






Article

Performance Evaluation of UDP-Based Data Transmission with Acknowledgment for Various Network Topologies in IoT Environments

Bereket Endale Bekele ¹, Krzysztof Tokarz ¹, Nebiyat Yilikal Gebeyehu ², Bolesław Pochopień ¹
and Dariusz Mrozek ^{2,*}

¹ Department of Graphics, Computer Vision and Digital Systems, Silesian University of Technology, 44-100 Gliwice, Poland; bereketendalebekele@gmail.com (B.E.B.); krzysztof.tokarz@polsl.pl (K.T.)

² Department of Applied Informatics, Silesian University of Technology, 44-100 Gliwice, Poland; nyilikal@gmail.com

* Correspondence: dariusz.mrozek@polsl.pl; Tel.: +48-32-237-13-39

Abstract: The rapid expansion of Internet-of-Things (IoT) applications necessitates a thorough understanding of network configurations to address unique challenges across various use cases. This paper presents an in-depth analysis of three IoT network topologies: linear chain, structured tree, and dynamic transition networks, each designed to meet the specific requirements of industrial automation, home automation, and environmental monitoring. Key performance metrics, including round-trip time (RTT), server processing time (SPT), and power consumption, are evaluated through both simulation and hardware experiments. Additionally, this study introduces an enhanced UDP protocol featuring an acknowledgment mechanism and a power consumption evaluation, aiming to improve data transmission reliability over the standard UDP protocol. Packet loss is specifically measured in hardware experiments to compare the performance of standard and enhanced UDP protocols. The findings show that the enhanced UDP significantly reduces packet loss compared to the standard UDP, enhancing data delivery reliability across dynamic and structured networks, though it comes at the cost of slightly higher power consumption due to additional processing. For network topology performance, the linear chain topology provides stable processing but higher RTT, making it suitable for applications such as tunnel monitoring; the structured tree topology offers low energy consumption and fast communication, ideal for home automation; and the dynamic transition network, suited for industrial Automated Guided Vehicles (AGVs), encounters challenges with adaptive routing. These insights guide the optimization of communication protocols and network configurations for more efficient and reliable IoT deployments.

Keywords: IoT network topologies; UDP server–client communication; RPL routing protocol; 6LoWPAN; performance analysis



Citation: Bekele, B.E.; Tokarz, K.; Gebeyehu, N.Y.; Pochopień, B.; Mrozek, D. Performance Evaluation of UDP-Based Data Transmission with Acknowledgment for Various Network Topologies in IoT Environments. *Electronics* **2024**, *13*, 3697. <https://doi.org/10.3390/electronics13183697>

Academic Editors: Minghui Li, Sye Loong Keoh and Djuradj Budimir

Received: 22 July 2024

Revised: 8 September 2024

Accepted: 14 September 2024

Published: 18 September 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The rise of the Internet of Things (IoT) has dramatically altered connectivity frameworks, redefining the way devices communicate and exchange information within an increasingly digital ecosystem [1]. The rapid proliferation of IoT technologies across domains such as smart homes, industrial automation, and environmental surveillance underscores the need for robust and efficient data transmission strategies [2,3]. A central challenge in the IoT field is achieving reliable, low-latency, and real-time data exchange across heterogeneous and frequently changing environments [4]. As IoT continues to extend its reach into diverse sectors, the demand for optimized communication protocols that cater to a variety of dynamic requirements has grown significantly [5]. Conventional protocols, designed for static network settings, often face limitations in addressing the distinctive challenges posed by dynamic IoT networks [5]. Recent research increasingly

highlights the inadequacies of these traditional protocols in managing high variability and fluid topologies within IoT applications [6].

The complexity inherent in IoT network topologies calls for a comprehensive understanding of data propagation methods and the development of flexible communication protocols that can maintain performance under varied conditions [7]. Among these, the User Datagram Protocol (UDP) is frequently chosen for IoT deployments due to its simplicity and minimal overhead. However, UDP's inherent lack of reliability, particularly in dynamic and resource-constrained scenarios, presents substantial hurdles. Studies have shown that although UDP is pivotal for facilitating streamlined communication in IoT contexts, its absence of built-in error correction mechanisms often results in data loss, which is detrimental to critical IoT use cases [8,9].

To mitigate these issues, this study proposes an enhanced version of UDP that integrates an acknowledgment mechanism to ensure data accuracy and a power consumption evaluation component utilizing the *Energest* module in Contiki OS. This improvement is aimed at delivering more reliable and energy-conscious data transmission, especially in dynamic networks where both data integrity and energy efficiency are of paramount concern. The advanced UDP protocol demonstrates considerable enhancements over the standard UDP by reducing data loss and providing more consistent data delivery across various IoT configurations, such as linear chain, structured tree, and dynamic transition networks.

By evaluating crucial performance indicators such as round-trip time (RTT), server processing duration (SPT), energy consumption, and packet loss, this research offers a detailed assessment of both standard and enhanced UDP protocols across different IoT network architectures. The results illustrate that while the standard UDP protocol's low overhead makes it attractive, its dependability diminishes in dynamic settings, limiting its suitability for applications where data integrity is critical. In contrast, the enhanced UDP protocol successfully addresses these limitations by minimizing packet loss and maintaining more stable performance across diverse topologies, albeit with a marginal increase in energy consumption due to the added acknowledgment feature.

This work contributes a novel perspective by focusing on both enhancing the User Datagram Protocol (UDP) and incorporating a power consumption assessment mechanism—an aspect that has been relatively underexplored in IoT data transmission research. The enriched UDP protocol's acknowledgment feature, combined with the *Energest* module in Contiki OS, facilitates a thorough analysis of power consumption based on the devices employed, such as Z1 motes and CC2650 Launchpad. This holistic evaluation, grounded in both simulation and hardware experiments, positions this study as a meaningful advancement over existing research.

The study includes both simulation and hardware experimentation to ensure a comprehensive analysis. The Contiki OS and Cooja simulator are utilized for simulations, while CC2650 Launchpad devices are employed for hardware testing, ensuring a detailed understanding of how UDP-based communication can be optimized in practical IoT scenarios. The findings provide valuable guidance for improving IoT communication methodologies by demonstrating the advantages of the enhanced UDP protocol in boosting reliability and energy efficiency, paving the way for future progress in adaptive and intelligent network systems.

The structure of this paper is organized as follows: Section 2 reviews the current literature on IoT data transmission and communication protocols, highlighting recent advancements and identifying research gaps that serve as a foundation for this work. Section 3 outlines the research methodology, including the experimental framework incorporating both simulation and hardware elements, the justification for selecting UDP, and the criteria used for the analysis such as the RTT, server response time (SRT), and energy consumption. This section also introduces the algorithm developed in this research and describes the scenarios analyzed, such as linear chain, structured tree, and dynamic transition networks. Section 4 presents the experimental outcomes, examining UDP's performance across different scenarios with specific metrics like RTT, server response time,

and energy use, assessed through CPU and low-power mode (LPM) metrics. The practical implications and new insights derived from the study are also discussed. Section 5 merges the discussion and conclusion, providing a thorough analysis of the results in comparison with related studies and emphasizing the broader implications for designing energy-efficient IoT systems. This section also summarizes the key findings, considers their potential impact, and outlines avenues for future research, reinforcing the study's contributions to advancing IoT data transmission and communication protocols for more effective, reliable, and scalable networks.

2. Related Works

Wireless communication is a foundational element for modern IoT networks, enabling device interconnectivity and data transmission [10]. The evolution of wireless technology, from 1G to 5G, has significantly impacted the IoT, facilitating new applications with enhanced connectivity, data rates, and reduced latency [11]. While each generation brought advancements, IoT networks present unique challenges for data transmission, particularly concerning protocol efficiency, reliability, and energy consumption.

Protocols like the User Datagram Protocol (UDP) are widely adopted in IoT environments due to their simplicity and low-latency characteristics, making them suitable for time-sensitive applications [12]. However, UDP lacks built-in error recovery, posing reliability challenges, especially in dynamic IoT environments [10,13]. To address these issues, enhancements such as adding acknowledgment mechanisms have been proposed to improve data integrity without significantly increasing overhead [14]. This study evaluates both standard and enhanced versions of UDP across different IoT network topologies and hardware setups to optimize protocol efficiency and reliability.

Optimizing communication protocols is crucial in the IoT to manage power consumption, minimize packet loss, and adapt to dynamic environments [15]. Techniques like adaptive duty cycling and the use of energy-efficient Medium Access Control (MAC) protocols help reduce energy consumption, particularly in low-power networks [16]. Recent advances in edge computing also provide opportunities for optimizing protocol performance by reducing latency and offloading computation closer to the source of data generation [17]. This research focuses on evaluating the performance of an enhanced UDP, considering energy efficiency as a critical factor for real-world IoT deployments.

Recent studies have highlighted the increasing popularity of UDP in IoT environments due to its low overhead and suitability for resource-constrained devices [18]. Enhancements such as hybrid UDP-TCP approaches [19] and integrated acknowledgment mechanisms [20] have been proposed to address UDP's reliability issues while maintaining its low latency. Furthermore, research on power-efficient communication strategies [21] and UDP optimizations for sensor networks [22] shows the protocol's adaptability to different IoT scenarios. Our work extends these studies by introducing specific enhancements to UDP, including acknowledgment mechanisms and power consumption metrics, and evaluates their performance across various IoT network topologies. By integrating these features, our research provides new insights into UDP's application in IoT environments and addresses gaps identified in the existing literature.

This study evaluates the performance of a UDP-based communication protocol with an acknowledgment mechanism for specific IoT applications through the comprehensive analysis of different network topologies, including linear chain networks, structured tree networks, and dynamic transition networks. The evaluation of key performance metrics, such as round-trip time (RTT), server processing time (SPT), and power consumption, illustrates the behavior of UDP server-client communication built on the RPL routing protocol and 6LoWPAN-based stack. Furthermore, the study demonstrates the efficiency of hierarchical network designs and dynamic routing protocols in various IoT scenarios, contributing to the development of energy-efficient communication strategies for resource-constrained devices.

3. Research Methodology

Within our work, we wanted to evaluate and optimize the performance of various network topologies for IoT applications, specifically focusing on linear chain networks, structured tree networks, and dynamic transition networks. The aim was to comprehensively analyze the impact of these topologies on key performance metrics such as round-trip time (RTT), server processing time (SPT), and power consumption, utilizing UDP server–client communication, the RPL routing protocol, and 6LoWPAN to identify the most efficient and reliable configurations for different real-world IoT scenarios.

3.1. Methodological Approach and Tools

This section provides a comprehensive rationale for the communication methodology, protocols, and tools chosen for the study. Our approach is grounded in UDP server–client communication and RPL routing, with enhancements specifically designed to address the reliability and efficiency of data exchange in IoT networks.

3.1.1. Communication Methodology and Protocol Enhancements

The UDP protocol was modified by incorporating an acknowledgment mechanism. Traditionally, UDP allows for unidirectional transmission where the client sends packets to the server without any form of acknowledgment from the server. Our modification introduces a bidirectional communication feature, enabling the server to send acknowledgments back to the client upon receiving a packet. This enhancement was implemented to address the research problem of establishing reliable communication in IoT networks, where acknowledgment is crucial for confirming data receipt and improving overall communication reliability. Figure 1 illustrates each client independently sending packets directly to the server and receiving individual acknowledgments.

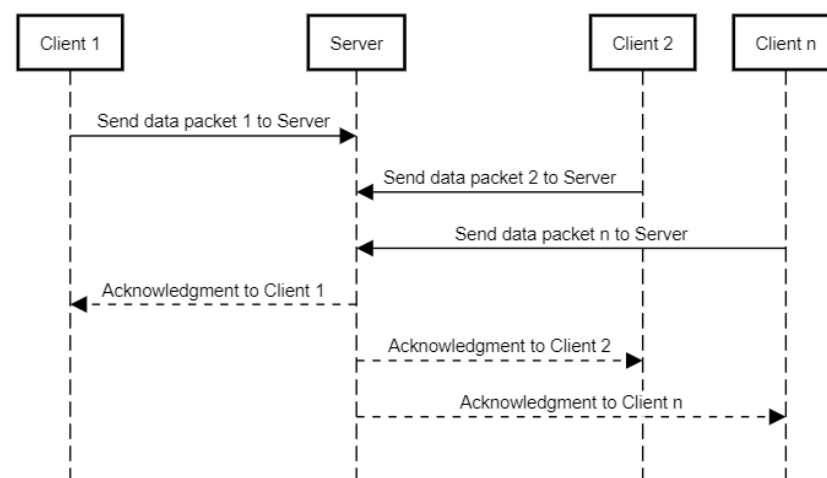


Figure 1. Direct communication of each client with the server sending acknowledgments.

