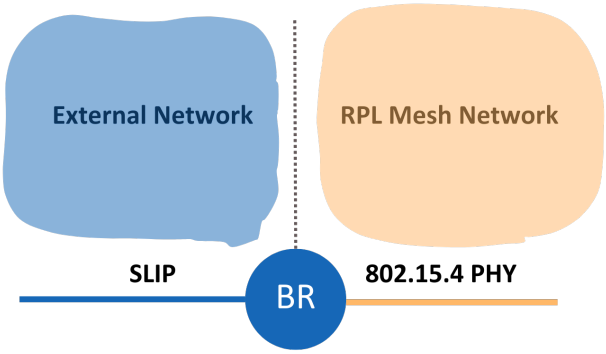

Figure 2.8: BR using different interfaces for individual networks

The SLIP functionality was implemented for every border router and primarily affected the Platform and Contiki-NG layers. Since the system was evaluated on two different platforms, the implementation required platform specific adaptations. On the STM32 board the hardware UART interface was used. In Cooja there is no physical UART, so the UART channel is emulated over TCP sockets. Each border router's UART stream was mapped to a dedicated socket that connected the simulated node to the host machine, effectively carrying UART traffic through a socket based transport.

On the Contiki-NG side a callback was added on top of the UART driver to capture incoming SLIP frames and deliver them to the appropriate network stack handlers. Outgoing packets were processed in the reverse direction, encapsulated and passed back to the SLIP layer. The Figure highlights where this implementation resides within the software stack and which components are involved in enabling bidirectional communication.

Cooja was selected as the primary evaluation platform since its ability to run many

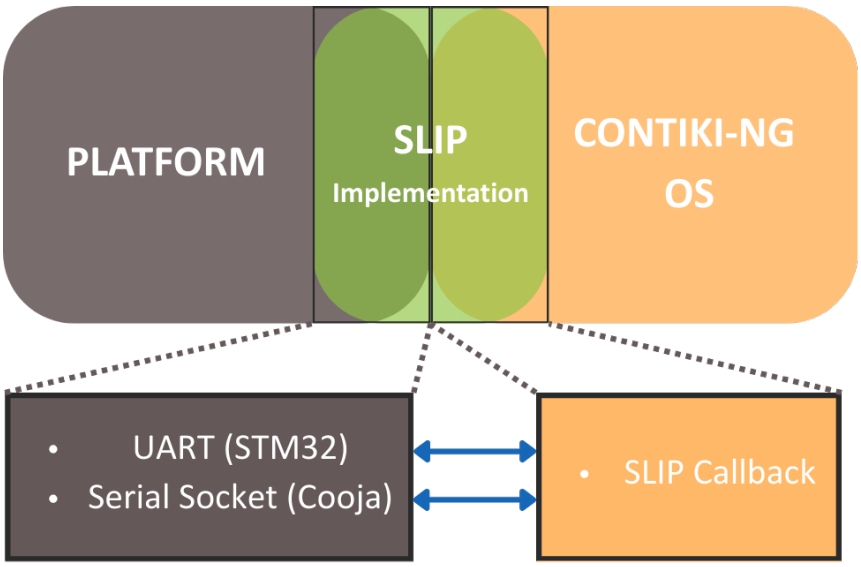

Figure 2.9: SLIP implementation and its corresponding layers

devices virtually provided clear advantages over physical hardware testing. For completeness and to observe real world behavior the implementations were also deployed on STM32 Nucleo F429ZI boards. The following role specific configurations can be used to enable the intended mechanisms in the mesh network. The implementation is not limited to the code listings shown here, but these examples illustrate how the design aligns with the requirements. The complete implementation appears in Chapter 4. The required configuration for each role is shown in the corresponding code listings. For the root node, the configuration is provided in code listings 2.1, 2.2, 2.3 and 2.5. Similarly, the configuration for the simple node is shown in code listing 2.6. The configuration for the bridge node appears in code listings 2.9, 2.10 and 2.11.

2.2.2 Mesh node implementations

The multi border router bridge relies on three node roles with distinct initialization and forwarding behavior. The following listings describe the root node, a representative mesh node, and the bridge node, and then connect these roles to the TCP/IP stack modification that enables SLIP based egress.

Listing 2.1: Root node process initialization sequence

```
1
2  PROCESS_THREAD(custom_slip_border_router_process, ev, data)
3  {
4    static struct etimer status_timer;
5    PROCESS_BEGIN();
6    /* Initialize SLIP architecture-specific layer (1) */
7    slip_arch_init();
8    /* Set our SLIP input callback (2)*/
9    slip_set_input_callback(slip_input_callback);
10   /* Become a DODAG root (3)*/
11   NETSTACK_ROUTING.root_start();
12   /* Set our global address (4)*/
13   set_own_addresses();
14   LOG_INFO("SLIP-based Border Router initialized\n");
15   ...
16 }
```

There are two Contiki-NG processes that play a crucial role. The first is named as the

custom BR process. The initialization sequence performs four critical operations: (1) initializing the SLIP architecture layer for serial communication, (2) registering the custom callback function to handle incoming SLIP packets, (3) invoking root_start() to establish this node as the DODAG root and (4) configuring the node's global IPv6 address. The root_start() call is the key differentiator from the bridge node, as it designates this node as the sole RPL network coordinator.

Listing 2.2: SLIP process implementation from os/dev/slip.c

```
PROCESS_THREAD(slip_process, ev, data)
{
  PROCESS_BEGIN();
  rxbuf_init();  /* Initialize receive buffer (1) */

  while(1) {
    /* Wait for poll event (2) */
    PROCESS_YIELD_UNTIL(ev == PROCESS_EVENT_POLL);
    /* Move packet from rxbuf to uIP buffer (3) */
    uip_len = slip_poll_handler(uip_buf, UIP_BUFSIZE);
    if(uip_len > 0) {
      if(input_callback) {
        input_callback();  /* Call custom callback if registered (4) */
      }
      tcpip_input();  /* Inject packet into IP stack (5) */
    }
  }
  PROCESS_END();
```

```
19 }
```

The second main process is named as slip_process and as the name suggests, it handles SLIP communication. This is a crucial part of the application that enables the border router to communicate with the internal and external networks. The slip_process is a built in Contiki-NG process that handles Serial Line Internet Protocol communication. The process operates as follows: (1) initializing the circular receive buffer at startup, (2) yielding until a poll event is triggered by incoming serial data, (3) copying the received packet from the internal buffer to the global uip_buf and setting uip_len accordingly, (4) invoking the registered input callback function if one exists allowing custom packet processing and (5) calling tcpip_input() to inject the packet into the standard IP networking stack for routing.

This architecture separates the low level serial byte handling from the high level packet processing. The slip_input_byte() function handles individual bytes from the serial interface detecting SLIP framing characters (END and ESC) and assembling complete packets in the receive buffer. When a complete packet is received the process is polled triggering the main loop to process the buffered data. This event driven design helps efficient Central Processing Unit (CPU) utilization on resource constrained embedded devices.

Listing 2.3: SLIP input callback and IP stack integration

```
void
slip_input_callback(void)
{
  slip_packets_received++;

  /* Log received packet information (1) */
  LOG_INFO("SLIP input received, packet #%u, length: %u bytes\n",
           (unsigned int)slip_packets_received, (unsigned int)uip_len);

  /* Debug: Print first few bytes of packet (2) */
  LOG_INFO("SLIP packet data (first 16 bytes): ");
  for(int i = 0; i < (uip_len > 16 ? 16 : uip_len); i++) {
    LOG_INFO_("%02x ", ((uint8_t*)uip_buf)[i]);
  }
  LOG_INFO_("\n");

  LOG_INFO("Forwarding packet to IP stack for normal routing\n");

  /* Packet data is already in uip_buf with length in uip_len (3) */
  /* After callback returns, SLIP process calls: (4) */
  /* tcpip_input(); */
}
```

The slip_input_callback() function handles incoming packets from the external network via the SLIP interface. The function performs the following operations: (1) incrementing the

packet counter and logging packet metadata, (2) printing the first 16 bytes for debugging purposes and (3) accessing the packet data which resides in uip_buf with its length stored in uip_len. The final step (4) involves calling tcpip_input() which injects the packet into the Contiki-NG IP stack for standard RPL routing. This function takes no arguments as it operates directly on the global uip_buf buffer containing the received packet. The function has shown here to highlight how the packet is forwarded to TCP/IP stack later, the actual forwarding is done by the slip_process.

Listing 2.4: Log level configuration in project-conf.h

```
/* Log level definitions (1) */
#define LOG_CONF_LEVEL_RPL    LOG_LEVEL_INFO
#define LOG_CONF_LEVEL_IPV6   LOG_LEVEL_INFO
#define LOG_CONF_LEVEL_ICMP6  LOG_LEVEL_INFO
#define LOG_CONF_LEVEL_TCPIP  LOG_LEVEL_INFO

/* Available log levels (2) */
/* LOG_LEVEL_NONE  - No logging */
/* LOG_LEVEL_ERR   - Errors only */
/* LOG_LEVEL_WARN  - Warnings and errors */
/* LOG_LEVEL_INFO  - Informational messages */
/* LOG_LEVEL_DBG   - Debug messages (most verbose) */
```

One important aspect to note is that log functions are computationally expensive because they inherently perform string formatting and issue system calls. The log levels can be configured in project-conf.h as shown in Listing 2.4. For tests where max-

imum performance is required the log level can be reduced to LOG_LEVEL_NONE or LOG_LEVEL_ERR (1) so that the system prints as little as possible. The available levels range from LOG_LEVEL_NONE (no output) to LOG_LEVEL_DBG (maximum verbosity) (2). Reducing log verbosity allows more packets to flow through the TCP/IP stack instead of wasting already constrained CPU cycles on logging operations.

Listing 2.5: Global IPv6 address configuration

```
1  static void
2  set_own_addresses(void)
3  {
4    uip_ipaddr_t ipaddr;
5
6    /* Construct address using default prefix (1) */
7    uip_ip6addr(&ipaddr, UIP_DS6_DEFAULT_PREFIX_0,
8                UIP_DS6_DEFAULT_PREFIX_1, 0, 0, 0, 0, 0, 0);
9
10   /* Set interface identifier from link layer address (2) */
11   uip_ds6_set_addr_iid(&ipaddr, &uip_lladdr);
12
13   /* Add address to node's address list (3) */
14   uip_ds6_addr_add(&ipaddr, 0, ADDR_AUTOCONF);
15
16   LOG_INFO("Node addresses set up\n");
17 }
```

The set_own_addresses() function constructs a global IPv6 address through three steps: (1)

creating the address using the default prefix (fd00::/64), (2) setting the Interface Identifier from the node's link layer address and (3) adding the constructed address to the node's address list with the autoconfiguration flag.

The second step is particularly noteworthy as it leverages the uniqueness of hardware addresses to generate collision free IPv6 addresses. Since each node possesses a unique link layer address the resulting interface identifier is guaranteed to be unique within the network. This approach eliminates the need for manual address assignment or centralized address management making network deployment straightforward and scalable. The process results in predictable and convenient address patterns such as fd00::201:1:1:1 for node ID 1 and fd00::202:2:2:2 for node ID 2. However users are free to introduce other address assignment mechanisms as they would prefer.

In combination, the application process and `slip_process` can establish a primary border router with bidirectional connectivity between the external and internal networks. Incoming SLIP packets can be inspected via the callback and then injected into the IP stack through `tcpip_input()` for standard RPL routing. Figure 2.10 illustrates the process sequence that BR handles during SLIP communication.

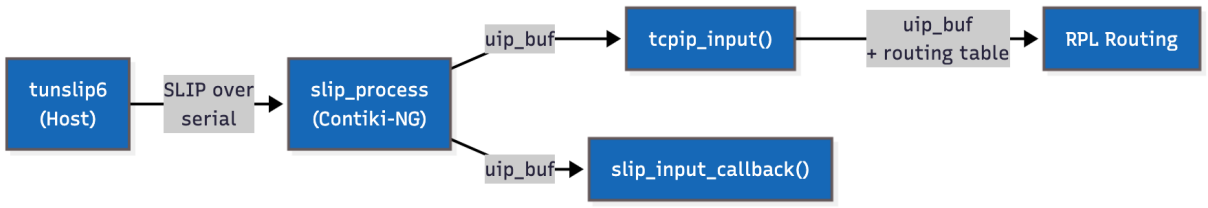

Figure 2.10: Processing sequence of a SLIP input by BR. Host corresponds to the host computer where the Cooja is run.

For nodes that only participate in mesh forwarding, the configuration is reduced to address setup and (in storing mode) route state monitoring.

Listing 2.6: Simple node process initialization

```
PROCESS_THREAD(receiver_node_process, ev, data)
{
  static struct uip_ds6_notification n;

  set_global_address();  /* Configure IPv6 address (1) */

  /* Register route callback (2) */
  #if RPL_WITH_STORING
  uip_ds6_notification_add(&n, route_callback);
  #endif

  LOG_INFO (("Simple node initialized\n");
  ...
}
```

The simple node initialization performs two key operations: (1) configuring the global IPv6 address using the same mechanism as the root node where the interface identifier is derived from the link layer address, (2) registering a notification callback to monitor route changes in storing mode. Unlike the root node or bridge node the simple node does not initialize any SLIP interface or invoke root_start() as it operates purely as an individual non root node within the mesh network.

One important aspect to note here is that in the RPL storing mode each node maintains its own routing table including the default route to the DODAG root. The route notification callback monitors changes to this default route providing real time awareness of network connectivity. When the default route is added, it indicates successful attachment to the DODAG and the node can begin transmitting data. Conversely when the default route is removed, it signals loss of connectivity either due to parent failure or network partitioning. This information is critical for the proper operation of the network as attempting to send packets without a valid route would result in packet loss.

Listing 2.7: RPL-Classic routing configuration in Makefile

```
1 # RPL-Classic Configuration for route injection support
2 MAKE_ROUTING = MAKE_ROUTING_RPL_CLASSIC
```

The storing mode is activated by selecting RPL-Classic as the routing protocol in the Makefile. RPL-Classic defaults to storing mode (MOP 2 - storing without multicast) as defined in the protocol configuration.

Listing 2.8: Default MOP configuration in rpl-conf.h

```
/* MOP Definitions (1) */
#define RPL_MOP_NO_DOWNWARD_ROUTES      0
#define RPL_MOP_NON_STORING             1
#define RPL_MOP_STORING_NO_MULTICAST    2
#define RPL_MOP_STORING_MULTICAST       3

/* Default MOP Selection (2) */
#ifdef RPL_CONF_MOP
#define RPL_MOP_DEFAULT  RPL_CONF_MOP
#else
#define RPL_MOP_DEFAULT  RPL_MOP_STORING_NO_MULTICAST  /* MOP 2 */
#endif

/* Storing Mode Enable Flag (3) */
#define RPL_WITH_STORING (RPL_MOP_DEFAULT != RPL_MOP_NON_STORING)
```

The RPL mode of operation determines how downward routes are managed. As shown in Listing 2.8: (1) four MOP values are defined ranging from no downward routes (MOP 0) to storing with multicast (MOP 3), (2) RPL-Classic defaults to MOP 2 (storing without multicast) unless explicitly overridden via RPL_CONF_MOP and (3) the RPL_WITH_STORING flag is automatically enabled when the default MOP is not non-storing. This configuration enables storing mode features throughout the codebase including the route notification callbacks used in simple nodes. Table 2.3 denotes different modes of operation and their respective downward route handling for the RPL protocol.

Table 2.3: RPL Mode of Operation (MOP) Values

MOP	Name	Downward Routing
0	No Downward Routes	None
1	Non-Storing	Source routing at root
2	Storing (No Multicast)	Distributed routing tables
3	Storing (Multicast)	Distributed plus multicast

To add a second ingress/egress point while preserving a single DODAG root, the bridge role combines SLIP support with periodic DAO advertisements.

Listing 2.9: Bridge node process initialization

```
PROCESS_THREAD(gateway_bridge_process, ev, data)
{
  static struct etimer status_timer, dao_timer;

  /* Initialize SLIP architecture-specific layer (1) */
  slip_arch_init();

  /* Set our SLIP input callback (2) */
  slip_set_input_callback(slip_input_callback);

  /* NOTE: root_start() is NOT called here (3) */

  /* Periodic status reporting */
  etimer_set(&status_timer, CLOCK_SECOND * 30);

  /* DAO announcement timer (4) */
```

```
  etimer_set(&dao_timer, CLOCK_SECOND * 60);
  ...
}
```

The bridge node initialization differs from the root node in one critical aspect: (1) SLIP architecture is initialized identically, (2) the SLIP input callback is registered for packet handling, (3) however root_start() is intentionally omitted meaning this node joins the DODAG as a regular child node rather than becoming the root and (4) a dedicated DAO timer is configured to periodically announce external host reachability every 60 seconds. This design maintains the single root constraint of RPL while enabling secondary gateway functionality.

Listing 2.10: External host reachability announcement via DAO

```
static void
announce_host_reachability(void)
{
  uip_ipaddr_t target_addr;

  /* Only announce if we're part of RPL DAG (1) */
  if(default_instance == NULL || default_instance->current_dag == NULL ||
     default_instance->current_dag->preferred_parent == NULL) {
    LOG_WARN("Not joined to RPL DAG, skipping host announcement\n");
    return;
  }

```

```
13    /* Parse target address from configuration (2) */
14    if(!uiplib_ipaddrconv(TUN11_HOST_ADDR, &target_addr)) {
15      LOG_ERR("Failed to parse target address: " TUN11_HOST_ADDR "\n");
16      return;
17    }
18
19    /* Send DAO to preferred parent announcing external host (3) */
20    dao_output_target(default_instance->current_dag->preferred_parent,
21                      &target_addr,
22                      default_instance->default_lifetime);
23
24    dao_announcements_sent++;
25  }
```

The announce_host_reachability() function is the core mechanism that enables the bridge node to act as a secondary gateway. It performs three operations: (1) verifying that the node has successfully joined the DODAG by checking for a valid preferred parent ensuring DAO messages can be forwarded toward the root, (2) parsing the external host address (fd00::2) from the compile time configuration and (3) sending a DAO message via dao_output_target() to announce that the external host is reachable through this bridge node. When the DODAG root receives this DAO it automatically creates a routing table entry directing traffic destined for fd00::2 to the bridge node.

Listing 2.11: Periodic DAO announcement in main loop

```
while(1) {
  PROCESS_WAIT_EVENT();

  if(etimer_expired(&status_timer)) {
    /* Status reporting every 30 seconds (1) */
    LOG_INFO("SLIP packets received: %u\n", slip_packets_received);
    LOG_INFO("DAO announcements sent: %u\n", dao_announcements_sent);
    etimer_reset(&status_timer);
  }

  if(etimer_expired(&dao_timer)) {
    /* Announce fd00::2 reachability via DAO (2) */
    announce_host_reachability();
    etimer_reset(&dao_timer);
  }
}
```

The main loop inside the bridge process handles two periodic tasks: (1) status reporting every 30 seconds for debugging and monitoring purposes and (2) DAO announcement every 60 seconds to refresh the external host route at the DODAG root. The periodic DAO refresh is necessary because RPL routes have a limited lifetime and must be renewed before expiration to maintain continuous connectivity to the external host.

Node level roles define SLIP and DAO signaling, but explicit SLIP egress for external destinations is required in the IPv6 output path.

The node level implementations described above handle SLIP initialization and DAO announcements but do not address the actual packet forwarding mechanism. When a packet destined for an external host (fd00::1 or fd00::2) arrives at a border router or bridge node the standard Contiki-NG routing logic does not automatically forward it via SLIP. To enable this functionality a custom modification was implemented in the core TCP/IP stack specifically in the `tcpip_ipv6_output()` function located in `os/net/ipv6/tcpip.c`.

Listing 2.12: Custom SLIP routing logic in tcpip_ipv6_output()

```
1  void
2  tcpip_ipv6_output(void)
3  {
4    /* Standard validation checks ... */
5
6    /* CUSTOM: SLIP routing for Border Router and Gateway (1) */
7    {
8      uip_ipaddr_t mote1_addr, mote2_addr, host1_addr, host2_addr;
9      uip_ds6_addr_t *my_addr;
10
11     /* Define addresses for both nodes and hosts (2) */
12     /* mote1: fd00::201:1:1:1, mote2: fd00::202:2:2:2 */
13     /* host1: fd00::1, host2: fd00::2 */
14     /* ... address initialization ... */
15
16     my_addr = uip_ds6_get_global(-1);
17     if(!my_addr) return;
18
```

```
19      /* Check if we are mote1 (Border Router) (3) */
20      if(uip_ipaddr_cmp(&my_addr->ipaddr, &mote1_addr)) {
21        if(uip_ipaddr_cmp(&UIP_IP_BUF->destipaddr, &host1_addr)) {
22          if(uip_is_proto_ext_hdr(UIP_IP_BUF->proto)) {
23            NETSTACK_ROUTING.ext_header_remove(); /* (4) */
24          }
25          slip_send();
26          uipbuf_clear();
27          return;
28        }
29      }
30      /* Check if we are mote2 (Gateway) (5) */
31      else if(uip_ipaddr_cmp(&my_addr->ipaddr, &mote2_addr)) {
32        if(uip_ipaddr_cmp(&UIP_IP_BUF->destipaddr, &host2_addr)) {
33          if(uip_is_proto_ext_hdr(UIP_IP_BUF->proto)) {
34            NETSTACK_ROUTING.ext_header_remove();
35          }
36          slip_send();
37          uipbuf_clear();
38          return;
39        }
40      }
41    }
42
43    /* Standard routing continues below ... */
44    if(!NETSTACK_ROUTING.ext_header_update()) { /* (6) */
```

```
45      ...
46    }
47  }
```

The custom SLIP routing logic is inserted at the beginning of `tcpip_ipv6_output()` before the standard routing operations. This placement is critical for the following reasons: (1) the entire custom block is enclosed in its own scope to avoid variable conflicts with the rest of the function, (2) both node addresses and external host addresses are defined to enable runtime identification of the current node's role, (3) the code first retrieves the node's global IPv6 address and compares it against known addresses to determine whether this node is the border router or the bridge, (4) before forwarding the packet via SLIP any RPL extension headers are removed using `ext_header_remove()` because external hosts do not understand RPL specific headers and would reject or misprocess such packets, (5) the same logic applies symmetrically to the bridge node with its corresponding addresses and (6) the custom block executes before `ext_header_update()` which would otherwise add or modify RPL headers that external hosts cannot interpret.

Listing 2.13: RPL extension header removal before SLIP transmission

```
1  /* CRITICAL: Clean RPL headers if present before sending to host */
2  if(uip_is_proto_ext_hdr(UIP_IP_BUF->proto)) {
3    /* Has extension headers - clean RPL headers (1) */
4    NETSTACK_ROUTING.ext_header_remove();
5  }
6  /* Direct SLIP transmission bypassing normal routing (2) */
```

```
7 slip_send();
8 uipbuf_clear();
9 return; /* (3) */
```

The header removal and transmission sequence performs three operations: (1) checking whether the packet contains IPv6 extension headers using `uip_is_proto_ext_hdr()` and if present invoking the routing protocol's header removal function to strip RPL specific options, (2) calling `slip_send()` to transmit the cleaned packet directly over the SLIP interface to the external host and (3) returning immediately after transmission to prevent the packet from entering the standard wireless routing path.

This modification enables a single compiled TCP/IP stack to serve both the border router and bridge nodes. At runtime each node identifies itself by comparing its global address against the predefined addresses and executes only the relevant forwarding logic. The border router forwards packets destined for fd00::1 via its SLIP interface while the bridge node forwards packets destined for fd00::2 via its own SLIP interface. Packets destined for other addresses bypass this custom block entirely and follow the standard RPL routing path.

Figure 2.11 illustrates the achieved network topology including external interfaces that are connected to individual BRs.

Overall, the bridge role can provide secondary gateway functionality while preserving the single root constraint. DAO advertisements can establish reachability for external targets, and the modified IPv6 output path can forward matching traffic via SLIP.

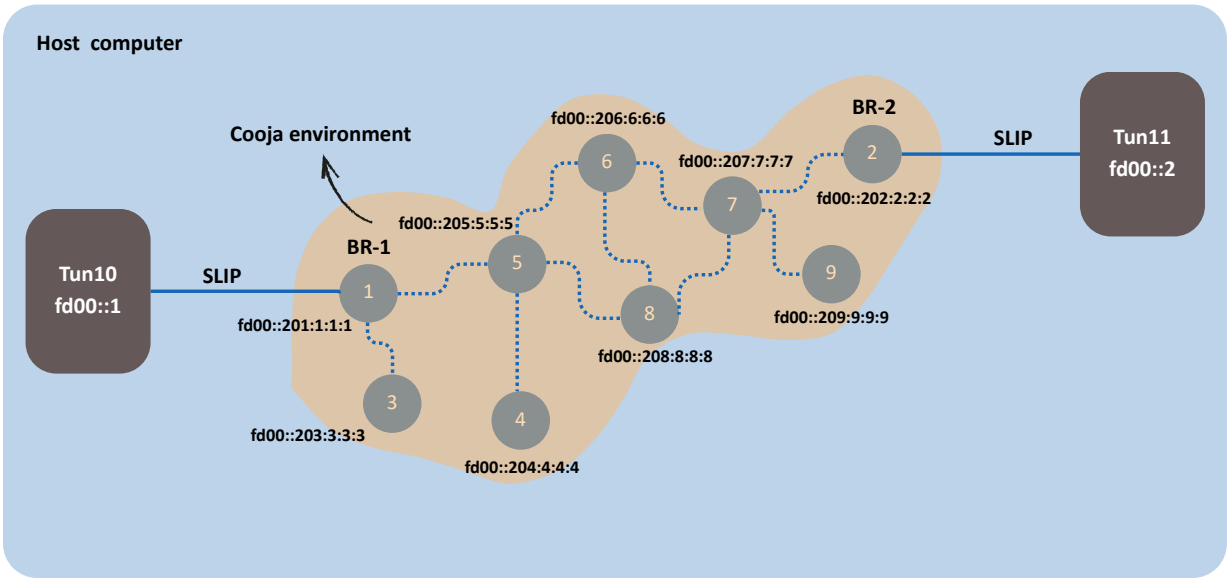

Figure 2.11: Achieved network topology including external interfaces and configured IP addresses.

Table 2.4 gives a comparison between the configurations of the root node, the simple node and the bridge node. Understanding how individual nodes differ is crucial to maintain the network in a predictable and scalable manner.

With multiple ingress and egress points available on the LoWPAN side, the bridge role can also be extended as an integration point for behind-the-meter assets.

2.2.3 Extending mesh network to behind the meter assets

The multi BR topology introduced earlier (Section 2.2) provides two ingress and egress points for the RPL mesh through dedicated SLIP interfaces. To demonstrate that this architecture can support behind the meter assets in a realistic manner, the bridge node is

Table 2.4: Comparison of node types in the custom mesh implementation

Operation	**Root Node**	**Bridge Node**	**Simple Node**
slip_arch_init()	Yes	Yes	No
slip_set_input_callback()	Yes	Yes	No
root_start()	Yes	No	No
DAO announcement	Automatic (as root)	Manual (dao_output_target)	Automatic (as child)
Route notification callback	No	No	Yes
DODAG role	Root	Child node	Child node
External interface	tun10 (fd00::1)	tun11 (fd00::2)	None
Primary function	Network coordinator and gateway	Secondary gateway	Extend network reach and relay packets

extended to operate as a gateway toward a local DER and EVSE segment. The goal is to enable native two way IP connectivity between the utility side and individual assets while preserving basic security and privacy requirements through segmentation. This design aligns with the broader gateway concept developed in the author's earlier work [220], but it is adapted here to the Contiki-NG based multi BR bridge implementation and its evaluation workflow.

In the proposed setup, the bridge node remains a non root member of the DODAG and continues to advertise reachability using DAO messages, as described earlier. In addition, it exposes a dedicated access point for a set of DER and EVSE endpoints located behind the bridge node. From the perspective of the RPL mesh, these endpoints are treated as external hosts reachable via the bridge node. From the perspective of the DER and EVSE side, the bridge node acts as the single attachment point that mediates inbound

and outbound traffic.

Access to individual assets is provided through a port based forwarding scheme. Each asset is assigned a unique service port on the gateway facing side. Inbound connections targeting a given port are translated to the internal Internet Protocol Version 4 (IPv4) address of the corresponding device on the DER and EVSE segment. This method provides a simple and deterministic addressing model for applications on the utility side, since a single gateway address can be used while device selection is handled by the destination port. Table 2.5 gives an example mapping used during evaluation.

Table 2.5: Example port mapping used by the gateway to reach individual DER and EVSE endpoints

Device	**Assigned Port**	**Internal IPv4 Address**	**VLAN ID**
Heat pump	14400	10.0.1.10	10
Inverter	14401	10.0.2.10	20
EV charger	14402	10.0.3.10	30

Segmentation between device groups is enforced using Virtual Local Area Network (VLAN)s on the gateway. Each VLAN forms an isolated local network for a specific asset class. This prevents unrelated devices from observing each other's traffic and reduces the blast radius of misconfigurations. In the example configuration, heat pumps, inverters and EV chargers are placed in three separate VLANs. Only traffic that is explicitly forwarded by the gateway crosses these boundaries.

The evaluation platform follows the same philosophy used throughout this chapter, where the network stack behavior is observed in a controlled environment. In Cooja, DER and EV charger endpoints are emulated on the host machine and attached to the bridge

node through its external interface. This allows repeatable experiments without requiring physical hardware for each asset type. Figure 2.12 summarizes the resulting architecture, highlighting the mesh side, the gateway functions at the bridge node and the segmented DER and EV charger networks.

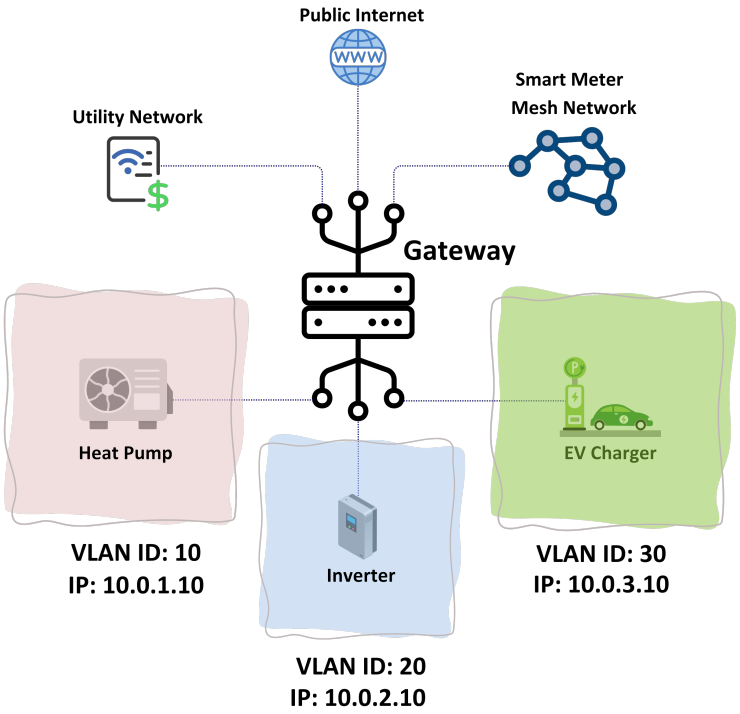

Figure 2.12: Gateway extension of the bridge node. The bridge node provides mesh connectivity toward the DODAG and segmented access toward DER and EV charger endpoints using VLAN based isolation and port based forwarding. Adapted from the author's prior publication [220]. © 2025 IEEE.

