USER-CENTRIC SECURE SERVICE PROVISIONING WITH SUSTAINABILITY IN RESOURCE CONSTRAINED PLATFORMS USING PREDICTIVE MACHINE LEARNING MODELS

by

DAMILOLA ADEDAMOLA ALAO

A dissertation submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND INFORMATICS

2025

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Amartya Sen, Ph.D., Chair
Sunny Raj, Ph.D.
Lanyu Xu, Ph.D.
Eralda Caushaj, Ph.D.

To all who design and leverage technology for safe, meaningful, life-enhancing impact. May your innovations lead to tangible improvements and empower communities. And especially to those who persist in pursuing their dreams, undeterred by setbacks. May your resilience inspire continued progress and future generations.

ACKNOWLEDGMENTS

This dissertation's completion reflects years of dedication and growth. It would not have been possible without the profound support and guidance from many individuals and institutions. My deepest gratitude extends to all who contributed.

My sincerest appreciation goes to my Academic Advisor and Dissertation Committee Chair, Dr. Amartya Sen. His unparalleled guidance, unwavering mentorship, and invaluable intellectual contributions were the bedrock of this research, profoundly shaping this dissertation and my development as a researcher. Immense gratitude goes to my Dissertation Committee: Dr. Sunny Raj, Dr. Lanyu Xu, and Dr. Eralda Caushaj. Their meticulous feedback and constructive criticism, from conceptualization to defense, were instrumental in refining this work.

I also appreciate other faculty for their invaluable support, including Dr. Guatam Singh for insightful advice, and Dr. Olawoyin (NSBE OU Chapter staff advisor) for guidance and fostering community. Oakland University, its faculty, and departments provided an exceptional academic environment and resources. Special gratitude is extended to the International Students and Scholars Office (ISSO), particularly David Archbold, Petra Knoche, and Lindy Hancock, whose unwavering support during a rough period as an international student was truly life-sustaining. Profound thanks for the essential financial support from the Graduate School and Department via Teaching and Research Assistantships, which were critical for my research. Sincere thanks to the external faculty collaborator, Dr. Kenneth Fletcher, for their contributions. I am also grateful to my REU mentees, Saman Bhat and Emily Hamrick, and international undergraduate collaborator, Esmay Azam Rochy, for their enriching collaborative efforts.

This journey was impossible without my family's steadfast love and support. To my beloved parents, Dr. Akinkunmi Adegbola Alao and Dr. Ireti Folasade Alao, your enduring love, prayers, financial sacrifices, and belief were my constant strength. Thanks to my sister, Dr. Tinuola Olorunsogbon, and her husband, Barrister Olutayo Olorunsogbon, for their continuous encouragement. My nephew, Obal'Olowa Olorunsogbon, brought immense joy with his smiles.

Eternal gratitude for the invaluable emotional support from my friends: Dr. Francisca Opoku-Boateng for constant encouragement; Lovelyn Epelle, Eucharia Ganda, Promice Mosely, Ayomide and Oluwakemi Yusuf, Esosa Ekhoragbon, Dotun Ojelabi, Tobi Adebisi, Folajimi Afolabi, Olugbadebo, Morayo Adenuga, Mary Ishaiku, Haripriya Sameer, Toye and Oyindamola Onabiyi, Frank Osei, and Afeez Abdulrasak, for their listening ears, motivation, and understanding during demanding times. My research group members and housemates, Oluwafeyisayo Oyeniyi and Victorine Wakam Younang, provided a supportive environment, offering intellectual camaraderie and proofreading. Thanks to fellow graduate students, Dr. Dafer Alali, Dr. Obieda Ananbeh, and Meena Nagabhushana, for shared experiences and encouragement.

I acknowledge the profound impact of my non-academic mentors, Dr. Ernest and Dr. Pat Emenyonu, for their incredible generosity and for embracing me as a daughter. Dr. Murali Mani for kindling my research interest. Dr. Charlotte Tang for inspiring me toward female research in AI/Machine Learning. Dr. Amal Alhosban for suggesting Oakland University for my Ph.D., a pivotal recommendation.