Figure 2 shows the sequential packet transmission from clients to the server through the relays of other clients and the acknowledgment sent back from the server to each client.

The rationale behind choosing UDP lies in its simplicity and low-overhead nature, which makes it suitable for resource-constrained IoT devices that require efficient data transmission without the extensive handshaking and error-checking overhead of TCP. UDP's connectionless protocol offers benefits in scenarios where real-time data transfer is critical, such as in smart homes and industrial automation systems [23]. RPL, or the Routing Protocol for Low-Power and Lossy Networks, was selected for its efficiency in handling the dynamic topology of IoT networks. RPL supports the creation of a robust and flexible network that can adapt to changes in the environment and node connectivity, which is essential for maintaining communication reliability in IoT applications [24].

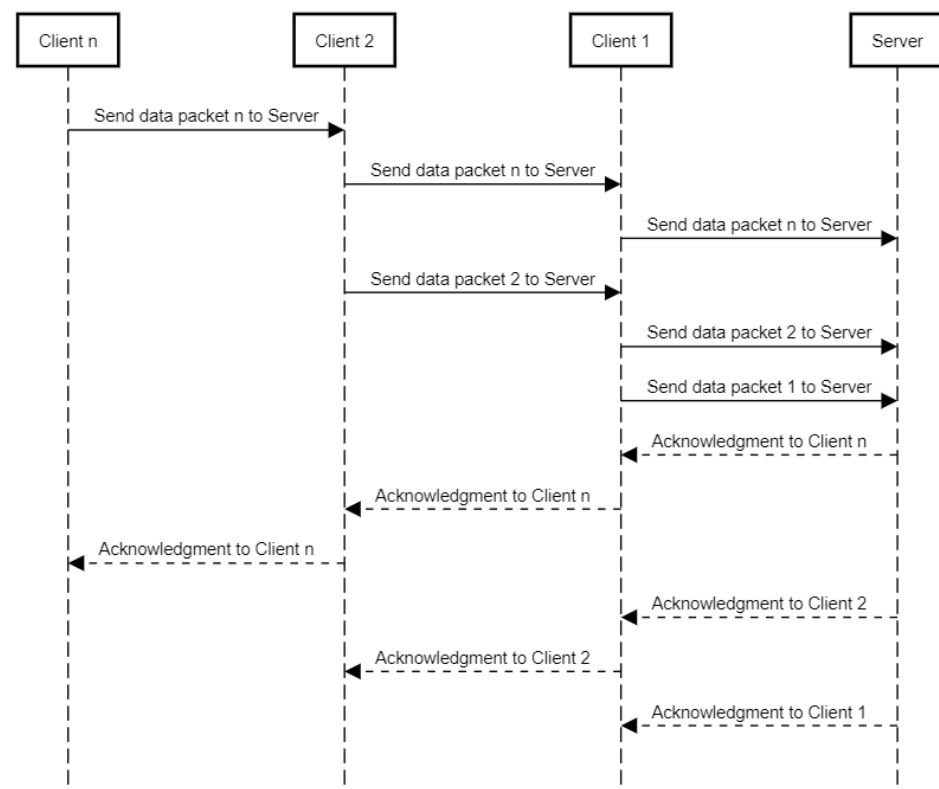


Figure 2. Serial (or indirect) communication of clients with the server sending acknowledgments.

3.1.2. Methodology for Network Topologies and Tools

The communication methodology explored three distinct scenarios to understand how UDP server–client communication functions in various network environments. Here is an overview of each scenario:

Linear chain network tunnel scenario (Figure 3): nodes are positioned in a straight line, each communicating only with its adjacent neighbor.

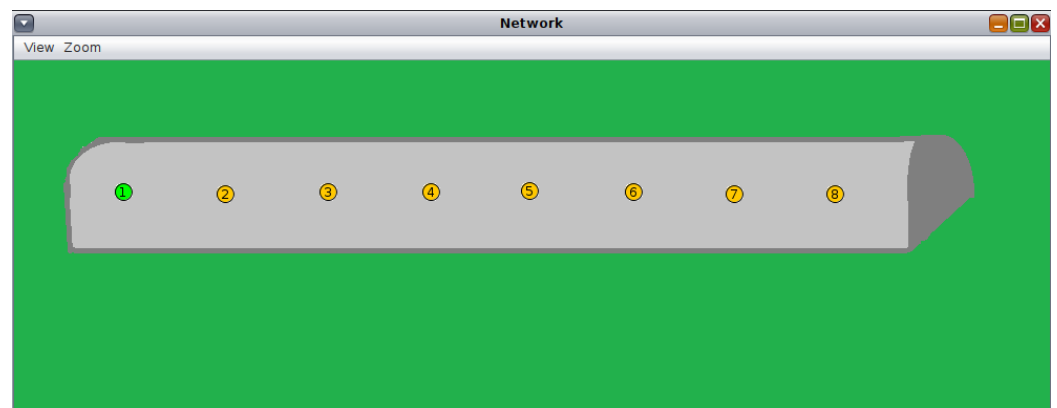


Figure 3. Linear chain network with nodes arranged in a series connection, starting from the server (note ID 1) and followed by the client nodes.

Structured tree network (home automation) scenario (Figure 4): nodes are arranged in a hierarchical tree structure, where the root node communicates with several child nodes, which may themselves have child nodes.

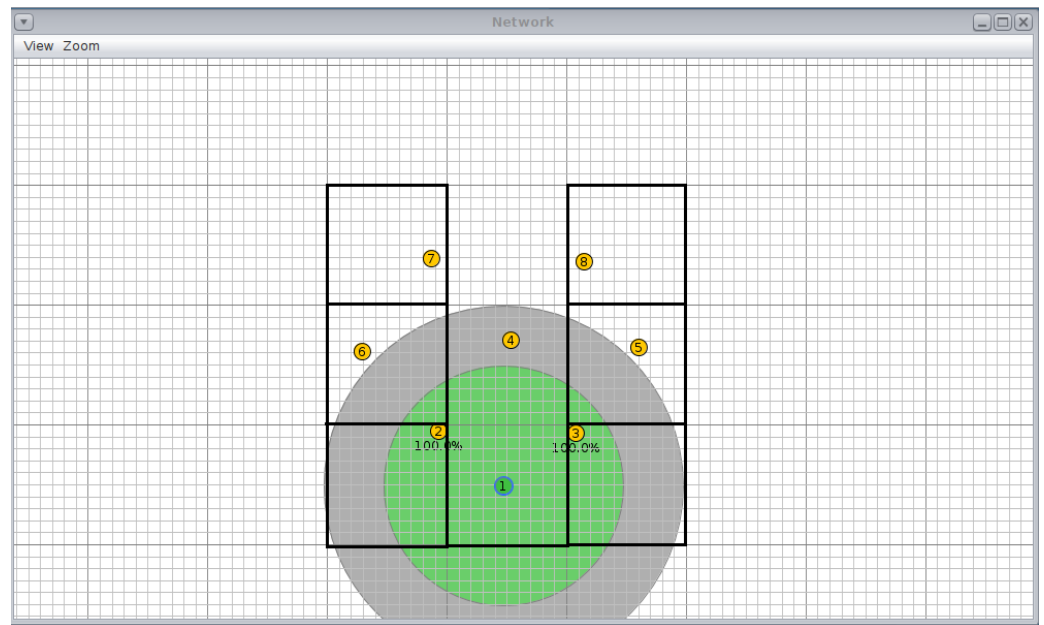


Figure 4. Node placement in a structured tree network with mote ID 1 as the server and other motes arranged hierarchically as clients.

Dynamic transition network (e.g., AGVs) scenario (Figures 5–7): nodes have dynamic and adaptive routing capabilities, allowing them to adjust their routing paths based on current network conditions to simulate a realistic and variable network environment.

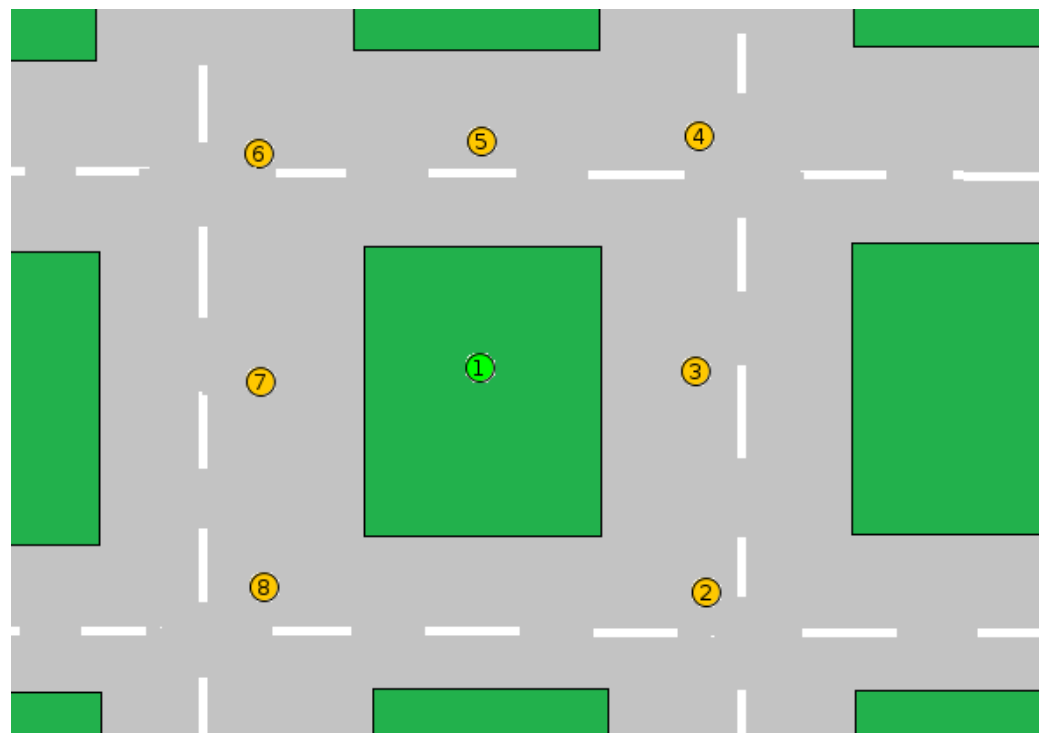


Figure 5. Client motes establishing a connection with the server (mote ID 1) at a stationary point.

In every task, each client mote started a movement and maintained reliable communication with the next client (Figure 6).

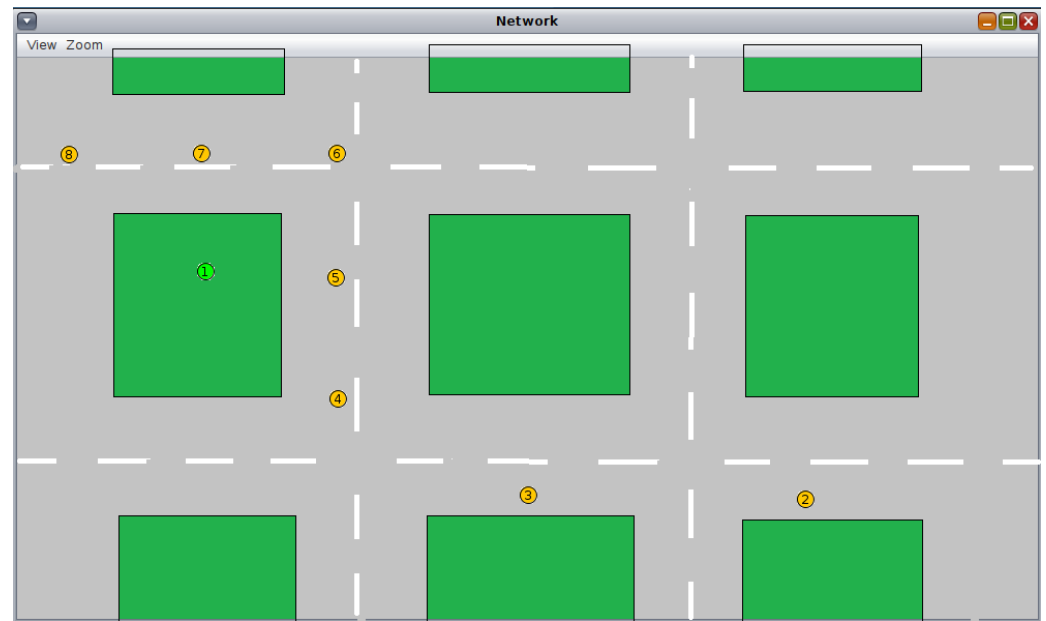


Figure 6. Clients start moving gradually to complete their tasks.