Overall, this extension turns the secondary border router into a practical integration point for behind-the-meter assets. The RPL mesh retains a single DODAG root and standard topology maintenance, while the bridge node provides controlled reachability

to specific endpoints. Figures 2.13 and 2.14 illustrates examples of such hypothetical communication paths on the unified LoWPAN involving some behind the meter assets, within the host OS environment.

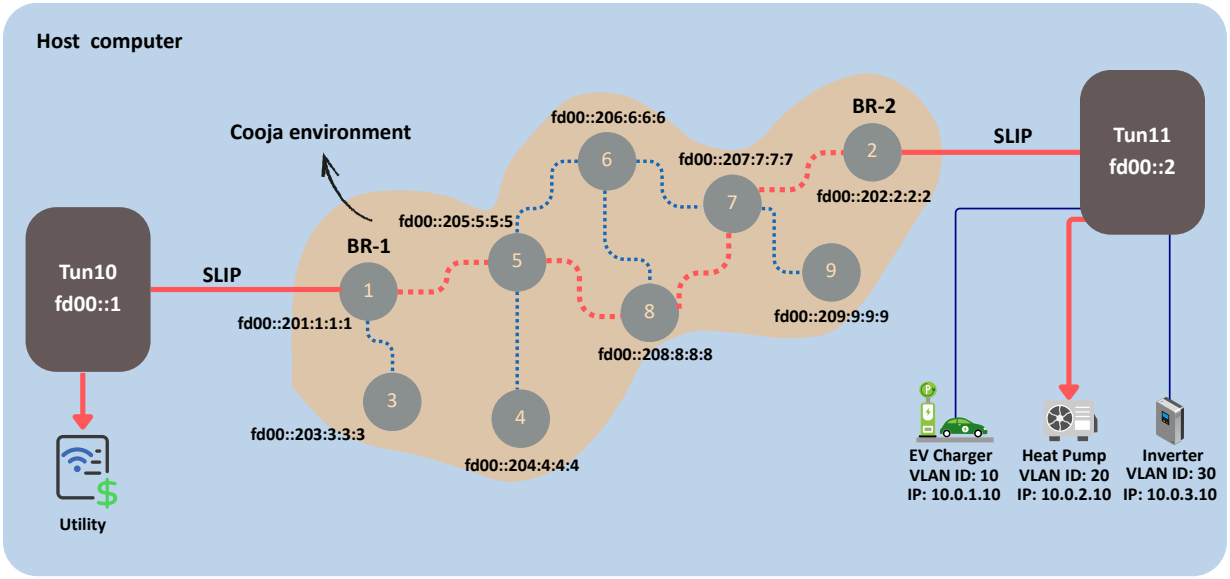

Figure 2.13: Illustrative end-to-end communication path from the utility side to a heat pump over the unified LoWPAN fabric.

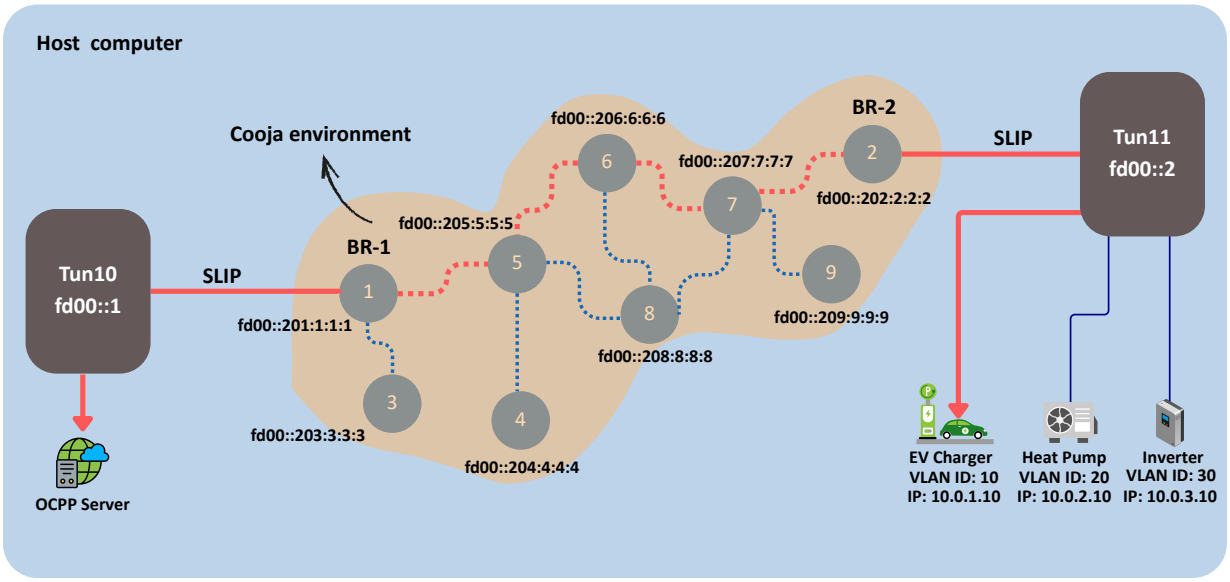

Figure 2.14: Illustrative end-to-end communication path from an OCPP server to an EV charger over the unified LoWPAN fabric.

Having this customizable environment makes it possible to evaluate monitoring and control traffic flows across floors or buildings using the same multi hop mesh while keeping the DER and EV charger side structured and observable. Subsequent sections can use this setup to assess representative traffic flows and configuration choices under controlled conditions.

2.3 The Implications of Relaying on the Mesh

Multi hop relaying in a unified LoWPAN fabric can introduce delay accumulation, higher loss exposure, and fragmentation sensitivity, which directly affect bidirectional interactions with behind-the-meter assets. The following evaluation therefore uses three representative

application flows to exercise downstream delivery and upstream reporting across the mesh and the gateway integrated DER segment. Several selected use cases are summarized in Table 2.6. First, a utility side control query is delivered to a residential heat pump and the corresponding telemetry response is returned over the same path. Second, an EV charging transaction is considered in which an EVSE exchanges OCPP messages with an OCPP server through the mesh, reflecting remote management and session level signaling. Third, an aggregator driven dispatch flow is examined where setpoints are delivered to a behind the meter PV inverter and operational telemetry is collected for closed loop coordination. Together, these flows provide realistic traffic patterns that exercise downstream delivery and upstream reporting across the mesh and the gateway integrated DER segment.

Table 2.6: Representative application flows used to evaluate the unified LoWPAN fabric.

Use case	**Field context**	**Purpose**	**Evaluation focus**
Utility to heat pump control and telemetry	Residential demand response, utility load control	Control request and telemetry return	Request response behavior, downstream reliability, hop sensitivity
EVSE to OCPP server transaction signaling	Managed EV charging networks, centralized OCPP backends	Session signaling and metering updates	Transaction completion, burst traffic, fragmentation effects
Aggregator to PV inverter dispatch and feedback	DER aggregation, feeder level PV coordination	Setpoint dispatch and feedback telemetry	Control loop latency, loss tolerance, reliability tradeoffs

To separate application effects from network effects, each flow is executed under a common set of stress conditions that vary hop count, loss, frame size, and scale. To observe how the unified LoWPAN fabric behaves under different network loads and impairments, each use case is evaluated under a set of stress conditions summarized in Table 2.7. A

baseline configuration is first established to provide a reference point. Hop stress increases path length to expose delay accumulation and multi hop reliability limits. Loss stress increases link level frame loss to capture throughput degradation and retransmission overhead. Fragmentation stress combines reduced frame size with larger application payloads to force 6LoWPAN fragmentation and reassembly. Scale stress increases the total node count to capture contention effects and routing state behavior in denser deployments. Use case definitions below specify message sets, sequence diagrams, and payload sizes, which are then used to interpret the experimental results reported in Chapter 4 under Section 4.5.

Table 2.7: Stress conditions used to evaluate the unified LoWPAN fabric.

Stress scenario	**Primary parameter**	**What it reveals**
Baseline	Nominal topology, nominal frame size, low loss, moderate scale	Reference performance for latency, delivery ratio and goodput
Hop stress	Path length, hop count	Delay accumulation, reliability limits over multi hop paths
Loss stress	Link level frame loss rate	Throughput degradation, retransmission overhead, delivery ratio impact
Fragmentation stress	Frame size and application payload size	6LoWPAN fragmentation and reassembly behavior, sensitivity to fragment loss
Scale stress	Total node count, traffic contention	Contention effects, routing state behavior and stability in denser deployments

The stress conditions in Table 2.7 are realized through a shared set of network scenarios that is reused across all application flows to enable consistent comparisons. These scenarios, summarized in Table 2.8, define the specific parameter settings used during evaluation.

Each use case is executed under the same scenario set to enable consistent comparison across traffic types.

Each test run captures bidirectional traffic over multiple hops and reports common performance indicators that reflect delay accumulation, reliability and retransmission overhead. The following metrics are collected for each test run:

- **One way latency:** End-to-end bidirectional one way latency measured for each request-response exchange, reported as minimum, maximum and mean values.
- **Delivery ratio:** The proportion of successfully delivered messages relative to the total number of messages sent.
- **MAC layer statistics:** Per node frame counters including transmitted frames, acknowledged frames and received frames. The retransmission count is derived as the difference between transmitted and acknowledged frames.

Figures 2.15, 2.16, and 2.17 show the topologies used for the baseline (S0), hop (S1), and scale (S4) scenarios, respectively. Scenario loss (S2) and fragmentation (S3) uses the same network topology with baseline(S0) but with different network parameters. Figure 2.18 summarizes the evaluation workflow that maps the three application flows onto scenarios S0–S4. Use case specific payload definitions are provided in the corresponding subsections.

Table 2.8: Network scenarios used in the experiments. Each scenario is applied to all three application flows.

ID	Stress focus	Topology	Nodes	Max hops	Link PRR	Frame Size
S0	Baseline	Tree	25	4	0.99	128 B
S1	Hop	Line or multi floor chain	11	10	0.99	128 B
S2	Loss	Tree	25	4	0.85	128 B
S3	Fragmentation	Tree	25	4	0.99	80 B
S4	Scale	Two cluster bottleneck	60	6	0.99	128 B

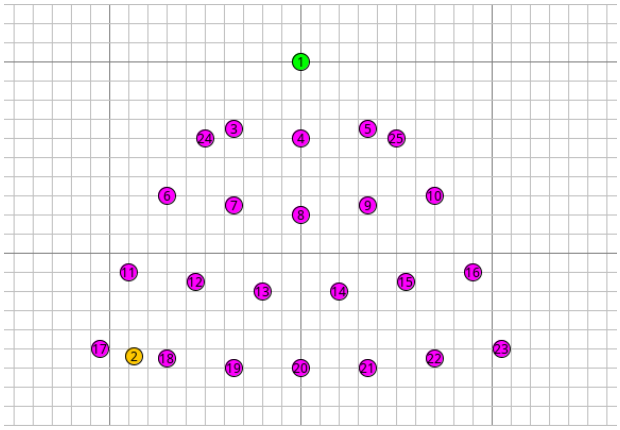

Figure 2.15: Network topology for baseline scenario, S0. Including 25 nodes consisting a communication path of maximum 4 hops.

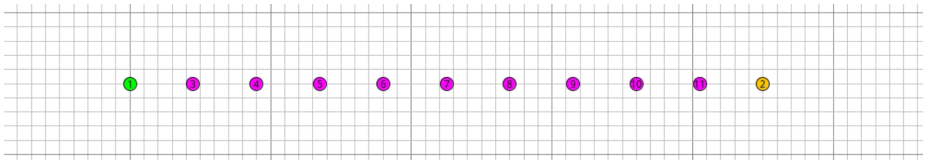

Figure 2.16: Network topology for the hop stress scenario S1. A linear chain topology with 11 nodes, where the communication path between the border router and the gateway spans a maximum of 10 hops.

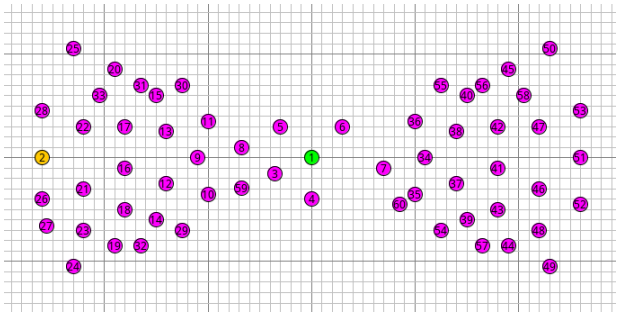

Figure 2.17: Network topology for the scale stress scenario S4. A two-cluster bottleneck topology with 60 nodes, where the communication path between the border router and the gateway spans a maximum of 6 hops.

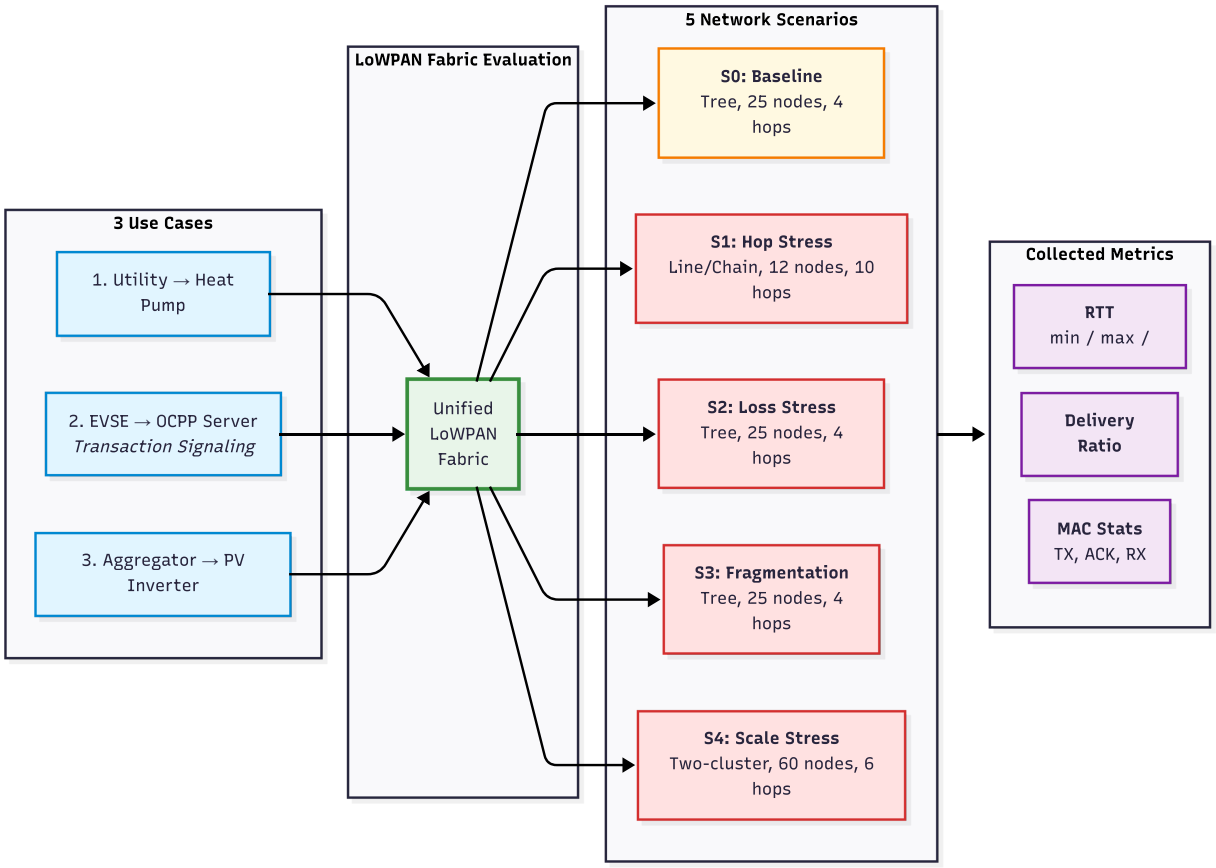

Figure 2.18: Evaluation methodology for the unified LoWPAN fabric. The three application flows are executed over the same fabric under five network scenarios from S0 to S4, which vary topology, node count, maximum hop count, link PRR and frame size to represent baseline operation and stress conditions.

2.3.1 Utility to heat pump control and telemetry exchange

This use case represents a common utility driven interaction with a residential heat pump during demand response or load control operation. The exchange follows a request response pattern where a control query is delivered from the utility side to the heat pump, then the heat pump returns an acknowledgement and an operational telemetry report. This structure captures the two dominant traffic directions observed in practice, downstream delivery of small control commands and upstream reporting of monitoring data. Table 2.9 summarizes the messages used in this flow.

Table 2.9: Message set used in the utility to heat pump control and telemetry exchange.

ID	Message	Function	Direction	Size class
HP1	Control query	Requests current operating state or a parameter update such as a setpoint limit	Utility → Heat pump	Small
HP2	Acknowledgement	Confirms receipt and acceptance of the control query	Heat pump → Utility	Small
HP3	Telemetry report	Returns operational telemetry such as mode, temperature, power draw and status flags	Heat pump → Utility	Medium

Heat pump communication follows the message sequence shown in Figure 2.19. A downstream control query originating at the utility control center enters the fabric through Tun10, is received by BR-1, and is forwarded along the RPL downward route across intermediate mesh nodes until it reaches BR-2. BR-2 then delivers the packet through Tun11 to

the heat pump on the appropriate VLAN segment. The acknowledgement and telemetry messages follow the RPL upward direction, traversing the mesh from BR-2 back to BR-1 before exiting through Tun10 toward the utility backend.

The heat pump flow is modeled using a lightweight request response application protocol carried over UDP. This choice reflects common IoT deployments where control commands and telemetry reports are exchanged using compact messages with low processing overhead. The control query and acknowledgement remain small, while telemetry messages are sized to represent typical monitoring reports.

Table 2.10: Application payload sizes used for the heat pump message set. Payload sizes exclude transport and lower layer headers.

ID	**Message**	**Payload size**
HP1	Control query	48 B
HP2	Acknowledgement	24 B
HP3	Telemetry report	200 B

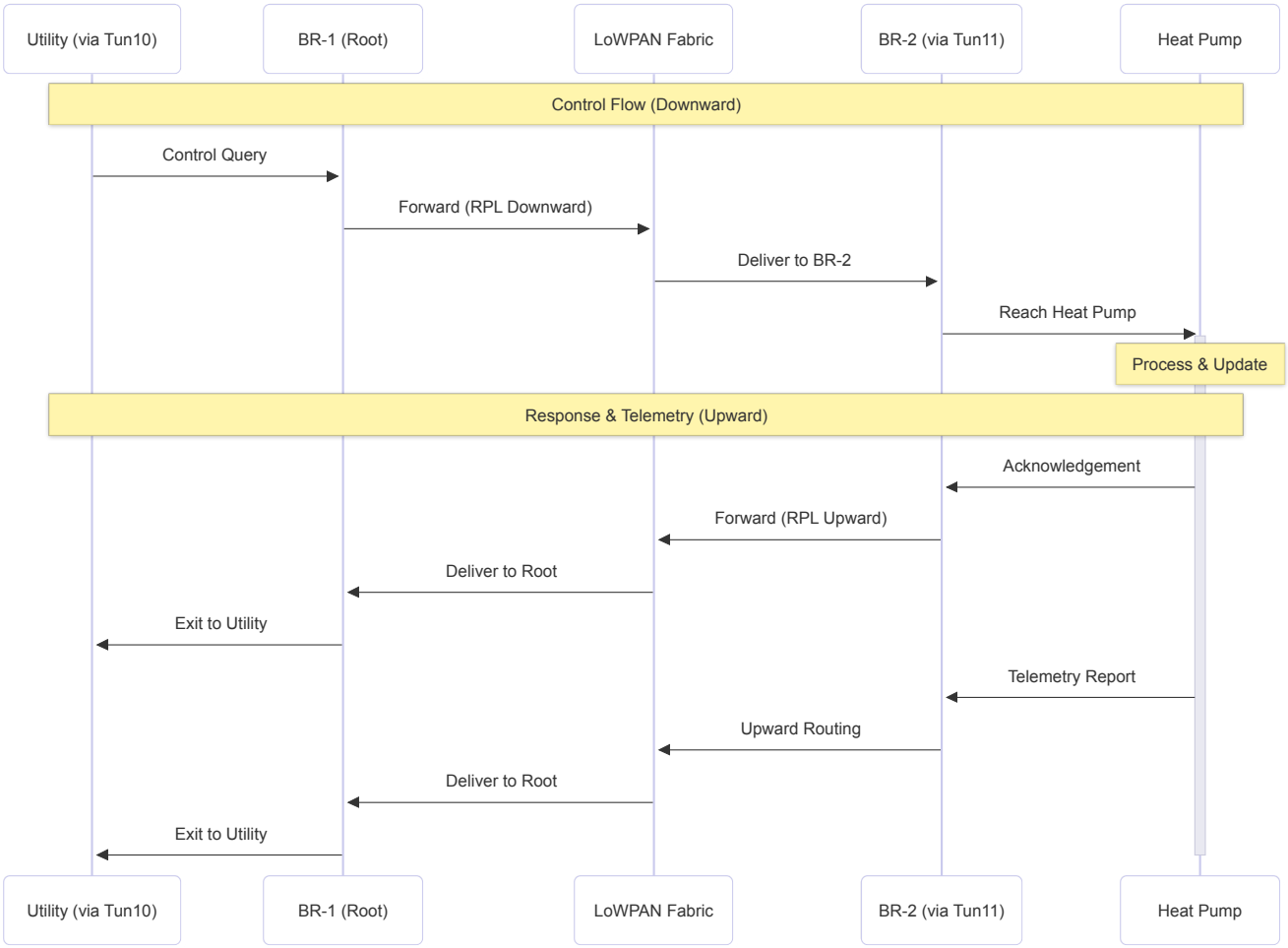

Figure 2.19: Heat pump control and telemetry message sequence over the unified LoWPAN fabric. A control query enters through Tun10 and BR-1, is forwarded along the RPL downward route to BR-2, and reaches the heat pump via Tun11. The acknowledgement and telemetry reports return via RPL upward routing from BR-2 to BR-1 and exit through Tun10.

2.3.2 Evse to ocpp server transaction over the mesh

This use case models a standard EV charging backend interaction where an EVSE exchanges OCPP messages with a remote management server to create and maintain a charging transaction. The flow includes status reporting, transaction initiation, periodic metering updates and transaction termination. This structure captures session oriented signaling with both bursty control messages and recurring upstream reports. Table 2.11 summarizes the messages used in this flow.

Table 2.11: Message set used in the EVSE to OCPP server transaction signaling flow.

ID	Message	Function	Direction	Size class
OC1	StatusNotification (Preparing)	Reports that the EVSE is preparing for a session after cable plug in	EVSE → OCPP server	Small
OC2	StartTransaction	Creates a new charging transaction, includes identifier and connector context	EVSE → OCPP server	Medium
OC3	StartTransaction.conf	Confirms transaction creation and returns a transaction identifier	OCPP server → EVSE	Small
OC4	MeterValues	Periodic metering updates during an active transaction	EVSE → OCPP server	Medium
OC5	MeterValues.conf	Confirms receipt of metering updates	OCPP server → EVSE	Small
OC6	StopTransaction	Terminates the transaction and reports final metering summary	EVSE → OCPP server	Medium
OC7	StopTransaction.conf	Confirms transaction termination	OCPP server → EVSE	Small
OC8	StatusNotification (Finishing, Available)	Reports end of session state transitions after stop	EVSE → OCPP server	Small

EVSE communication follows the message sequence shown in Figure 2.20. Uplink

messages originate at the EVSE and enter the fabric through Tun11 at BR-2, then traverse the RPL mesh toward BR-1 and exit through Tun10 to reach the OCPP server. Downlink confirmations follow the reverse direction, entering through Tun10 at BR-1 and reaching the EVSE through BR-2 and Tun11. This sequence exercises transaction completion behavior and periodic reporting under multi hop conditions and the stress scenarios defined in the experimental methodology.

OCPP is carried over a persistent application session. In this work, OCPP message content is modeled as JavaScript Object Notation (JSON) payloads to reflect common deployments where EVSE to backend traffic includes identifiers, status codes and metering samples. Table 2.12 defines the application payload sizes used for each message.

Table 2.12: Application payload sizes used for the EVSE to OCPP message set. Payload sizes exclude transport and lower layer headers.

ID	**Message**	**Payload size**
OC1	StatusNotification (Preparing)	180 B
OC2	StartTransaction	320 B
OC3	StartTransaction.conf	120 B
OC4	MeterValues	420 B
OC5	MeterValues.conf	80 B
OC6	StopTransaction	360 B
OC7	StopTransaction.conf	90 B
OC8	StatusNotification (Finishing, Available)	200 B

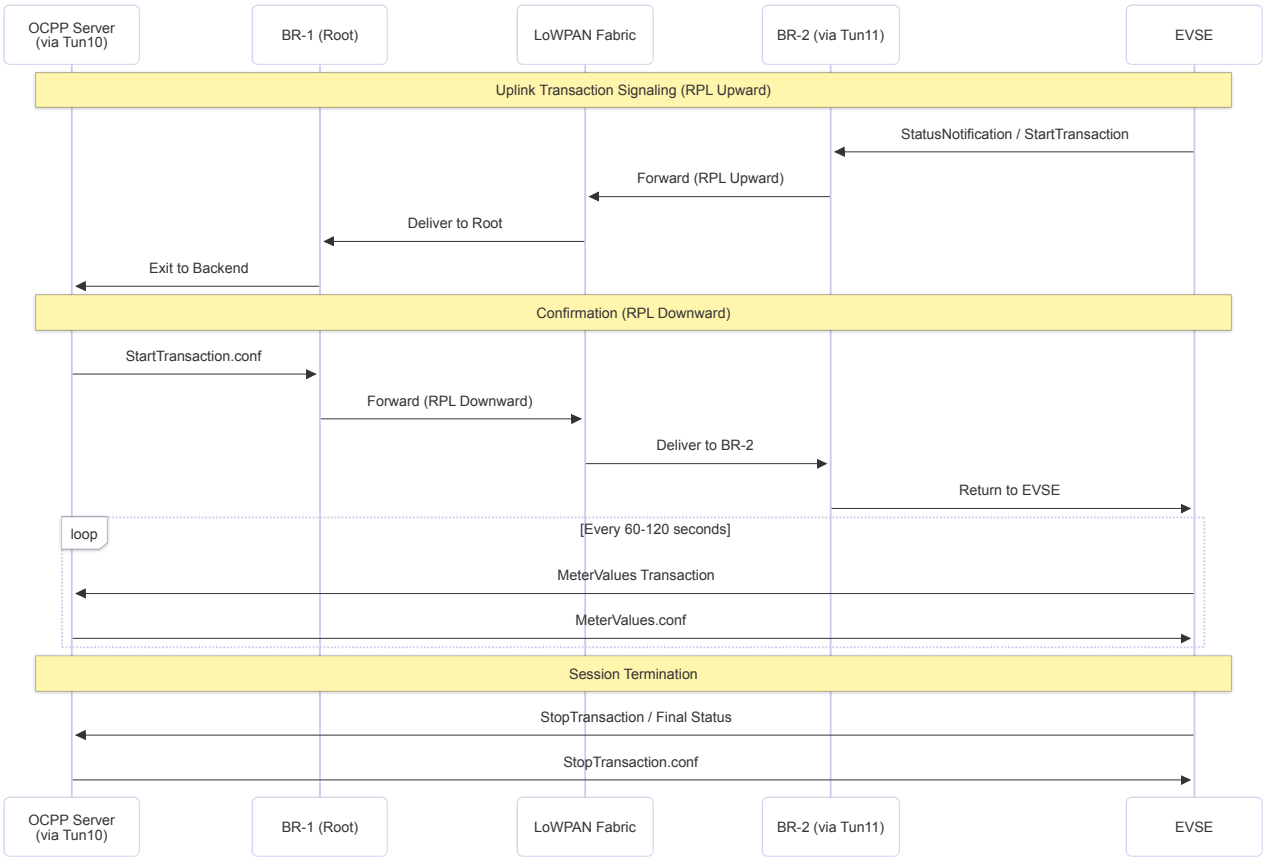

Figure 2.20: EVSE to OCPP server transaction signaling over the unified LoWPAN fabric. Uplink OCPP messages enter through Tun11 at BR-2, traverse the RPL mesh toward BR-1 and exit through Tun10 to reach the backend. Confirmation messages follow the reverse direction back to the EVSE.

2.3.3 Aggregator to pv inverter dispatch and feedback telemetry

This use case models an aggregator driven DER dispatch interaction where active power or reactive power setpoints are delivered to a behind the meter PV inverter and operational telemetry is collected for verification and closed loop coordination. The exchange follows a poll based pattern where the aggregator initiates all transactions. Unlike event driven protocols where devices push telemetry autonomously, Modbus TCP requires the controller to issue explicit requests and await responses. Table 2.13 summarizes the messages used in this flow.

Table 2.13: Message set used in the aggregator to PV inverter dispatch and feedback flow. All transactions are initiated by the aggregator as Modbus TCP follows a request response model.

ID	Message	Function	Direction
INV1	Setpoint write request	Writes a real power limit or reactive power setpoint to the inverter holding registers	Agg → Inv
INV2	Setpoint write response	Modbus write response confirming the setpoint was accepted	Inv → Agg
INV3	Telemetry read request	Requests current measurements and status from the inverter input registers	Agg → Inv
INV4	Telemetry read response	Returns voltage, current, power, frequency, temperature and status values	Inv → Agg

The inverter flow is modeled using Modbus TCP which reflects common field practice for inverter monitoring and control on local IP networks. In this protocol the aggregator

acts as the Modbus client and the inverter acts as the server. Setpoint delivery uses a write multiple registers operation consisting of an INV1 request followed by an INV2 response. Telemetry collection uses a read input registers operation where the aggregator sends an INV3 request and the inverter returns an INV4 response containing measurement data. The telemetry read sequence is repeated at regular intervals during the dispatch window to verify that the inverter has responded to the setpoint and to collect ongoing measurements for closed loop coordination.

Table 2.14 defines the application payload sizes used for this message set. The setpoint write request carries register values for power limit and control mode. The write response is a compact Modbus acknowledgement. The telemetry read request specifies the register range to read while the read response returns a block of input registers containing operational measurements.

Table 2.14: Application payload sizes used for the PV inverter message set. Payload sizes exclude transport and lower layer headers.

ID	**Message**	**Payload size**
INV1	Setpoint write request	24 B
INV2	Setpoint write response	16 B
INV3	Telemetry read request	12 B
INV4	Telemetry read response	120 B

In Figure 2.21 a setpoint dispatch originates at the aggregator backend. The INV1 request is delivered downstream through BR-1 and the RPL mesh toward BR-2 then forwarded to the inverter through Tun11. The inverter processes the setpoint and returns an

INV2 response which traverses the mesh in the upstream direction from BR-2 to BR-1 before reaching the aggregator. Following the setpoint exchange the aggregator issues a series of telemetry reads at fixed intervals. Each INV3 request travels downstream through the mesh and the corresponding INV4 response returns upstream carrying current measurements. This polling sequence exercises both downstream command delivery and upstream data reporting under the stress scenarios defined in the experimental methodology.