Finally, to those who contributed to my well-being: Pa. and Ma. Awoyele (my grandparents), Mrs. Funmilayo Ogungbemi, The Oluwatosin and Ife Awoyele families, Dr. Olufunke Ezekiel, Dr. Olabisi Loto (Late), Dr. Mrs. Anthonia Loto, Dr. Seun Loto, Mrs. Bada, Mr. and Mrs. Akeredolu, Drs. Kofi Daniel and Sarah Opoku-Boateng, Mrs.

Oyeniyi, Mrs. Faderera Johnson, Abiola Opeyemi, Olabisi Okunbolade, Mrs. Mojisola Alimi (Nee Gbolade), Drs. Abiola and Fiyin Olaitan, Dr. Temitope Richard Banji, Mr. Babatunde Majolagbe, Dr. Babasinmisola Fadirepo, Mr. Dipo Kolawole, and Mr. Mayowa Onokoya. Your kindness and support provided invaluable solace and strength.

To all mentioned, and to those whose support, though unlisted, was deeply felt, thank you. This achievement is as much yours as it is mine.

Damilola Adedamola Alao

ABSTRACT

USER-CENTRIC SECURE SERVICE PROVISIONING WITH SUSTAINABILITY IN RESOURCE CONSTRAINED PLATFORMS USING PREDICTIVE MACHINE LEARNING MODELS

by

Damilola Adedamola Alao

Adviser: Amartya Sen, Ph.D.

This dissertation addresses the challenges of resource constraints and security vulnerabilities inherent in the increasing deployment of IoT devices and services. It embarks on a journey to optimize IoT service provisioning by recognizing the dynamic nature of user preferences and the need for robust security measures compatible with the limited computational, energy, and storage capacities of IoT devices. This foundational understanding leads to the initial contribution: a novel IoT simulation framework. This framework uniquely integrates lightweight cryptography - NtruEncrypt, within a user-centric service provisioning model, simulated using CupCarbon. The data collected from this framework validates the efficacy of lightweight security in enhancing Quality of Service (QoS) and also highlights the need for sophisticated resource management. Insights from the initial simulation inform subsequent works, driving the research towards comprehensive IoT optimization. Recognizing the urgency for improved resource management, the dissertation presents a two-phase IoT resource management optimization framework. This solution leverages predictive analytics of user service requests combined with a service recommendation methodology to dynamically reallocate IoT resources, thereby minimizing service delays and optimizing device efficiency.

Focusing on sustained device operation, particularly in the context of emerging energy harvesting solutions, the work further develops an enhanced energy management framework. This framework dynamically optimizes the power distribution of harvested radio Frequency (RF) energy using a multi-objective optimization algorithm (NSGA-III) and a custom energy allocation algorithm, ensuring continuous service availability despite intermittent energy supplies. The final contribution is a secure energy scheduling framework for Energy Harvesting (EH)-enabled Sensor Cloud platforms. Directly countering energy depletion attacks by implementing dynamic, user-centric secure admission control, projecting each request's net energy impact to block malicious requests before they can compromise the energy integrity of IoT devices. These contributions form a coherent and robust solution for dynamic, user-centric, and secure IoT service provisioning, resource management, and energy sustainability.

TABLE OF CONTENTS

TABLE OF CONTENTS—Continued

TABLE OF CONTENTS—Continued

TABLE OF CONTENTS—Continued

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES—Continued

CHAPTER ONE

INTRODUCTION

The network of physical objects or "things", embedded with sensors and software to connect and exchange data with other devices and systems over the Internet, are referred to as Internet of Things (IoT) devices [1]. The Internet of Things (IoT) environment has helped to expand the connection between the physical and the digital world. Raw physical data are converted into digital signals and transmitted by specialized devices within the IoT environment known as actuators and sensors. These transmissions enable wireless control and equipment monitoring [1] [2]. Consequently, IoT has become widely adopted in several aspects of daily life and industrial sectors.