Despite moving in series, the clients established a reliable communication path with the server, facilitating seamless task completion (Figure 7).

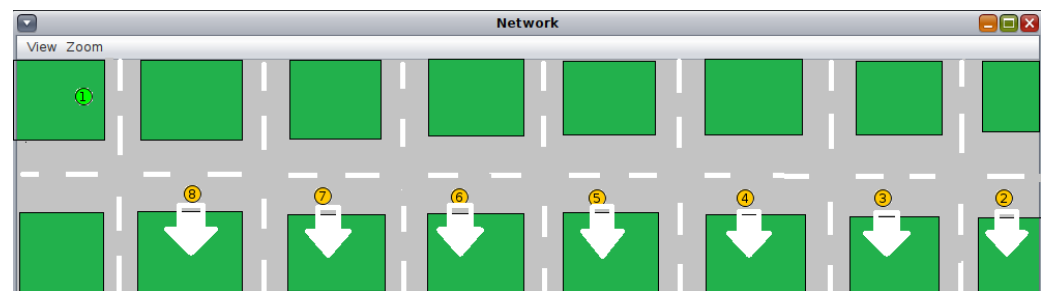


Figure 7. All clients (e.g., AGVs) reached their destination and completed their tasks.

After completing the task, clients (e.g., AGVs) returned to their station point in the same way, in reverse, and waited for the next task, which may differ depending on the requirements. In terms of tools, the Contiki OS was utilized due to its lightweight nature and its provision of a comprehensive environment for simulating and testing IoT applications. Contiki's extensive support for various communication protocols, including UDP and 6LoWPAN, made it a fitting choice for this research. The COOJA simulator, an integral part of the Contiki ecosystem, facilitated the modeling and testing of network protocols and their performance under different conditions. The CC2650 Launchpad devices were employed for their compatibility with Contiki OS and their support for low-power communication protocols, making them a good platform for practical implementation and comparison of real-world results with simulation outcomes [25].

3.1.3. Algorithm for Performance Evaluation

In the context of this research, we developed an algorithm specifically designed to evaluate the performance of UDP when integrated with a modified acknowledgment mechanism across various IoT network topologies. The algorithm systematically assessed key performance metrics such as packet delivery success rate, round-trip time (RTT), server processing time (SPT), and power consumption. By analyzing these metrics within different topologies, the algorithm identified potential issues related to packet delivery, particularly in scenarios with varying levels of network interference or device placement challenges. The algorithm also incorporated steps to optimize network performance by adjusting

device placements and communication parameters when the initial evaluation indicates suboptimal results. This approach ensured a thorough examination of UDP's effectiveness in diverse IoT environments, offering valuable insights for optimizing communication protocols and energy consumption. Algorithm 1 presents the step-by-step procedure for conducting this comprehensive performance evaluation.

Algorithm 1: Performance evaluation of UDP with modified acknowledgment mechanism

```

1: Input:
2:   Set of topologies  $\{T_1, T_2, \dots, T_n\}$ 
3:   UDP packet sequences
4:   Power consumption data
5: Output:
6:   Performance metrics for each topology
7:
8: for each topology  $T_i$  in  $\{T_1, T_2, \dots, T_n\}$  do
9:   Evaluate UDP performance:
10:  Compute the packet delivery success rate for UDP packets in  $T_i$ 
11:  if the delivery success rate meets the required threshold then
12:    Proceed with performance evaluation:
13:    Measure power consumption during packet transmission and reception in  $T_i$ 
14:    Compute performance metrics: round-trip time (RTT), server processing time (SPT), and
        power consumption for  $T_i$ 
15:  else
16:    Investigate issues:
17:    Adjust device placement:
18:    Optimize the physical placement of devices to improve communication success, ensuring
        minimal distance and line-of-sight between nodes.
19:    Evaluate communication factors:
20:    Analyze and adjust factors such as signal strength, interference sources, and network
        congestion to enhance communication reliability.
21:    Implement network adjustments:
22:    Modify network parameters like transmission power or data rates if necessary to improve
        communication performance.
23:    Document observations: Include details on the adjusted device placements, communication
        factors addressed, and any remaining challenges or improvements achieved.
24:    Compute performance metrics: include observed RTT, SPT, and power consumption after
        adjustments.
25:  end if
26: end for
27: Return: Summary of performance metrics for all topologies evaluated, including detailed
        findings for those with performance issues

```

3.2. UDP Enhancement with Acknowledgment and Power Consumption Metrics for IoT Networks

In this study, we introduced enhancements to the traditional UDP protocol to support acknowledgment and power consumption metrics specifically for IoT networks. The primary enhancement was the addition of an acknowledgment mechanism to the standard UDP protocol. This mechanism allowed the server to send acknowledgment (ACK) packets back to the client upon receiving data packets. The ACK packets included an acknowledgment header that contained a sequence number to identify the corresponding data packet and an optional status code indicating the receipt status. This addition helped confirm successful data transmission and improved the reliability of communication [18,26].

Alongside this, we integrated power consumption monitoring into the UDP communication process to evaluate energy usage. This involved recording CPU and low-power mode (LPM) power consumption using the *Energest* module in Contiki OS [25]. The CPU power consumption was measured during data transmission and acknowledgment han-

dling, while LPM power consumption was assessed during periods when the device was idle, reflecting the efficiency of power management. These enhancements allowed for a comprehensive evaluation of UDP's performance, focusing on both data reliability and energy efficiency in various IoT network topologies. The detailed structure of these enhancements ensured that while UDP retained its low-overhead nature, it also provided crucial feedback on communication reliability and energy usage.

3.3. Research Approach

The communication methodologies of UDP server–client communication and RPL routing were selected for their suitability in addressing real-time communication demands and optimizing routing paths within low-power IoT networks. UDP's connectionless nature allows for faster data transmission with minimal overhead, making it a good choice for time-sensitive applications where delays can be detrimental. This is particularly important in scenarios involving both dynamic and static placement of devices, where real-time data exchange is critical for navigation and coordination [27]. RPL routing, on the other hand, is designed to handle the complexities of low-power and lossy networks typical of IoT environments. Its capability to create a Destination-Oriented Directed Acyclic Graph (DODAG) ensures efficient data routing, minimizing power consumption and maximizing network lifespan. This is especially beneficial in applications like smart home automation, where devices frequently communicate with each other and with central hubs [28]. Furthermore, combining UDP and RPL is particularly advantageous in tunnel monitoring systems, where the network topology can change dynamically due to environmental factors. RPL's adaptability ensures reliable data transmission despite these changes, while UDP ensures that the data are transmitted swiftly and with minimal delay.

In this study, three specific network topologies were chosen to comprehensively evaluate the performance of these communication methodologies: linear chain network, structured tree network, and dynamic transition network.

The linear chain network was selected for its simplicity, allowing us to establish baseline performance metrics and understand basic data propagation delays. This straightforward configuration provided clear insights into fundamental communication behaviors and protocol efficiency.

The structured tree network was included to examine the effects of hierarchical node placement on performance. This topology helped in analyzing how well UDP and RPL handled structured, hierarchical routing scenarios, which are common in many IoT applications where data aggregation and organized communication paths are crucial.

The dynamic transition network was chosen to simulate real-world environments where network conditions change dynamically. This topology was essential for evaluating how well UDP and RPL adapted to varying network conditions, reflecting scenarios where routing paths and network configurations are not static.

By employing these three topologies, the research aimed to cover a broad spectrum of network scenarios, providing a comprehensive evaluation of the communication methodologies' strengths and limitations in different IoT environments. The multi-layered approach, from the physical layer with the COOJA simulator and CC2650 Launchpad to the transport layer with UDP, ensured that the findings were robust and applicable to a range of real-world applications [29]. Figure 8 shows the implementation of these protocols, highlighting their integration for robust IoT communication.



Figure 8. Implementation of selected supported protocols.

3.4. Key Parameters for Performance Evaluation

This section outlines the key parameters used for performance evaluation in our study. The primary metrics considered were the round-trip time (RTT), server processing time (SPT), and power consumption. These metrics are essential for assessing the performance and efficiency of the network in different scenarios [25]. RTT is a critical performance metric that measures the time a signal takes to travel from the source node to the destination node and back again [30]. It is a crucial indicator of network latency and is influenced by numerous factors such as network topology, node processing delays, and routing protocols [31]. RTT is calculated using the following formula:

$$RTT = T_{received} - T_{sent} \quad (1)$$

where $T_{received}$ is the timestamp when the packet is received back at the source node, and T_{sent} is the timestamp when the packet is originally sent from the source node. A lower RTT value signifies a more responsive and efficient network, which is particularly important for real-time applications [32].

SPT refers to the time taken by the server to process a request from a node and send back the response [33]. It is an essential metric for understanding the computational load and efficiency of the server in handling network requests [34]. SPT is calculated as follows:

$$SPT = T_{response\ sent} - T_{request\ received} \quad (2)$$

where $T_{response\ sent}$ is the timestamp when the server sends the response, and $T_{request\ received}$ is the timestamp when the server receives the request from the node [35]. Lower SPT values indicate faster server processing, which is crucial for maintaining overall network performance [36].

Power consumption is a vital metric for evaluating the energy efficiency of the network, especially in resource-constrained environments such as wireless sensor networks [37]. We focused on two primary components of power consumption: CPU power and low-power mode (LPM) power [38]. The total power consumption is the sum of the power the CPU uses when it is active and the power used when the nodes are in a low-power state [39]. To calculate power consumption, we used the energy consumption values obtained from the Energest module in Contiki, which provides data in terms of ticks. These ticks are then converted to power consumption using the supply voltage (V) and the current (I) [40].

CPU power consumption is the power the CPU uses when it is active, processing data, or performing tasks. It is measured in milliwatts (mW) and calculated as follows:

$$\text{CPU Power} = \left(\frac{\text{CPU ticks}}{\text{RTIMER ticks per second}} \right) \times V \times I_{\text{CPU}} \quad (3)$$

where *CPU ticks* is the number of ticks the CPU is active, *RTIMER ticks per second* is the number of real-time timer ticks per second, *V* is the supply voltage, and *I_{CPU}* is the current consumed by the CPU [41].

LPM power consumption is the power the nodes use when they are in a low-power state, conserving energy. It is also measured in milliwatts (mW) and calculated as follows:

$$\text{LPM Power} = \left(\frac{\text{LPM ticks}}{\text{RTIMER ticks per second}} \right) \times V \times I_{\text{LPM}} \quad (4)$$

where *LPM ticks* is the number of ticks the node is in low-power mode, *RTIMER ticks per second* is the number of real-time timer ticks per second, *V* is the supply voltage, and *I_{LPM}* is the current consumed in low-power mode [42]. The total power consumption is then the sum of CPU and LPM power consumption over the entire network [43]. Monitoring and minimizing power consumption is crucial for extending the battery life of nodes and ensuring the sustainability of the network.

3.5. Simulation and Experimental Setup

This section details the simulation and hardware setups employed to evaluate the performance of three specific network topologies: linear chain network, structured tree network, and dynamic transition network. The evaluations were conducted using the Cooja simulator alongside the Zolertia Z1 mote (Zolertia, Barcelona, Spain) and the Texas Instruments CC2650 LaunchPad (Texas Instruments, Dallas, TX, USA).

In the linear chain network, nodes were positioned in a straight line, each communicating only with its adjacent neighbor. This configuration assessed the cumulative delay and processing time as data passed through multiple nodes [25]. The structured tree network organized nodes in a hierarchical tree structure, where the root node communicated with several child nodes, which may themselves have child nodes. This setup investigated the impact of hierarchical node placement on network performance metrics such as round-trip time (RTT) and server processing time (SPT) [30]. The dynamic transition network featured nodes with dynamic and adaptive routing capabilities, allowing them to adjust their routing paths based on current network conditions to simulate a realistic and variable network environment [38].