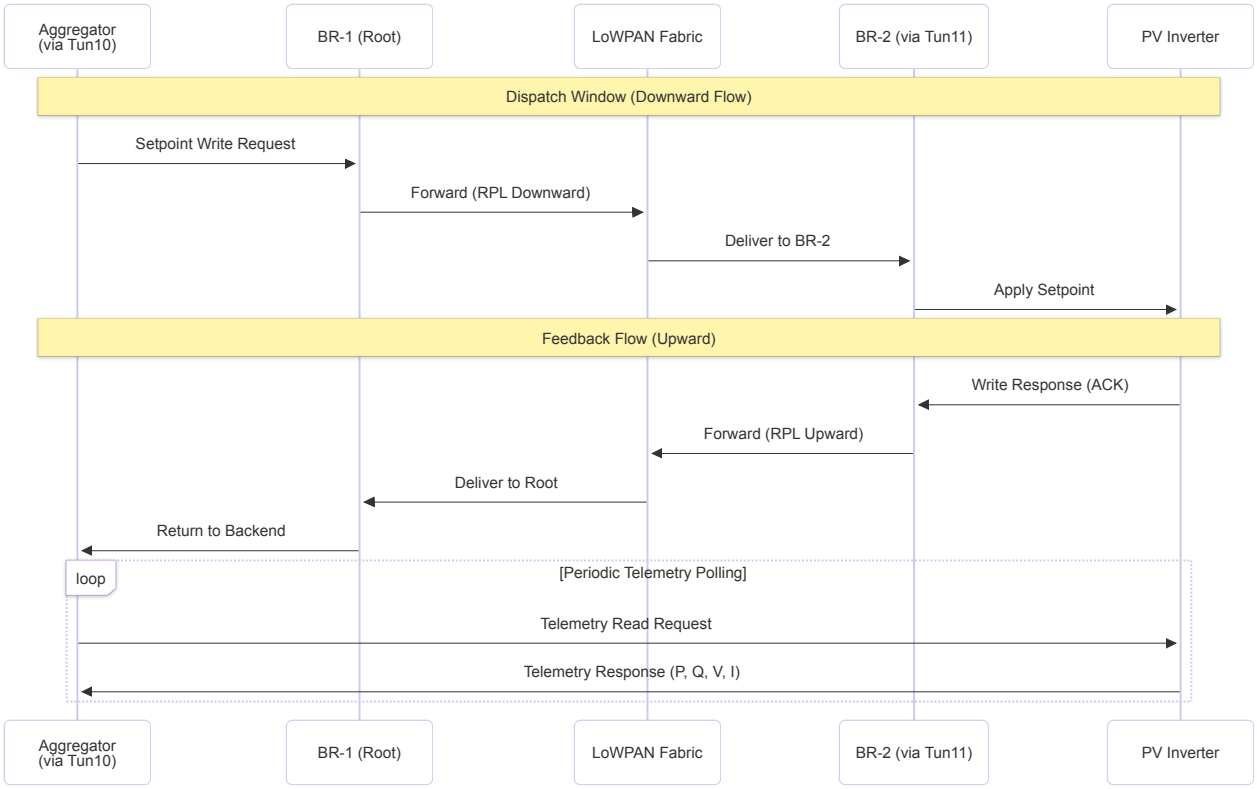

Figure 2.21: Aggregator to PV inverter dispatch and feedback over the unified LoWPAN fabric. Setpoint and telemetry requests are delivered downstream through BR-1 and the RPL mesh to BR-2 then forwarded to the inverter via Tun11. Write and read responses return upstream through BR-2 and the mesh toward BR-1 before exiting through Tun10 to reach the aggregator backend.

3. Data Imputation in Narrowband Networks

3.1 Overview

This chapter evaluates an online data imputation methodology for telemetry collected over a narrowband IPv6 mesh network. The focus is the Electric Vehicle Load Management System (EVLMS), a gateway device that continuously collects measurements from field devices and compensates for missing segments on the fly. In this workflow, the EVLMS receives phase resolved smart meter updates over the mesh network and maintains a time indexed data stream that is used by the EVLMS for real time forecasting and load management functions. The goal is to transform an irregular arrival stream affected by loss and delay into a representation that remains usable for real time analytics and control. Because narrowband telemetry streams can exhibit different missingness mechanisms and rates (e.g., missing completely at random under transient link loss, or bursty loss under congestion), the imputation approach is evaluated across varying missing rates with an emphasis on low latency operation.

The workflow follows two stages. First, incoming updates are organized into a consistent representation and missing entries are identified. Second, short gaps are filled using an online imputation method operating on a rolling window. This design bounds memory and compute requirements and supports near real time operation.

The data collection and imputation workflow was previously presented in the author's publication [145] and is summarized here for completeness. Section 3.2 describes telemetry acquisition at the gateway. Section 3.3 describes missingness detection using a rolling window in Section 3.3.1 and online imputation in Section 3.3.2.

3.2 Data Collection and Measurement Pipeline

The primary data source is an AMI 2.0 smart meter that reports phase resolved electrical consumption. The EVLMS communicates with the meter over Wi-SUN and receives periodic updates. The EVLMS also interfaces with local assets that are relevant to load management. This includes monitoring OCPP messages exchanged with EVSEs to support charge profile adjustments and collecting rooftop solar inverter measurements through Modbus RS-485. These measurements and messages are handled as multiple channels within the same monitoring pipeline.

During continuous monitoring and collection, missing data and latency in receiving updates may occur. In this workflow, the EVLMS maintains a rolling window of recent consumption and solar production values that is used to track recent changes and identify missing entries. When missing values are detected, online imputation is applied as described in Section 3.3.2, and storage is updated with the imputed entries. The rolling window size and the gap ratio can be configured to fit the operating environment.

Figure 3.1 illustrates the telemetry path used in this work. Measurements from the smart meter and local assets are aggregated at the EVLMS gateway and written to storage,

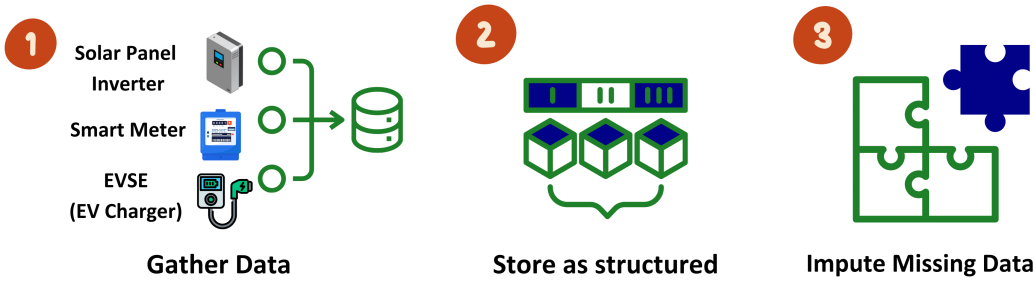

Figure 3.1: Communication architecture used by the EVLMS gateway. Phase resolved smart meter telemetry is collected over Wi-SUN and combined with local signals such as EVSE OCPP messages and rooftop solar inverter measurements over Modbus RS-485.

while missing data detection and online imputation are applied in subsequent parts.

3.3 Missing Data Handling

3.3.1 Rolling window gap detection

As monitoring and collection progress, missing data or latency in receiving updates may occur. A time indexed representation of recent measurements is maintained within a rolling window, and the active window is continuously checked to detect missing entries. When an expected update is not observed within the active window, the corresponding time index is treated as missing and is passed to the online imputation step described in Section 3.3.2.

Missingness is tracked within a rolling window to provide a short term view of data availability with bounded memory. Let the rolling window be represented by a matrix $W \in \mathbb{R}^{k \times d}$, where k is the window length in time steps and d is the measurement dimension (e.g.,

phases and channels). A binary mask $M \in \{0,1\}^{k\times d}$ with entries $M_{i,j} \in \{0,1\}$ indicates whether measurement in channel j, at time index i is observed for $i \in \{1,\ldots,k\}$ and $j \in \{1,\ldots,d\}$, and $\bar{M}$ denotes its complement. For a single time step, let $\mathbf{X} = (X_1,\ldots,X_d) \in \mathbb{R}^d$ denote the measurement vector and $\mathbf{M} \in \{0,1\}^d$ denote the corresponding mask. A partially observed vector $\hat{\mathbf{X}} = (\hat{X}_1,\ldots,\hat{X}_d)$ is formed by replacing missing entries with a sentinel symbol $*$ that denotes an unobserved value. This masking operation follows the entrywise definition in (3.3). In online operation, each new time step contributes a new partially observed sample; equivalently, over a window the gateway maintains a collection of partially observed samples paired with their masks.

Over a rolling window W of length k whose most recent time index is t, the total number of missing measurements in the window is computed directly from the mask M as

$$\mathcal{S} = \sum_{i=t-k+1}^{t} \sum_{j=1}^{d} (1 - M_{i,j}) \tag{3.1}$$

The corresponding missingness ratio is

$$\rho = \frac{\mathcal{S}}{kd} \tag{3.2}$$

These expressions quantify missingness within the rolling window. Specifically, $\mathcal{S}$ gives the total number of missing entries across the $k \times d$ window, while ρ normalizes this count to yield the fraction of missing measurements in the window. In deployment, k and the missingness ratio threshold on ρ are treated as configuration parameters and can be

adjusted to match the characteristics of the target environment. A visual illustration of an imputation process that operates on this defined rolling window is presented in Figure 3.2

3.3.2 Online imputation methods

When missing entries are detected within the rolling window, missing values are imputed to maintain a continuous data stream for real time operation. Three online imputation methods are considered to reflect different modeling assumptions and computational trade-offs. These methods are a Generative Adversarial Imputation Network, k nearest neighbors and MissForest. The three methods represent distribution learning, local similarity based estimation and ensemble based regression.

Using the notation in Section 3.3.1, the objective is to infer hidden entries in near real time while preserving observed entries and updating the active window with the imputed values. For completeness, the entrywise masking operation for a single sample can be written as

$$\hat{X}_j = \begin{cases} X_j, & \text{if } M_j = 1 \\ *, & \text{if } M_j = 0. \end{cases} \tag{3.3}$$

Many classical methods such as random forests and k-nearest neighbors can be viewed through a Fully Conditional Specification (FCS) lens, where each target variable is iteratively predicted from the remaining variables. While effective, this can increase computation when many variables require repeated model fitting. In contrast, generative approaches

such as GAIN train a single model to capture the joint distribution of the data and can therefore remain robust when multiple components are missing simultaneously.

GAIN learns multivariate dependencies from partially observed samples and generates missing entries conditioned on observed entries and the mask. In contrast, the k nearest neighbors baseline imputes missing entries by selecting similar samples based on observed components and aggregating their corresponding values. MissForest performs iterative imputation using tree based ensembles that predict missing entries from observed entries within the window. Figure 3.2 illustrates the rolling window based monitoring and imputation process.

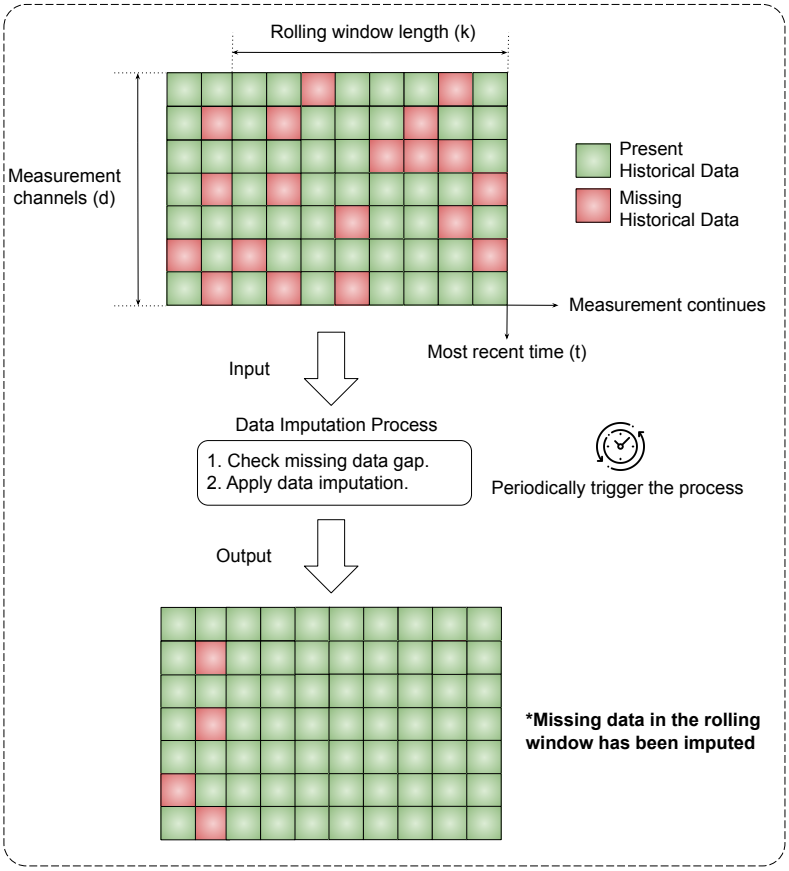

Figure 3.2: Rolling window based missingness monitoring and online imputation process. Missing entries in the active window are detected using a binary mask and are imputed. Adapted from the author's prior publication [145]. © 2025 IEEE.

In deployment, a simple selection policy can be used to choose an imputation method as a function of rolling window size and the observed missingness ratio. In the evaluated workflow, k-NN was used for small windows (e.g., ≈100 points) under low missingness (e.g., $< 10\%$ missing values) due to low query time. For small to moderate windows (e.g., 100–300 points) under moderate missingness (e.g., 10–30%), MissForest was used as a balance between accuracy and runtime. For higher missingness levels or operating regimes characterized by bursty loss, GAIN was used to improve robustness under severe loss.

Table 3.1: Comparison of online imputation technologies

Criteria	**k-NN**	**MissForest**	**GAIN**
Training speed	Fastest	Moderate	Slow
Loss of information	Moderate	High	Low
Learn complex targets	Low	Moderate	High
Query time	Fastest	Moderate	Slow
Scalability	High	Moderate	Low

For completeness, the distance computation used by k-NN can be expressed using the Minkowski family. For two vectors x_i and x_j, the distance can be written as

$$d(x_i, x_j) = \left(\sum_{\ell=1}^{d} |x_{i,\ell} - x_{j,\ell}|^p \right)^{1/p}, \tag{3.4}$$

which recovers Manhattan distance for $p = 1$ and Euclidean distance for $p = 2$. In this work, p is exclusively bounded as $p \in \{1, 2\}$ corresponding to the Manhattan and Euclidean distances. While k-NN is simple, it can become expensive for large windows because it requires scanning the active window at query time, and in high dimensional

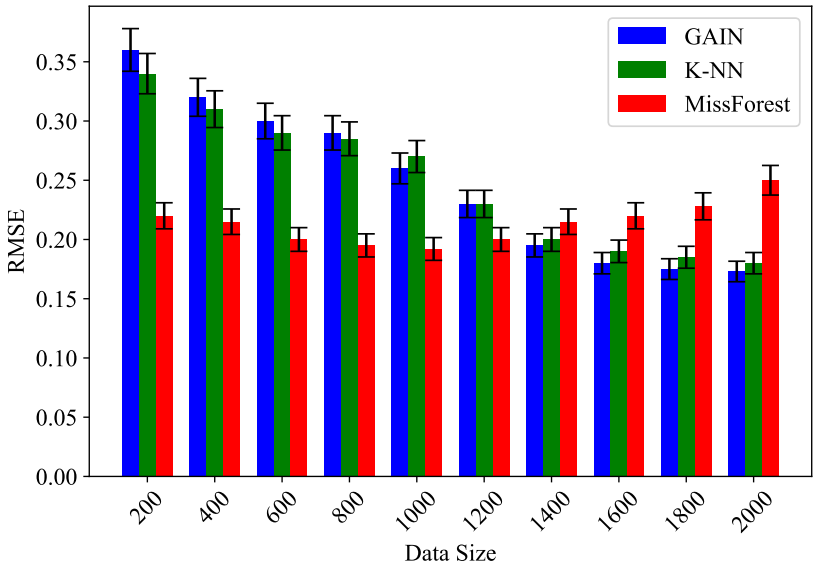

Figure 3.3: Root mean square error comparison of generative adversarial imputation, k nearest neighbors and MissForest under different rolling window sizes for a fixed missingness level. Reproduced from the author's prior publication [145]. © 2025 IEEE.

settings neighbor distances can become less discriminative.

MissForest relies on random forest regressors/classifiers trained on bootstrap samples; at each split, a random subset of predictors is considered (often on the order of $\sqrt{p}$ when p predictors are available). MissForest iterates through variables with missingness and refines imputations until convergence, which can preserve cross variable structure but may increase runtime as window sizes and missing rates grow.

GAIN employs two neural networks, a generator $\mathbb{G}$ that proposes imputations and a discriminator $\mathbb{D}$ that attempts to distinguish observed from imputed entries. Training follows an adversarial minimax objective; at convergence, the generator produces imputa-

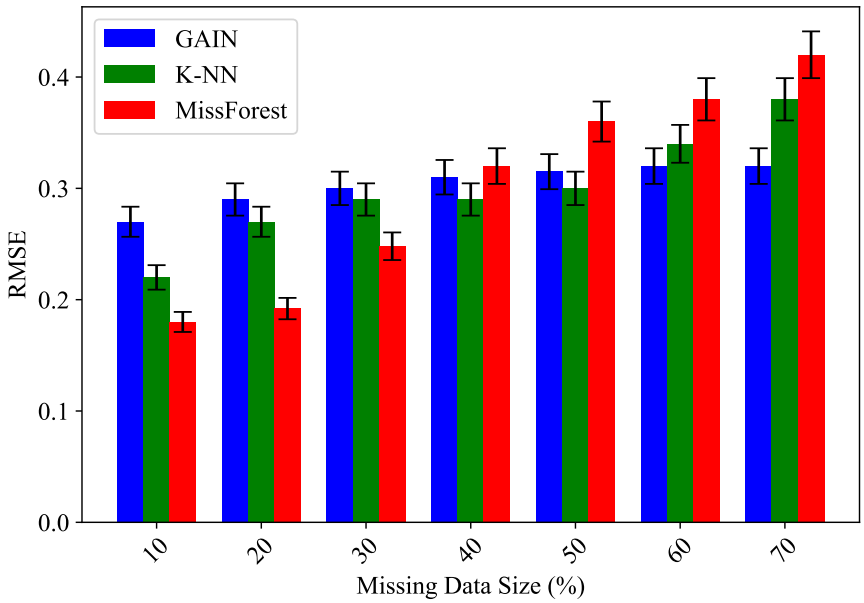

Figure 3.4: Root mean square error comparison of generative adversarial imputation, k nearest neighbors and MissForest under increasing missingness for a fixed rolling window size. Reproduced from the author's prior publication [145]. © 2025 IEEE.

tions that are statistically consistent with the observed data distribution, which improves robustness under higher missingness but typically increases computational cost.

Algorithms 1, 2, and 3 summarize the MissForest procedure, random forest prediction, and GAIN based online imputation used in this Thesis. For clarity, the main steps of each algorithm and the meaning of the key parameters are described below.

MissForest is an iterative imputation procedure that fills missing values by repeatedly training a random forest model for each feature that contains missing entries. The inputs are a data matrix X and a binary mask M, where an entry of M equal to one indicates that the corresponding value in X is observed, and an entry of M equal to zero indicates that

it is missing. The output is an imputed matrix X_{imp} in which missing entries have been replaced by predicted values. The procedure begins with an initialization step in which missing entries in X_{imp} are filled using a simple rule such as the mean or median. The algorithm then cycles through each feature that contains missing entries. For a selected feature, the rows are separated into two groups based on the mask: rows where that feature is observed and rows where it is missing. A random forest model is trained using the observed rows, with the selected feature as the prediction target and the remaining features as predictors. The trained model is then applied to the rows where the selected feature is missing, and the predicted values are written into the corresponding locations of X_{imp}. This process is repeated across features and iterated until the imputed values change only slightly between iterations, which is treated as convergence.

Algorithm 2 summarizes how a random forest produces a prediction for a single query input. The training data are denoted by $\mathbf{X}$, the query sample is denoted by x, and the number of bootstrap samples is denoted by B. A bootstrap sample is formed by randomly sampling rows from $\mathbf{X}$ with replacement, and one decision tree is trained on each bootstrap sample. During tree construction, each split considers only a random subset of the available predictor variables, which reduces correlation across trees and improves generalization. After training, each tree produces an individual prediction for the query sample. The forest output is obtained by aggregating the tree predictions. For regression, this aggregation is the average of the B tree predictions. For classification, aggregation is typically performed by majority vote or by averaging class probabilities and selecting the most likely class.

GAIN is a generative imputation method that uses two neural networks, a generator

and a discriminator, to infer missing entries in partially observed data. The input to Algorithm 3 is a rolling window of measurements denoted by $\hat{X}$ with dimensions k by d, where k is the number of time steps in the window and d is the number of measurement channels. The corresponding binary mask is denoted by M and indicates which entries are observed and which are missing. The algorithm also takes a noise source denoted by Z, a hint rate denoted by p_h, and trained network parameters for the generator $\mathbb{G}$ and discriminator $\mathbb{D}$. During training or fine tuning, noise is inserted into the missing positions so that the generator receives a complete input tensor, while the mask is provided so the generator can condition its imputations on which entries are observed. A hint matrix is formed by revealing a random subset of mask entries to the discriminator, controlled by the hint rate p_h. The discriminator is trained to identify which entries are observed and which entries were generated, while the generator is trained to produce imputations that resemble observed data and are difficult for the discriminator to distinguish. After optimization, imputation is performed by running the generator once on the current window input and replacing only the missing entries, while keeping observed entries unchanged. The resulting imputed window is denoted by X_{imp} and can be used to update the rolling buffer for downstream analytics and control.

The three methods are evaluated under different rolling window sizes and missing rates to characterize their behavior in the target operating conditions. Across the evaluated conditions, GAIN exhibited more stable performance as missingness increased. The comparative performance of the three methods is summarized in Figures 3.3 and 3.4, which compare root mean square error under different rolling window sizes and missingness rates.

Algorithm 1: MissForest Algorithm

Input: Dataset X with missing values and mask M

Output: Imputed dataset X_{imp}

Initialization: Initialize missing entries in X_{imp} (e.g., mean/median);

while not converged **do**

  for each feature j with missing entries **do**

    $I_{\text{obs}}^{(j)} \leftarrow \{i : M_{i,j} = 1\}, \quad I_{\text{mis}}^{(j)} \leftarrow \{i : M_{i,j} = 0\}$;

    $X_{\text{train}}^{(j)} \leftarrow X_{\text{imp}}[I_{\text{obs}}^{(j)}, \{-j\}]$;

    $y_{\text{train}}^{(j)} \leftarrow X_{\text{imp}}[I_{\text{obs}}^{(j)}, j]$;

    Train a random forest model f_j using $(X_{\text{train}}^{(j)}, y_{\text{train}}^{(j)})$;

    $X_{\text{test}}^{(j)} \leftarrow X_{\text{imp}}[I_{\text{mis}}^{(j)}, \{-j\}]$;

    Predict missing values $\hat{y}^{(j)} \leftarrow f_j(X_{\text{test}}^{(j)})$;

    Update $X_{\text{imp}}[I_{\text{mis}}^{(j)}, j] \leftarrow \hat{y}^{(j)}$;

  end

end

Algorithm 2: Random forest prediction

Input: Training data $\mathbf{X} = \{(x_i, y_i)\}_{i=1}^{N}$, query sample x, number of trees B, number of candidate predictors per split m

Output: Forest prediction $\hat{y}$

for $b = 1$ **to** B **do**

  Draw a bootstrap sample $\mathbf{Z}_b^{\star}$ of size N from $\mathbf{X}$ (sampling with replacement);

  Train a decision tree T_b on $\mathbf{Z}_b^{\star}$ using the following procedure at each node until a stopping criterion is met:;

    (i) Select m predictors at random from the p available predictors;

    (ii) Choose the best split (variable and split-point) among the m candidates according to the split criterion;

    (iii) Split the node into two child nodes;

end

for $b = 1$ **to** B **do**

  $\hat{y}_b \leftarrow T_b(x)$;

end

return Aggregated prediction $\hat{y} = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_b$;

Algorithm 3: GAIN based online imputation over a rolling window

Input: Partially observed window $\hat{X} \in \mathbb{R}^{k \times d}$, mask $M \in \{0,1\}^{k \times d}$, noise prior Z, hint rate p_h, trained networks $\mathbb{G}, \mathbb{D}$

Output: Imputed window X_{imp}

;

Fine tune $\mathbb{G}, \mathbb{D}$ on the current window for T steps;

for $t = 1$ **to** T **do**

  Sample minibatch $(\hat{X}_b, M_b)$ from $(\hat{X}, M)$;

  Sample noise $Z_b \sim p(Z)$ with shape of $\hat{X}_b$;

  Construct corrupted input $X_b^{\text{in}} \leftarrow M_b \odot \hat{X}_b + (1 - M_b) \odot Z_b$;

  Sample hint selector $B_b \sim \text{Bernoulli}(p_h)$ (entrywise);

  Construct hint matrix $H_b \leftarrow B_b \odot M_b + (1 - B_b) \odot 0.5$;

  `// Discriminator update`

  $\tilde{X}_b \leftarrow M_b \odot \hat{X}_b + (1 - M_b) \odot \mathbb{G}(X_b^{\text{in}}, M_b)$;

  Update $\mathbb{D}$ to minimize

  $$L_D = -\mathbb{E}\left[M_b \odot \log \mathbb{D}(\tilde{X}_b, H_b) + (1 - M_b) \odot \log(1 - \mathbb{D}(\tilde{X}_b, H_b))\right];$$

  `// Generator update`

  Update $\mathbb{G}$ to minimize

  $$L_G = -\mathbb{E}\left[(1 - M_b) \odot \log \mathbb{D}(\tilde{X}_b, H_b)\right] + \alpha\, \mathbb{E}\left[M_b \odot \|\hat{X}_b - \tilde{X}_b\|_2^2\right];$$

end

;

Imputation (forward pass):;

Sample noise $Z \sim p(Z)$;

$X^{\text{in}} \leftarrow M \odot \hat{X} + (1 - M) \odot Z$;

$X_{\text{imp}} \leftarrow M \odot \hat{X} + (1 - M) \odot \mathbb{G}(X^{\text{in}}, M)$;

return X_{imp};

When GAIN is used for online imputation, storage is updated with the filled entries.

To quantify imputation performance, Root Mean Square Error (RMSE) is reported for numerical variables and micro averaged F_1 is reported for categorical variables (or downstream classification tasks that rely on imputed categorical inputs). For numerical

evaluation, RMSE is computed over the imputed entries (i.e., indices where $M_{i,j} = 0$) and is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{3.5}$$

where N is the number of evaluated points, y_i denotes the ground truth value, and $\hat{y}_i$ denotes the imputed estimate.

For categorical evaluation, the micro averaged F_1 score is used, which aggregates true positives, false positives, and false negatives across classes before computing a single F_1 value:

$$F_1^{\mu} = \frac{2\sum_{c=1}^{C} TP_c}{2\sum_{c=1}^{C} TP_c + \sum_{c=1}^{C} FP_c + \sum_{c=1}^{C} FN_c}. \tag{3.6}$$

For single label multi class categorical prediction, F_1^{μ} is equal to overall classification accuracy, since each sample contributes exactly one predicted label.

3.4 Data Imputation on Constrained Edge Platforms

The online imputation workflow described in Section 3.3 is presented under the assumption that the EVLMS gateway may provide sufficient compute and memory to execute the selected imputation method in near real time. In some deployments, parts of the telemetry pipeline can be placed closer to field devices (e.g., embedded controllers integrated with meters, power electronics, or EVSE subsystems), where compute, memory, and energy budgets are substantially smaller. This section summarizes model compression and deployment techniques by which imputation models is executed on resource limited

ARM-based microcontrollers while maintaining acceptable accuracy under missingness.

Edge imputation can be shaped by constraints that differ from server-class execution. Limited Flash and Static Random Access Memory (SRAM) may bound model size and intermediate activation buffers, real-time operation may bound inference latency, and energy limits may constrain continuous inference duty cycles. Under these constraints, compact parametric imputers and post training transformations can be considered to reduce memory footprint and arithmetic cost.

3.4.1 Lightweight imputation models for microcontrollers

Resource constrained microcontrollers impose strict limits on Flash and SRAM, which makes many classical imputation methods difficult to deploy. In streaming settings, a history buffer is typically required regardless of the imputation strategy; consequently, query time computation often becomes a primary constraint rather than storage alone. Instance based approaches such as k nearest neighbors require a distance scan over the retained window at each query, which leads to an $O(ND)$ cost and latency that increases with window size. Iterative ensemble based methods such as MissForest further increase computational burden because ensembles are repeatedly evaluated across variables until convergence. Under these properties, compute latency and energy demand increase and firmware integration is complicated; consequently, compact forward pass models with predictable execution cost are often preferred in resource limited deployments.

An alternative approach is to learn an imputation function offline and to deploy only

a compact forward pass on device. Under this paradigm, training and model selection are performed in a higher resource environment, and the embedded artifact is limited to deterministic inference code and a small set of parameters. The on device implementation also includes the required preprocessing so that inference is executed from raw features without reliance on a separate runtime.

In this work, a gradient boosted tree regressor is trained in Python together with a preprocessing pipeline, and the resulting model is exported to pure C for embedded inference. Model capacity is controlled through the number of trees so that smaller variants can be obtained that fit within typical microcontroller memory budgets. The role of numeric representation and compression is treated separately in Section 3.4.2, where reduced precision and fixed point encodings are introduced as additional mechanisms by which footprint is reduced and arithmetic cost is modified. Deployment level considerations and quantitative results are then discussed in Section 3.4.3, which reports Flash and SRAM footprints, accuracy, and per inference latency under an Arm Cortex M4 oriented evaluation setup.

3.4.2 Compression and encoding for ARM-based microcontrollers

On ARM microcontrollers, dominant deployment constraints are set by Flash capacity for storing code and parameters and by SRAM capacity for transient state during inference. Under these constraints, compression and numeric encoding schemes can be used to fit learned models within a given memory budget while maintaining acceptable accuracy. Two practical levers that are available are model structure compression and arithmetic representation.

Structure level compression can reduce the number of operations and the number of stored parameters. For tree ensembles, this effect can be obtained by limiting the number of trees and restricting tree depth, thereby reducing the number of branch nodes and leaf values that are stored and evaluated. This behavior can be interpreted as pruning-like, in the sense that lower utility decision paths are removed or avoided and overall model capacity is reduced, which can decrease both Flash footprint and evaluation cost at inference time. Unlike approaches that rely on a dedicated inference runtime, exporting a fixed model structure to pure C may allow the footprint to be made explicit and audited through the resulting binary sections.