Fig. 1.1 shows IoT applications in different fields, including Healthcare, Smart Cities, Industry 4.0, Smart Agriculture, Remote Monitoring, and Logistics [3] [4], creating an increasing dependence on IoT devices in modern applications. In the following sections, we discuss concepts and issues around IoT service provisioning, including user-centric provisioning, security, service simulation, resource management, energy harvesting, and enhanced energy allocation strategies for IoT service provisioning.

1.1 IoT Service Provisioning

Service provisioning is the process of configuring, connecting, and making IoT devices available to support effective communication and data sharing within an IoT network to perform the services that they have been designed for [5]. To improve user experiences and ensure smooth interactions with applications on IoT devices, it is crucial to understand how interconnected devices work. Since most IoT devices have to transfer data over the internet without data encryption, security is also an issue in IoT service provisioning. In addition,

[1] www.oracle.com/in/internet-of-things/ what-is-iot/

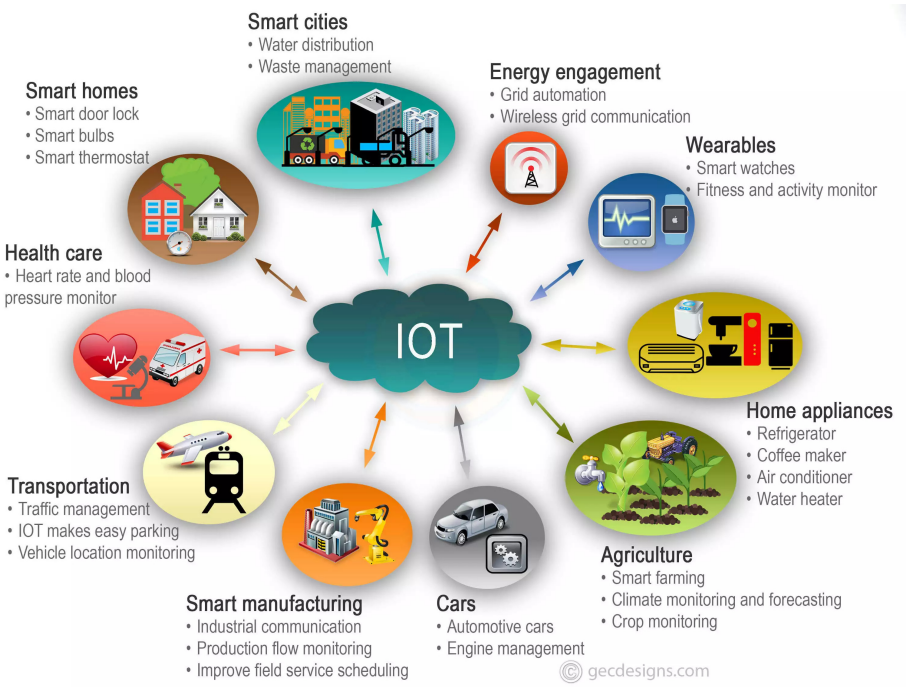

Figure 1.1: IoT Applications and Services

knowing that the preferences of users who subscribe to services on applications supported by IoT devices change over time, the interactions between IoT devices become complex.

1.1.1 Secure and User-Centric Service Provisioning

Security vulnerabilities that affect service provisioning include inadequate device security, vulnerabilities to network attacks, unsecured data transmission, and lack of encryption. User-centric service provisioning currently faces the challenge of considering security provisioning for different users as a one-size-fits-all paradigm. In reality, users' preferences on parameters like QoS and security are dynamic and change over time, with the strength of service security inversely affecting QoS. Dynamic, user-centric, and secure service provisioning frameworks allow users to express varying QoS and security preferences when selecting IoT devices and services to support their applications [6]. The

satisfaction of a combination of personalized user preferences on functional, non-functional Quality of Service (QoS), and security requirements during service provisioning is referred to as secure user-centric service provisioning.