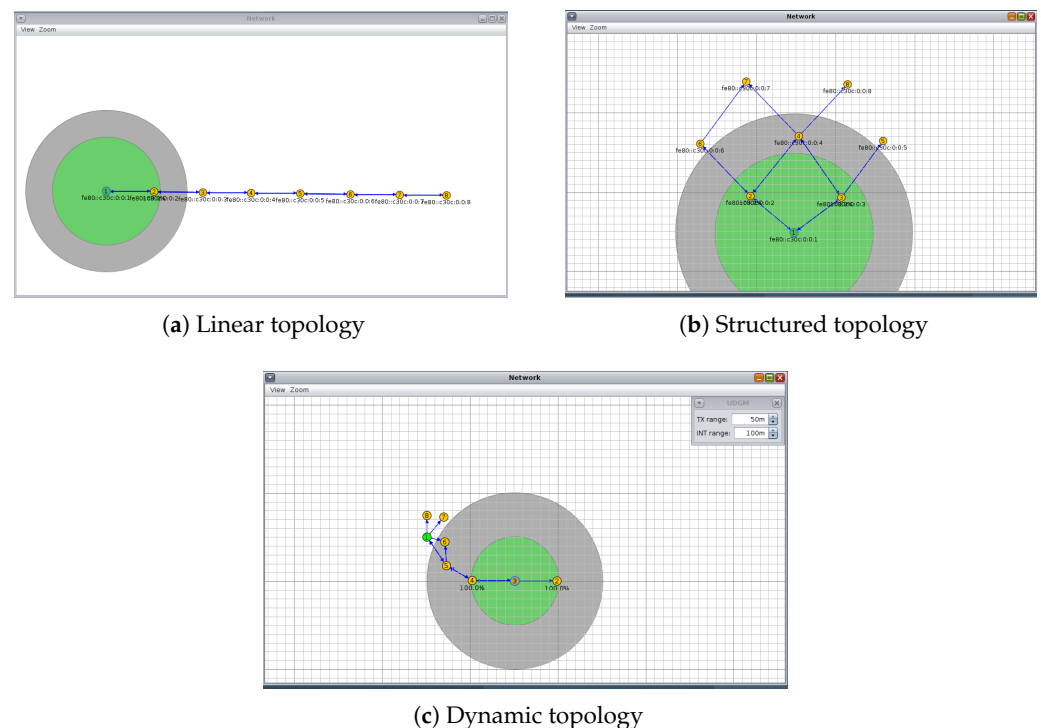
The Cooja simulator, a Java-based tool designed to emulate the behavior of IoT devices running Contiki OS, was used for these simulations. Cooja's capability for detailed and cycle-accurate simulations makes it a good platform for performance evaluation [40]. The Zolertia Z1 mote was used as the hardware platform for the simulations. Key features of the Z1 mote include the MSP430 microcontroller, renowned for its energy efficiency. The operating voltage ranges from 1.8 V to 3.6 V for the MCU and 2.1 V to 3.6 V for the transceiver, with an operational temperature range of $-40\text{ }^{\circ}\text{C}$ to $+85\text{ }^{\circ}\text{C}$ and a maximum clock frequency of 16 MHz [44]. The Z1 mote exhibits varying power consumption levels in different operational modes: 2 mA in active mode (MCU), 0.5 μA in low-power mode (MCU), 17.4 mA during radio transmission, 18.8 mA during radio reception, 426 μA in idle mode, and 0.1 μA in off mode [44]. These specifications were essential for accurately modeling node power consumption in different network scenarios. As detailed in Table 1, the simulation parameters include settings such as the transmission range and the number of motes.

Table 1. Simulation Parameters.

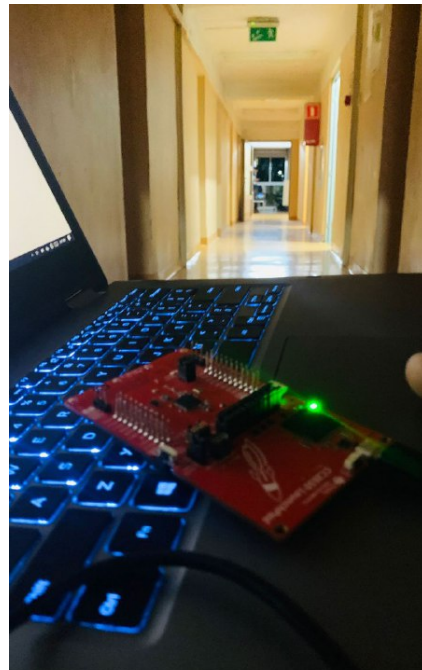
Parameter	Configuration
Transmission range	50 m
Server mote	One
Client motes	Seven
Radio medium	Unit disk graph medium
Transport layer protocol	UDP
PHY and MAC Layer	IEEE 802.15.4
Simulation time	One thousand seconds

For hardware experiments, the three network topologies used in simulations were replicated: the linear chain network, structured tree network, and dynamic transition network. Each CC2650 LaunchPad was configured as a mote using the Contiki-NG operating system, designed for IoT devices [45]. Power consumption was calculated by combining current consumption values from the datasheet, supplied voltage, and tick values from experiments, focusing on CPU and low-power mode (LPM) power. Nodes communicated via the IEEE 802.15.4 protocol, common in low-power wireless networks [25]. Data on round-trip time (RTT), server processing time (SPT), and power consumption (CPU and LPM) were collected and analyzed. Hardware experiment results were compared with simulations to validate the simulation models' accuracy and reliability. Experiments were conducted in controlled indoor environments for structured tree network scenarios and outdoor environments for linear chain and dynamic transition network scenarios, facilitating the precise measurement and analysis of network performance metrics across configurations [46].

The simulation setups illustrated in Figure 9 provide a clear view of how the motes were arranged for each topology. Figure 9a shows the linear topology, where motes were aligned in a straight line to mimic a linear chain. In contrast, Figure 9b demonstrates the structured topology with motes placed in a more gridlike formation, representing a typical structured network layout. Figure 9c depicts the dynamic topology, designed to simulate mobility and varying network conditions.

**Figure 9.** Simulation setup for different topologies: (a) linear, (b) structured, (c) dynamic.

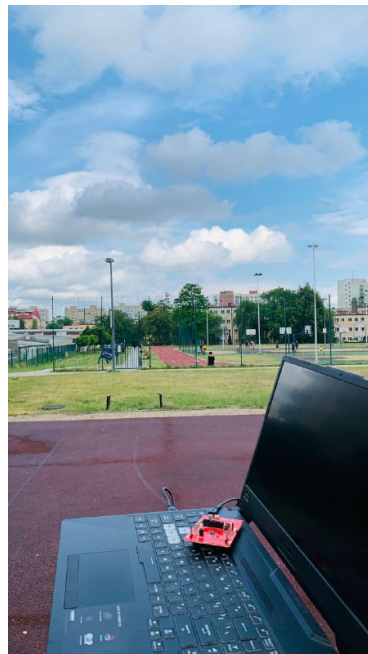
Similarly, the hardware experimental setups shown in Figure 10 capture the real-world environments used for testing. Figure 10a presents the linear topology setup in a long corridor, Figure 10b shows the structured topology with devices placed in different rooms, and Figure 10c illustrates the dynamic topology conducted in an open stadium, representing a scenario with movement and a variable distance among devices.



(a) Linear topology



(b) Structured topology



(c) Dynamic topology

Figure 10. Hardware experimental setup for different topologies: (a) linear, (b) structured, (c) dynamic.

3.6. Implementation and Protocol Application

Throughout the implementation process, various communication protocols were employed to facilitate data exchange between IoT devices. The study focused on three

primary protocols: UDP, RPL, and 6LoWPAN, each operating at different layers of the OSI model and contributing uniquely to the network's functionality.

UDP (User Datagram Protocol) operates at the transport layer, providing a lightweight and connectionless communication protocol. It is characterized by minimal overhead and lack of connection establishment, making it particularly suitable for IoT applications where low latency and reduced protocol complexity are crucial. In this study, UDP was utilized for its simplicity and efficiency in scenarios where timely data transmission was essential. A modified UDP protocol, incorporating an acknowledgment mechanism similar to TCP but without retransmission, was implemented to evaluate performance under different network topologies and hardware setups.

RPL (Routing Protocol for Low-Power and Lossy Networks) operates at the network layer, offering robust and energy-efficient routing capabilities tailored for low-power IoT networks. It utilizes a structured approach to route data packets efficiently across a network of constrained devices, optimizing battery life and network performance. The implementation of RPL aimed to assess its effectiveness in managing communication paths and ensuring reliable data delivery in various IoT scenarios.

6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) functions at the adaptation layer, providing mechanisms for header compression and fragmentation to facilitate IPv6 communication over low-power wireless networks. By compressing IPv6 headers, 6LoWPAN reduces the quantity of data transmitted over the air, which is critical for maintaining efficiency in resource-constrained environments. This protocol was employed to enable seamless integration of IoT devices into larger IPv6 networks, ensuring compatibility and interoperability across different network layers.

The implementation process utilized the Contiki-NG operating system (OS), a lightweight, open-source platform that supports various hardware platforms and facilitates the development of low-power wireless communication protocols. Contiki-NG's modular architecture allowed for the easy integration of communication protocols such as UDP, RPL, and 6LoWPAN, enabling seamless communication across IoT networks. The COOJA simulator, integral to Contiki-NG, provided a powerful tool for simulating various network topologies and scenarios. COOJA's capability to emulate network behavior and visualize protocol interactions facilitated comprehensive testing and evaluation.

The CC2650 Launchpad from Texas Instruments served as the development platform for implementing and testing IoT solutions. This versatile platform, featuring an ARM Cortex-M3 processor and integrated radio transceiver, supported Contiki-NG and ensured compatibility with the communication protocols under study. Terminals such as Tera Term played a crucial role in interfacing with the CC2650 Launchpad, providing a user-friendly interface for sending commands, receiving data, and analyzing experimental results. Additionally, Smart RF Programming Tools were used to configure the radio transceiver settings, optimizing communication performance for the specific IoT applications.

By incorporating these protocols and tools, the study achieved efficient and reliable communication within IoT networks. The integration of UDP, RPL, and 6LoWPAN, along with the use of Contiki-NG, COOJA, CC2650 Launchpad, and various terminals and programming tools, contributed to the development of a robust IoT communication framework. This approach provided valuable insights into protocol performance and interactions, advancing the understanding and deployment of IoT technology.

3.7. Tools and Software Utilized

Contiki-NG operating system (OS) has emerged as a prominent choice for IoT development due to its lightweight, open-source nature, and support for various hardware platforms [47]. It facilitates the development of low-power wireless communication protocols, making it a good choice for resource-constrained IoT devices [48]. Contiki-NG's modular architecture allows for easy integration of communication protocols such as UDP, RPL (Routing Protocol for Low-Power and Lossy Networks), and 6LoWPAN (IPv6 over Low-power Wireless Personal Area Networks), enabling seamless communication across

IoT networks [49]. At the physical layer, Contiki-NG provides support for various radio transceivers, ensuring compatibility with a wide range of IoT hardware [50]. It abstracts the complexities of wireless communication, enabling developers to focus on higher-level protocol implementations [51]. In the data link layer, Contiki-NG implements protocols like IEEE 802.15.4, facilitating reliable communication over low-power wireless networks [52]. For this study, Contiki-NG served as the foundation for implementing communication protocols and conducting experiments on the CC2650 Launchpad. Its versatility and robustness made it an ideal choice for exploring UDP communication and evaluating protocol performance [53].

COOJA (Contiki's Network Simulator) is an integral part of the Contiki-NG ecosystem, providing a powerful tool for simulating IoT networks [54]. It allows the emulation of various network topologies and scenarios, enabling comprehensive testing and evaluation of communication protocols [55]. At the network layer, COOJA simulates the behavior of routing protocols like RPL, enabling the assessment of performance under different network conditions [56]. It also provides visualization capabilities, allowing researchers to observe network behavior and analyze protocol interactions [57]. COOJA's integration with Contiki-NG ensures compatibility with the platform's communication stack, including support for UDP, RPL, and 6LoWPAN [58]. This seamless integration facilitates the development and testing of communication protocols in a controlled environment, reducing the time and resources required for experimental setup and execution [59]. In our work, the COOJA simulator was utilized to simulate IoT network scenarios and evaluate the performance of UDP communication protocols. Its capability to replicate real-world conditions made it a valuable tool for validating protocol implementations and assessing their suitability for practical deployment [60].

The CC2650 Launchpad from Texas Instruments is a versatile development platform that supports Contiki-NG, simplifying the implementation and testing of IoT solutions [61]. It features an ARM Cortex-M3 processor and integrated radio transceiver, providing the necessary hardware capabilities for IoT applications [62]. At the physical layer, the CC2650 Launchpad supports IEEE 802.15.4 radio communication, ensuring compatibility with Contiki-NG's communication stack [63]. This integration enables developers to deploy and evaluate IoT applications directly on the Launchpad, streamlining the development process and reducing time-to-market [64]. In this study, the CC2650 Launchpad served as the target platform for implementing UDP communication protocols developed in Contiki-NG. Its compatibility with Contiki-NG simplified the deployment of experimental setups and facilitated real-world testing, ensuring the validity and relevance of research findings [65].

Terminals, such as Tera Term, play a crucial role in interfacing with the CC2650 Launchpad and monitoring communication between devices. They provide a user-friendly interface to send commands and receive data, facilitating the debugging and analysis of experimental results [66]. In this study, terminals were utilized to observe the behavior of UDP communication protocols, aiding in the identification of potential issues and optimization of protocol parameters for enhanced performance in IoT deployments [67].