Numeric encoding may target the representation of thresholds, intermediate computations, and outputs. Floating point arithmetic can offer implementation simplicity but may incur higher compute cost and code size on microcontrollers, particularly when double precision is used. Single precision float32 is reduce arithmetic cost relative to float64 in many toolchains. Fixed point encodings can further reduce Flash and SRAM usage by representing inputs and thresholds as integers with an associated scale factor and by implementing the forward pass using integer operations. In this work, two fixed point variants are considered, fxp16 and fxp8, in which inputs and thresholds are quantized to int16 and int8, respectively, while accumulations are kept in wider integer types to support numerical stability.

These encoding choices are involve an accuracy–footprint–latency tradeoff. Coarser quantization can reduce parameter size and could improve throughput. Compression and encoding are evaluated jointly with model capacity control, and the resulting design points

are compared in a later subsection using Flash and SRAM footprints, accuracy metrics, and per inference latency under a Cortex M4 oriented evaluation setup.

3.4.3 Embedded deployment considerations and empirical results

This subsection summarizes the deployment pipeline and the evaluation protocol used to characterize accuracy and resource usage of the proposed lightweight models under microcontroller oriented constraints. The objective is to quantify tradeoffs between memory footprint, accuracy, and inference latency across model capacity and numeric encoding choices.

Model training and preprocessing are performed offline in Python. The preprocessing pipeline is treated as part of the model artifact so that the deployed implementation receives a fixed size feature vector with consistent semantics. After training, the learned model is exported to pure C and integrated into a minimal firmware. Under this approach, dependence on external inference runtimes is avoided and the resulting binary can be inspected directly.

Evaluation is conducted in a Cortex M4 Quick Emulator (QEMU) environment to provide a reproducible embedded oriented benchmark while keeping the toolchain and measurement pipeline consistent across variants. The evaluation setup and measurement configuration are summarized in Table 3.2. Memory footprint is reported using Flash for weights and code and SRAM for state and activations, where SRAM is derived from static

Table 3.2: Cortex M4 QEMU evaluation setup and measurement configuration.

Item	Value
QEMU binary	`qemu-system-arm`
Machine	`mps2-an386`
CPU	`cortex-m4`
Console and I O	`-nographic` with semihosting enabled
Firmware interface	bare metal ELF loaded via `-kernel`
Embedded test slice	2100 rows, 16 features per row
Latency metric	QEMU host wall time per inference
Memory metrics	Flash uses ELF sizes and SRAM uses `.data+.bss` plus stack peak.
Numeric variants	f32, fxp16, fxp8

sections and the stack high water mark. Accuracy is assessed using mean squared error and R^2 on a fixed test slice, and inference latency is reported as QEMU host wall time per inference. Variants are produced by controlling tree count to emulate pruning like structural compression and by switching between float32 and fixed point implementations such as fxp16 and fxp8 to evaluate encoding effects.

Table 3.3 outlines memory footprint, latency, and predictive accuracy for the exported embedded models across two controllable factors, ensemble capacity (tree count) and numeric representation (float32, fxp16, fxp8). This tabulation supports a direct, configuration level comparison in which Flash and SRAM requirements are read alongside R^2 and max normalized Mean Square Error (MSE), with lower MSE and higher R^2 indicating better predictive performance under the same test slice. At fixed tree count, numeric precision primarily affects Flash footprint while leaving accuracy largely unchanged: for example, at 20 trees Flash decreases from 325.19 kB (float32) to 67.38 kB (fxp8), while R^2 remains essentially unchanged (0.851628 vs. 0.851868) and max normalized MSE remains at the

same order. Similar Flash reductions are observed at 50 trees (406.81 kB to 114.23 kB) and at 10 trees (293.61 kB to 49.22 kB), indicating that reduced precision encodings capture substantial Flash savings with minimal impact on predictive metrics in this evaluation. Reducing tree count lowers Flash footprint but is accompanied by a degradation in predictive performance: R^2 decreases and MSE increases as the ensemble is reduced (e.g., from 50 trees to 5 trees). The SRAM footprint remains within a narrow range (approximately 1.17–1.41 kB) across all configurations, indicating that SRAM usage is dominated by a small fixed working state and stack/static overheads rather than by ensemble size. The QEMU wall time latency remains in the microsecond range across all variants, with values spanning roughly 11–41 μs per inference; the table therefore establishes that the evaluated configurations lie within a consistently low latency regime while still exhibiting clear Flash–accuracy tradeoffs across tree count and precision.

Figure 3.5 summarizes Flash footprint across model configurations. Flash usage is observed to increase with tree count for all precision modes, which is consistent with larger ensembles requiring more stored thresholds, leaf values, and associated control logic in the exported C implementation. For a fixed tree count, lower precision variants (fxp16 and fxp8) are observed to require substantially less Flash than float32, which is consistent with thresholds and leaf values being represented more compactly and with reduced dependence on floating point support code. This figure therefore highlights Flash as a primary constraint that is directly affected by both model capacity and numeric encoding.

Table 3.3: QEMU Cortex M4 inference results (2100 each). Flash reflects weights and code. SRAM reflects static state plus stack high water mark. Model IDs omit the common prefix `xgb_lag5_v1` for readability. MSE values are max normalized.

Model ID	**Trees (-)**	**Precision (-)**	**Flash (kB)**	**SRAM (kB)**	**QEMU (μs/inf)**	**R^2 (-)**	**MSE (Max-Norm)**
t50_d3	50	f32	406.81	1.414	28.99	0.857556	2.048×10^{-3}
t50_d3	50	fxp16	169.74	1.309	40.79	0.857511	2.048×10^{-3}
t50_d3	50	fxp8	114.23	1.309	39.54	0.856574	2.062×10^{-3}
t20_d3	20	f32	325.19	1.297	20.67	0.851628	2.133×10^{-3}
t20_d3	20	fxp16	106.65	1.191	36.31	0.851633	2.133×10^{-3}
t20_d3	20	fxp8	67.38	1.195	12.91	0.851868	2.129×10^{-3}
t10_d3	10	f32	293.61	1.273	14.23	0.839621	2.305×10^{-3}
t10_d3	10	fxp16	84.69	1.172	12.19	0.839621	2.305×10^{-3}
t10_d3	10	fxp8	49.22	1.172	11.70	0.843687	2.247×10^{-3}
t7_d3	7	f32	287.84	1.258	13.48	0.802280	2.842×10^{-3}
t7_d3	7	fxp16	80.60	1.172	11.43	0.802280	2.842×10^{-3}
t7_d3	7	fxp8	46.72	1.172	11.29	0.811633	2.708×10^{-3}
t5_d3	5	f32	285.76	1.258	13.05	0.713672	4.116×10^{-3}
t5_d3	5	fxp16	78.66	1.172	11.52	0.713672	4.116×10^{-3}
t5_d3	5	fxp8	45.39	1.172	11.07	0.728972	3.896×10^{-3}

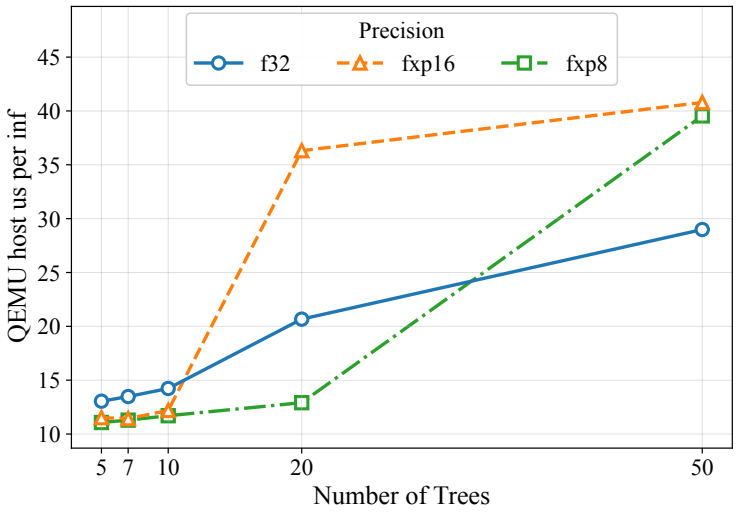

Figure 3.8: QEMU host wall time per inference as a function of tree count for each numeric precision mode.

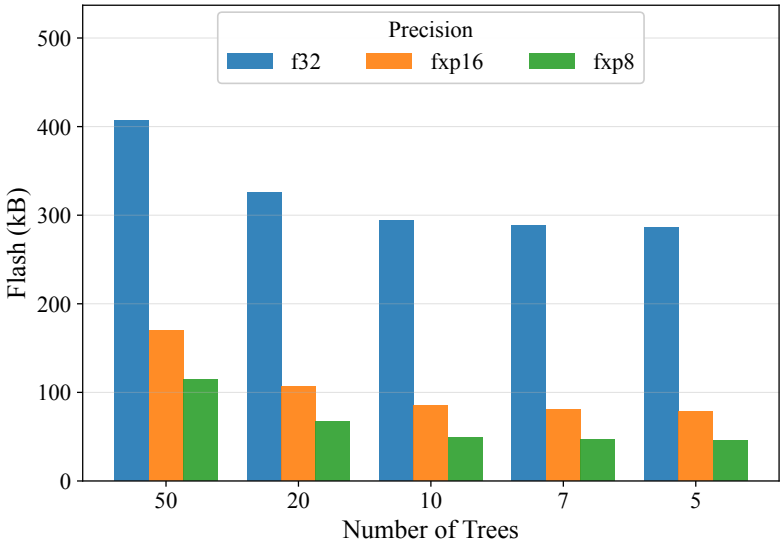

Figure 3.5: Flash footprint across model configurations. Flash denotes weights and code size. Bars are grouped by tree count and colored by numeric precision.

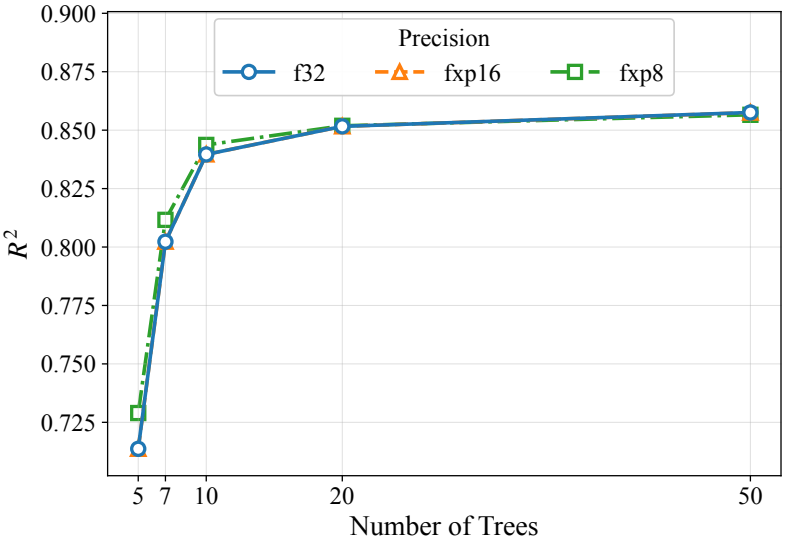

Figure 3.9: Predictive performance measured by R^2 as a function of tree count for each numeric precision mode.

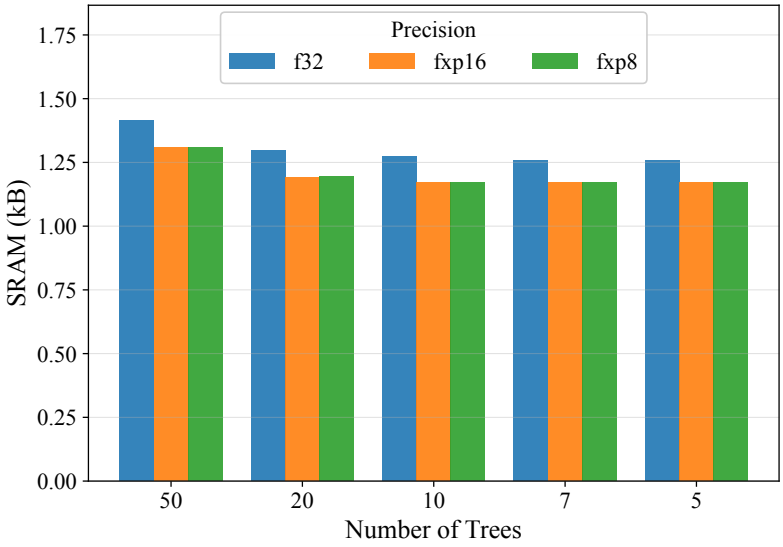

Figure 3.6: SRAM footprint across model configurations. SRAM denotes static state plus stack high water mark. Bars are grouped by tree count and colored by numeric precision.

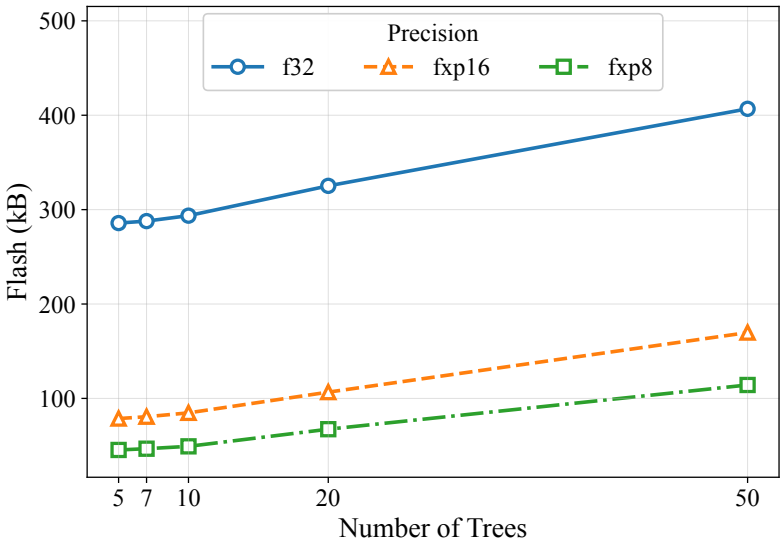

Figure 3.7: Flash footprint as a function of tree count for each numeric precision mode.

Figure 3.6 summarizes SRAM footprint across the same configurations. SRAM usage is observed to vary only slightly with tree count and precision mode and remains within a narrow range across all evaluated variants. This pattern is consistent with inference requiring a small fixed working state and with SRAM being dominated by static buffers and stack usage in the evaluation harness rather than by the number of trees or the numeric precision of stored parameters. The figure therefore indicates that, under the evaluated deployment pipeline, SRAM is comparatively insensitive to capacity control and encoding choices.

Figure 3.7 presents Flash footprint as a function of tree count for each numeric precision mode and provides the corresponding trend view. For each mode, Flash footprint is observed to increase approximately monotonically with tree count, thereby making the Flash cost of additional model capacity explicit. The separation between curves also shows that, at any fixed tree count, reduced precision encodings remain associated with lower Flash footprint than float32, reinforcing the effect observed in Figure 3.5.

Figure 3.8 reports QEMU host wall time per inference as a function of tree count for each precision mode. Latency values are observed to remain on the order of tens of microseconds across the evaluated configurations, which corresponds to on the order of 10^3 CPU cycles at typical Cortex M4 clock rates. While an overall increase with tree count is expected because more trees imply more node evaluations, small non monotonic variations are also present and can be attributed to microsecond scale measurement noise and host load effects in the QEMU based setup. In this figure, the primary value is therefore the empirical confirmation that inference remains within a narrow low latency range for the

tested capacity and encoding settings.

Figure 3.9 reports predictive performance measured by R^2 as a function of tree count for each precision mode. R^2 is observed to increase as tree count increases, with the largest gains occurring when moving from very small ensembles to moderate ensembles and smaller incremental improvements at higher tree counts. This saturation behavior indicates diminishing returns in accuracy with additional trees beyond moderate ensemble sizes. In addition, the proximity of the curves across precision modes indicates that reduced precision encodings maintain similar R^2 to float32 under the evaluated conditions, suggesting that substantial Flash reductions are attainable with limited impact on predictive performance.

4. SG-Mesh Test Suite

4.1 Overview

Conducting wireless network research in the smart grid domain involves challenges related to physical device access, configuration, deployment and reliable data collection. The effort is further increased by the need to cover large geographical areas while integrating diverse device types with varying communication interfaces. These practical constraints can limit the scope and reproducibility of experimental studies.

Model-based environments can be used to address some of these limitations. The Cooja network simulator provides a platform for emulating wireless communication dynamics within constrained IPv6 networks. However, Cooja operates at a fundamental level that requires low level system programming in C for every application component. In addition, due to its emulation scope, Cooja limits simulations to devices that reside within the mesh network. Interactions with elements outside the mesh network cannot be represented natively. These constraints can increase the effort required to evaluate higher layer protocols or application level behaviors when the full stack must be implemented within the simulator.

To bridge this gap, an integrated test suite(SG-Mesh) was developed that preserves the flexibility of system design within Cooja while providing higher layer tools to reduce implementation effort and extend evaluation capabilities. The resulting infrastructure enables holistic observation of behind the meter assets alongside 6LoWPAN network dynamics.

The framework supports iteration across configurations and collection of measurement results without requiring firmware modifications for each experiment.

The test suite was designed to support extension and is made available as an open resource for the research community. This availability is intended to support reuse and adaptation of the infrastructure across different use cases. The test suite is publicly available as open source software.[1]

The test suite provides an end to end environment for evaluating behind the meter asset communication over IPv6 mesh networks. The IPv6 mesh communication is simulated in the Cooja framework. The test suite builds on top of Cooja and adds host-based components that enable holistic evaluation of more complex scenarios, including integration of behind the meter assets. The system runs across both Cooja and the host operating system. Its core capabilities span network emulation, device simulation and measurement instrumentation.

At the network layer, the test suite implements a custom dual border router architecture within the Cooja framework, as described in Chapter 2. The test suite provides two independent SLIP tunnels that connect the simulated mesh to the host system. Real-world behind the meter communication is executed on the host, on both the utility side and the device side, and is routed through the emulated mesh network. Linux network namespaces isolate the utility side from the device side along the communication path. This design ensures that all traffic traverses the complete mesh network rather than being

[1] `https://github.com/BaturayOnural/SG-Mesh-Test-Suite`

routed through local shortcuts.

For device emulation, the test suite includes simulators for three representative behind the meter assets: a heat pump using UDP based telemetry, a photovoltaic inverter using Modbus TCP and an electric vehicle charger using WebSocket based OCPP. Each simulator operates within an isolated VLAN namespace and implements realistic message structures with configurable payload sizes.

The measurement framework captures per message timestamps at multiple points along the communication path. One way latency is computed using embedded timestamps rather than round trip estimates. Additionally, MAC layer statistics including transmission counts, acknowledgments and retransmissions are collected from each node in the mesh.

Test automation is provided through an interactive script that orchestrates the entire workflow from firmware compilation to result collection. Multiple stress scenarios can be selected at startup to evaluate performance under varying hop counts, frame loss ratios, fragmentation constraints and network scale.

The remainder of this chapter is structured as follows. Section 4.2 presents the overall system architecture and explains how the simulated IPv6 mesh is integrated with host-based device emulators. Section 4.2.1 details the network topology and the complete protocol stack, including the dual border router setup, SLIP tunnels and VLAN segmented device namespaces. Section 4.2.2 describes the software components that comprise the test suite, covering the Contiki-NG firmware variants, Python based device simulators, utility side controllers and the orchestration layer. Section 4.3 explains the measurement

methodology, including embedded timestamp based one way latency calculation, application layer delivery metrics and MAC layer statistics collection. Section 4.4 provides the operational workflow and command line interface for executing experiments and collecting results. Finally, Section 4.5 summarizes the experimental results and discusses the impact of hop count, frame loss, fragmentation and network scale across the evaluated protocols.

4.2 System Architecture

The test suite architecture integrates a simulated IPv6 mesh network with host based device emulators through a layered design. The architecture supports evaluation of smart grid communication patterns while providing control over network conditions and measurement instrumentation. This section describes the network topology, protocol stack and software components that comprise the system.

4.2.1 Network topology and protocol stack

The network topology implements a dual border router configuration that creates physically separated communication endpoints for utility side and device side applications. This design reflects deployment scenarios where aggregator systems communicate with distributed energy resources through intermediate wireless infrastructure. Figure 4.1 illustrates the complete network topology. The system consists of four primary segments: the utility network, the IPv6 mesh, the gateway bridge and the device networks.

On the utility side, backend controller applications execute on the host system and

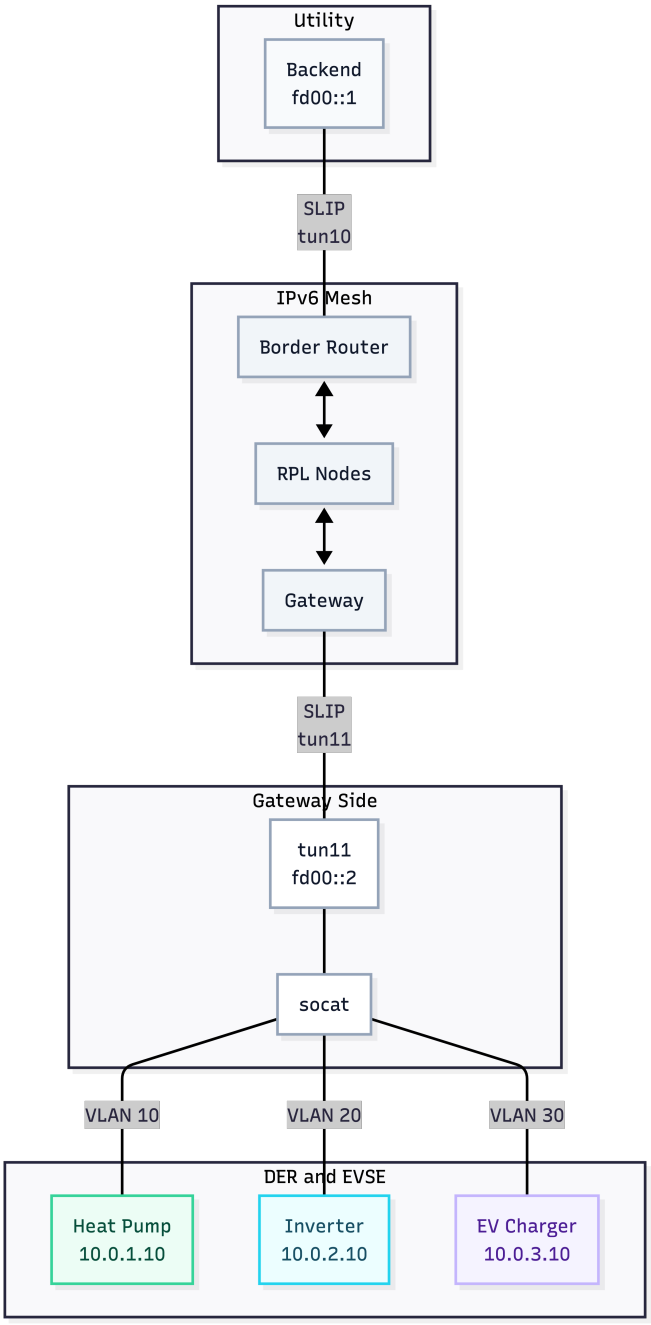

Figure 4.1: Network topology of the test suite showing dual border router architecture with VLAN segmented device networks. Traffic from the utility backend traverses the mesh path before reaching device simulators.

communicate through a Network Tunnel (TUN) interface designated as `tun10`. This interface is created by the `tunslip6` tool which establishes a SLIP connection to the border router node running within the Cooja simulator. The interface is configured with the IPv6 address `fd00::1/64` and serves as the source address for all outgoing control traffic.

The border router node operates as the RPL DODAG root and maintains routing information for all nodes in the mesh network. When packets arrive through the SLIP interface, the border router forwards them into the mesh according to the RPL routing table.

The mesh network operates within the Cooja network simulator using emulated motes. The simulator supports cycle accurate execution of Contiki-NG firmware while modeling radio propagation characteristics including transmission range, interference and frame reception probability.

Intermediate relay nodes participate in RPL routing and forward packets along the path between the border router and the gateway. The number of relay nodes and their spatial arrangement determine the hop count and network diameter. The test suite includes predefined simulation configurations for various topology patterns including tree, linear and clustered arrangements.

The gateway node serves as the exit point from the mesh network toward the device networks. It maintains a SLIP connection to the host system through a second TUN interface designated as `tun11` with the IPv6 address `fd00::2/64`.

To ensure that traffic between `fd00::1` and `fd00::2` traverses the actual mesh network

rather than being routed locally on the host, the `tun11` interface operates within a dedicated Linux network namespace called `mesh11`. This namespace isolation requires packets destined for `fd00::2` to enter through the SLIP tunnel rather than a local routing path.

The gateway firmware periodically transmits RPL DAO messages to advertise reachability of the `fd00::2` address. When the other nodes receive these announcements, they install routing table entries that direct traffic through the mesh path to the gateway node.

Behind the gateway, three isolated VLAN segments host the device simulators. Each VLAN is implemented as a separate Linux network namespace connected to the host through virtual Ethernet pairs.

VLAN 10 operates within the `vlan10` namespace using the `10.0.1.0/24` subnet and hosts the heat pump simulator at address `10.0.1.10`. VLAN 20 operates within the `vlan20` namespace using the `10.0.2.0/24` subnet and hosts the photovoltaic inverter simulator at address `10.0.2.10`. VLAN 30 operates within the `vlan30` namespace using the `10.0.3.0/24` subnet and hosts the electric vehicle charger simulator at address `10.0.3.10`.

The namespace isolation ensures that devices cannot communicate directly with each other, reflecting the segmentation policies typically enforced in operational environments. All device traffic must traverse the gateway and mesh network to reach the utility backend.

Communication between the IPv6 mesh and IPv4 device networks requires protocol translation at the gateway boundary. This translation is performed by `socat` instances running within the `mesh11` namespace. Table 4.1 shows the port mapping configuration for each device type.

Table 4.1: Port mapping for IPv6 to IPv4 protocol translation

Device	IPv6 Endpoint	IPv4 Target	Protocol	Transport
Heat Pump	`[fd00::2]:14400`	`10.0.1.10:14400`	Custom	UDP
PV Inverter	`[fd00::2]:14401`	`10.0.2.10:14401`	Modbus	TCP
EV Charger	`[fd00::2]:14402`	`10.0.3.10:14402`	OCPP	TCP

For UDP based communication, `socat` operates in datagram forwarding mode where each incoming packet on the IPv6 endpoint is translated and forwarded to the IPv4 target. For TCP based protocols, `socat` establishes bidirectional stream forwarding between the address families.

The test suite implements a complete protocol stack spanning from physical layer emulation through application layer protocols. Figure 4.2 illustrates the layered architecture on both the mesh network side and the device network side.

The Cooja simulator emulates IEEE 802.15.4 radio communication at the physical layer. The simulator models unit disk graph radio propagation where nodes within transmission range receive frames with a configurable success probability. The frame reception ratio parameter allows evaluation of lossy channel conditions without modifying node firmware.

At the link layer, the Contiki-NG Carrier Sense Multiple Access (CSMA) implementation provides medium access control with exponential backoff for collision avoidance. Frame acknowledgments and retransmissions are handled automatically by the link layer. The MAC layer maintains statistics counters for received frames, transmitted frames and received acknowledgments which are exposed through the instrumentation interface.

IPv6 networking is provided by the Contiki-NG uIP stack. The 6LoWPAN adaptation

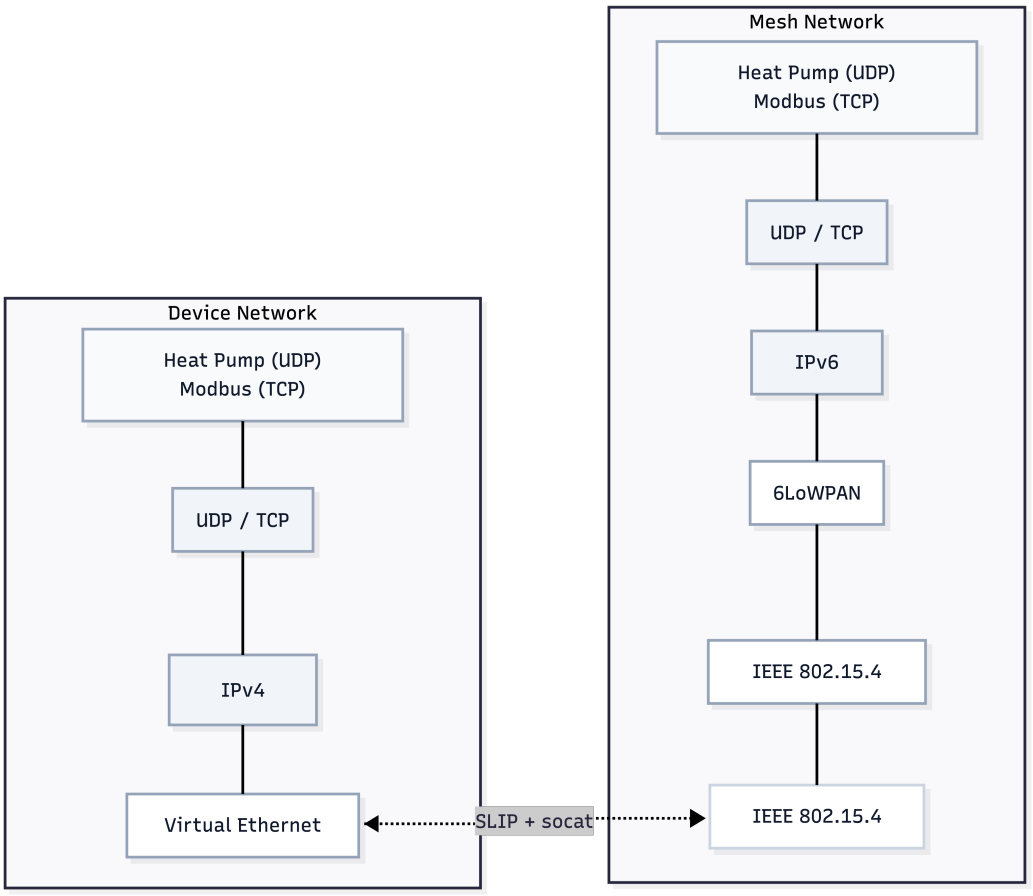

Figure 4.2: Protocol stack diagram showing the layered architecture. The right side shows the constrained mesh network stack while the left side shows the device network stack. Protocol translation occurs at the gateway boundary.

layer handles the impedance mismatch between IPv6 packet sizes and the constrained IEEE 802.15.4 frame payload. Header compression reduces the overhead of IPv6 and UDP headers while fragmentation and reassembly handle packets that exceed the link Maximum Transmission Unit (MTU).

The test suite also supports configurable frame sizes through compile time parameters. Reducing the frame size forces further fragmentation of application payloads. This capability enables evaluation of fragmentation overhead and reassembly reliability under various payload size scenarios.

Routing within the mesh network is provided by the RPL protocol operating in storing mode. The border router serves as the DODAG root and maintains downward routes to all nodes. Intermediate nodes store routing tables for their subtrees enabling direct forwarding without root involvement for intra subtree traffic.

The transport layer supports both UDP and TCP communication. UDP is used for the heat pump protocol where low latency and tolerance for occasional packet loss are acceptable characteristics. TCP is used for Modbus and OCPP protocols where reliable ordered delivery is required.