1.1.2 Simulating Secure and Dynamic User-centric Service Provisioning

Traditional security and encryption measures are not feasible with IoT devices as they require high computational abilities and energy usage. Specific security protocols, referred to as Lightweight cryptography protocols, have been developed to overcome these issues [7] [8] [9] [10]. Given the adaptive nature of these protocols and how costly their deployment is, extensive testing is required before they are deployed in the real world. IoT simulation tools are designed to enable the modeling and testing of IoT systems and applications, with use case scenarios, before deployment in target environments [11]. IoT simulation tools also test the deployment performance of IoT applications by determining and comparing key performance indices. A bottleneck in developing IoT simulation tools is that existing frameworks treat service security as a static parameter during IoT service provisioning. This negatively impacts performance, as the strength of service security is inversely related to service QoS parameters. Additionally, for user-centric service provisioning, one cannot assume that each user's security requirements will remain static. There is, therefore, limited discussion on how existing simulation tools address the issue of lightweight security affecting service performance in the use case for a user-centric IoT service provisioning.

1.2 IoT Resource Management

Another aspect of IoT service provisioning is resource management. The number of IoT devices in our environment is projected to increase to 75 billion devices in 2025 [12]. These devices facilitate seamless integration and automation across diverse sectors, including

healthcare [13], agriculture [14], transportation [15], Industry 4.0 [16], and smart cities [17]. They support applications such as home security systems, wearable healthcare monitors, and industrial automation devices [18], enabling efficient data exchange and real-time decision-making. However, this growth introduces significant challenges related to resource constraints. The term "resource constraints" refers to the finite and limited computational power, energy capacity, storage availability, and sensing capabilities of IoT devices due to their miniature and heterogeneous nature [19] [20]. Optimally providing user-centric services by resource-constrained IoT devices makes service provisioning challenging. The massive data generated —up to 50 trillion GB annually—further amplifies these constraints [21], necessitating efficient resource management to maintain Quality of Service (QoS) and user satisfaction. The challenge is maximizing the utilization of resource-constrained devices while ensuring a high QoS for each user, typically referred to as resource optimization. Existing IoT resource management techniques often fail to address evolving user needs, security concerns, and real-time QoS requirements during provisioning, particularly when historical data about users' interests is sparse or unavailable. Users are also sometimes unaware of, or unable to identify, the most beneficial IoT services by themselves. Therefore, user service QoS requirements are initially determined using predictive analysis for efficient service and device selection. Thereafter, personalized recommendations are made based on user behavior and preferences [22] to match services and devices to users for optimized resource management.

1.2.1 Service Prediction and Recommendation to Extend Device Network

For IoT service providers to efficiently manage and allocate devices suitable for users' service requests, IoT resource management frameworks using service prediction and recommendation can be employed. Knowing the future service demands of a user aids the

service provider in making informed recommendations about devices capable of providing those services and ensuring their availability. Traditional prediction and recommendation systems employ several techniques to select services and devices [23–25], and match services and devices to users [26–28]. Two traditional techniques are Collaborative Filtering (CF) and Content-based Filtering [22]. These techniques prove effective when sufficient historical information about services, devices, users, and behavior is available. Otherwise, they suffer from data sparsity, scalability, and cold-start problems [29]. Making accurate predictions and recommendations becomes difficult where necessary information is lacking or the decision-making data is constantly changing, as in the case of dynamic IoT-based recommendations [30]. To address this, ML-based prediction and recommendation techniques can be leveraged to create an optimized resource management framework that increases the efficiency of constrained IoT resources.

1.3 Energy Harvesting and Resource Management

The association of resource-constrained IoT devices with energy consumption and efficiency issues [31] is a limiting factor in their continuous functionality. This is because IoT devices mainly rely on batteries to power their operations. These finite energy sources require that they be recharged or replaced often, making frequent maintenance impractical or costly in the case of remote or large-scale deployments. Since the longevity of a device is directly dependent on the battery capacity, regular energy loss negatively impacts the lifespan and performance of the IoT network and service provisioning. To solve the energy constraint challenge, Energy Harvesting (EH) has been widely considered as a practical approach to power management for IoT service provisioning. EH involves the conversion of ambient or naturally existing energy sources such as solar, vibration, Radio Frequency

(RF), and Thermal energy into electric energy to extend the lifetime of IoT devices. EH is considered an environmentally friendly power solution for IoT [32] [33].