Smart RF Programming Tools are essential for programming and configuring the CC2650 Launchpad's radio transceiver [68]. These tools enable developers to customize radio settings and parameters, optimizing communication performance for specific IoT applications [69]. In our study, Smart RF Programming Tools were utilized to fine-tune the radio settings to ensure compatibility with Contiki-NG's communication stack and enhance the reliability and efficiency of UDP communication protocols [70].

4. Results and Discussion

The objective of our experiments was to evaluate and compare the performance and energy consumption of wireless sensor networks (WSNs) using both simulation and real hardware components. The experiments aimed to provide insights into the energy efficiency and latency of different network topologies and to validate the simulation results against real-world hardware performance. Each of these scenarios was tested using both

the Contiki-NG Cooja Simulator with Z1 (Zolertia) motes and real hardware experiments using the CC2650 Launchpad. For the simulation setup, we used the Cooja simulator to model networks comprising up to eight Z1 motes. The Z1 mote features an MSP430 microcontroller with a 16-bit architecture, 10 KB of RAM, and 48 KB of flash memory, which are representative of resource-constrained IoT devices. The motes were configured to use the 6LoWPAN protocol at the data link layer, RPL for routing, and UDP for transport layer communication. Simulation parameters included a communication range of 50 m and data transmission intervals set at 60 s. In the real hardware experiments, we utilized the TI CC2650 Launchpad, which included an ARM Cortex-M3 microcontroller with 20 KB of RAM and 128 KB of flash memory. The hardware setup mirrored the simulation environment, with the same network configurations and communication protocols. Each scenario (linear chain, structured tree, and dynamic transition networks) was tested under identical conditions to ensure comparability. The results from these tests were analyzed and compared to understand the differences in performance and power consumption between simulated and real-world conditions, providing a comprehensive evaluation of WSN performance in various network topologies.

4.1. Round-Trip Time (RTT) Analysis

First, we performed the RTT analysis for three different network scenarios, including linear chain network, structured tree network, and dynamic transition networks. We discuss the RTT results obtained from both simulation and hardware experiments and provide visual representations for each of the eight motes in each scenario. Figure 11a,b show the RTT results for the linear chain network scenario from simulation and hardware experiments, respectively.

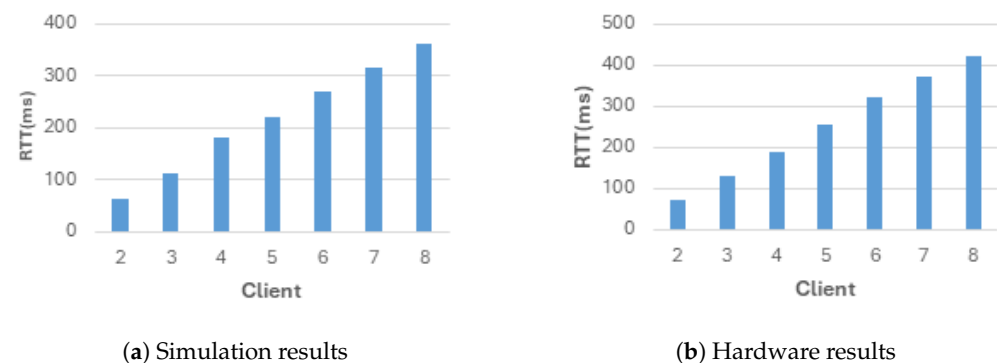


Figure 11. RTT for the linear chain network scenario.

Mote 1 served as the base node and did not measure RTT. From the simulation results shown in Figure 11a, the RTT for Mote 2 started at 62.0 ms and increased progressively to 362.3 ms for Mote 8. This incremental rise in RTT highlighted the cumulative delay as data traversed more nodes in the linear chain. Like the simulation, the hardware results displayed an increasing trend in RTT with the addition of each mote. The linear increase in RTT for both simulation and hardware setups reflected the anticipated behavior of a linear chain topology where each node adds a fixed processing and transmission delay.

Figure 12a,b depict the RTT results for the structured tree network scenario from simulation and hardware experiments, respectively.

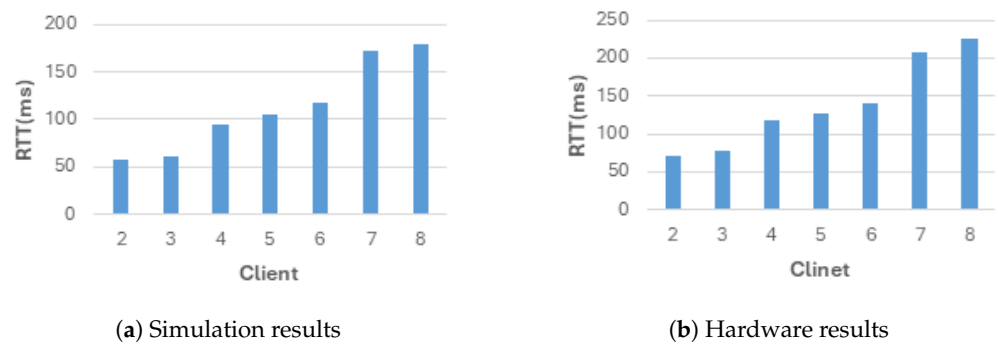


Figure 12. RTT for the structured tree network scenario.

The RTT values showed a structured increment, reflecting the hierarchical arrangement where motes closer to the root node experienced a lower RTT. The RTT for Mote 2 was 69.96 ms, escalating to 224.56 ms for Mote 8, according to the simulation arrangement shown in Figure 12a. The hardware experimental results shown in Figure 12b confirmed the increase in RTT, which was similar to the simulation one, though slightly higher, likely due to real-world factors such as interference and processing overhead. The structured tree network showed that RTT grew with the depth of the tree, illustrating the impact of hierarchical node placement on communication delay.

Figure 13a,b illustrate the RTT results for the dynamic transition network scenario from simulation and hardware experiments, respectively.

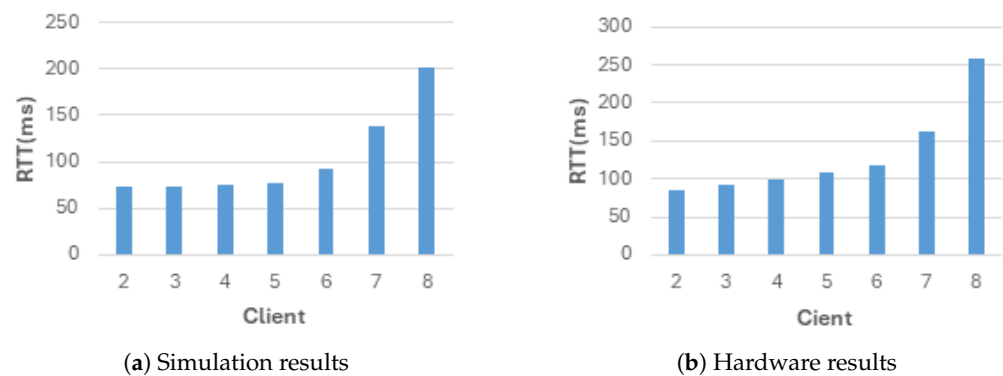


Figure 13. RTT for the dynamic transition network scenario.

These results demonstrated dynamic routing changes affecting RTT, with variability introduced by transitioning paths. The RTT for Mote 2 was 85.67 ms, increasing to 258.36 ms for Mote 8, as shown in Figure 13a. The hardware RTTs were higher than the simulation ones, indicating additional delays likely due to real-time network reconfiguration and environmental factors. The dynamic transition network highlighted the effects of adaptive routing on RTT, with noticeable increases as nodes dynamically switched paths, reflecting real-world variability and adaptability challenges.

The RTT analysis across all three scenarios demonstrated the expected influence of network topology and real-world conditions on communication delay. The linear chain network confirmed a clear linear increase in RTT with each additional mote, which is important when designing these types of topologies in tunnels or long corridors. The structured tree network reflected hierarchical delays, with RTT increasing based on the depth of the node. The dynamic transition network, with its adaptive routing, showed higher and more variable RTTs due to the dynamic nature of path transitions. In all tested scenarios, the results obtained with the real devices had similar tendencies, although the RTTs were a bit higher than those from the simulation.

4.2. Server Processing Time (SPT) Analysis

During our experiments, we also analyzed the server processing time (SPT) for the three network topologies and usage scenarios, i.e., linear chain network, structured tree network, and dynamic transition network. The SPT results are reported and discussed for both simulation and hardware experiments and supported by visual representations for each of the eight motes in each scenario. Figure 14 shows the SPT results for the linear chain network scenario from simulation and hardware experiments.

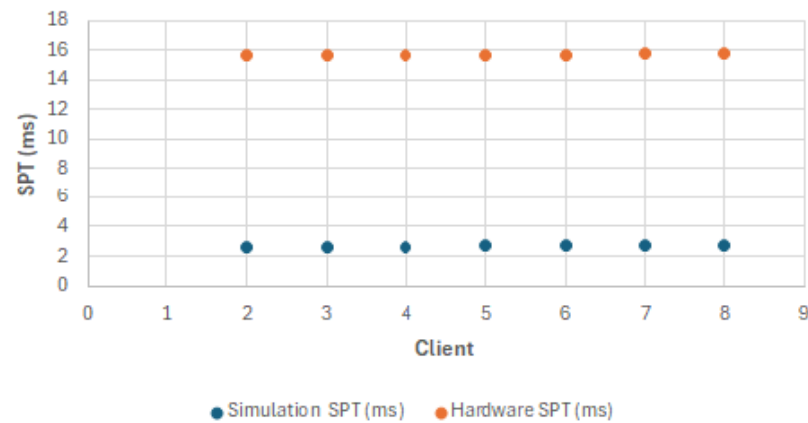


Figure 14. SPT for the linear chain network scenario.

In the linear chain network scenario, Mote 1 served as the base node and did not measure server processing time (SPT). The SPT for Mote 2 started at 2.62 ms and remained consistent, reaching 2.75 ms for Mote 8. This minimal variation in SPT indicated a stable processing time across the motes, reflecting the straightforward nature of the linear topology. In the hardware results, the SPT for Mote 2 started at 15.6 ms and increased slightly to 15.8 ms for Mote 8. The hardware results showed a similarly stable trend, with slightly higher values due to real-world processing overhead. The SPT values in both simulation and hardware experiments for the linear chain network showed minimal variation, consistent with the expected behavior of a simple linear topology where each mote processes data uniformly. This stability underscored the efficiency of linear topologies in maintaining consistent processing times across nodes.

Figure 15 depicts the SPT results for the structured tree network scenario from simulation and hardware experiments.

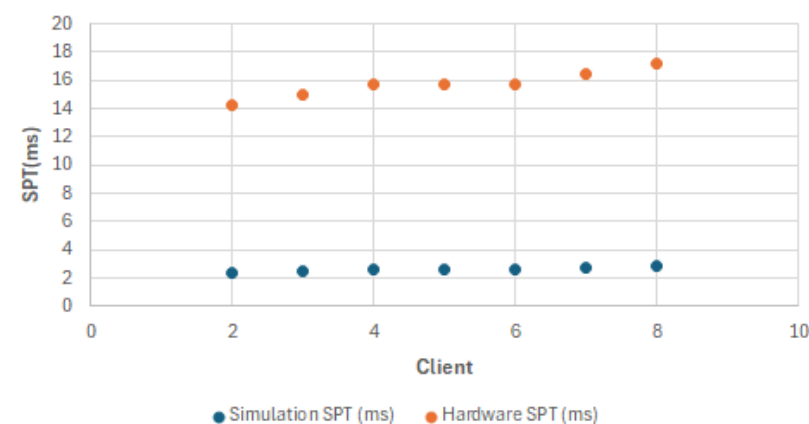


Figure 15. SPT for the structured tree network scenario.

In the structured tree network scenario, the simulation results showed the server processing time (SPT) for Mote 2 was 2.37 ms, increasing marginally to 2.87 ms for Mote 8.

The slight incremental rise in SPT values reflected the hierarchical structure, where each additional level added a minor processing delay. Hardware experiments provided the SPT for Mote 2 at the level of 14.22 ms, rising to 17.22 ms for Mote 8. These results were slightly higher than the simulation results, likely due to real-world factors such as network interference and processing overhead. The structured tree network results indicated that the SPT increased modestly as motes were added, which was expected due to the hierarchical structure of the network. The consistent processing times across the levels illustrated the efficiency of the structured approach.

Figure 16 illustrates the SPT results for the dynamic transition network scenario from simulation and hardware experiments.