For TCP connections traversing the mesh network, the constrained environment is associated with behavior related to retransmission timeouts and congestion control. The test suite uses default Contiki-NG TCP parameters that are intended for constrained networks. These parameters can be adjusted at the firmware level if required by a given experimental configuration.

Three application layer protocols are implemented to represent diverse smart grid communication patterns including heat pump protocol, Modbus TCP protocol and OCPP protocol.

The heat pump device uses a custom UDP based binary protocol with four message types. Control queries are 48 bytes and carry command identifiers with optional payloads. Acknowledgments are 24 bytes and confirm command reception. Telemetry reports are 200 bytes and carry comprehensive device state including temperatures, power consumption and operational status. Periodic telemetry uses the same format as telemetry reports but is transmitted at regular intervals during monitoring sessions.

The photovoltaic inverter implements standard Modbus TCP for register based communication. Holding registers provide read write access to setpoint values including active power limits, reactive power setpoints and ramp rates. Input registers provide read only access to telemetry values including Alternating Current (AC) and Direct Current (DC) electrical measurements, temperature and energy counters. Message sizes follow the Modbus specification with typical transactions requiring 12 to 24 bytes for requests and 16 to 120 bytes for responses depending on the number of registers accessed.

The electric vehicle charger implements Open Charge Point Protocol version 1.6 over WebSocket connections. OCPP uses JSON formatted messages for operations including boot notification, heartbeat, status notification and meter values reporting. The WebSocket transport adds framing overhead and provides mechanisms for connection management and message integrity.

4.2.2 Component design

The software architecture consists of four component categories: mesh firmware, device simulators, backend controllers and test orchestration. Figure 4.3 shows the relationships between these components and their execution environments.

Three firmware variants are implemented in C for the Contiki-NG platform. All variants share a common MAC statistics module that tracks reception counts, transmission counts and acknowledgment counts at the link layer. These statistics are periodically captured and made ready for post processing.

The border router firmware extends the standard Contiki-NG RPL border router with SLIP connectivity and MAC statistics collection. The node initializes as the RPL DODAG root and advertises the `fd00::/64` prefix to the mesh network. Packets received from the SLIP interface are injected into the network stack for routing while packets destined for external addresses are forwarded to the tunnel interface.

The gateway firmware implements bidirectional SLIP bridging with route announcement capability. Unlike the border router, the gateway joins the existing DODAG as a regular node rather than serving as the root. To enable reachability of the `fd00::2` address, the gateway periodically transmits DAO messages advertising this address as a routing target. The implementation uses the Contiki-NG RPL Application Programming Interface (API) to generate these announcements at configurable intervals.

Relay nodes execute minimal firmware consisting only of the RPL routing stack and

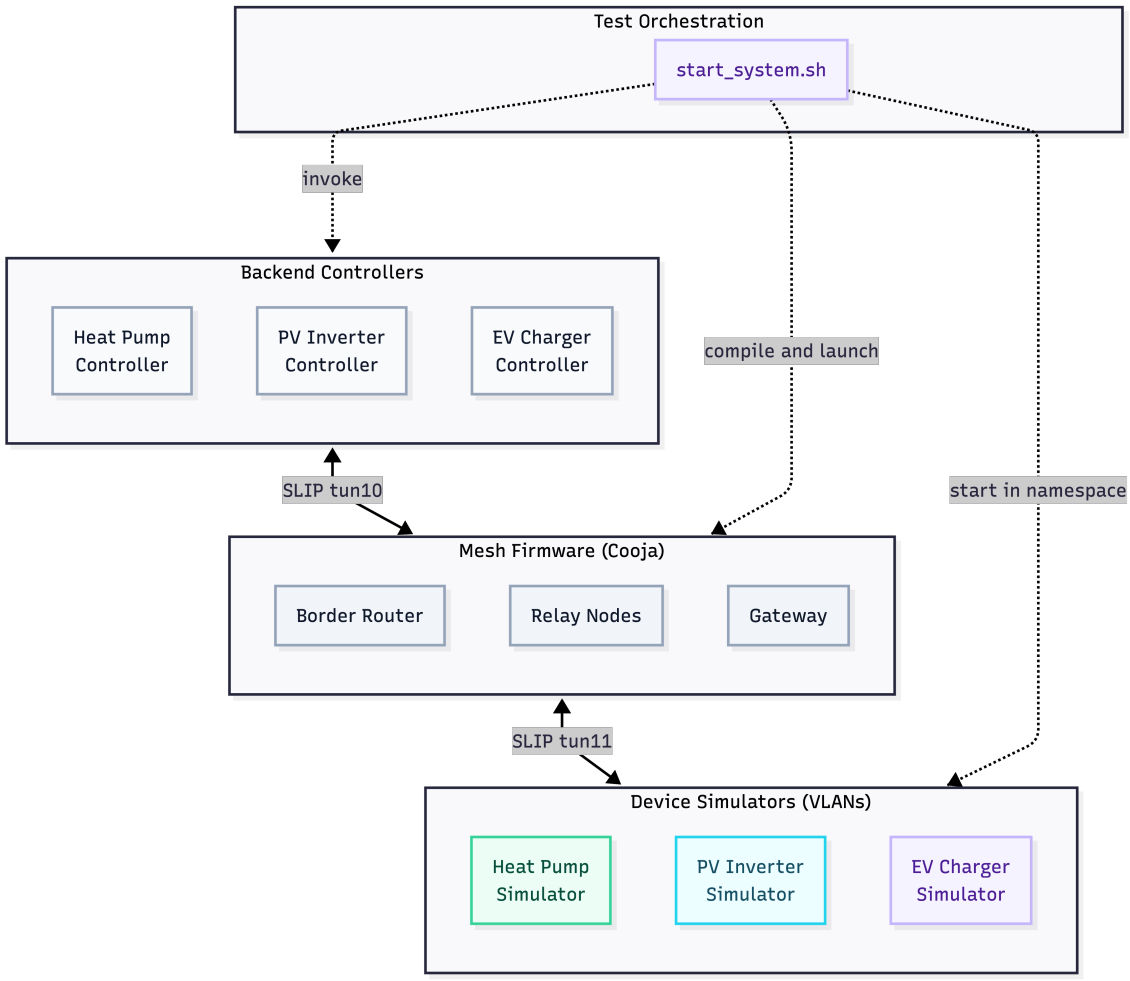

Figure 4.3: Software component diagram showing the four main categories: mesh firmware executing in Cooja, device simulators in VLAN namespaces, backend controllers on the host and the orchestration script coordinating the system.

MAC statistics collection. No application logic is included, so the measured performance primarily reflects routing overhead rather than application processing. The firmware can be extended to include application layer communication when evaluation of background traffic generated by intermediate nodes is required.

Device simulators are implemented in Python and execute within isolated VLAN namespaces. Each simulator maintains internal state representing the physical device characteristics and responds to incoming requests according to the defined protocol. Simulators embed timestamps in outgoing messages to enable one way latency measurements.

The heat pump simulator listens for UDP packets on port 14400 and implements the four message type protocol. Internal state includes operating mode, target temperature, current temperature, power consumption and various status flags. A background thread updates the internal state to represent behaviors such as temperature drift and compressor cycling. The simulator supports both request response interaction and continuous monitoring mode.

The inverter simulator implements a Modbus TCP server using the `pymodbus` library. Holding registers store setpoint values while input registers contain simulated telemetry data. A background thread updates telemetry values periodically to represent measurement variation. The simulator responds to standard Modbus function codes for reading and writing registers.

The charger simulator implements an OCPP 1.6 charge point using the `websockets` library. In test mode, the simulator connects to the central system, completes the boot

notification handshake and executes a predefined measurement sequence. During this sequence, the simulator transmits a configurable number of meter value messages at specified intervals while timestamps are recorded for latency analysis.

Backend controllers represent utility side applications that initiate communication with devices through the mesh network. Each controller is implemented in Python with a command line interface supporting both manual interaction and automated test modes. The test suite initiates and orchestrates the individual controllers accordingly.

The heat pump controller provides commands for status queries, temperature setpoint adjustment, mode changes and monitoring activation. The automated test mode transmits a configurable number of control queries and records per message timestamps for latency calculation. Results are written to Comma-Separated Values (CSV) files including raw timing data and aggregated statistics.

The inverter controller provides commands for telemetry reading, setpoint writing and dispatch window execution. A dispatch window consists of an initial setpoint write followed by periodic telemetry reads for the specified duration. The automated test mode exercises this pattern and records timing data for each Modbus transaction.

The charger controller implements an OCPP central system that accepts connections from charge points. During test execution, the controller completes the protocol handshake and receives meter value messages from the connected charge point. Timestamps are recorded for each received message to enable latency calculation.

A shell script provides automated orchestration of the complete test workflow. The

script performs the following operations in sequence: firmware compilation for the Cooja target platform, simulator launch with the selected scenario configuration, SLIP tunnel creation and IPv6 address assignment, network namespace creation for VLAN isolation, device simulator startup in respective namespaces, `socat` initialization for protocol translation and finally presentation of an interactive menu for test execution.

The script supports scenario selection at startup where each scenario corresponds to a specific scenario file with specific network topology and radio configuration. Fragmentation settings in the firmware are automatically adjusted when fragmentation scenarios are selected. Signal handlers perform cleanup of processes, namespaces and interfaces upon termination.

4.3 Instrumentation Procedure

The test suite incorporates measurement instrumentation at multiple points along the communication path. This section describes the mechanisms for latency measurement, delivery ratio calculation and MAC layer statistics collection.

Latency measurement relies on timestamps embedded within application layer messages rather than external packet capture. This approach enables one way latency calculation without requiring clock synchronization infrastructure since both the backend controllers and device simulators execute on the same host system.

Figure 4.4 illustrates the timestamp embedding strategy using the heat pump protocol as an example. The measurement process operates as follows. When the backend controller

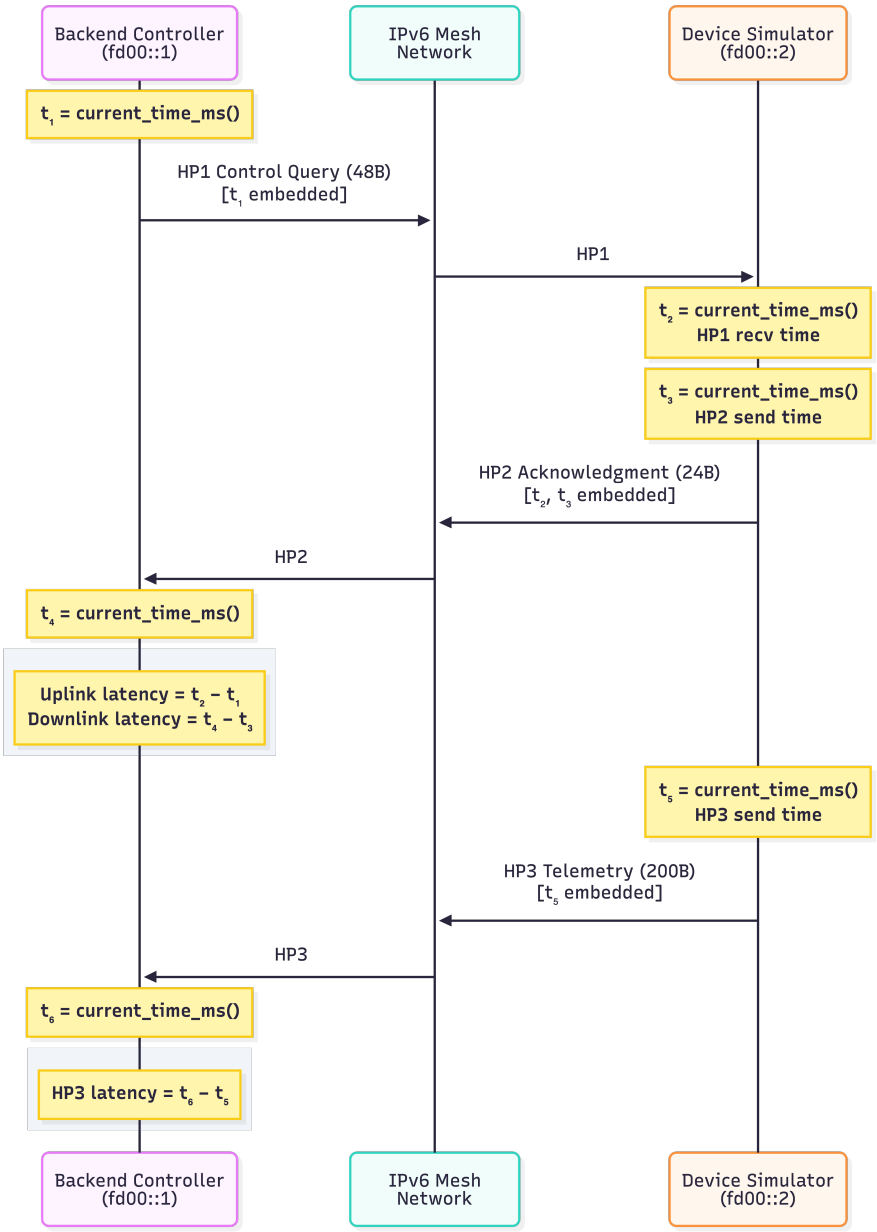

Figure 4.4: Message sequence diagram showing timestamp embedding points for the heat pump protocol. Each message carries send and receive timestamps that enable one way latency calculation for both uplink and downlink directions.

transmits a control query (HP1), it records the current system time in milliseconds and embeds this value in the message payload. Upon reception, the device simulator immediately records its local receive time. The difference between these values represents the one way uplink latency from the controller to the device. The device simulator then constructs an acknowledgment message (HP2) containing both the recorded HP1 receive time and its own HP2 send time. When the controller receives this acknowledgment, it calculates the one way downlink latency as the difference between its local receive time and the embedded HP2 send time.

For telemetry messages (HP3), the device embeds its send timestamp which allows the controller to calculate the downlink latency upon reception. This procedure yields three latency measurements per transaction: HP1 uplink, HP2 downlink and HP3 downlink. The Modbus TCP protocol uses a similar strategy where timestamps are stored in designated input registers. The controller writes its send timestamp to holding registers before initiating a transaction and reads the device timestamps from input registers in the response. The OCPP protocol embeds timestamps within the JSON message payload and calculates one way message latencies in the same manner.

Delivery ratio is calculated as the proportion of successfully received messages relative to the total number transmitted. The test suite tracks delivery independently for each message type in a protocol exchange. A message is considered successfully delivered only if it arrives within the configured timeout period and contains valid data. For the heat pump protocol, three delivery ratios are computed per test session. The HP1 delivery ratio indicates successful uplink transmission of control queries. The HP2 delivery ratio

indicates successful acknowledgment reception. The HP3 delivery ratio indicates successful telemetry reception. For the PV inverter protocol, delivery ratios are computed for Modbus transactions. The write transaction ratio tracks successful setpoint commands where both the request reaches the device and the response returns to the controller. The read transaction ratio tracks successful telemetry retrievals. For the EV charger protocol, delivery is measured for OCPP message exchanges. The boot notification ratio tracks successful connection establishment. The meter values ratio tracks successful telemetry transmissions from the charge point to the central system. WebSocket connection failures and message timeouts are recorded as delivery failures.

The automated test mode transmits a configurable number of message sequences with a fixed interval between transmissions. The interval is configured to reduce queuing effects at the SLIP interface and to allow the mesh network to reach steady operation between transactions. The interval can be adjusted for a given experiment to evaluate different traffic rates. Results are aggregated into summary statistics including total messages sent, total messages received and the resulting delivery ratio. These metrics are computed separately for each message type to identify whether losses occur preferentially in the uplink or downlink direction and to compare reliability across different protocol types.

The MAC layer instrumentation provides visibility into link layer behavior that is not observable from application layer measurements alone. Each node in the mesh network maintains counters for received and transmitted frames along with received acknowledgments. The firmware implements a statistics module that periodically prints counter values to the serial output. These values are captured by the Cooja log listener and written to

individual log files for each node. The statistics represent cumulative counts since node initialization. A post processing script parses the log files and extracts the final counter values from each node. The script computes derived metrics including the retransmission count and retransmission ratio. The retransmission count is calculated as the difference between transmitted frames and received acknowledgments. The retransmission ratio expresses this value as a proportion of total transmissions.

Table 4.2: MAC layer metrics collected from each mesh node

Metric	**Source**	**Description**
TX Count	Hardware counter	Total frames transmitted including retransmissions
ACK Count	Hardware counter	Acknowledgments received for transmitted frames
RX Count	Hardware counter	Total frames received from other nodes
Retransmissions	Derived	TX Count minus ACK Count
Retransmission Ratio	Derived	Retransmissions divided by TX Count

Table 4.2 summarizes the MAC layer metrics collected from each node. Statistics are reported separately for the border router, gateway and relay nodes to support analysis of load concentration and packet forwarding behavior within the mesh topology.

Test results are organized in a hierarchical directory structure under the `test-runs` folder. Each test execution creates a new directory named according to the scenario identifier, device type and execution timestamp.

```
test-runs/
├── S0-baseline/
│   ├── heatpump/
│   │   └── 20251217_125143/
│   │       ├── raw_data.csv
│   │       ├── summary.csv
│   │       └── mac_stats.csv
│   ├── pvinverter/
│   └── evcharger/
├── S1-hop-stress/
└── S2-loss-stress/
```

Each test session generates three output files in comma separated value format. The `raw_data.csv` file records per-message timing data, including send and receive timestamps, latency and delivery status. The `summary.csv` file provides aggregated statistics including message counts, delivery ratios and latency percentiles across the entire test session. The `mac_stats.csv` file contains link layer metrics including transmission counts, acknowledgment counts and retransmission ratios for each node in the mesh.

4.4 Execution Workflow

The operational workflow used to conduct experiments with the test suite is described in this section. System initialization is performed first, after which test execution is carried out through an interactive interface. The test suite is initialized through a shell script that orchestrates the complete setup sequence. Execution begins by invoking the script from the `scripts` directory:

```
$ cd scripts
$ bash start_system.sh
```

Upon invocation, the script presents a scenario selection menu. Five predefined scenarios are available corresponding to different network configurations:

- **S0-baseline**: Tree topology with 25 nodes, 4 hops maximum, 99% frame reception ratio, frame size 128 B
- **S1-hop-stress**: Linear topology with 11 nodes, 10 hops maximum, 99% frame reception ratio, frame size 128 B
- **S2-loss-stress**: Tree topology with 25 nodes, 4 hops maximum, 85% frame reception ratio, frame size 128 B
- **S3-fragmentation**: Tree topology with 25 nodes, 4 hops maximum, 99% frame reception ratio, frame size 80 B
- **S4-scale-stress**: Two-cluster topology with 60 nodes, 6 hops maximum, 99% frame reception ratio, frame size 128 B

After scenario selection, the script executes the following initialization sequence. First, the Contiki-NG firmware is compiled for the Cooja target platform. The compilation automatically incorporates the frame configuration appropriate for the selected scenario. Second, the Cooja simulator is launched with the corresponding simulation file. The simulation is then started within the Cooja interface, after which the serial socket servers become available.

Once the simulation is running, the script creates the SLIP tunnel interfaces and con-

figures IPv6 addressing. Network namespaces are established for the gateway and VLAN segments. Device simulators are started within their respective namespaces and the `socat` instances are initialized for protocol translation. Status messages are printed during initialization to indicate completion of each step.

After initialization completes, the script presents an interactive menu for test execution:

```
Interactive Test Menu
================================================================================
 1) Ping test: fd00::1 -> fd00::2 (through mesh)
 2) Heat Pump Latency/Delivery Test (UDP)
 3) PV Inverter Latency/Delivery Test (Modbus TCP)
 4) EV Charger Latency/Delivery Test (WebSocket OCPP)
 5) Check network interfaces
 6) Check VLAN isolation
 7) Exit
================================================================================
```

The ping test is used to assess basic connectivity through the mesh network by transmitting Internet Control Message Protocol version 6 (ICMPv6) echo requests from the utility endpoint to the gateway endpoint. This test provides an indication that the SLIP tunnels, mesh routing and namespace configuration are operating before application layer tests are executed.

The device specific tests prompt for configuration parameters including the number of messages to transmit and the timeout period for responses. After the parameters are provided, the measurement sequence is executed and progress information is printed, including per message latencies and delivery status. After completion, summary statistics are printed to the console and detailed results are written to the output directory.

The network interface check displays the current state of tunnel interfaces and VLAN configurations. The VLAN isolation check verifies that devices in different segments cannot communicate directly with each other.

The test suite performs cleanup upon termination. When the user exits the interactive menu or interrupts the script with a termination signal, the cleanup handler stops all running processes including the Cooja simulator, SLIP tunnels, device simulators and `socat` instances. Network namespaces are deleted and virtual interfaces are removed. This cleanup procedure restores the host network configuration by removing interfaces and namespaces created for the test session.

4.5 Experimental Results

This section presents the experimental results obtained from the test suite. Each subsection follows a consistent structure: introduction of test conditions, presentation of measurement data, observations from the results and technical analysis of the findings.

The evaluation is organized as follows: Section 4.5.1 reports reference results under

nominal network conditions. Section 4.5.2 examines the impact of increased hop count on communication latency. Section 4.5.3 evaluates protocol behavior under lossy channel conditions. Section 4.5.4 investigates the effect of fragmentation on message delivery. Section 4.5.5 assesses performance under increased network density.

4.5.1 Baseline characterization

The baseline scenario (S0) provides reference values under nominal network conditions. The configuration uses a tree topology with 25 nodes, maximum 4 hops and 99% frame reception ratio. The baseline delivery performance, latency characteristics and MAC layer statistics under scenario S0 are summarized in Tables 4.3–4.5 and Figures 4.5 and 4.6.

Table 4.3: Application layer delivery performance under baseline conditions (S0). Each message type was transmitted 250 times.

Protocol	**Message**	**Sent**	**Received**	**Delivery (%)**
UDP	HP1	250	244	97.6
UDP	HP2	250	244	97.6
UDP	HP3	250	244	97.6
Modbus TCP	INV3	250	250	100.0
Modbus TCP	INV4	250	250	100.0
WebSocket	OC4	250	250	100.0
WebSocket	OC5	250	250	100.0

Table 4.4: One-way latency measurements under baseline conditions (S0). Values represent median, 95th percentile and maximum latency in milliseconds.

Protocol	**Message**	**Median (ms)**	**P95 (ms)**	**Max (ms)**
UDP	HP1	133	216	351
UDP	HP2	130	353	537
UDP	HP3	397	555	844
Modbus TCP	INV3	147	223	381
Modbus TCP	INV4	358	467	2270
WebSocket	OC4	246	510	1675
WebSocket	OC5	143	430	2924

Table 4.5: MAC layer transmission statistics under baseline conditions (S0). Retransmission ratio indicates the percentage of frames that required retransmission at the link layer.

Protocol	**TX**	**Retrans**	**Retrans (%)**
UDP	6,838	192	2.81
Modbus TCP	12,906	312	2.42
WebSocket	20,777	536	2.58

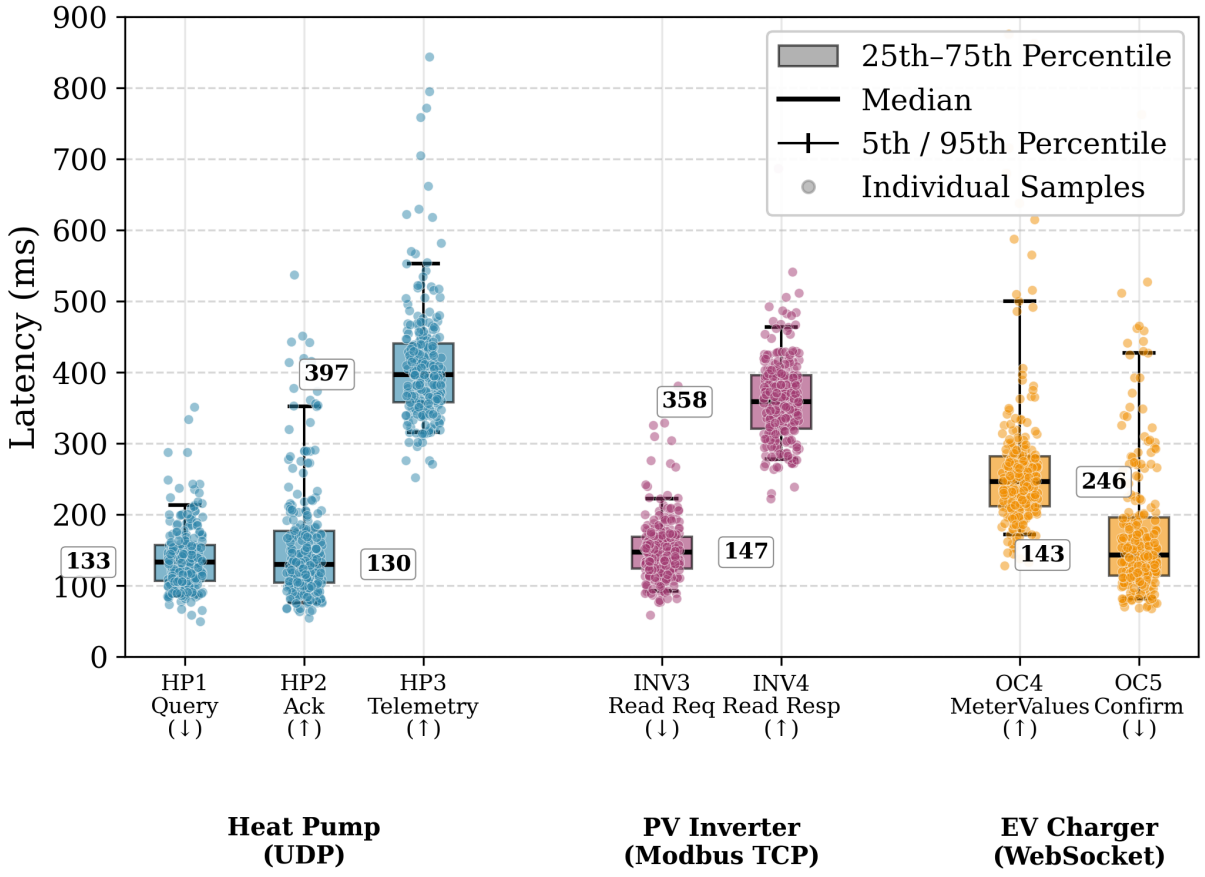

Figure 4.5: One way latency distribution under S0 conditions. Box plots show the interquartile range (Q1 to Q3) with median values annotated. Whiskers extend to the 5th and 95th percentiles. Individual data points are overlaid to show the complete distribution. Message directions are indicated as uplink (device to controller) or downlink (controller to device). Outliers above 900ms: INV4: 1 outliers (max 2270ms); OC4: 3 outliers (max 1675ms); OC5: 4 outliers (max 2924ms).

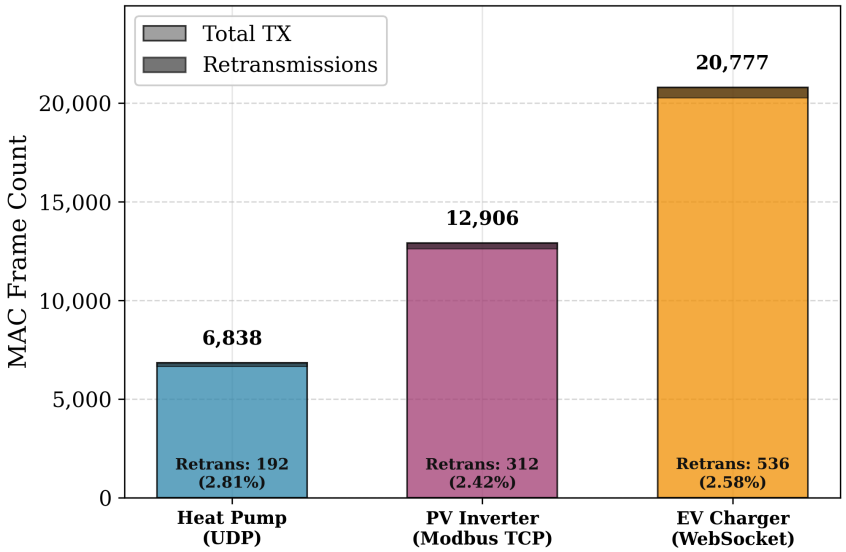

Figure 4.6: MAC layer transmission overhead under baseline conditions (S0). Stacked bars show successful transmissions and retransmissions for each use case. The retransmission ratio (percentage of total transmissions) is annotated for each bar.

The heat pump use case shows a 97.6% delivery ratio across all three message types, with 6 out of 250 packets not received. These losses are consistent with the absence of transport layer recovery in UDP. Latency performance varies by message size: HP1 and HP2 have median latencies of 133 ms and 130 ms respectively, while HP3 shows a higher median of 397 ms which is consistent with its larger payload size. The P95 latency ranges from 216 ms (HP1) to 555 ms (HP3), with maximum values remaining below 850 ms. MAC layer retransmission ratio is observed as 2.81%.

The inverter use case shows a 100% delivery ratio for both message types under the tested conditions. This result is consistent with TCP providing reliable delivery. An asymmetry is observed between uplink and downlink: INV3 (uplink) shows a median latency of 147 ms while INV4 (downlink) reaches 358 ms. The maximum latency of 2270 ms for INV4 is consistent with occasional TCP retransmissions. MAC layer overhead is the lowest among all protocols at 2.42% retransmission ratio.

The OCPP-based charging station shows a 100% delivery ratio under the tested conditions, with median latencies of 246 ms (OC4, uplink) and 143 ms (OC5, downlink). Maximum latencies of 1675 ms (OC4) and 2924 ms (OC5) indicate increased tail latency relative to the median. The WebSocket protocol generates the highest MAC layer frame count (20,777 frames) compared to Modbus TCP (12,906) and UDP (6,838), which is consistent with overhead from the Hypertext Transfer Protocol (HTTP) upgrade, WebSocket framing and JSON message encoding. Despite this overhead, the retransmission ratio remains comparable at 2.58%.

The observed delivery failures occur at the application layer due to transaction timeouts. The test framework employs a 5 second timeout for the complete HP1-HP2-HP3 exchange. When end-to-end latency exceeds this threshold, the transaction is recorded as failed. Contributing factors may include transient routing changes during RPL convergence, queueing at the SLIP interface, or cumulative delays across multiple mesh hops.

The HP3 message exhibits 3× higher median latency (397 ms) compared to HP1 and HP2 (130–133 ms). This difference is consistent with the message size. When the payload exceeds the IEEE 802.15.4 frame payload budget (102 bytes for a 128 byte raw frame), 6LoWPAN fragmentation is required. Each fragment must be transmitted, acknowledged and reassembled independently, which increases per-hop delay. The increased P95 (555 ms) and maximum (844 ms) values are consistent with fragmentation-related latency accumulation.

The 3× difference in MAC frame counts between UDP (6,838) and WebSocket (20,777) can be attributed to overhead across multiple protocol layers. TCP requires connection establishment (Synchronize (SYN), Synchronize Acknowledgment (SYN-ACK), Acknowledgment (ACK)), per-segment acknowledgments and connection teardown. WebSocket adds HTTP upgrade negotiation (GET, 101 Switching Protocols) and frame headers for each message. JSON text encoding in OCPP produces larger payloads than binary Modbus registers, increasing fragment count. These effects add to the total number of transmitted frames along the end-to-end path.