1.3.1 Integrating Energy Allocation Strategy with RF Energy Harvesting

Although ambient energy naturally exists, other factors such as low power density, complexities within the environment, and intermittent supply could cause variations in its availability. This makes the efficiency and control of the harvesting process across various harvesting technologies difficult. To improve energy Harvesting, some strategies to enhance the generation, allocation, and use of ambient energy in IoT service provisioning have been developed [34]. The methods include adaptive power management, where energy consumption is dynamically regulated based on its availability in real-time to extend device lifespan and improve the reliability of IoT networks. Optimization of power distribution and energy consumption patterns has also been achieved by integrating energy harvesting systems with machine learning techniques. Intelligent energy management systems leverage machine learning techniques by analyzing the complexity of the network dynamics and predicting energy availability to improve efficient power allocation [35].

1.3.2 Energy Harvesting and Energy Security

Energy security is quickly becoming as important as securing user information when energy harvesting methods are deployed to power IoT devices during service provisioning. Additionally, with user-centric implementations, where user information is needed for efficient energy allocation and distribution, the attack surface for energy attacks widens [36]. Such attacks could lead to energy starvation and denial of service [37], creating the need to implement energy security methods to prevent energy attacks during service provisioning with energy harvesting.

1.4 Shortcomings of IoT Service Provisioning

A fully user-centric IoT service provisioning framework should accommodate users' personalized and dynamic preferences on functional (region of interest, service duration, and sensing frequency) and non-functional (QoS and security) parameters. In addition, more effective IoT simulation tools should be developed to ensure safer, scalable, secure user-centric IoT solutions. Furthermore, the existing methods used in the prediction of user services and the recommendation of devices often cannot accommodate the dynamic and user-specific QoS demands and simultaneously prioritize resource allocation for real-time user requests, creating a gap in the efficient management of IoT resources. Although

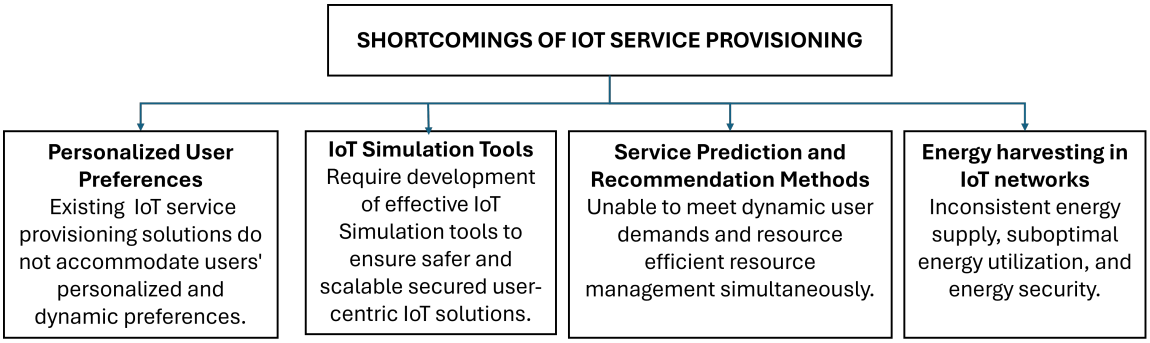

Figure 1.2: Shortcomings in various aspects of IoT service provisioning

Energy harvesting in IoT networks is proposed as a promising solution to enable IoT devices to operate autonomously, existing adaptive power management and intelligent energy management methods still struggle with inconsistent energy supply, inefficient resource allocation, and suboptimal energy utilization. Minimal research has also explored mitigating the energy security issues that could arise during the allocation and distribution of harvested energy. This research aims to fill the gaps in IoT service provisioning presented in 1.2 with the research objectives discussed in the next section.