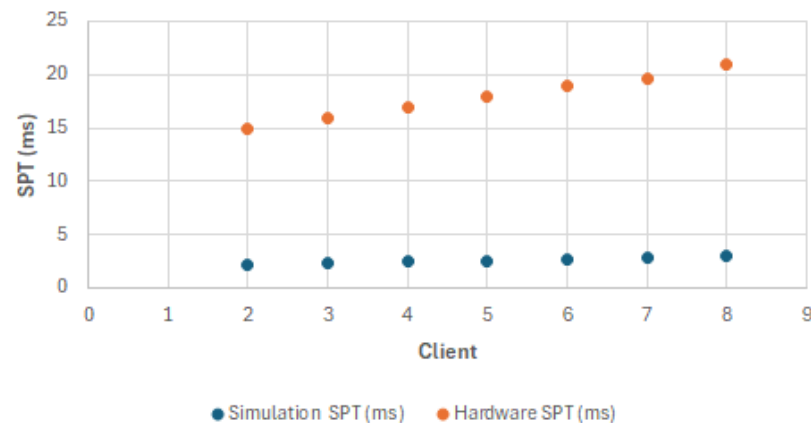


Figure 16. SPT for the dynamic transition network scenario.

In the dynamic transition network scenario, the simulation results showed a gradual increase in server processing time (SPT) from 2.14 ms for Mote 2 to 3.00 ms for Mote 8. This incremental rise in SPT values reflected the dynamic nature of the network, where each mote transition added a slight processing delay. The hardware experiments indicated a similar trend but with higher SPT values, starting from 14.98 ms for Mote 2 and increasing to 21.00 ms for Mote 8. The discrepancy between the simulation and hardware results could be attributed to real-world factors such as network interference and processing overhead. These findings demonstrated that while the dynamic transition approach effectively managed processing times, real-world conditions must be considered for accurate performance assessment. The results highlighted the importance of hardware validation in conjunction with simulation for comprehensive network evaluation. Moreover, the hardware SPT values illustrated the need to optimize dynamic network deployments to minimize latency. Overall, the dynamic transition network exhibited a predictable increase in processing times, reinforcing the necessity for good design to handle varying network conditions efficiently.

The SPT analysis across all three scenarios demonstrated the influence of network topology and real-world conditions on server processing times. The linear chain network showed stable SPT values due to its straightforward topology. The structured tree network reflected a hierarchical increase in SPT, consistent with its structured arrangement. The dynamic transition network, with its adaptive routing, showed higher and more variable SPT values, highlighting the challenges and complexities of dynamic path adjustments.

4.3. Power Consumption Analysis

Finally, we examined the power consumption focusing on CPU and low-power mode (LPM) for the three different network scenarios, i.e., linear chain network, structured tree network, and dynamic transition network. Again, we analyzed the power consumption results coming from both simulation and hardware experiments, providing a visual repre-

sensation for each of the eight motes in each scenario. CPU and LPM power consumption values are depicted together in the same figures for a comprehensive comparison.

Figure 17a,b show the CPU and LPM power consumption results for the linear chain network scenario from simulation and hardware experiments, respectively.

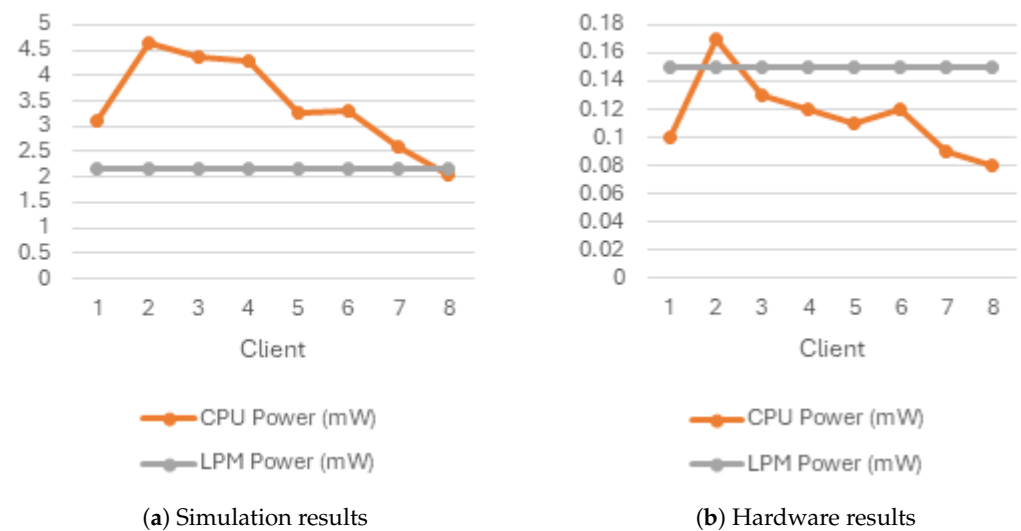


Figure 17. CPU and LPM power consumption for the linear chain network scenario.

The simulation results showed that CPU consumption started at 3.12 mW for Mote 1 and increased to 4.64 mW for Mote 2, then gradually decreased to 2.03 mW for Mote 8. LPM consumption was stable across all motes, around 2.18 mW. The initial increase in CPU power consumption reflected the additional processing required as data passed through each mote, while the subsequent decrease indicated stabilization in processing requirements. The stable LPM consumption indicated that motes spent a significant portion of their time in low-power mode, reflecting efficient energy usage. The hardware experimental results showed that CPU consumption started at 0.10 mW for Mote 1, peaked at 0.17 mW for Mote 2, and gradually decreased to 0.08 mW for Mote 8. LPM consumption was consistent at 0.15 mW for all motes. The decrease in CPU consumption for higher motes may be attributed to efficiency improvements in real hardware as the network stabilized. The consistent LPM power consumption suggested a similar energy usage pattern in hardware as seen in the simulation [71].

Figure 18a,b show the CPU and LPM power consumption results for the structured tree network scenario from simulation and hardware experiments, respectively.

The simulation results in this scenario showed that the CPU consumption varied, with Mote 1 at 3.48 mW and Mote 3 at 5.26 mW, decreasing to 1.74 mW for Mote 7. LPM consumption was stable, around 2.16 mW for all motes. The variation in CPU consumption reflected the hierarchical structure of the network, where higher-level nodes had more processing load. The stable LPM usage indicated efficient energy management across the structured tree network. The hardware experiments showed that the CPU consumption started at 0.60 mW for Mote 1, peaked at 0.90 mW for Mote 3, and decreased to 0.40 mW for Mote 8. LPM consumption was consistent at 0.15 mW for all motes. The pattern was similar to that obtained from the simulation, with CPU consumption varying due to the hierarchical structure and different processing loads. The stability in LPM power consumption reflected efficient energy management, as in the case of the simulation results. These findings underscored the effectiveness of both simulation and hardware setups in replicating the energy consumption characteristics of the structured tree network. The consistency between simulated and real-world results validated the reliability of the methods used for performance evaluation in this research.

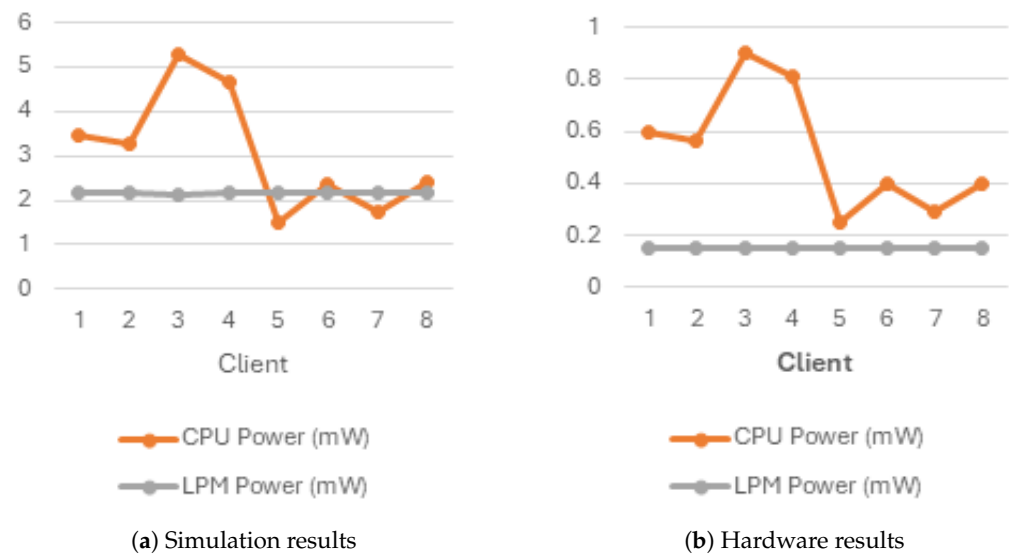


Figure 18. CPU and LPM power consumption for the structured tree network scenario.

Figure 19a,b show the CPU and LPM power consumption results for the dynamic transition network scenario from simulation and hardware experiments, respectively.

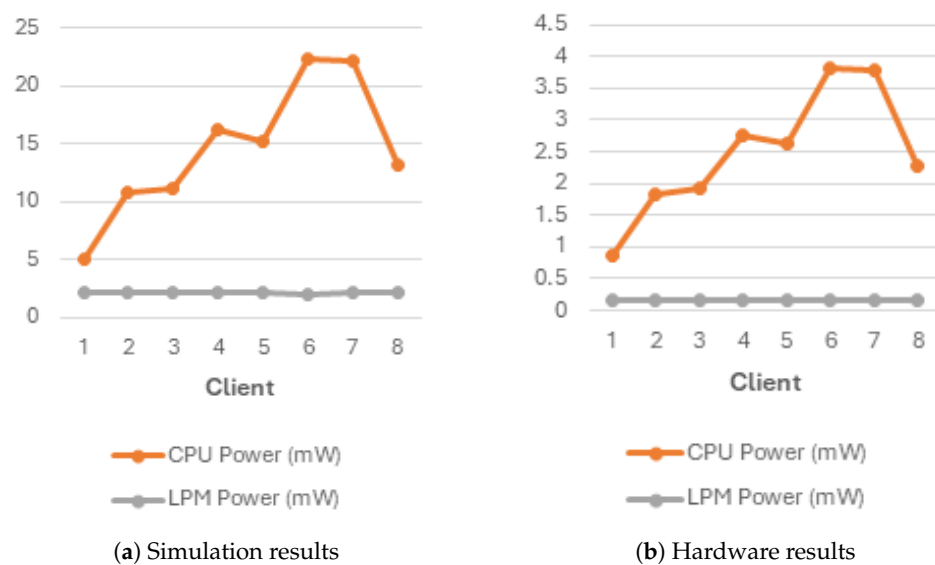


Figure 19. CPU and LPM power consumption for the dynamic transition network scenario.

In this scenario, CPU consumption exhibited variability across motes based on simulation results. It started at 5.09 mW for Mote 1, peaked at 22.28 mW for Mote 6, and decreased to 13.21 mW for Mote 8. Conversely, LPM consumption remained relatively stable, starting at 2.16 mW for Mote 1, slightly decreasing to 2.09 mW for Mote 6, and stabilizing around 2.14 mW for Mote 8. The high values and variability in CPU consumption reflected the dynamic nature of the network, with frequent changes in routing paths necessitating more processing power. However, the stable LPM consumption suggested that even with dynamic routing, motes managed to maintain a significant portion of their time in low-power mode. From the obtained hardware experimental results, we can see that CPU consumption started at 0.87 mW for Mote 1, peaked at 3.82 mW for Mote 6, and decreased to 2.27 mW for Mote 8. LPM consumption remained consistent at 0.15 mW for all motes. This pattern mirrored the simulation findings, with higher CPU consumption attributed to dynamic

adjustments in the network. Nonetheless, the stable LPM power consumption in hardware reflected efficient energy management despite the dynamic nature of the network.

The power consumption analysis across all three scenarios demonstrated the impact of network topology and real-world conditions on CPU and LPM usage. The linear chain network showed a gradual change in CPU consumption with stable LPM usage. The structured tree network benefited from a hierarchical structure, effectively distributing processing loads. The dynamic transition network exhibited higher and more variable CPU consumption due to frequent adjustments in routing paths, with stable LPM consumption reflecting efficient energy management.

4.4. Comparative Analysis of Standard and Enhanced UDP

This section presents a comparative analysis of the standard and enhanced UDP protocols in terms of power consumption and packet loss across different IoT network topologies. The analysis was conducted using hardware-based evaluations, which provided more reliable insights into real-world scenarios. We considered three representative topologies linear, structured, and dynamic and measured the average performance across multiple motes for each topology. The comparison was crucial to understanding the pros and cons of enhancing the standard UDP protocol with an acknowledgment mechanism.