Maximum latencies exceeding 2 seconds (INV4: 2270 ms, OC5: 2924 ms) are consis-

tent with TCP retransmission timeout (Retransmission Timeout (RTO)) behavior. When a segment is lost and no duplicate ACKs trigger fast retransmit, TCP waits for RTO expiration (typically 1–3 seconds in constrained implementations) before resending. The observed maximum values on the order of 2–3× the base RTO are consistent with single retransmission events rather than multiple consecutive losses, given the 99% channel reception ratio.

4.5.2 Hop count impact

This experiment examines the effect of increased hop count on communication performance. The S1 scenario uses a linear topology with 11 nodes and 10 hops maximum compared to 4 hops in the baseline. The scenario uses 99% frame reception ratio. A comparison of application layer delivery performance, one way latency metrics and MAC-layer transmission behavior between S0 and S1 is summarized in Tables 4.6–4.8 and Figures 4.7 and 4.8.

Table 4.6: Application layer delivery performance comparison between S0 and S1 conditions. Each message type was transmitted 250 times per scenario.

Protocol	**Msg**	**S0**			**S1**		
		Sent	**Recv**	**%**	**Sent**	**Recv**	**%**
UDP	HP1	250	244	97.6	250	245	98.0
UDP	HP2	250	244	97.6	250	245	98.0
UDP	HP3	250	244	97.6	250	245	98.0
Modbus TCP	INV3	250	250	100.0	250	250	100.0
Modbus TCP	INV4	250	250	100.0	250	250	100.0
WebSocket	OC4	250	250	100.0	250	240	96.0
WebSocket	OC5	250	250	100.0	250	239	95.6

Table 4.7: One-way latency measurements comparison between S0 and S1 conditions. Values represent median, 95th percentile and maximum latency in milliseconds.

Protocol	**Msg**	**S0 (ms)**			**S1 (ms)**		
		Med	**P95**	**Max**	**Med**	**P95**	**Max**
UDP	HP1	133	216	351	293	398	666
UDP	HP2	130	353	537	298	467	1171
UDP	HP3	397	555	844	881	1040	1945
Modbus TCP	INV3	147	223	381	296	569	2569
Modbus TCP	INV4	358	467	2270	837	1152	4904
WebSocket	OC4	246	510	1675	560	1463	4819
WebSocket	OC5	143	430	2924	308	1430	5964

Table 4.8: MAC layer transmission statistics comparison between S0 and S1 conditions.

Protocol	**S0**			**S1**		
	TX	**Retrans**	**%**	**TX**	**Retrans**	**%**
UDP	6,838	192	2.81	13,933	370	2.66
Modbus TCP	12,906	312	2.42	28,610	655	2.29
WebSocket	20,777	536	2.58	46,545	1,139	2.45

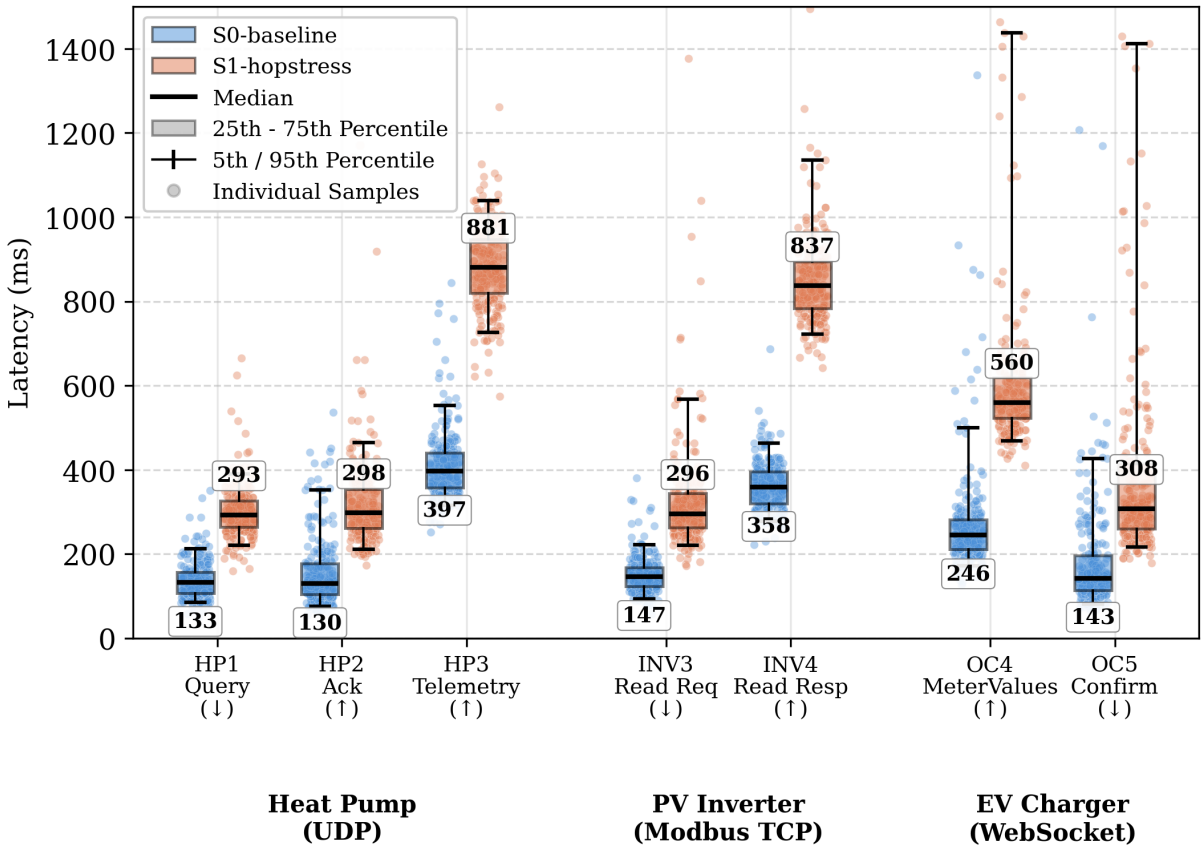

Figure 4.7: Latency distribution comparison between S0 (blue) and S1 (orange) conditions. Box plots show 25th to 75th percentile with whiskers extending to 5th and 95th percentiles. Median values are annotated above each box. Outliers above 1500ms: HP3 (S1-hopstress): 2 (max 1945ms); INV3 (S1-hopstress): 3 (max 2569ms); INV4 (S0-baseline): 1 (max 2270ms).

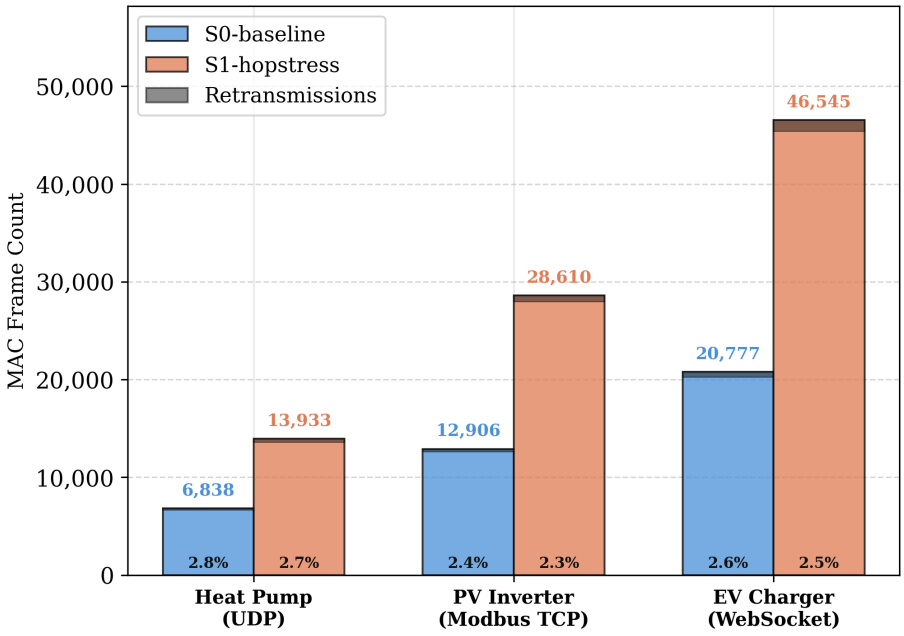

Figure 4.8: MAC layer transmission comparison between S0 and S1 conditions. Retransmission percentages are shown inside bars.

All protocols exhibit increased latency when hop count extends from 4 to 10. UDP messages show a 120–129% increase, with median latency rising from 133 ms to 293 ms for HP1, from 130 ms to 298 ms for HP2, and from 397 ms to 881 ms for HP3. Modbus TCP follows a similar pattern with a 101–134% increase, with median latency rising from 147 ms to 296 ms for INV3 and from 358 ms to 837 ms for INV4. WebSocket latencies increase by 115–128%, with median latency rising from 246 ms to 560 ms for OC4 and from 143 ms to 308 ms for OC5. The similarity in relative increases across protocols is consistent with hop count being a primary contributor to median latency in this scenario.

Protocol behavior differs under extended hop conditions. UDP delivery changes from 97.6% to 98.0%, which is within expected variation for the sample size. Modbus TCP maintains 100% delivery. WebSocket decreases from 100% to 96.0% (OC4) and 95.6% (OC5), representing 10–11 lost messages out of 250. This pattern is consistent with protocol-dependent sensitivity to increased path length and associated timeout behavior.

MAC layer transmission counts approximately double across all protocols: UDP increases from 6,838 to 13,933 (+103.8%), Modbus TCP from 12,906 to 28,610 (+121.7%) and WebSocket from 20,777 to 46,545 (+124.0%). Despite this increase, retransmission ratios remain stable or slightly decrease from 2.81% to 2.66% for UDP, from 2.42% to 2.29% for Modbus, and from 2.58% to 2.45% for WebSocket.

Maximum latency values show larger relative increases than medians. INV4 maximum increases from 2270 ms to 4904 ms (+116%), while OC5 maximum grows from 2924 ms to 5964 ms (+104%). P95 values exhibit similar amplification: OC4 P95 increases from

510 ms to 1463 ms (+187%), indicating increased dispersion in the latency distribution under the extended path.

The observed latency increase when hop count increases from 4 to 10 yields an estimated per-hop delay of 25–30 ms for small messages. This value encompasses MAC layer channel access time, frame transmission duration and processing delay at each intermediate node. The approximately linear relationship between hop count and median latency is consistent with additive forwarding delay under the tested traffic conditions.

The 4% delivery loss in WebSocket is consistent with connection-oriented operation combined with application layer timeouts. Extended round-trip times in 10-hop paths may exceed OCPP transaction timeouts or WebSocket keep-alive intervals, leading to connection termination. Additionally, the 3× higher frame count of WebSocket increases the number of opportunities for frame loss along the path, which can increase the likelihood of transport-layer retransmission events.

The 2× increase in MAC transmissions is consistent with hop-by-hop forwarding in IEEE 802.15.4 mesh networks. Each frame is transmitted and acknowledged at every hop, so a packet traversing 10 hops corresponds to 10 link-layer transmissions compared to 4 transmissions in the baseline topology. The near-linear increase (101–123%) indicates that link-layer transmissions scale approximately linearly with hop count for the tested topologies.

The marginal decrease in retransmission ratio (approximately 0.13–0.15 percentage points) despite doubled hop count is consistent with similar link quality across the tested

path lengths. The linear topology reduces hidden-node effects relative to branched topologies, which can reduce collision probability. Additionally, the 11 node network in S1 versus 25 nodes in S0 reduces aggregate channel utilization, which can offset the increase in per-packet transmissions caused by a longer path.

The disproportionate growth in P95 and maximum values is consistent with a higher probability of retransmission events along longer paths. With 10 opportunities for frame loss versus 4, TCP segments face higher cumulative loss probability, which can trigger retransmission timeouts more frequently. The 5964 ms maximum for OC5 is on the order of two RTO intervals (approximately 2–3 seconds each in constrained implementations), which is consistent with one segment requiring multiple retransmissions along the 10-hop path.

4.5.3 Frame loss impact

In this section we examine the performance under lossy channel conditions. The S2 scenario reduces the frame reception ratio from 99% to 85% while maintaining the same topology as the baseline. A comparison of application layer delivery performance, one way latency metrics and MAC layer transmission behavior between S0 and S2 is summarized in Tables 4.9–4.11 and Figures 4.9 and 4.10.

Table 4.9: Application layer delivery performance comparison between S0 and S2 conditions. Each message type was transmitted 250 times per scenario.

Protocol	Msg	S0			S2		
		Sent	Recv	%	Sent	Recv	%
UDP	HP1	250	244	97.6	250	249	99.6
UDP	HP2	250	244	97.6	250	249	99.6
UDP	HP3	250	244	97.6	250	249	99.6
Modbus TCP	INV3	250	250	100.0	250	250	100.0
Modbus TCP	INV4	250	250	100.0	250	250	100.0
WebSocket	OC4	250	250	100.0	250	250	100.0
WebSocket	OC5	250	250	100.0	250	250	100.0

Table 4.10: One-way latency measurements comparison between S0 and S2 conditions. Values represent median, 95th percentile and maximum latency in milliseconds.

Protocol	Msg	S0 (ms)			S2 (ms)		
		Med	P95	Max	Med	P95	Max
UDP	HP1	133	216	351	135	223	289
UDP	HP2	130	353	537	136	255	416
UDP	HP3	397	555	844	479	648	752
Modbus TCP	INV3	147	223	381	145	238	398
Modbus TCP	INV4	358	467	2270	420	586	702
WebSocket	OC4	246	510	1675	288	465	1196
WebSocket	OC5	143	430	2924	147	325	839

Table 4.11: MAC layer transmission statistics comparison between S0 and S2 conditions.

Protocol	S0			S2		
	TX	Retrans	%	TX	Retrans	%
UDP	6,838	192	2.81	9,226	2,345	25.42
Modbus TCP	12,906	312	2.42	17,225	4,375	25.40
WebSocket	20,777	536	2.58	33,625	8,263	24.57

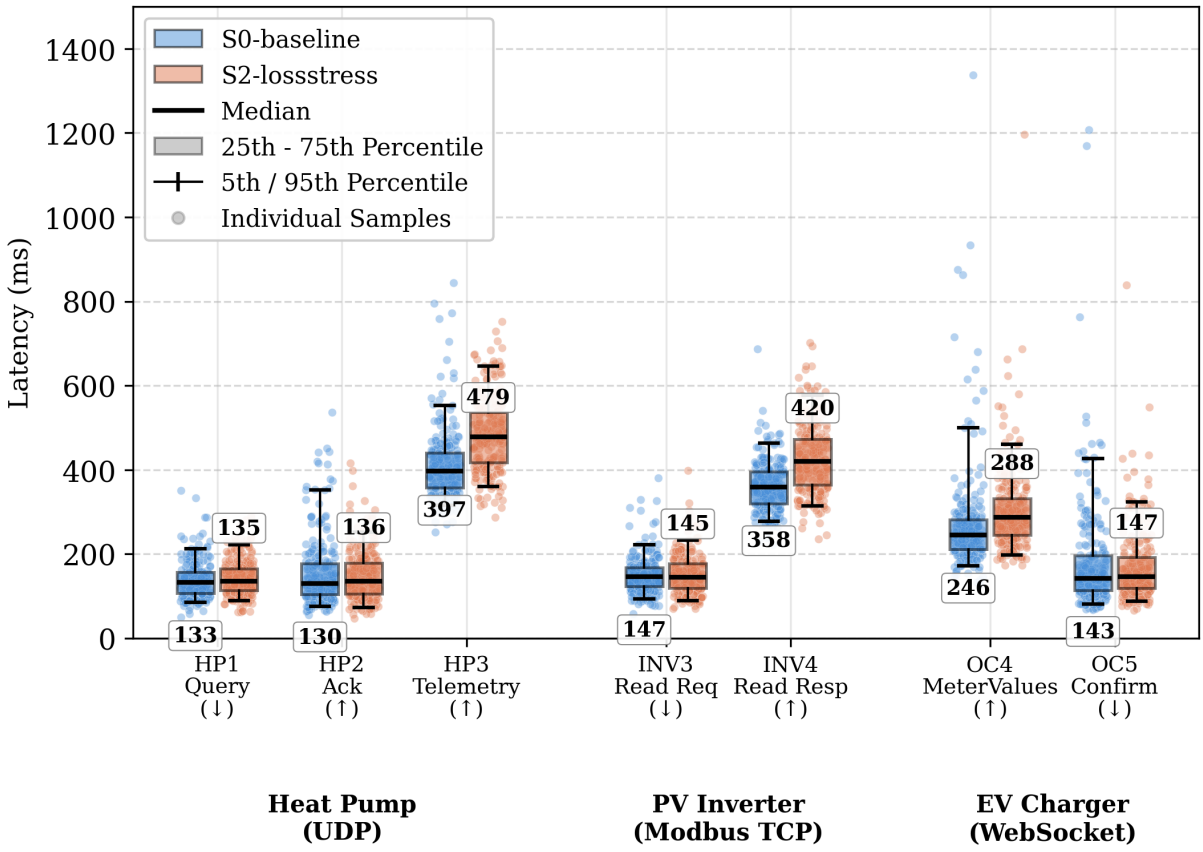

Figure 4.9: Latency distribution comparison between S0 (blue) and S2 (orange) conditions. Box plots show 25th to 75th percentile with whiskers extending to 5th and 95th percentiles. Median values are annotated above each box. Outliers above 1500ms: INV4 (S0-baseline): 1 (max 2270ms); OC4 (S0-baseline): 1 (max 1675ms); OC5 (S0-baseline): 2 (max 2924ms).

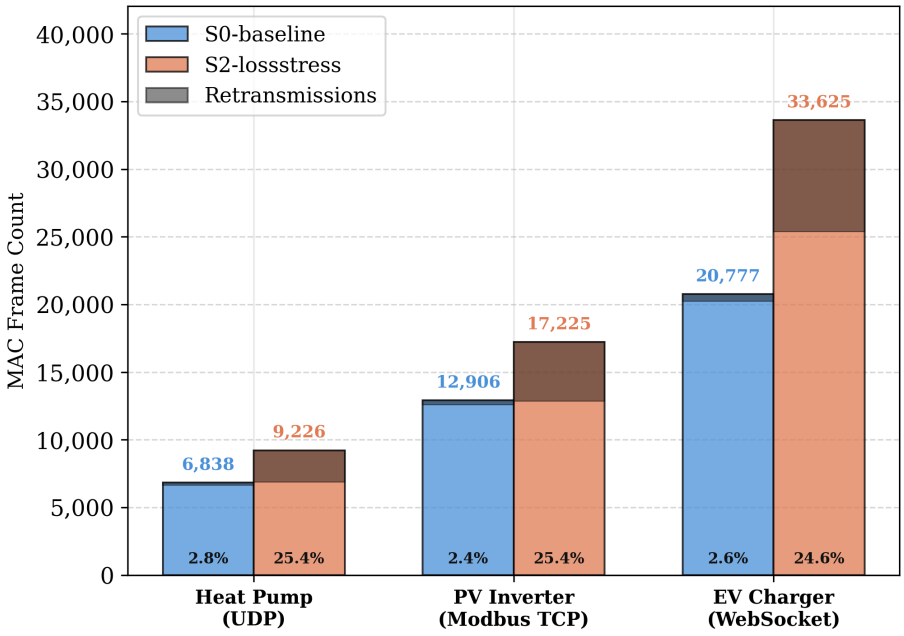

Figure 4.10: MAC layer transmission comparison between S0 and S2 conditions. Retransmission percentages are shown inside bars.

Delivery ratios remain stable under the tested 15% channel loss conditions. UDP delivery varies from 97.6% (S0) to 99.6% (S2), with a difference of 5 packets, which is consistent with test-to-test variation for a 250 message sample size. TCP-based protocols maintain 100% delivery for both Modbus TCP and WebSocket across both scenarios, which is consistent with transport-layer retransmissions compensating for increased channel errors.

Median latencies show small changes under channel degradation. Small messages exhibit minimal change: HP1 increases by 1.5%, with the median rising from 133 ms to 135 ms; INV3 decreases by 1.4%, with the median decreasing from 147 ms to 145 ms; and OC5 increases by 2.8%, with the median rising from 143 ms to 147 ms. Larger messages show moderate increases: HP3 rises by 20.7%, with the median increasing from 397 ms to 479 ms; INV4 increases by 17.2%, with the median rising from 358 ms to 420 ms; and OC4 increases by 17.1%, with the median rising from 246 ms to 288 ms.

MAC layer retransmission ratios increase by approximately an order of magnitude, from 2.4–2.8% in S0 to 24.6–25.4% in S2. Absolute retransmission counts rise substantially: UDP from 192 to 2,345, Modbus TCP from 312 to 4,375, and WebSocket from 536 to 8,263. A retransmission ratio near 25% is consistent with the configured 15% frame loss rate over a multi-hop path. Each hop independently experiences frame loss, triggering link-layer retransmission before forwarding. This link-layer recovery limits error propagation to transport and application layers, which is consistent with the observed delivery ratios under degraded channel conditions.

The lower maximum latency values under S2 relative to S0 are consistent with fewer

transport-layer retransmission timeout (RTO) events. In S0, sporadic MAC layer failures can exhaust retransmission limits, leading to segment loss and triggering TCP's RTO mechanism (typically 1–3 seconds). In S2, the configured 15% loss rate results in MAC retransmissions that are observed within the link layer before segments are lost, reducing the likelihood of escalation to transport-layer recovery. The S2 maximum latencies of 700–1200 ms are consistent with accumulated MAC retry delays rather than RTO events.

Larger messages exhibit greater latency sensitivity to channel loss (HP3: +20.7%, INV4: +17.2%) compared to small messages (HP1: +1.5%, OC5: +2.8%). This difference is consistent with 6LoWPAN fragmentation. Messages exceeding the 102-byte IEEE 802.15.4 payload require multiple fragments, each independently subject to channel loss and retransmission. The cumulative delay across fragments is consistent with the larger relative latency increases for larger payloads. A retransmission ratio near 25% combined with high delivery rates is consistent with IEEE 802.15.4 link-layer recovery mechanisms under the tested loss rate. Under these conditions, channel degradation is reflected primarily as increased latency rather than delivery failure.

4.5.4 Fragmentation impact

We evaluate the impact of 6LoWPAN fragmentation on message delivery and latency. The S3 scenario maintains the same topology as S0 but reduces the effective link-layer payload budget by decreasing the packet buffer size from 128 bytes to 80 bytes. This reduction lowers the maximum payload that can be carried in a single IEEE 802.15.4

frame and results in additional fragmentation by the 6LoWPAN adaptation layer for the same IPv6 and application traffic. Table 4.12 reports the application payload sizes for the message types shown in the figures and the corresponding estimated minimum number of IEEE 802.15.4 MAC frames required under S0 and S3, using the measured per-frame MAC payload budgets (104 B and 59 B, respectively). A comparison of application layer delivery performance, one way latency metrics and MAC layer transmission behavior between S0 and S3 is summarized in Tables 4.13–4.15 and Figures 4.11 and 4.12.

Table 4.12: Message sizes for the message types shown in the figures and the calculated minimum number of IEEE 802.15.4 MAC frames required. Calculations use the measured per-frame MAC payload budget (S0: 104 B, S3: 59 B) and $N = \lceil \text{Size}/B \rceil$.

Protocol	**Msg**	**Size (B)**	**S0 Frames**	**S3 Frames**
UDP	HP1	48	1	1
UDP	HP2	24	1	1
UDP	HP3	200	2	4
Modbus TCP	INV3	120	2	3
Modbus TCP	INV4	120	2	3
WebSocket	OC4	420	5	8
WebSocket	OC5	80	1	2

Delivery ratios remain stable under reduced MTU conditions. UDP delivery shows 98.8% in S3 versus 97.6% in S0, a difference of 3 packets that is consistent with test-to-test variation for a 250 message sample size. TCP-based protocols maintain 100% delivery for both Modbus TCP and WebSocket, which is consistent with fragmentation not introducing additional losses under the tested channel conditions.

Latency increases differ across message types. HP2 exhibits no change, with the median remaining at 130 ms. HP1 increases by 61.7%, with the median rising from 133 ms to

Table 4.13: Application layer delivery performance comparison between S0 and S3 conditions. Each message type was transmitted 250 times per scenario.

Protocol	Msg	S0			S3		
		Sent	Recv	%	Sent	Recv	%
UDP	HP1	250	244	97.6	250	247	98.8
UDP	HP2	250	244	97.6	250	247	98.8
UDP	HP3	250	244	97.6	250	247	98.8
Modbus TCP	INV3	250	250	100.0	250	250	100.0
Modbus TCP	INV4	250	250	100.0	250	250	100.0
WebSocket	OC4	250	250	100.0	250	250	100.0
WebSocket	OC5	250	250	100.0	250	250	100.0

Table 4.14: One-way latency measurements comparison between S0 and S3 conditions. Values represent median, 95th percentile and maximum latency in milliseconds.

Protocol	Msg	S0 (ms)			S3 (ms)		
		Med	P95	Max	Med	P95	Max
UDP	HP1	133	216	351	215	302	475
UDP	HP2	130	353	537	130	280	777
UDP	HP3	397	555	844	575	731	949
Modbus TCP	INV3	147	223	381	211	274	369
Modbus TCP	INV4	358	467	2270	548	646	806
WebSocket	OC4	246	510	1675	431	579	3304
WebSocket	OC5	143	430	2924	220	451	2385

Table 4.15: MAC layer transmission statistics comparison between S0 and S3 conditions.

Protocol	S0			S3		
	TX	Retrans	%	TX	Retrans	%
UDP	6,838	192	2.81	11,191	250	2.23
Modbus TCP	12,906	312	2.42	22,122	433	1.96
WebSocket	20,777	536	2.58	36,800	792	2.15

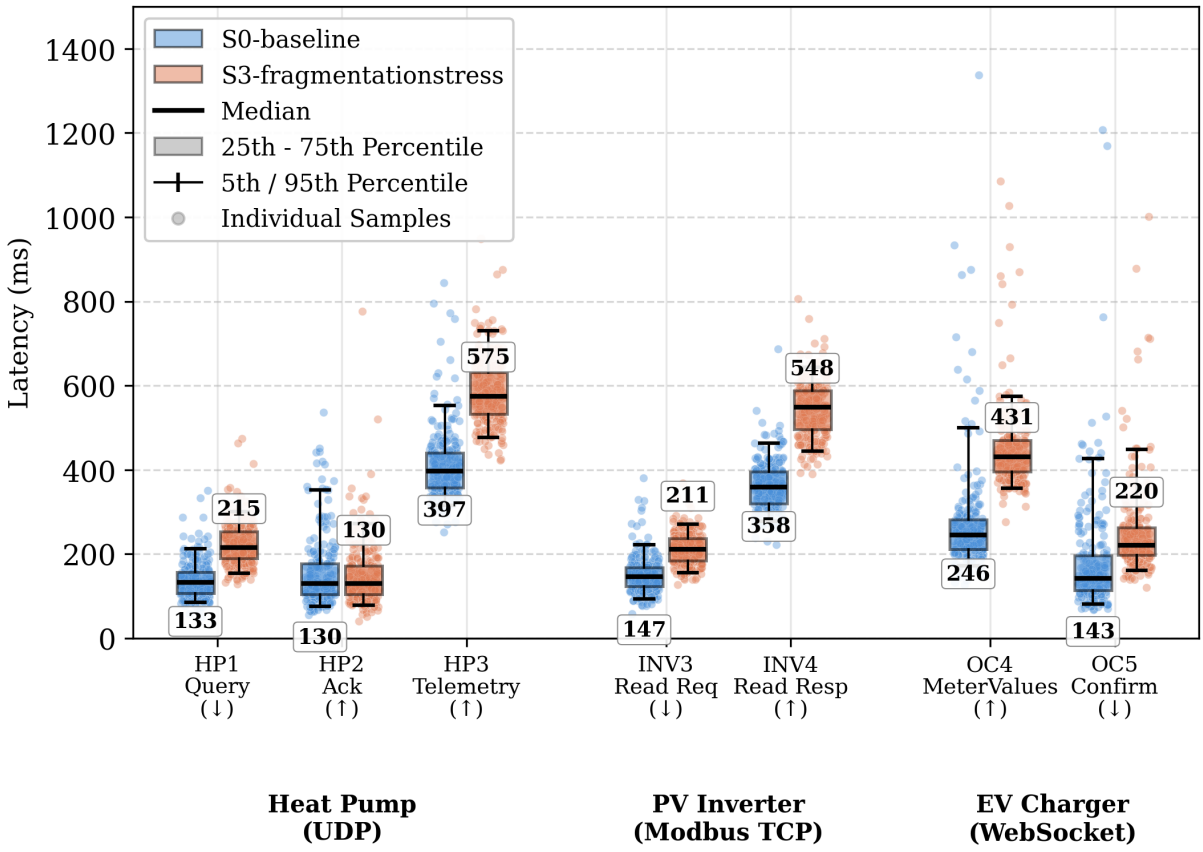

Figure 4.11: Latency distribution comparison between S0 (blue) and S3 (orange) conditions. Box plots show 25th to 75th percentile with whiskers extending to 5th and 95th percentiles. Median values are annotated above each box. Outliers above 1500ms: INV4 (S0-baseline): 1 (max 2270ms); OC4 (S0-baseline): 1 (max 1675ms); OC4 (S3-fragmentationstress): 1 (max 3304ms).

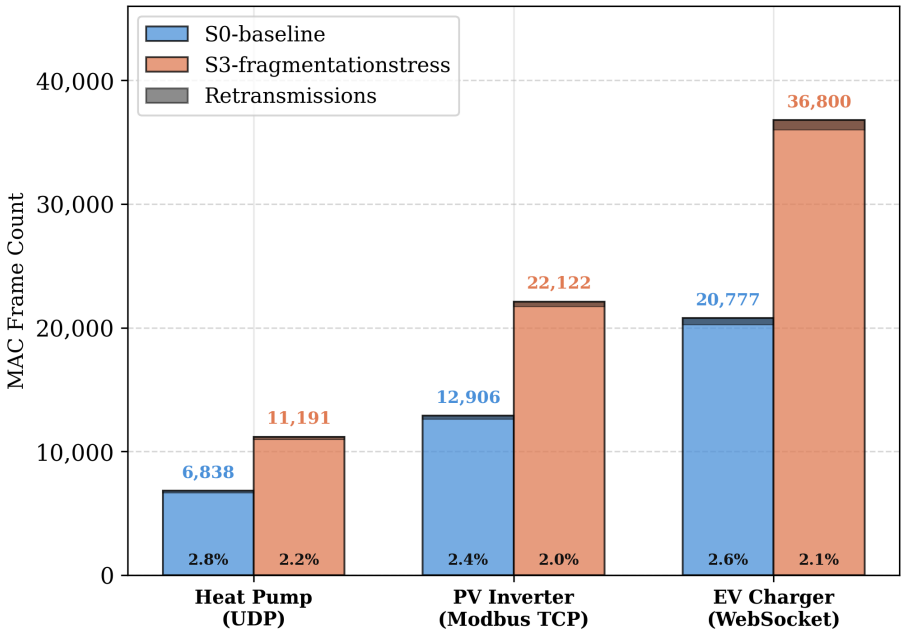

Figure 4.12: MAC layer transmission comparison between S0 and S3 conditions. Retransmission percentages are shown inside bars.