1.5 Research Objectives

Understanding a user's desires for personalized IoT service provisioning, coupled with the fact that requirements vary among users over time, is beneficial to making service provisioning user-centric. Designing IoT simulation tools prioritizing users' service provisioning QoS and security preferences will help increase trust in the provisioned services. As more users adopt IoT-enabled applications and demand real-time services with dynamic QoS preferences, the need for the available IoT devices to support those applications and provide such services increases. This means devices must support more applications and provide diverse services [38]. As such, the objectives for this research are outlined as follows:

1.5.1 RO1: IoT Simulator Tool for Provisioning Secure and Dynamic User-centric Services

IoT simulation tools are developed to simulate implementations of protocols specific to IoT environment. To keep these implementations secure, lightweight security (encryption and decryption) protocols have been implemented. However, these simulation tools hardly address how implementing lightweight security protocols affects service performance in user-centric IoT service provisioning. To address the shortcomings of existing simulation tools, this research proposes the design of a simulation tool that takes the changing security preferences of users into account during service provisioning.

1.5.2 RO2: Resource Management with Service Prediction and Recommendation

Efficient resource management involves discovery, selection, provisioning, and monitoring to meet user-specific needs. Complexities in discovery and selection increase with the scale and heterogeneity of IoT devices and services [39]. Creating an urgency for improved IoT resource management based on users' service preferences to improve

their experience. This research is driven by the need to anticipate user demands and dynamically reallocate IoT resources, thereby minimizing service delays and optimizing device efficiency.

1.5.3 RO3: Enhanced Energy Management with Energy Harvesting

Intermittent energy supply remains a challenge in IoT service provisioning. Energy Harvesting (EH) sources such as solar or Radio Frequency (RF) energy are inherently unpredictable and variable, making it difficult to maintain consistent power levels in IoT devices. Additionally, efficiency limitations hinder the widespread adoption of energy harvesting methods, as the conversion efficiency of EH technologies remains relatively low, often failing to meet the continuous power demands of specific IoT applications. This research explores the development of an energy allocation algorithm to effectively balance energy harvesting, storage, and consumption in response to fluctuating service demands that arise.

1.5.4 RO4: Integrating Energy Harvesting Management with Energy Security Techniques

Energy harvesting for IoT has mainly focused on the optimization of energy efficiency, the improvement of service quality, and extending network lifetime. Neglecting the aspect of energy security opens up energy management to potential threats and vulnerabilities. Safeguarding the energy harvesting and distribution process from attacks such as eavesdropping, wiretapping, and denial of energy attacks is the future focus of this research.

CHAPTER TWO

BACKGROUND

Prior studies have explored resource tracking and scheduling as a solution to IoT resource optimization [40–42]. To deal with the heterogeneity and dynamic nature of IoT resources, resource modeling and allocation were proposed to improve the management of IoT resources [43–48]. These solutions are not globally optimal as they fall short of addressing the issues of real-time processing or the dynamic nature of user QoS requirements [43]. Such problems are addressed by applying blockchain technology, Deep Reinforcement Learning, and Machine Learning methods for real-time resource optimization [21, 49–52]. Additionally, users are sometimes unaware of, or unable to identify, the most beneficial IoT services. Therefore, user service QoS requirements are initially determined using predictive analysis for efficient service and device selection. Thereafter, personalized recommendations are made based on user behavior and preferences [22] to match services and devices to users for optimized resource management.

2.1 Existing IoT Simulation Tools

Understanding a user's desires for personalized IoT service provisioning, coupled with the fact that requirements vary among users over time, is beneficial to making service provisioning user-centric. This means giving users power over their service requests by ensuring that the user can specify their dynamic and variable preferences on service input parameters. A fully user-centric IoT service provisioning framework should accommodate users' personalized and dynamic preferences for functional (region of interest, service duration, and sensing frequency) and non-functional (QoS and security) parameters. We proposed such a framework in our previous work [6], where we designed an algorithm,

developed in Java, to simulate service requests and the provisioning environment. Taking as input both user functional and non-functional requirements, including varying QoS and security user preferences. The algorithm utilizes existing NSGA-II (Non-Dominant Sorting Genetic Algorithm II) for service provisioning, to determine an optimal set of suitable IoT devices to fulfill user service requests. However, the previously proposed framework did not incorporate the simulation of real-life networking or security protocols.