4.4.1. Power Consumption Comparison

The power consumption of both the standard and enhanced UDP protocols was measured across all topologies, focusing specifically on CPU power consumption and calculated using Equation (3), as low-power mode (LPM) power consumption remained constant. The analysis used the average power consumption measured for each mote to provide a comprehensive view of the protocol performance under varying network conditions.

Let P_{std} and P_{enh} represent the CPU power consumption of standard and enhanced UDP, respectively. The percentage increase in power consumption was calculated using the formula:

$$\text{Percentage Increase} = \left(\frac{P_{enh} - P_{std}}{P_{std}} \right) \times 100\%. \quad (5)$$

Table 2 provides a summary of the average CPU power consumption for both the standard and enhanced UDP protocols across all topologies.

Table 2. Average CPU power consumption for standard and enhanced UDP protocols across different topologies.

Topology	Standard UDP CPU Power (mW)	Enhanced UDP CPU Power (mW)	% Increase in CPU Power Consumption
Linear	0.08	0.12	33.33%
Structured	0.37	0.53	30.19%
Dynamic	1.86	2.61	28.73%

From Table 2, we observe that the enhanced UDP protocol consumed more power due to the additional processing required for the acknowledgment mechanism. In a linear topology, where network conditions are more stable, the percentage increase was relatively higher compared to the dynamic topology. This was attributed to the frequent state changes and higher processing demands in dynamic networks, which led to naturally higher baseline power consumption.

4.4.2. Packet Loss Comparison

Packet loss is a critical metric for evaluating the reliability of data transmission in networked systems. The packet loss rates for both the standard and enhanced UDP

protocols were evaluated based on 100 packet transmissions from all clients in each topology. Due to the lack of an acknowledgment mechanism, the standard UDP protocol has no way to verify whether a packet reaches its destination, which can lead to higher packet loss rates, especially in more challenging environments such as dynamic networks.

Let L_{std} and L_{enh} denote the packet loss percentages for standard and enhanced UDP, respectively. The packet loss was calculated based on the ratio of lost packets to total sent packets:

$$\text{Packet Loss Percentage} = \left(\frac{\text{Lost Packets}}{\text{Total Sent Packets}} \right) \times 100\%. \quad (6)$$

Table 3 summarizes the packet loss percentages observed for both protocols across the different topologies:

Table 3. Packet loss percentages for standard and enhanced UDP protocols across different topologies.

Topology	Standard UDP Packet Loss (%)	Enhanced UDP Packet Loss (%)
Linear	5%	1%
Structured	15%	6%
Dynamic	30%	12%

As we can see in Table 3, the enhanced UDP protocol significantly reduced packet loss across all topologies. The impact of the acknowledgment mechanism was more pronounced in the dynamic topology, where frequent node mobility and environmental changes resulted in higher packet loss for the standard UDP protocol. In contrast, the enhanced UDP protocol effectively adapted to these conditions by retransmitting lost packets or adjusting the communication strategy based on acknowledgment feedback.

4.5. Topology Suitability in Real-World Applications

The versatility and adaptability of IoT networks make them suitable for a wide range of applications across various domains. This subsection explores three distinct use cases where different network topologies, including linear chain, structured tree, and dynamic transition networks, are leveraged to meet specific requirements. By analyzing these use cases, we can better understand how network structure impacts performance metrics such as round-trip time (RTT), server processing time (SPT), power consumption, and overall network efficiency. The selected use cases include Automated Guided Vehicles (AGVs) in industrial settings, home automation systems, and environmental monitoring in tunnels, each illustrating the unique benefits and challenges associated with different network configurations.

Automated Guided Vehicles (AGVs) are increasingly employed in industrial environments for tasks such as material handling, warehouse automation, and assembly line operations [72]. AGVs navigate through factories and warehouses, transporting goods efficiently and autonomously. These vehicles rely heavily on robust and efficient communication networks to ensure real-time control and coordination [73]. In an industrial setting, AGVs require a communication network that can handle dynamic environments where devices move continuously [74]. The dynamic transition network scenario is particularly applicable here. In this scenario, client devices (representing AGVs) and a central server maintain a mesh topology, supporting constant communication as the AGVs move. Data are collected at regular intervals, with performance metrics such as round-trip time (RTT), server processing time (SPT), power consumption, and packet loss being critical [72]. The higher RTT and SPT values observed indicate the potential latencies and increased energy demands associated with mobile AGVs, emphasizing the need for optimized network protocols and efficient power management to maintain operational efficiency.

Home automation systems integrate various smart devices to provide enhanced control, convenience, and energy efficiency within a residential environment [75]. Devices such

as smart thermostats, lighting systems, security cameras, and door locks communicate with a central hub or server to perform automated tasks based on user preferences and sensor data [76]. The structured tree network scenario is highly relevant for home automation applications. In this setup, devices are strategically positioned in a hierarchical tree topology with one server managing the network [77]. Each device communicates with the server and other devices, ensuring efficient data transmission and processing. The observed lower RTT and SPT values suggest that this topology supports rapid communication and quick responses, crucial for real-time home automation tasks [75]. Additionally, the reduced power consumption metrics indicate that the tree topology is energy-efficient, which is essential for battery-operated smart home devices [76]. This ensures that devices can operate for extended periods without frequent battery replacements, enhancing user convenience and overall system reliability.

Monitoring environmental conditions in tunnels, such as air quality, temperature, and structural health, is vital for ensuring safety and operational efficiency [78]. Sensors deployed along the length of the tunnel collect and transmit data to a central monitoring system [79]. These sensors need to operate reliably despite the elongated and constrained environment of tunnels. The linear chain network scenario reflects such cases well. Devices are arranged in a straight line with significant gaps between them, reflecting the physical layout of tunnel monitoring systems [80]. Each sensor sends data packets to the server, and the network's performance metrics are analyzed. The higher RTT values observed indicate potential communication delays due to the elongated paths (which should be taken into account while designing such network infrastructures), but the stable SPT values suggest consistent data processing times [78]. Moreover, the lower power consumption metrics highlight the energy efficiency of this topology, which is crucial for sensors that need to operate for extended periods with limited power sources [79]. This makes the linear chain network an effective model for ensuring reliable and efficient environmental monitoring in tunnels, where power efficiency and consistent data transmission are paramount.

These use cases illustrate the diverse applications of different network topologies in real-world scenarios. The dynamic transition network fits the dynamic industrial applications involving AGVs well, the structured tree network excels in energy-efficient home automation systems, and the linear chain Network is well suited for environmental monitoring in tunnels. By understanding the specific requirements and performance metrics of each scenario, it is possible to optimize communication protocols and network configurations to enhance the efficiency and reliability of IoT deployments across various application areas.

5. Discussion and Conclusions

Our study provided valuable insights into IoT network performance by measuring the time, energy, and packet delivery effectiveness of the enhanced UDP-based data transmission for different network topologies in an IoT resource-constrained environment. The findings indicated that CPU power consumption increased initially with additional nodes but stabilized over time, reflecting the balance between data handling needs and network stability. While these results align with established theories on network performance [1,8], they also underscore the importance of our approach. Measuring these patterns was essential to validate our assumptions and provide empirical evidence for the effects of topology on performance metrics.

The comparison between simulation and hardware results highlighted the necessity of accounting for real-world factors like network interference and processing overhead. Simulations, although valuable for initial design, often present idealized conditions that differ from hardware experiments. This discrepancy emphasizes the need for comprehensive testing in actual environments to ensure reliable performance.

Our comparative analysis of standard and enhanced UDP protocols showed that the enhanced UDP protocol, despite its higher CPU power consumption, significantly reduced packet loss. Specifically, power consumption increased by 33.33% in linear, 30.19%

in structured, and 28.73% in dynamic topologies, while packet loss decreased notably, particularly in dynamic environments where it dropped from 30% to 12%.

These results highlight the trade-off between energy consumption and data reliability. Although the enhanced UDP protocol introduced additional processing overhead, it offered substantial improvements in data integrity, which is crucial for applications requiring reliable data transmission.

Overall, this research enhanced our understanding of how network topology impacts power consumption and reliability, offering practical implications for IoT deployments. Looking ahead, future work should leverage machine learning and edge-cloud solutions to optimize network protocols. Machine learning can predict network conditions and adjust parameters dynamically, while edge-cloud integration can enhance data processing and scalability. These advancements will improve performance and energy efficiency, leading to more adaptive and intelligent IoT network architectures.

Author Contributions: Conceptualization, B.E.B., B.P. and K.T.; methodology, formal analysis, B.E.B. and D.M.; investigation, B.E.B. and N.Y.G.; resources, K.T. and D.M.; writing—original draft preparation, B.E.B., K.T. and D.M.; writing—review and editing, B.E.B., K.T. and D.M.; visualization, B.E.B.; supervision, K.T. and B.P.; project administration, D.M.; funding acquisition, D.M. All authors have read and agreed to the published version of the manuscript.

Funding: The research was supported by the ReActive Too project that has received funding from the European Union’s Horizon 2020 Research, Innovation, and Staff Exchange Programme under the Marie Skłodowska-Curie Action (grant agreement no. 871163), partially by a pro-quality grant for highly scored publications or issued patents (grant no. 02/100/RGJ23/0026), Statutory Research funds of Department of Applied Informatics, Silesian University of Technology, Gliwice, Poland (grant no. 02/100/BK_24/0035). Scientific work was published as part of an international project co-financed by the program of the Polish Minister of Science and Higher Education entitled “PMW” in the years 2021–2025 (contract no. 5169/H2020/2020/2).

Data Availability Statement: All data underlying the results are available as part of the article and no additional source data are required.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AGV	Automated Guided Vehicle
API	Application Programming Interface
CPU	Central Processing Unit
DNS	Domain Name System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
LPM	Low-power mode
MAC	Media Access Control
OS	Operating System
RAM	Random Access Memory
RFID	Radio-Frequency Identification
RTOS	Real-Time Operating System
SPT	Server processing time
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
VPN	Virtual Private Network
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access

XML	Extensible Markup Language
XSS	Cross-Site Scripting
YAML	Yet Another Markup Language
Zigbee	Zigbee Alliance
IoT	Internet of Things
VoIP	Voice over Internet Protocol
SMTP	Simple Mail Transfer Protocol