215 ms, and HP3 increases by 44.8%, with the median rising from 397 ms to 575 ms. Modbus TCP messages show 43.5–53.0% increases, with INV3 rising from 147 ms to 211 ms and INV4 rising from 358 ms to 548 ms. WebSocket shows larger relative increases, with OC4 rising by 75.4% from 246 ms to 431 ms and OC5 rising by 54.2% from 143 ms to 220 ms. MAC layer transmission counts increase by 64–77% across protocols. UDP increases from 6,838 to 11,191, corresponding to a 63.7% increase. Modbus TCP increases from 12,906 to 22,122, corresponding to a 71.4% increase. WebSocket increases from 20,777 to 36,800, corresponding to a 77.1% increase. Despite increased frame count, retransmission ratios decrease slightly. UDP decreases from 2.81% to 2.23%, Modbus TCP decreases from 2.42% to 1.96%, and WebSocket decreases from 2.58% to 2.15%.

Maximum latency behavior differs across protocols. Modbus TCP shows reduced maximum values, with INV4 decreasing from 2270 ms to 806 ms, corresponding to a 65% reduction. UDP shows mixed changes, with HP2 increasing from 537 ms to 777 ms while HP3 increases from 844 ms to 949 ms. WebSocket shows higher dispersion, with OC4 increasing from 1675 ms to 3304 ms while OC5 decreases from 2924 ms to 2385 ms.

The unchanged median latency for HP2 is consistent with the message fitting within a single 6LoWPAN fragment under both configurations. Messages that require additional fragments under reduced frame size show higher latency because each fragment is transmitted and acknowledged independently and may be retransmitted. The observed 44–75% median latency increases are consistent with messages transitioning from one or two fragments to three to five fragments. The 64–77% increase in MAC frame count is consistent

with fragmentation overhead. With frame size reduced, the effective 6LoWPAN payload decreases proportionally, requiring more frames per message. The larger increase for WebSocket is consistent with larger average message size due to JSON encoding and WebSocket framing, which increases the number of fragments per OCPP transaction.

The slight decrease in retransmission ratio can occur under fragmentation conditions if per-frame loss probability decreases. Smaller fragments have shorter transmission durations, which can reduce collision probability during Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) channel access. Shorter frames also reduce the number of bits exposed to channel errors per transmission, which can increase frame check sequence success probability at the receiver. The increase in OC4 maximum latency from 1675 ms to 3304 ms is consistent with WebSocket message fragmentation interacting with transport-layer retransmissions. OCPP messages contain JSON payloads that can expand to multiple fragments, and WebSocket framing adds per-message overhead. When a fragment loss triggers TCP retransmission, message completion is delayed. A maximum latency on the order of 3304 ms is consistent with a retransmission timeout event combined with accumulated fragment transmission delay across the multi-hop path.

The stable delivery ratios are consistent with successful 6LoWPAN reassembly at destination nodes under the tested conditions. The reassembly buffer holds fragments until the final fragment arrives, with a typical timeout of 60 seconds. Under favorable channel conditions, fragment loss is infrequent, and reassembly completes. The increased latency dispersion is consistent with fragment arrival timing and retransmission events having

greater influence as the number of fragments increases.

4.5.5 Scale impact

Performance under increased network density is evaluated in this experiment. The S4 scenario expands the network from 25 to 60 nodes using a two cluster topology with 6 hops maximum. The scenario uses 99% frame reception ratio. A comparison of application layer delivery performance, one way latency metrics and MAC layer transmission behavior between S0 and S4 is summarized in Tables 4.16–4.18 and Figures 4.13 and 4.14.

Table 4.16: Application layer delivery performance comparison between S0 and S4 conditions. Each message type was transmitted 250 times per scenario.

Protocol	**Message**	**S0**			**S4**		
		Sent	**Recv**	**%**	**Sent**	**Recv**	**%**
UDP	HP1	250	244	97.6	250	247	98.8
UDP	HP2	250	244	97.6	250	247	98.8
UDP	HP3	250	244	97.6	250	247	98.8
Modbus TCP	INV3	250	250	100.0	250	250	100.0
Modbus TCP	INV4	250	250	100.0	250	250	100.0
WebSocket	OC4	250	250	100.0	250	250	100.0
WebSocket	OC5	250	250	100.0	250	250	100.0

Table 4.17: One-way latency measurements comparison between S0 and S4 conditions. Values represent median, 95th percentile and maximum latency in milliseconds.

Protocol	**Msg**	**S0 (ms)**			**S4 (ms)**		
		Med	**P95**	**Max**	**Med**	**P95**	**Max**
UDP	HP1	133	216	351	205	322	719
UDP	HP2	130	353	537	203	407	1088
UDP	HP3	397	555	844	639	837	1107
Modbus TCP	INV3	147	223	381	210	410	1500
Modbus TCP	INV4	358	467	2270	595	759	2715
WebSocket	OC4	246	510	1675	400	626	2460
WebSocket	OC5	143	430	2924	207	535	2005

Table 4.18: MAC layer transmission statistics comparison between S0 and S4 conditions.

Protocol	**S0**			**S4**		
	TX	**Retrans**	**%**	**TX**	**Retrans**	**%**
UDP	6,838	192	2.81	11,268	407	3.61
Modbus TCP	12,906	312	2.42	22,106	601	2.72
WebSocket	20,777	536	2.58	35,063	903	2.58

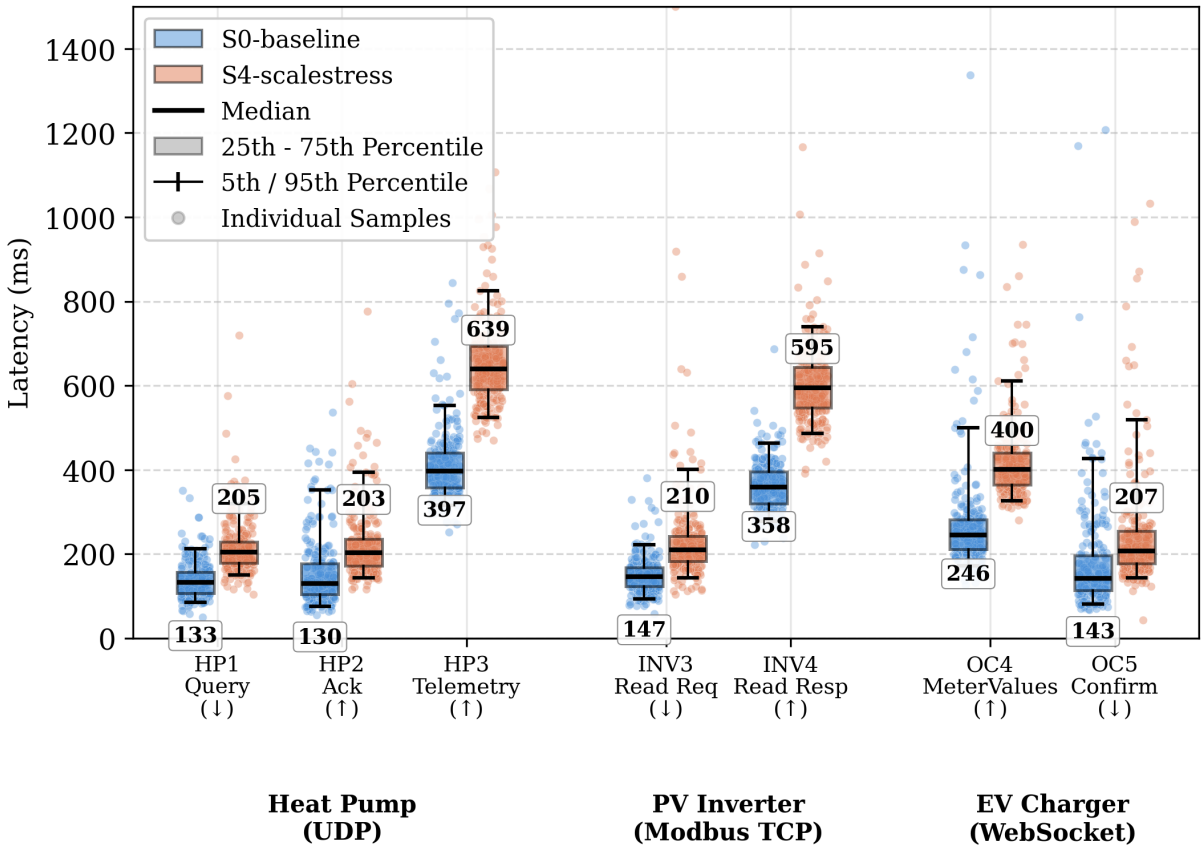

Figure 4.13: Latency distribution comparison between S0 (blue) and S4 (orange) conditions. Box plots show 25th to 75th percentile with whiskers extending to 5th and 95th percentiles. Median values are annotated above each box. Outliers above 1500ms: INV4 (S0-baseline): 1 (max 2270ms); INV4 (S4-scalestress): 2 (max 2715ms); OC4 (S0-baseline): 1 (max 1675ms).

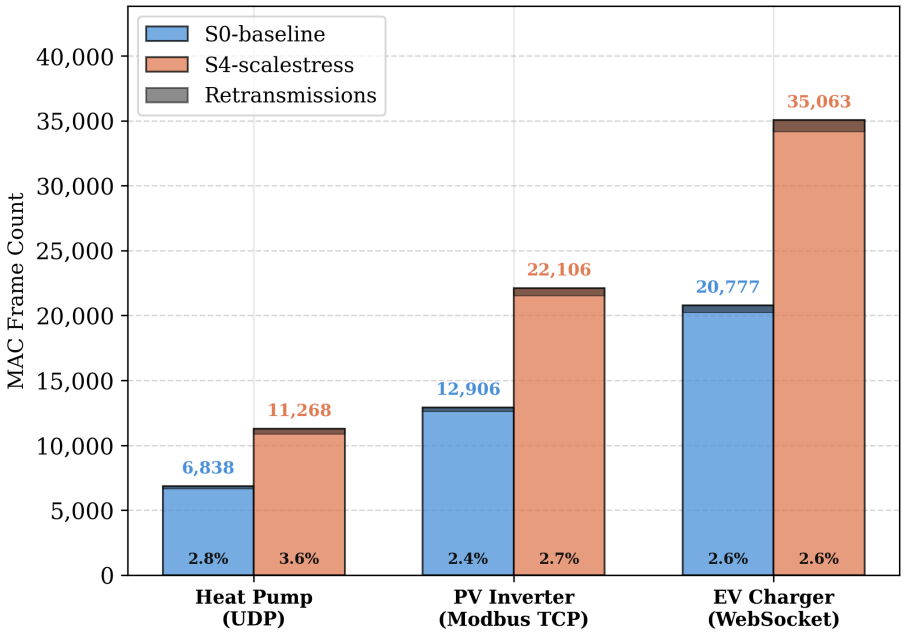

Figure 4.14: MAC layer transmission comparison between S0 and S4 conditions. Retransmission percentages are shown inside bars.

Delivery ratios remain stable under increased network scale. UDP delivery shows 98.8% in S4 versus 97.6% in S0, a difference of 3 packets that is consistent with test-to-test variation for a 250 message sample size. TCP-based protocols maintain 100% delivery for both Modbus TCP and WebSocket across both scenarios, which is consistent with network scaling from 25 to 60 nodes not introducing additional packet loss under the tested traffic patterns.

All message types exhibit latency increases in the 43–66% range. UDP messages show 54–61% increases, with HP1 median latency rising from 133 ms to 205 ms, HP2 median latency rising from 130 ms to 203 ms, and HP3 median latency rising from 397 ms to 639 ms. Modbus TCP follows a similar pattern with 43–66% increases, with INV3 median latency rising from 147 ms to 210 ms and INV4 median latency rising from 358 ms to 595 ms. WebSocket exhibits 45–63% increases, with OC4 median latency rising from 246 ms to 400 ms and OC5 median latency rising from 143 ms to 207 ms. The similarity in relative increases across protocols is consistent with shared network-level contributors to latency. MAC layer transmission counts increase by 65–71% across protocols. UDP increases from 6,838 to 11,268, corresponding to a 64.8% increase. Modbus TCP increases from 12,906 to 22,106, corresponding to a 71.3% increase. WebSocket increases from 20,777 to 35,063, corresponding to a 68.8% increase. Retransmission ratios show modest increases. UDP rises from 2.81% to 3.61%, Modbus TCP rises from 2.42% to 2.72%, while WebSocket remains at 2.58%.

Maximum latency values show varied responses to scale. INV3 maximum increases from

381 ms to 1500 ms, while HP2 maximum increases from 537 ms to 1088 ms. OC5 maximum decreases from 2924 ms to 2005 ms, and INV4 maximum increases from 2270 ms to 2715 ms. P95 values increase more uniformly, with INV3 increasing from 223 ms to 410 ms and OC4 increasing from 510 ms to 626 ms. The increase in INV3 maximum is consistent with intermittent queueing or contention events under the S4 traffic concentration conditions. The S4 scenario uses a two-cluster topology that routes traffic through a constrained relay corridor rather than providing multiple parallel paths. The border router (node 1) is positioned at the network center, while the gateway (node 2) is located at the far edge and connected through relay nodes 3–8. This topology concentrates forwarding load near the gateway and corridor relays, which is consistent with the observed latency increases and the increase in MAC retransmissions.

The latency increase is consistent with a combination of longer paths and increased contention near the bottleneck corridor. With 6 hops maximum in S4 compared to 4 hops in S0, the additional hops contribute to increased propagation and forwarding delay. The remaining increase is consistent with additional contributors such as routing lookup overhead in larger networks, increased CSMA-CA backoff under contention near the corridor, and queueing at relays that serve multiple flows. The 60-node configuration also requires larger routing state, which can increase per-packet routing lookup and control-plane processing overhead. While RPL's distance-vector approach limits per-node state, the increased number of candidate parents and neighbors in clustered regions can increase the time spent in parent selection and next-hop lookup. The relatively uniform latency increase across protocols is consistent with these effects applying primarily at the network

and link layers rather than at the transport layer.

Across the tested conditions, delivery remains stable while latency increases are observed with increased network scale. The observed median latency increases in the 43–66% range quantify the change in performance between S0 and S4 for the selected topology and traffic patterns. These results are specific to the evaluated topology and parameters and may differ under alternative network layouts, traffic rates, or channel conditions.

5. Conclusion

5.1 Summary

This thesis investigated the connectivity challenges of LoWPAN-based smart grid applications from the perspective of multi-residential building deployments. The work identified the key actors involved in behind-the-meter communication and examined the inherent characteristics of low-power wireless networks, including packet loss patterns, network dynamics and the structural properties of application protocols operating over constrained links.

These aspects were addressed through a holistic approach that combined theoretical analysis with practical experimentation. To enable systematic investigation of dynamic network behavior, a configurable test suite was developed and released as open-source software. This framework provides researchers and practitioners with a reproducible environment for evaluating smart grid communication under various stress conditions. Additionally, data imputation techniques were explored to address measurement gaps arising from packet loss and network instability, demonstrating methods for recovering missing telemetry data in constrained environments.

The experimental evaluation demonstrated the test suite capabilities through five distinct scenarios examining baseline performance, hop count scaling, channel loss resilience, fragmentation overhead and network density effects. The results revealed protocol-specific behaviors and quantified the impact of network conditions on latency, delivery ratio and

MAC layer overhead for UDP, Modbus TCP and WebSocket-based OCPP communication.

By providing an integrated testing methodology with configurable scenarios, automated measurement collection and data recovery capabilities, this work contributes toward interoperable, practical and reliable communication solutions for electrification initiatives in multi-residential settings. The findings and tools presented here aim to support the deployment of scalable smart grid communication infrastructure in dense urban environments.

5.2 Future work

The current implementation relies on SLIP to connect mesh network nodes to the external host system. While functional, SLIP introduces limitations in bandwidth and concurrent access capabilities. One potential extension is to implement a custom socket-handling mechanism in Contiki-NG, enabling the host operating system to establish socket communication with nodes without a SLIP intermediary. This approach could support scaling the test suite workload beyond current limits and enable the use of standard socket tools and protocols for debugging and monitoring. The external interface type could be made user-configurable, offering both SLIP and socket-based options depending on experimental requirements.

In addition to transport-level changes, further work may focus on deeper integration of data imputation into the test suite workflow. Integrating on-the-fly data imputation could increase topology awareness at border routers and support workload-aware behavior during experiments. Exposing imputation within the interactive test menu could allow

users to observe the impact of missing-data recovery while running scenarios, rather than only during post-processing.

The current simple node implementation transmits only essential network telemetry to minimize interference with test measurements. This choice supports scenario interpretation by reducing background traffic that is unrelated to the introduced stress conditions. For large-scale evaluation scenarios, the simple node behavior could be extended to generate application-layer traffic. Configurable roles such as smart meters, environmental sensors, or other smart grid elements could enable experimentation with heterogeneous traffic patterns. Scenario-configurable node behavior could broaden the set of deployment models that can be evaluated using the framework.

Beyond node behavior and traffic models, the framework could also be extended to support more complex routing architectures. The current test suite operates within a single DODAG with all nodes joining a common RPL instance. Supporting multiple DODAGs with separate root nodes could enable experimentation with multi-root deployments and segmented network domains. Layer 3 Virtual Private Network (VPN)-like tunneling mechanisms between DODAGs could be implemented to evaluate communication between network segments and to study load distribution and failover behavior under multi-root architectures. Such tunnels could also allow communication across isolated network segments in scenarios where multiple administrative domains or geographic regions must coordinate. This extension would allow the test suite to be used to evaluate federated smart grid deployments spanning multiple network boundaries.

References

1. Li, X., Lepour, D., Heymann, F. & Maréchal, F. Electrification and digitalization effects on sectoral energy demand and consumption: A prospective study towards 2050. *Energy* **279,** 127992 (June 2023).
2. Park, T., Touchie, M., O'Brien, W., Lee, T. & Peters, T. *Identifying Policy Barriers to Multi-Unit Residential Building Design Elements that Promote Good Indoor Environmental Quality in Canada* in *Proceedings of eSim 2024: 13th Conference of IBPSA-Canada* **13** (IBPSA-Canada, Alberta, Canada, 2022).
3. Karmaker, A. K., Prakash, K., Siddique, M. N. I., Hossain, M. A. & Pota, H. Electric vehicle hosting capacity analysis: Challenges and solutions. *Renewable and Sustainable Energy Reviews* **189,** 113916 (2024).
4. Tun, T. P., Ceylan, O. & Pisica, I. A Real-World Case Study Towards Net Zero: EV Charger and Heat Pump Integration in End-User Residential Distribution Networks. *Energies* **18,** 2510 (2025).
5. Fournier, E. D., Cudd, R., Smithies, S. & Pincetl, S. Quantifying the electric service panel capacities of California's residential buildings. *Energy Policy* **192,** 114238 (2024).
6. Sandoval, N. *et al.* Achieving equitable widespread residential building electrification–examining barriers, strategies, and opportunities using Los Angeles as a case study. *Applied Energy* **384,** 125498 (2025).
7. Hong, S.-K., Lee, S. G. & Kim, M. Assessment and mitigation of electric vehicle charging demand impact to transformer aging for an apartment complex. *Energies* **13,** 2571 (2020).
8. *Open Charge Point Protocol (OCPP) 1.6* Version 1.6, Open Charge Alliance, Arnhem, The Netherlands. Open Charge Alliance (2017).
9. *Open Charge Point Protocol (OCPP) 2.0.1* Version 2.0.1, Open Charge Alliance, Arnhem, The Netherlands. Open Charge Alliance (Mar. 2020).
10. Neal, D. C., Rogers, D. J. & McCulloch, M. A techno-economic analysis of communication in low-voltage islanded microgrids. *Renewable and Sustainable Energy Reviews* **209,** 115031 (2025).
11. Chai, Y. & Zeng, X.-J. The development of green wireless mesh network: A survey. *Journal of Smart Environments and Green Computing* **1,** 47–59 (2021).
12. Chai, Y., Zeng, X.-J. & Liu, Z. The future of wireless mesh network in next-generation communication: a perspective overview. *Evolving Systems* **15,** 1635–1648 (2024).
13. Lei, L. & Wang, X. Scaling laws of average end-to-end delay in multi-tier wireless mesh networks. *Computer Networks,* 111566 (2025).
14. Hassan, N., Fernando, X., Woungang, I. & Anpalagan, A. User association performance trade-offs in integrated RF/mmWave/THz communications. *Future Internet* **15,** 376 (2023).
15. Gupta, R., Naware, V., Vattem, A. & Hussain, A. M. *A wi-sun network-based electric vehicle charging station using open charge point protocol (ocpp) and onem2m platform* in *2024 IEEE Applied Sensing Conference (APSCON)* (2024), 1–4.
16. Wang, J., Hao, Y. & Yang, C. The current progress and future prospects of path loss model for terrestrial radio propagation. *Electronics* **12,** 4959 (2023).

17. Maggio, M., Hamann, A., Mayer-John, E. & Ziegenbein, D. *Control-system stability under consecutive deadline misses constraints* in *32nd euromicro conference on real-time systems (ECRTS 2020)* (2020), 21–1.

18. Fattahi, J. *Real-Time Data Imputation for Low Inertia Nanogrid Digital Twins* in *2024 IEEE 10th World Forum on Internet of Things (WF-IoT)* (2024), 654–659.

19. Samadi, M. & Fattahi, J. Low-inertia microgrid synchronization using data-driven digital twins. *IEEE Access* **12,** 78534–78548 (2024).

20. Abrahamsen, F. E., Ai, Y. & Cheffena, M. Communication technologies for smart grid: A comprehensive survey. *Sensors* **21,** 8087 (2021).

21. Natgunanathan, I., Fernando, N., Loke, S. W. & Weerasuriya, C. Bluetooth low energy mesh: Applications, considerations and current state-of-the-art. *Sensors* **23,** 1826 (2023).

22. Seferagić, A., Famaey, J., De Poorter, E. & Hoebeke, J. Survey on wireless technology trade-offs for the industrial internet of things. *Sensors* **20,** 488 (2020).

23. Mozaffariahrar, E., Theoleyre, F. & Menth, M. A survey of Wi-Fi 6: Technologies, advances, and challenges. *Future Internet* **14,** 293 (2022).

24. IEEE Standard for Low-Rate Wireless Networks Corrigendum 1: Correction of Errors Preventing Backward Compatibility. *IEEE Std 802.15.4-2020/Cor 1-2022 (Corrigendum to IEEE Std 802.15.4-2020 as amended by IEEE Std 802.15.4z-2020, IEEE Std 802.15.4w-2020, IEEE Std 802.15.4y-2021, and IEEE Std 802.15.4aa-2022),* 1–22 (2023).

25. Natkaniec, M. & Bieryt, N. An analysis of the mixed IEEE 802.11 ax wireless networks in the 5 GHz band. *Sensors* **23,** 4964 (2023).

26. Adame, T., Carrascosa-Zamacois, M. & Bellalta, B. Time-sensitive networking in IEEE 802.11 be: On the way to low-latency WiFi 7. *Sensors* **21,** 4954 (2021).

27. Oughton, E. J. *et al.* Revisiting wireless internet connectivity: 5G vs Wi-Fi 6. *Telecommunications Policy* **45,** 102127 (2021).

28. International Telecommunication Union, Radiocommunication Sector (ITU-R). *Prediction of building entry loss* tech. rep. P.2109-2 (International Telecommunication Union (ITU), Geneva, Switzerland, Aug. 2023).

29. Doppler, K. *et al.* Future indoor network with a sixth sense: Requirements, challenges and enabling technologies. *Pervasive and Mobile Computing* **83,** 101571 (2022).

30. *General technical requirements (for U-NII devices)* Code of Federal Regulations, Title 47, Telecommunication, Part 15, Subpart E. Federal Communications Commission (FCC), 2025.

31. Rostamikafaki, Z., Chan, F. & D'amours, C. 5G New Radio Signal Propagation and Ground-to-Air Channel Modeling at 3.565 GHz Based on Extensive Measurements. *IEEE Access* (2024).

32. Ali, A. I. & Zorlu Partal, S. Development and performance analysis of a ZigBee and LoRa-based smart building sensor network. *Frontiers in Energy Research* **10,** 933743 (2022).

33. Eltholth, A. A. Improved spectrum coexistence in 2.4 GHz ISM band using optimized chaotic frequency hopping for Wi-Fi and Bluetooth signals. *Sensors* **23,** 5183 (2023).

34. Lazaro, M., Lazaro, A., Gonzalez, B., Villarino, R. & Girbau, D. Long-Range Wireless System for U-Value Assessment Using a Low-Cost Heat Flux Sensor. *Sensors* **22,** 7259 (2022).

35. Netshikweta, N. R. & Sumbwanyambe, M. *Power Control Techniques for Interference Management—A Systematic Review* in *Telecom* **6** (2024), 2.

36. Gohary, R. H., Huang, Y., Luo, Z.-Q. & Pang, J.-S. A generalized iterative water-filling algorithm for distributed power control in the presence of a jammer. *IEEE Transactions on Signal Processing* **57,** 2660–2674 (2009).

37. Alzubaidi, O. T. H., Alheejawi, S., Hindia, M. N., Dimyati, K. & Noordin, K. A. Interference mitigation strategies in beyond 5G wireless systems: A review. *Electronics* **14,** 2237 (2025).

38. Shao, C., Park, H., Roh, H., Lee, W. & Kim, H. PolarScout: Wi-Fi interference-resilient ZigBee communication via shell-shaping. *IEEE/ACM Transactions on Networking* **28,** 1587–1600 (2020).

39. Dash, B. K. & Peng, J. Zigbee Wireless Sensor Networks: Performance Study in an Apartment-Based Indoor Environment. *Journal of Computer Networks and Communications* **2022,** 2144702 (2022).

40. Saavedra, E., Mascaraque, L., Calderon, G., Del Campo, G. & Santamaria, A. The smart meter challenge: Feasibility of autonomous indoor iot devices depending on its energy harvesting source and iot wireless technology. *Sensors* **21,** 7433 (2021).

41. *Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz; Harmonised Standard for access to radio spectrum; Part 2: Non-specific radio equipment* version V3.3.1. European Telecommunications Standards Institute (ETSI), 2025.

42. *Licence-Exempt Radio Apparatus: Category I Equipment* RSS-210. Innovation, Science and Economic Development Canada (ISED) (2024).

43. *ERC Recommendation (70-03) Relating to the Use of Short Range Devices (SRD)* Edition of February 2025. European Conference of Postal and Telecommunications Administrations (CEPT), 2025.

44. *47 CFR § 15.247: Operation within the bands 902–928 MHz, 2400–2483.5 MHz, and 5725–5850 MHz* 47 CFR Ch. I (10-1-24 Edition). U.S. Government Publishing Office (2024).

45. *Digital Transmission Systems (DTSs), Frequency Hopping Systems (FHSs) and Licence-Exempt Local Area Network (LE-LAN) Devices* RSS-247. Regulations covering the devices operate in 902–928 MHz band in Canada. Innovation, Science and Economic Development Canada (ISED) (2025).

46. *IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 3: Physical Layer (PHY) Specifications for Low-Data-Rate, Wireless, Smart Metering Utility Networks* Published 2012-04-27. Institute of Electrical and Electronics Engineers (IEEE), 2012.

47. Lavric, A., Petrariu, A. I. & Popa, V. *SigFox Communication Protocol: The New Era of IoT?* in *2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI)* (2019), 1–4.

48. Rahman, H. U., Ahmad, M., Ahmad, H. & Habib, M. A. *LoRaWAN: State of the Art, Challenges, Protocols and Research Issues* in *2020 IEEE 23rd International Multitopic Conference (INMIC)* (2020), 1–6.

49. Thrane, J., Malarski, K. M., Christiansen, H. L. & Ruepp, S. *Experimental Evaluation of Empirical NB-IoT Propagation Modelling in a Deep-Indoor Scenario* in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference* (2020), 1–6.

50. International Telecommunication Union. *Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range from 300 MHz to 450 GHz* Recommendation P.1238-11. (09/2021) (ITU-R, Sept. 2021).

51. Tian, L., Santi, S., Seferagić, A., Lan, J. & Famaey, J. Wi-Fi HaLow for the Internet of Things: An up-to-date survey on IEEE 802.11 ah research. *Journal of Network and Computer Applications* **182,** 103036 (2021).

52. Kane, L., Liu, V., McKague, M. & Walker, G. An Experimental Field Comparison of Wi-Fi HaLow and LoRa for the Smart Grid. *Sensors* **23,** 7409 (2023).

53. Huang, C. *et al.* Smart meter pinging and reading through AMI two-way communication networks to monitor grid edge devices and DERs. *IEEE Transactions on Smart Grid* **13,** 4144–4153 (2021).

54. Myoung, N., Kwon, Y., Park, M. & Eun, C. Data interworking model and analysis for harmonization of smart metering protocols in IoT-based AMI system. *Sensors* **23,** 2903 (2023).

55. García-Martín, J. P. & Torralba, A. Energy consumption analytical modeling of NB-IoT devices for diverse IoT applications. *Computer Networks* **232,** 109855 (2023).

56. Mochinski, M. A. *et al.* Towards an efficient method for large-scale wi-SUN-enabled AMI network planning. *Sensors* **22,** 9105 (2022).

57. Quispe, A. A. *et al.* DLMS over Wi-SUN FAN Networks: Performance Evaluation. *Applied Sciences* **15,** 12499 (2025).

58. Quispe, A. A. *et al.* Parallel Rendezvous Strategy for Node Association in Wi-SUN FAN Networks. *Sensors* **25,** 6213 (2025).

59. Piyare, R., Oikonomou, G. & Elsts, A. TSCH for long range low data rate applications. *IEEE Access* **8,** 228754–228766 (2020).

60. Dobrilović, D., Mazalica, M. & Gecin, G. The performance analyses of ieee 802.15. 4g sun low-power wireless networks and their application. *Interdisciplinary Description of Complex Systems: INDECS* **20,** 250–256 (2022).

61. Sanz, A., Ibar, J. C. & Lacasa, L. *PLC-RF hybrid communication systems, model and simulation* in *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)* (2021), 158–163.

62. *IEEE Standard for Wireless Smart Utility Network Field Area Network (FAN)* Published 2021-06-25. Institute of Electrical and Electronics Engineers (IEEE), 2021.

63. Maleki, A., Nguyen, H. H., Bedeer, E. & Barton, R. A tutorial on chirp spread spectrum modulation for LoRaWAN: Basics and key advances. *IEEE Open Journal of the Communications Society* (2024).

64. Basford, P. J., Bulot, F. M., Apetroaie-Cristea, M., Cox, S. J. & Ossont, S. J. LoRaWAN for smart city IoT deployments: A long term evaluation. *Sensors* **20,** 648 (2020).

65. Diane, A., Diallo, O. & Ndoye, E. H. M. A systematic and comprehensive review on low power wide area network: characteristics, architecture, applications and research challenges. *Discover Internet of Things* **5,** 7 (2025).

66. *Communication systems for meters and remote reading of meters - Part 4: Wireless meter reading (Wireless M-Bus)* Approved by CEN on 2019-02-25. European Committee for Standardization (CEN), 2019.

67. *IEEE Standard for Information technology–Telecommunications and information exchange between systems - Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1 GHz License Exempt Operation* Published 2017-05-05. Institute of Electrical and Electronics Engineers (IEEE), 2017.

68. Farhad, A. & Pyun, J.-Y. Resource management for massive internet of things in IEEE 802.11 ah wlan: Potentials, current solutions, and open challenges. *Sensors* **22,** 9509 (2022).

69. Quispe, A. A., Riella, R. J., Iantorno, L. M., Mariani, L. S. & Fernandez, E. M. G. Analysis of Wi-SUN FAN network formation time. *Sensors* **24,** 1142 (2024).

70. Hirakawa, R., Mizutani, K. & Harada, H. Specification and performance analysis of Wi-SUN FAN. *IEEE Open Journal of Vehicular Technology* **4,** 849–866 (2023).

71. Uribe-Pérez, N., Hernández, L., De la Vega, D. & Angulo, I. State of the art and trends review of smart metering in electricity grids. *Applied Sciences* **6,** 68 (2016).

72. Lichtensteiger, B., Bjelajac, B., Müller, C. & Wietfeld, C. *RF mesh systems for smart metering: System architecture and performance* in *2010 first IEEE international conference on smart grid communications* (2010), 379–384.

73. Alavikia, Z. & Shabro, M. A comprehensive layered approach for implementing internet of things-enabled smart grid: A survey. *Digital Communications and Networks* **8,** 388–410 (2022).

74. Kirmani, S., Mazid, A., Khan, I. A. & Abid, M. A survey on IoT-enabled smart grids: technologies, architectures, applications, and challenges. *Sustainability* **15,** 717 (2022).

75. Veenadhari, C. L. & Kandasamy, S. *A Comprehensive Survey on IPv6 Over Low Power Wireless Personal Area Networks (6LoWPAN)* in *2024 International Conference on Social and Sustainable Innovations in Technology and Engineering (SASI-ITE)* (2024), 78–83.

76. Zahra, S. R. & Chishti, M. A. Packet header compression in the Internet of Things. *Procedia Computer Science* **173,** 64–69 (2020).

77. Tömösközi, M., Reisslein, M. & Fitzek, F. H. Packet header compression: A principle-based survey of standards and recent research studies. *IEEE Communications Surveys & Tutorials* **24,** 698–740 (2022).

78. Dumay, M. *et al.* Enabling extremely energy-efficient end-to-end secure communications for smart metering internet of things applications using static context header compression. *Applied Sciences* **13,** 11921 (2023).

79. Xiaoyu, W., Xu, L., Chuan, L., Jing, T. & Zhonghua, L. *IPv6 Header Compression Scheme for Power Internet of Things* in *International Conference on Smart Grid Inspired Future Technologies* (2020), 107–116.

80. Haggag, A. Implementation and evaluation of IPv6 with compression and fragmentation for throughput improvement of internet of things networks over IEEE 802.15. 4. *Wireless Personal Communications* **130,** 1449–1477 (2023).

81. Bruniaux, A., Koutsiamanis, R.-A., Papadopoulos, G. Z. & Montavont, N. Defragmenting the 6lowpan fragmentation landscape: A performance evaluation. *Sensors* **21,** 1711 (2021).

82. Ayers, H. *et al. Design considerations for low power internet protocols* in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems* (2018), 317–318.

83. Alyami, S., Alharbi, R. & Azzedin, F. Fragmentation attacks and countermeasures on 6LoWPAN Internet of Things networks: Survey and simulation. *Sensors* **22,** 9825 (2022).

84. Hong, Y.-G., Gomez, C., Choi, Y., Sangi, A. R. & Chakrabarti, S. *Applicability and Use Cases for IPv6 over Networks of Resource-constrained Nodes (6lo)* RFC 9453. Sept. 2023.

85. Negirla, P., Druță, R. & Silea, I. Availability Improvements through data slicing in PLC smart grid networks. *Sensors* **20,** 7256 (2020).

86. Abood, A. M., Hasan, W. K. & Khaled, H. *6LoWPAN-Technical Features and Challenges in IoT: A Review* in *2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA)* (2024), 476–481.

87. Liu, Zhiruo (CAICT) and Xie, Chongfeng (China Telecom) and Li, Cong (China Telecom) and Wetterwald, Patrick (Cisco) and Thubert, Pascal (Cisco) and Clad, Francois (Cisco) and Pouffary, Yanick (Hewlett-Packard Enterprise) and Fioccola, Giuseppe (Huawei) and Xiao, Xipeng (Huawei) and Karagiannis, Georgios (Huawei) and Liu, Shucheng (Huawei) and Metz, Eduard (KPN) and Ladid, Latif (Luxembourg University) and Cananao, Jose (PT Telecom) and Palma, Jose (PT Telecom) and Lourdez, Sébastien (Post Luxembourg) and Contreras, Luis M. (Telefonica). *IPv6 Best Practices: Benefits, Transition Challenges and the Way Forward* White Paper No. 35 (European Telecommunications Standards Institute (ETSI), 2020).

88. Raptis, T. P., Passarella, A. & Conti, M. A survey on industrial internet with ISA100 wireless. *IEEE Access* **8,** 157177–157196 (2020).

89. Rashid, A. & Pecorella, T. Is 6LoWPAN-ND necessary?(Spoiler alert: Yes). *Computer Networks* **250,** 110535 (2024).

90. Rashid, A., Pecorella, T. & Chiti, F. Toward resilient wireless sensor networks: A virtualized perspective. *Sensors* **20,** 3902 (2020).

91. Chen, Z., Amani, A. M., Yu, X. & Jalili, M. Control and optimisation of power grids using smart meter data: A review. *Sensors* **23,** 2118 (2023).

92. Hassan, A. *et al.* A survey and bibliometric analysis of different communication technologies available for smart meters. *Cleaner Engineering and Technology* **7** (2022).

93. *Protocol Specification for Interfacing to Data Communication Networks* ANSI C12.22-2012 (R2020). Reaffirmation approved 2020-06-18. National Electrical Manufacturers Association (NEMA), ASC C12 (2020).

94. *Electricity metering data exchange - The DLMS/COSEM suite - Part 5-3: DLMS/COSEM application layer* International Electrotechnical Commission (IEC), 2023.

95. Gopstein, A., Nguyen, C., O'Fallon, C., Hastings, N., Wollman, D., *et al. NIST framework and roadmap for smart grid interoperability standards, release 4.0* (Department of Commerce. National Institute of Standards and Technology ..., 2021).

96. Yadav, S. & Anand, R. Analysis of advancing paradigms of smart grid innovations, applications, challenges, future trends and strategic implementations. *Discover Applied Sciences* **7,** 1380 (2025).

97. *IEEE Standard for Interconnection and Interoperability of Distributed Energy Resources with Associated Electric Power System Interfaces* Institute of Electrical and Electronics Engineers (IEEE), 2018.

98. World Bank Group. *Survey of International Experience in Advanced Metering Infrastructure and its Implementation* tech. rep. Energy and Extractives Global Practice Group, South Asia Region (SAR) (World Bank Group, Nov. 2018).

99. Lavenu, C., Chauvenet, C., Treffiletti, P., Varesio, M. & Hueske, K. Standardization challenges, opportunities and recent evolutions for the G3-PLC technology. *Energies* **14,** 1937 (2021).

100. Department of Energy and Climate Change (DECC). *Government Response to the Consultation on Home Area Network (HAN) Solutions: Implementation of 868MHz and Alternative HAN solutions* tech. rep. 15D/520 (Department of Energy and Climate Change (DECC), 2015).

101. International Energy Agency (IEA). *Unlocking the Potential of Distributed Energy Resources: Power system opportunities and best practices* tech. rep. Published 2022-05-06 (International Energy Agency (IEA), Paris, May 2022).

102. Parag, Y., Zemah-Shamir, S., Shaviv, E. & Teschner, N. The evolving long tail at the edge of the grid: Benefits and concerns. *Joule* **8,** 1567–1578 (2024).

103. Ealey, B., Brewster, C., Forbes, W., Trueblood, C. & Huque, A. *Assessment of Interoperability Achieved through IEEE Std 1547-2018 and IEEE P1547.1: Results from EPRI Interoperability Testing and Market Research* tech. rep. 3002013473. Product Type: Technical Update (Electric Power Research Institute (EPRI), Dec. 2018).

104. *BACnet–A Data Communication Protocol for Building Automation and Control Networks* American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE), 2020.

105. Kurella, P. K. & Mikkili, S. A comprehensive review of DERMS for smart and secure energy networks: standards, protocols, and security considerations. *International Journal of Ambient Energy* **46,** 2506703 (2025).

106. *IEEE Standard for Smart Energy Profile Application Protocol for Electric Power Applications* Institute of Electrical and Electronics Engineers (IEEE), 2018.

107. Luwaca, E. & Krishnamurthy, S. Realization of a Gateway Device for Photovoltaic Application Using Open-Source Tools in a Virtualized Environment. *Computers* **14,** 524 (2025).

108. Baptista, P., Rosado, J, Silva, M., Caldeira, F. & Cardoso, F. *IEVCC-A Mesh Managed Network for Electric Vehicle Charging* in *2022 IEEE Vehicle Power and Propulsion Conference (VPPC)* (2022), 1–5.

109. Harris, R. *Interoperability of distributed energy resources: Benefits, challenges, and solutions* tech. rep. (Carbon Trust, June 2023).

110. Pohl, E. & McKenna, K. *Planning Roadmap for DER Integration in India: Industry Best Practices and Resource Guide* tech. rep. NREL/TP-6A40-89603 (National Renewable Energy Laboratory (NREL), Oct. 2024).

111. Sethuvenkatraman, S. *et al. Standardisation and interoperability challenges in achieving demand side flexibility* tech. rep. EP2025-0887 (Commonwealth Scientific and Industrial Research Organisation (CSIRO), Dec. 2024).

112. González, I., Calderón, A. J. & Portalo, J. M. Innovative multi-layered architecture for heterogeneous automation and monitoring systems: Application case of a photovoltaic smart microgrid. *Sustainability* **13,** 2234 (2021).

113. Hosseinzadeh, N., Al Maashri, A., Tarhuni, N., Elhaffar, A. & Al-Hinai, A. A real-time monitoring platform for distributed energy resources in a microgrid—pilot study in oman. *Electronics* **10,** 1803 (2021).

114. U.S. DEPARTMENT OF ENERGY, OFFICE OF ELECTRICITY. *Guidelines for Next-Generation Grid Communications Architecture* tech. rep. (U.S. Department of Energy (DOE), 2024).

115. Rounaq, S. & Iqbal, M. Vision, challenges and future perspectives of low constrained devices iot operating systems: A systematic mapping review. *European Journal of Engineering and Technology Research* **5,** 107–115 (2020).

116. Oikonomou, G. *et al.* The Contiki-NG open source operating system for next generation IoT devices. *SoftwareX* **18,** 101089 (2022).

117. Donta, P. K., Srirama, S. N., Amgoth, T. & Annavarapu, C. S. R. Survey on recent advances in IoT application layer protocols and machine learning scope for research directions. *Digital Communications and Networks* **8,** 727–744 (2022).

118. Hmissi, F. & Ouni, S. A survey on application layer protocols for iot networks. *arXiv preprint arXiv:2405.15901* (2024).

119. Petrescu, I., Niculae, E., Vulturescu, V., Dimitrescu, A. & Ungureanu, L. M. Transport and Application Layer Protocols for IoT: Comprehensive Review. *Technologies* **13.** ISSN: 2227-7080 (2025).

120. Alnajjar, I. A. A Comprehensive Survey on Objective Functions in RPL Routing with Various Networking and Application Scenarios. *EAI Endorsed Transactions on Internet of Things* **11** (2024).

121. Darabkh, K. A., Al-Akhras, M., Zomot, J. N. & Atiquzzaman, M. RPL routing protocol over IoT: A comprehensive survey, recent advances, insights, bibliometric analysis, recommendations, and future directions. *Journal of Network and Computer Applications* **207,** 103476 (2022).

122. Winter, T. *et al.* RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. *RFC 6550* (2012).

123. Banaszek, M., Schuß, M., Boano, C. A. & Iwanicki, K. *RPL at Scale: Experiences from a Performance Evaluation on up to 700 IEEE 802.15. 4 Devices* in *NOMS 2024-2024 IEEE Network Operations and Management Symposium* (2024), 1–6.

124. Mahyoub, M. & Mahmoud, A. *Assessing Contiki-NG's Reliability for RPL-based Trickle Algorithm in IoT Networks* in *2025 International Wireless Communications and Mobile Computing (IWCMC)* (2025), 73–78.

125. Echoukairi, H., Ouacha, A., Oubaha, J. & El Ghmary, M. A New Objective Function for RPL Based on Combined Metrics in Mobile IoT. *J. Commun.* **18,** 301–309 (2023).

126. Stocker, A. & de Meer, H. *A Tutorial on Resilience in Smart Grids* in *2022 12th International Workshop on Resilient Networks Design and Modeling (RNDM)* (2022), 1–14.

127. Younesi, A. *et al.* Trends in modern power systems resilience: State-of-the-art review. *Renewable and Sustainable Energy Reviews* **162,** 112397. ISSN: 1364-0321 (2022).

128. Duenas Santos, C. L. *et al.* Q-RPL: Q-learning-based routing protocol for advanced metering infrastructure in smart grids. *Sensors* **24,** 4818 (2024).

129. Boonkajay, A., Tan, P. H., Goh, L. K., Ahmed, S. N. A. & Sun, S. *Enhancing Wi-SUN AMI Network Resilience by using Emergency Gateway with Optimal Placement* in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)* (2021), 1–5.

130. Krivohlava, Z., Chren, S. & Rossi, B. Failure and fault classification for smart grids. *Energy Informatics* **5,** 33 (2022).

131. Liu, M. *et al.* Enhancing cyber-resiliency of der-based smart grid: A survey. *IEEE Transactions on Smart Grid* **15,** 4998–5030 (2024).

132. Zabihi, A., Parhamfar, M. & Khodadadi, M. Strengthening Resilience: A Brief Review of Cybersecurity Challenges in IoT-Driven Smart Grids. *Journal of Modern Technology,* 106–120 (2024).

133. Sung, T.-W., Li, W., Lee, C.-Y., Chen, Y. & Fang, Q. Data Aggregation Point Placement and Subnetwork Optimization for Smart Grids. *Computers, Materials & Continua* **83** (2025).

134. Nait Belaid, Y. *et al.* Resilience Quantification of Smart Distribution Networks—A Bird's Eye View Perspective. *Energies* **14.** ISSN: 1996-1073 (2021).

135. Sterbenz, J. P. *et al.* Evaluation of network resilience, survivability, and disruption tolerance: analysis, topology generation, simulation, and experimentation. *Telecommunication systems* **52,** 705–736 (2013).

136. Abdel Hakeem, S. A., Hady, A. A. & Kim, H. RPL routing protocol performance in smart grid applications based wireless sensors: Experimental and simulated analysis. *Electronics* **8,** 186 (2019).

137. Ghanbari, Z., Navimipour, N. J., Hosseinzadeh, M., Shakeri, H. & Darwesh, A. The applications of the routing protocol for low-power and lossy networks (RPL) on the internet of mobile things. *International Journal of Communication Systems* **35,** e5253 (2022).

138. Ashraf, U., Ahmed, A., Al-Naeem, M. & Masood, U. Reliable and QoS aware routing metrics for wireless Neighborhood Area Networking in smart grids. *Computer Networks* **192,** 108051 (2021).

139. Alameri, I., Komarkova, J. & Al-Hadhrami, T. Fuzzy-based optimization of AODV routing for efficient route in wireless mesh networks. *PeerJ Computer Science* **9,** e1508 (2023).

140. Verma, A. & Ranga, V. Security of RPL based 6LoWPAN Networks in the Internet of Things: A Review. *IEEE Sensors Journal* **20,** 5666–5690 (2020).

141. Vedavalli, P. & Ch, D. A deep learning based data recovery approach for missing and erroneous data of IoT nodes. *Sensors* **23,** 170 (2022).

142. Cheng, Y., Foggo, B., Yamashita, K. & Yu, N. Missing value replacement for pmu data via deep learning model with magnitude trend decoupling. *IEEE Access* **11,** 27450–27461 (2023).

143. Lai, T. T., Tran, T. P., Cho, J. & Yoo, M. Noise-tolerant data reconstruction based on convolutional autoencoder for wireless sensor network. *Applied Sciences* **13,** 10090 (2023).

144. Berger, C. *et al.* A survey on resilience in the iot: Taxonomy, classification, and discussion of resilience mechanisms. *ACM Computing Surveys (CSUR)* **54,** 1–39 (2021).

145. Onural, E. B., Bazyari, M., Jamadi, B. & Fattahi, J. *Real-Time AI-Driven EV Charger Load Management for Residential Electrification* in *2025 IEEE Power & Energy Society General Meeting (PESGM)* (2025), 1–5.

146. Tightiz, L. & Yang, H. A comprehensive review on IoT protocols' features in smart grid communication. *Energies* **13,** 2762 (2020).

147. Farag, H., Österberg, P. & Gidlund, M. Congestion control and traffic differentiation for heterogeneous 6tisch networks in IIoT. *Sensors* **20,** 3508 (2020).

148. Kubaszek, M., Macheta, J., Krzak, Ł. & Worek, C. The analysis of energy consumption in 6TiSCH network nodes working in sub-GHz band. *International Journal of Electronics and Telecommunications* **66,** 201–210 (2020).

149. Demertzis, K. *et al.* Communication network standards for smart grid infrastructures. *Network* **1,** 132–145 (2021).

150. EU-SysFlex Consortium. *Proposal for data exchange standards and protocols* tech. rep. Deliverable D5.5 (H2020 EU-SysFlex, Apr. 2021).

151. Novo, O. & Francesco, M. D. Semantic interoperability in the IoT: Extending the web of things architecture. *ACM Transactions on Internet of Things* **1,** 1–25 (2020).

152. Hazra, A., Adhikari, M., Amgoth, T. & Srirama, S. N. A comprehensive survey on interoperability for IIoT: Taxonomy, standards, and future directions. *ACM Computing Surveys (CSUR)* **55,** 1–35 (2021).

153. Saavedra, E., Santamaria, A., del Campo, G. & Gomez, I. Leveraging iot harmonization: An efficacious nb-iot relay for integrating 6lowpan devices into legacy ipv4 networks. *Applied Sciences* **14,** 3411 (2024).

154. Chicco, G. Data consistency for data-driven smart energy assessment. *Frontiers in Big Data* **4,** 683682 (2021).

155. Almutairi, H. & Zhang, N. A survey on routing solutions for low-power and lossy networks: Toward a reliable path-finding approach. *Network* **4,** 1–32 (2024).

156. Marais, J. M., Abu-Mahfouz, A. M. & Hancke, G. P. A Survey on the Viability of Confirmed Traffic in a LoRaWAN. *Ieee Access* **8,** 9296–9311 (2020).

157. Adhikari, D. *et al.* A comprehensive survey on imputation of missing data in internet of things. *ACM Computing Surveys* **55,** 1–38 (2022).

158. Xuegang, L., Junrui, L. & Juan, W. Missing Data Reconstruction Based on Spectral k-Support Norm Minimization for NB-IoT Data. *Mathematical Problems in Engineering* **2021,** 1336900 (2021).

159. Emmanuel, T. *et al.* A survey on missing data in machine learning. *Journal of Big data* **8,** 140 (2021).

160. Ribeiro, S. M. & Castro, C. Missing data in time series: A review of imputation methods and case study. *Learn. Nonlinear Model* **20,** 31–46 (2022).

161. Aslam, S. *et al.* A survey on deep learning methods for power load and renewable energy forecasting in smart microgrids. *Renewable and Sustainable Energy Reviews* **144,** 110992 (2021).

162. Habbak, H., Mahmoud, M., Metwally, K., Fouda, M. M. & Ibrahem, M. I. Load forecasting techniques and their applications in smart grids. *Energies* **16,** 1480 (2023).

163. Iversen, M., Khan, M., Miraki, A. & Arghandeh, R. Advancements in super-resolution methods for smart meter data. *Frontiers in Energy Research* **11,** 1288683 (2023).

164. Schreiber, J. F., Sausen, A., De Campos, M., Sausen, P. S. & Ferreira Filho, M. T. D. S. Data imputation techniques applied to the smart grids environment. *IEEE Access* **11,** 31931–31940 (2023).

165. Fattahi, J. Data-Driven Fault Recovery With Software-Defined Smart Transmission Grids. *IEEE Access* **PP,** 1–1 (Jan. 2024).

166. Sartipi, A., Fernandez, J. D., Potenciano Menci, S. & Magitteri, A. *Bridging Smart Meter Gaps: A Benchmark of Statistical, Machine Learning and Time Series Foundation Models for Data Imputation* in *2025 IEEE Kiel PowerTech* (2025), 1–7.

167. Syed, D. *et al.* Smart grid big data analytics: Survey of technologies, techniques, and applications. *IEEE Access* **9,** 59564–59585 (2020).

168. Li, B. *et al.* A Large-Scale Residential Load Dataset in a Southern Province of China. *Scientific Data* **12,** 1–14 (2025).

169. Enrich, J., Li, R., Mizrahi, A. & Reguant, M. Measuring the impact of time-of-use pricing on electricity consumption: Evidence from Spain. *Journal of Environmental Economics and Management* **123,** 102901 (2024).

170. Yang, X., Sun, Y., Yuan, X. & Chen, X. Frequency-aware generative models for multivariate time series imputation. *Advances in Neural Information Processing Systems* **37,** 52595–52623 (2024).

171. Wang, Y. *et al.* Attention-based message passing and dynamic graph convolution for spatiotemporal data imputation. *Scientific Reports* **13,** 6887 (2023).

172. Das, L., Munikoti, S., Natarajan, B. & Srinivasan, B. Measuring smart grid resilience: Methods, challenges and opportunities. *Renewable and Sustainable Energy Reviews* **130,** 109918 (2020).

173. Xu, L., Guo, Q., Sheng, Y., Muyeen, S. & Sun, H. On the resilience of modern power systems: A comprehensive review from the cyber-physical perspective. *Renewable and Sustainable Energy Reviews* **152,** 111642 (2021).

174. Pazhoohesh, M., Allahham, A., Das, R. & Walker, S. Investigating the impact of missing data imputation techniques on battery energy management system. *IET Smart Grid* **4,** 162–175 (2021).

175. Himeur, Y., Alsalemi, A., Al-Kababji, A., Bensaali, F. & Amira, A. Data fusion strategies for energy efficiency in buildings: Overview, challenges and novel orientations. *Information Fusion* **64,** 99–120 (2020).

176. Azevedo, J. A. & Mendonça, F. A critical review of the propagation models employed in LoRa systems. *Sensors* **24,** 3877 (2024).

177. Molokomme, D. N., Chabalala, C. S. & Bokoro, P. N. A review of cognitive radio smart grid communication infrastructure systems. *Energies* **13,** 3245 (2020).

178. Fadel, E. *et al.* A survey on wireless sensor networks for smart grid. *Computer Communications* **71,** 22–33 (2015).

179. Alsukayti, I. S. Quality of service support in RPL networks: Standing state and future prospects. *Journal of Computer Science and Technology* **37,** 344–368 (2022).

180. Zainab, A. *et al.* Big data management in smart grids: Technologies and challenges. *IEEE Access* **9,** 73046–73059 (2021).

181. Yadav, P. K., Biswal, M. & Vemuganti, H. in *Smart Metering: Infrastructure, Methodologies, Applications, and Challenges* 221–256 (Elsevier, 2024).

182. Liu, Y. *et al.* Time Synchronization Techniques in the Modern Smart Grid: A Comprehensive Survey. *Energies* **18,** 1163 (2025).

183. Bhadra, D. R., Joshi, C. A., Soni, P. R., Vyas, N. P. & Jhaveri, R. H. *Packet loss probability in wireless networks: A survey* in *2015 International Conference on Communications and Signal Processing (ICCSP)* (2015), 1348–1354.

184. Widhalm, D. *Sensor node fault detection in wireless sensor networks: an immune-inspired approach* PhD thesis (Technische Universität Wien, 2022).

185. Li, Y. & Tu, W. *Traffic modelling for IoT networks: A survey* in *Proceedings of the 10th International Conference on Information Communication and Management* (2020), 4–9.

186. Wang, W., Xu, Y. & Khanna, M. A survey on the communication architectures in smart grid. *Computer networks* **55,** 3604–3629 (2011).

187. Radhoush, S., Bahramipanah, M., Nehrir, H. & Shahooei, Z. A review on state estimation techniques in active distribution networks: existing practices and their challenges. *Sustainability* **14,** 2520 (2022).

188. Alwateer, M., Atlam, E.-S., Abd El-Raouf, M. M., Ghoneim, O. A. & Gad, I. Missing data imputation: A comprehensive review. *Journal of Computer and Communications* **12,** 53–75 (2024).

189. Zhou, Y., Aryal, S. & Bouadjenek, M. R. *Review for Handling Missing Data with special missing mechanism* 2024. arXiv: 2404.04905.

190. Jaisankar, R., Sathishkumar, K. & Ramamoorthy, T. A Review of Statistical Methods for Addressing Missing Data. *Advances and Applications in Statistics* **92,** 1201–1218 (July 2025).

191. Fang, C. & Wang, C. Time series data imputation: A survey on deep learning approaches. *arXiv preprint arXiv:2011.11347* (2020).

192. Pereira, R. C., Santos, M. S., Rodrigues, P. P. & Abreu, P. H. Reviewing autoencoders for missing data imputation: Technical trends, applications and outcomes. *Journal of Artificial Intelligence Research* **69,** 1255–1285 (2020).

193. Sadegh, R., Mohameden, A., Salihi, M. L. & Nanne, M. F. A survey of missing data imputation techniques: statistical methods, machine learning models, and GAN-based approaches. *IAES International Journal of Artificial Intelligence (IJ-AI)* **14,** 2876–2888 (Aug. 2025).

194. Jäger, S., Allhorn, A. & Bießmann, F. A benchmark for data imputation methods. *Frontiers in big Data* **4,** 693674 (2021).

195. Alghamdi, T. A. & Javaid, N. A survey of preprocessing methods used for analysis of big data originated from smart grids. *Ieee Access* **10,** 29149–29171 (2022).

196. Dahunsi, F. M., Olawumi, A. E., Ale, D. T. & Sarumi, O. A. A systematic review of data preprocessing methods and unsupervised mining methods used in profiling smart meter data. *AIMS Electronics and Electrical Engineering* **5,** 284–314 (2021).

197. Goudarzi, A., Ghayoor, F., Waseem, M., Fahad, S. & Traore, I. A survey on IoT-enabled smart grids: emerging, applications, challenges, and outlook. *Energies* **15,** 6984 (2022).

198. Wu, J., Koirala, A. & Van Hertem, D. *Review of statistics based coping mechanisms for Smart Meter Missing Data in Distribution Systems* in *2022 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)* (2022), 1–6.

199. Alonso, S. *et al.* Gap imputation in related multivariate time series through recurrent neural network-based denoising autoencoder. *Integrated Computer-Aided Engineering* **31,** 157–172. eprint: https://journals.sagepub.com/doi/pdf/10.3233/ICA-230728 (2024).

200. Weber, M. *et al.* Data-Driven Copy-Paste Imputation for Energy Time Series. *IEEE Transactions on Smart Grid* **12,** 5409–5419 (2021).

201. Duarte, O., Duarte, J. E. & Rosero-Garcia, J. Data Imputation in Electricity Consumption Profiles through Shape Modeling with Autoencoders. *Mathematics* **12.** ISSN: 2227-7390 (2024).

202. Ryu, S., Kim, M. & Kim, H. Denoising autoencoder-based missing value imputation for smart meters. *IEEE Access* **8,** 40656–40666 (2020).

203. Buwaneswaran, M. *Missing Data Imputation for Smart Meters: Conditional Denoising Diffusion Model & Temporally Chained Equations* MA thesis (The University of Western Ontario (Canada), 2024).

204. Nair, V., Litjens, R. & Zhang, H. Optimisation of NB-IoT deployment for smart energy distribution networks. *Eurasip journal on wireless communications and networking* **2019,** 186 (2019).

205. Al-Sammak, K. A. *et al.* Optimizing IoT Energy Efficiency: Real-Time Adaptive Algorithms for Smart Meters with LoRaWAN and NB-IoT. *Energies* **18,** 987 (2025).

206. Pei, J., Wang, J., Shi, D. & Wang, P. Detection and imputation-based two-stage denoising diffusion power system measurement recovery under cyber-physical uncertainties. *IEEE Transactions on Smart Grid* **15,** 5965–5980 (2024).

207. Cini, A., Marisca, I. & Alippi, C. Multivariate time series imputation by graph neural networks. *arXiv preprint arXiv:2108.00298* (2021).

208. Ferrer-Cid, P., Barcelo-Ordinas, J. M. & Garcia-Vidal, J. A review of graph-powered data quality applications for IoT monitoring sensor networks. *Journal of Network and Computer Applications,* 104116 (2025).

209. Österlind, F. *A sensor network simulator for the Contiki OS* (Swedish Institute of Computer Science, 2006).

210. Mehmood, T. Cooja network simulator: Exploring the infinite possible ways to compute the performance metrics of iot based smart devices to understand the working of iot based compression & routing protocols. *arXiv preprint arXiv:1712.08303* (2017).

211. Zhang, T. & Li, X. *Evaluating and Analyzing the Performance of RPL in Contiki* in *Proceedings of the first international workshop on Mobile sensing, computing and communication* (2014), 19–24.

212. Ali, H. A performance evaluation of rpl in contiki: A cooja simulation based study. *School of Computing, Blekinge Institute of Technology* (2012).

213. Lamaazi, H., Benamar, N. & Jara, A. J. RPL-based networks in static and mobile environment: A performance assessment analysis. *Journal of King Saud University-Computer and Information Sciences* **30,** 320–333 (2018).

214. Ajayy, V. & Ranga, V. Performance analysis of RPL protocol in different nodes positioning using Contiki Cooja. *International Journal of Information Technology* **16,** 3683–3689 (2024).

215. Elyengui, S., Bouhouchi, R. & Ezzedine, T. LOADng routing protocol evaluation for bidirectional data flow in AMI mesh networks. *arXiv preprint arXiv:1506.06357* (2015).

216. Shakya, N. M., Mani, M. & Crespi, N. *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), 1–6.

217. Said, A. M., Yahyaoui, A., Yaakoubi, F. & Abdellatif, T. *Machine learning based rank attack detection for smart hospital infrastructure* in *International Conference on Smart Homes and Health Telematics* (2020), 28–40.

218. Abosata, N., Al-Rubaye, S. & Inalhan, G. Lightweight payload encryption-based authentication scheme for advanced metering infrastructure sensor networks. *Sensors* **22,** 534 (2022).

219. Dua, S., Kadyan, V., Bhogal, R. & Dua, M. *Detection and Mitigation of NDP Attacks for AMI Devices Using Cooja Simulator* in *International Conference On Artificial Intelligence, Computing, IOT and Data Analytics* (2023), 85–99.

220. Onural, E. B., Jamadi, B. & Fattahi, J. Rethinking Smart Grid Mesh Network with Dual Border Routers for Asset Integration. *2025 IEEE Electrical Power and Energy Conference (EPEC),* 237–241 (2025).