References

- Smith, J.; Jones, A.B. IoT Communication Protocols: A Comprehensive Review. *IEEE Internet Things J.* **2018**, *5*, 1234–1256.
- Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [\[CrossRef\]](#)
- Perera, C.; Zaslavsky, A.; Christen, P.; Georgakopoulos, D. Context-aware computing for the Internet of Things: A survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 414–454. [\[CrossRef\]](#)
- Martínez, G.; Hernández, J.A.; Reviriego, P.; Reinheimer, P. Round Trip Time (RTT) Delay in the Internet: Analysis and Trends. *IEEE Netw.* **2024**, *38*, 280–285. [\[CrossRef\]](#)
- Perera, C.; Liu, C.H.; Jayawardena, S.; Chen, M. A Survey on Internet of Things From Industrial Market Perspective. *IEEE Access* **2014**, *2*, 1660–1679. [\[CrossRef\]](#)
- Sharma, V.; Tiwari, R. A review of protocols used in Internet of Things (IoT) and low power wide area network (LPWAN). *Procedia Comput. Sci.* **2019**, *167*, 2151–2160.
- Garcia, R.; Martinez, L. UDP-based Communication Architecture for IoT Gateway under Intermittent Connectivity. *IEEE Trans. Ind. Inform.* **2021**, *17*, 2102–2113.
- Qays, M.O.; Ahmad, I.; Abu-Siada, A.; Hossain, M.L.; Yasmin, F. Key communication technologies, applications, protocols and future guides for IoT-assisted smart grid systems: A review. *Energy Rep.* **2023**, *9*, 2440–2452. [\[CrossRef\]](#)
- Chen, M.; Wan, J.; Gonzalez-Sanchez, J.L.; Liao, X.; Li, J. Machine-to-machine communications: Architectures, standards, and applications. *KSII Trans. Internet Inf. Syst.* **2014**, *8*, 1022–1039. [\[CrossRef\]](#)
- Jong, G.-J.; Wang, Z.-H.; Hendrick, H.; Hsieh, K.-S.; Horng, G.-J. A Novel Adaptive Optimization of Integrated Network Topology and Transmission Path for IoT System. *IEEE Sens. J.* **2019**, *19*, 6452–6459. [\[CrossRef\]](#)
- Andrews, J.G.; Buzzi, S.; Choi, W.; Hanly, S.V.; Lozano, A.; Soong, A.C.; Zhang, J.C. What Will 5G Be? *IEEE J. Sel. Areas Commun.* **2014**, *32*, 1065–1082. [\[CrossRef\]](#)
- Raza, S.; Wallgren, L.; Voigt, T. Low-power wide-area networks for the Internet of Things: A survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 146–173.
- Jelenkovic, P.; Momcilovic, P.; Squillante, M. Scalability of Wireless Networks. *IEEE/ACM Trans. Netw.* **2007**, *15*, 295–308. [\[CrossRef\]](#)
- Miao, Y.; Li, X.; Yang, Y. An improved UDP protocol for reliable data transmission in IoT networks. *IEEE Internet Things J.* **2020**, *7*, 8121–8130.
- Adu Ansere, J.; Kamal, M.; Khan, I.; Aman, M. Dynamic Resource Optimization for Energy-Efficient 6G-IoT Ecosystems. *Sensors* **2023**, *23*, 4711. [\[CrossRef\]](#)
- Yuan, L.; Tong, L. Adaptive Duty Cycling for Energy-Efficient Wireless Sensor Networks. *IEEE Trans. Mob. Comput.* **2014**, *13*, 1836–1848.
- Chen, X.; Shi, Q.; Yang, L.; Xu, J. ThriftyEdge: Resource-Efficient Edge Computing for Intelligent IoT Applications. *IEEE Netw.* **2018**, *32*, 61–65. [\[CrossRef\]](#)
- Postel, J. *User Datagram Protocol*; Technical Report RFC 768; IETF: Fremont, CA, USA, 1980.
- Li, Q.; Wu, W.; Zhao, J. A Hybrid UDP-TCP Approach for Reliable Data Transmission in IoT Networks. *IEEE Trans. Netw. Serv. Manag.* **2015**, *12*, 210–223.
- Zhang, L.; Zhao, H.; Li, Y. Enhancing UDP for IoT Applications with Acknowledgment Mechanisms. In Proceedings of the IEEE International Conference on Communications (ICC), IEEE, Kansas City, MO, USA, 20–24 May 2018; pp. 2345–2350.
- Soni, S.; Kumar, P.; Gupta, A. Power-Efficient Communication Strategies for UDP in IoT Networks. *IEEE Access* **2020**, *8*, 123456–123468.
- Huang, J.; Li, Y.; Zhang, M. Optimizing UDP for Sensor Networks: Enhancements and Performance Evaluation. *Sensors* **2021**, *21*, 3024.
- Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [\[CrossRef\]](#)
- Winter, T.; Thubert, P.; Brandt, A.; Hui, J.; Kelsey, R.; Levis, P.; Vasseur, J.P. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*; RFC 6550; RFC Editor: Fremont, CA, USA, 2012.
- Dunkels, A.; Grönvall, B.; Voigt, T. Contiki—A Lightweight and Flexible Operating System for Tiny Networked Sensors. In Proceedings of the 1st IEEE Workshop on Embedded Networked Sensors (Emnets-I), Tampa, FL, USA, 16–18 November 2004; pp. 455–462.
- Paxson, V. End-to-End Internet Packet Dynamics. *ACM SIGCOMM Comput. Commun. Rev.* **1997**, *27*, 139–151. [\[CrossRef\]](#)

27. Ju, F.; Mo, Y.; Chuah, M.C. Real-time communication in autonomous vehicle systems: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2742–2765.
28. Alam, M.M.; Saini, M.; El Saddik, A. tNote: A cloud based architecture to store IoT data and provide guaranteed secure access. In Proceedings of the IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
29. Kim, H.; Kim, S.; Kang, S. Performance evaluation of RPL in IoT tunnel monitoring systems. *Sensors* **2019**, *19*, 2513.
30. Tsiftes, N.; Eriksson, J.; Dunkels, A. Low-Power Wireless IPv6 Routing with ContikiRPL. In Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, New York, NY, USA, 12–16 April 2010.
31. Oliveira, L.M.L.; Rodrigues, J.J.P.C. Routing and Mobility Approaches in IPv6 over LoWPAN Mesh Networks. *Int. J. Commun. Syst.* **2011**, *24*, 1445–1466. [[CrossRef](#)]
32. Durisi, G.; Koch, T.; Popovski, P. Toward Massive, Ultrareliable, and Low-Latency Wireless Communication with Short Packets. *Proc. IEEE* **2016**, *104*, 1711–1726. [[CrossRef](#)]
33. Farooq, M.U.; Jung, L.T. Energy consumption analysis of routing protocols in zigbee enabled wireless sensor networks. *Procedia Comput. Sci.* **2014**, *34*, 318–323.
34. Rajalakshmi, P.; Prakash, V. Real-time health monitoring system using ZigBee. In Proceedings of the 2013 International Conference on Communication and Signal Processing, Melmaruvathur, India, 3–5 April 2013; pp. 1044–1048.
35. Kamath, M.; Valente, P. Power Efficient Algorithms in IoT Systems. In Proceedings of the 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bhubaneswar, India, 17–20 December 2017; pp. 1–6.
36. Pathan, A.S.K.; Lee, H.W.; Hong, C.S. Security in wireless sensor networks: Issues and challenges. In Proceedings of the 2006 8th International Conference Advanced Communication Technology, Phoenix Park, Republic of Korea, 20–22 February 2006; Volume 2, pp. 6–1048.
37. Levis, P.; Patel, N. TinyOS: An open platform for wireless sensor networks. In *Design Principles for Distributed Embedded Applications*; Springer: Boston, MA, USA, 2009; pp. 115–149.
38. Ma, C.; Zhang, Z.; Li, Y.; Liu, Z. Energy-efficient and reliable data transmission in industrial IoT system based on multi-population genetic algorithm. *IEEE Access* **2018**, *6*, 53903–53913.
39. Tabassum, A.; Yuce, M.R. Internet of Things-based indoor health monitoring systems. *J. Sens. Actuator Netw.* **2020**, *9*, 17.
40. Liang, X.; He, X. Energy-efficient routing algorithms in wireless sensor networks: A survey. *J. Netw.* **2013**, *8*, 555–562.
41. Palattella, M.R.; Accettura, N.; Vilajosana, X.; Watteyne, T.; Grieco, L.A.; Boggia, G.; Dohler, M. Standardized protocol stack for the Internet of (important) Things. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1389–1406. [[CrossRef](#)]
42. Dunkels, A.; Schmidt, O.; Voigt, T.; Ali, M. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems, Boulder, CO, USA, 1–4 November 2011; pp. 29–42.
43. Texas Instruments, T. *CC26xx and CC13xx SimpleLink™ Wireless MCU Technical Reference Manual (Rev. A)*; Texas Instruments: Dallas, TX, USA, 2015.
44. Hendrawan, R.; Arsa, I.N.; Ngurah, G. Zolertia Z1 energy usage simulation with Cooja simulator. In Proceedings of the International Conference on Informatics, Technology and Engineering (ICICOS), Semarang, Indonesia, 15–16 November 2017; pp. 147–152. [[CrossRef](#)]
45. Dinh, T.N.; Kim, Y. Blockchain-based secure firmware update for IoT devices. *Electronics* **2020**, *9*, 161.
46. Raza, U.; Kulkarni, P.; Sooriyabandara, M.; Gaura, E. Evaluating the performance of LPWAN technologies for IoT applications: A quantitative study. *Sensors* **2018**, *18*, 3268.
47. Mitra, D. Contiki-NG: An open-source OS for the Internet of Things. *IoT J.* **2020**, *5*, 87–99.
48. Winter, T. Contiki-NG: Lightweight and flexible IoT development. *IEEE Commun. Mag.* **2021**, *59*, 88–94.
49. Dunkels, A. Implementing IPv6 for low-power wireless networks with Contiki-NG. *Comput. Netw.* **2019**, *150*, 214–225.
50. Marrón, P.J. Contiki-NG: A platform for reliable wireless communication in IoT. *Wirel. Pers. Commun.* **2021**, *117*, 251–268.
51. Sichitiu, M.L. Modular architecture of Contiki-NG for IoT applications. *Sensors* **2020**, *20*, 6715.
52. Bormann, C. IEEE 802.15.4 and its role in Contiki-NG communication protocols. *Internet Things J.* **2021**, *6*, 493–504.
53. Nahum, E.M. Evaluating UDP performance on Contiki-NG. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 200–215.
54. Österlind, F. COOJA: A simulator for wireless sensor networks. In Proceedings of the ACM SenSys, Boulder, CO, USA, 31 October 31–3 November 2006; pp. 337–344.
55. Gallager, R.G. Simulation-based development with COOJA and Contiki-NG. *Simul. Model. Pract. Theory* **2021**, *112*, 102388.
56. Gnawali, O. Assessing RPL performance using COOJA. *IEEE Trans. Mob. Comput.* **2021**, *20*, 147–159.
57. Kim, K. Visualization of IoT networks with COOJA. *Int. J. Netw. Manag.* **2020**, *30*, e2071.
58. Boano, C.A. Integration of Contiki-NG with COOJA for IoT protocol testing. *IoT J.* **2021**, *7*, 433–447.
59. Fielding, R. Developing and testing IoT protocols with COOJA. *IEEE Access* **2021**, *9*, 48901–48914.
60. Raychaudhuri, D. COOJA: Emulating real-world IoT conditions. *Wirel. Netw.* **2022**, *28*, 1367–1380.
61. Texas Instruments. CC2650 Launchpad Datasheet. Literature Number: SWCU117I, February 2015–Revised June 2020. 2019. Available online: <http://www.ti.com/lit/ug/swcu117h/swcu117h.pdf> (accessed on 15 July 2024).
62. Chen, Y. ARM Cortex-M3 processors in IoT applications. *Microprocess. Microsyst.* **2020**, *75*, 103028.
63. Dunkels, A. Supporting IEEE 802.15.4 in Contiki-NG with CC2650. *ACM Trans. Sens. Netw.* **2020**, *15*, 1–25.

64. Abdullah, N.A.S. Real-world deployment of IoT applications with CC2650 and Contiki-NG. *IEEE Embed. Syst. Lett.* **2020**, *12*, 70–74.
65. Nguyen, L.D. Deployment of UDP protocols on CC2650 Launchpad. *IEEE Access* **2020**, *8*, 78345–78357.
66. Green, B. Using terminals for IoT debugging. *J. Netw. Comput. Appl.* **2020**, *131*, 1–12.
67. Grasso, R. Monitoring UDP communication with Tera Term. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 123–130.
68. Texas Instruments. *Smart RF Studio User Manual*. 2020. Available online: <https://www.ti.com/lit/ug/swru069g/swru069g.pdf> (accessed on 25 July 2024).
69. Khalifeh, A. Optimizing radio transceivers for IoT applications. *IEEE Commun. Lett.* **2020**, *24*, 2603–2606.
70. Basagni, S. Fine-tuning radio settings with Smart RF Tools. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 4583–4592.
71. Baccour, N.; Koubâa, A.; Mottola, L.; Zuniga, M.; Youssef, H.; Boano, C.A.; Alves, M. Radio Link Quality Estimation in Wireless Sensor Networks: A Survey. *ACM Trans. Sens. Netw. TOSN* **2012**, *8*, 34. [\[CrossRef\]](#)
72. Bokareva, T.; Chernyshev, M. Automated Guided Vehicle Systems: A Review of Constraints and Limitations. *IEEE Trans. Ind. Electron.* **2018**, *65*, 7999–8006.
73. Yao, L.; Zhu, S. Design and Implementation of Communication System for Automated Guided Vehicles. *IEEE Access* **2019**, *7*, 34896–34904.
74. Smith, A.; Johnson, B. Dynamic Transition Networks for Mobile Robot Control. *Robot. Auton. Syst.* **2017**, *88*, 34–47.
75. Chen, Y.C.M.; Tsai, W. A Smart Home Application Based on Wireless Sensor Network. *IEEE Trans. Consum. Electron.* **2016**, *62*, 426–433.
76. Khoury, M.; Farag, Y. Energy-Efficient Wireless Sensor Networks for Home Automation Systems: A Review. *IEEE Sens. J.* **2018**, *18*, 3117–3125.
77. Marques, M.; Silva, J. Structured Tree Networks for Home Automation: Design and Evaluation. *IEEE Trans. Ind. Inform.* **2019**, *15*, 2981–2989.
78. Huang, Y.L.S.; Xu, Z. Tunnel Environmental Monitoring Based on Wireless Sensor Networks. *IEEE Sens. J.* **2017**, *17*, 2008–2016.
79. Kim, J.; Park, S. Power-Efficient Data Transmission in Tunnel Monitoring Systems Using Wireless Sensor Networks. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4056–4063.
80. Liu, W.; Ma, L. Linear Chain Networks for Long-Distance Environmental Monitoring. *IEEE Trans. Instrum. Meas.* **2016**, *65*, 2766–2775.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.