2.2 Existing Prediction and Recommendation Systems

The two main traditional techniques, Collaborative Filtering (CF) and Content-based Filtering, are effective when sufficient historical data on services, devices, and user behavior is available [22]. However, they face significant challenges such as data sparsity, scalability issues, and the cold-start problem, particularly when information is incomplete or rapidly changing [29]. These limitations make accurate predictions difficult in dynamic IoT environments, highlighting the need for adaptive frameworks like ours to address such challenges [30]. Machine Learning(ML)-based techniques like Deep Neural Networks [53] and Reinforcement Learning [54] overcome these shortcomings by making logical inferences about user-desired services and predicting those services based solely on recognizing patterns within data sets with little input from users [30, 53]. With ML-based techniques, recommendation systems also learn from the represented data using multiple processing layers of the computational model through several levels of abstraction. The model speed and accuracy that can be achieved using ML-based techniques make them ideal for dynamic IoT-based predictions and recommendations [55] [53].

2.3 Existing Energy Harvesting and Management Systems

Although ambient energy naturally exists, other factors such as low power density, complexities within the environment, and intermittent supply could cause fluctuations in their availability, making the efficiency and control of the harvesting process across various harvesting technologies difficult. Some research has therefore gone into the development of strategies to enhance the generation, allocation, and use of ambient energy in IoT service provisioning [34]. A form of enhancement strategy is referred to as adaptive power management, where energy consumption is dynamically regulated based on its availability in real-time to extend device lifespan and improve the reliability of IoT networks. Optimization of power distribution and energy consumption patterns has also been achieved by integrating energy harvesting systems with machine learning techniques. Intelligent energy management systems leverage machine learning techniques by analyzing the complexity of the network dynamics and predicting energy availability to improve efficient power allocation [35]. To enhance network performance, security, and network sustainability in Wireless Sensor Networks (WSN), energy-efficient methods have been used [56]. The development of algorithms that combine intelligent and adaptive power management approaches can provide some benefits for IoT Networks by becoming self-sufficient and extending their operational lifetime as a result of enhanced energy harvesting processes.

2.4 Existing Energy Harvesting Management with Energy Security Systems

The integration of EH with IoT devices has introduced additional attack vectors that an attacker can exploit for energy-based attacks, which compromise the quality of service (QoS), reliability, and lifetime of the IoT devices. A significant form of these attacks is the Energy Depletion Attack (EDA), also referred to as sleep deprivation or vampire attack [57].

Examples of attacks that can be categorized as EDA attacks include flooding [58], jamming [59], and spoofing [60] attacks. Therefore, there is a need for a robust energy security framework that protects EH-powered IoT devices from EDA, while ensuring the user's QoS is not compromised. Device-centric approaches or detection schemes cannot entirely prevent persistent attacks. Therefore, there is a need for a security framework that integrates the request behavior of the user, alongside the profile of the device. The development of a secure energy scheduling framework for EH-assisted Sensor Cloud platforms that enables dynamic, user-centric secure admission control for IoT devices is a viable solution to the energy security problem.

CHAPTER THREE

LITERATURE REVIEW

The existing literature is categorized and discussed through the following domains: IoT simulation tools, resource optimization, service prediction, and service recommendation in IoT regarding IoT resource management with prediction and recommendation. The latter part of the literature review covers domains such as energy harvesting in IoT, RF Energy Harvesting with Wireless Power Transfers (WPT) in IoT, and adaptive and intelligent RF energy management in IoT regarding enhanced energy management with energy harvesting. The following sub-sections review research efforts in these domains that are important for designing and developing the proposed optimized resource management framework for IoT applications and an enhanced energy management framework for IoT service provisioning.

3.1 IoT Simulation Tools and Lightweight Cryptography Protocols

IoT simulation tools, unlike traditional tools, have to handle the heterogeneity of devices, scalability, and realistic IoT parameters [11]. Lightweight cryptography protocols, in addition, have to be adaptable to the dynamic nature of users' requests and network demands. In this section, we discuss various IoT simulator tools and Lightweight cryptographic protocols that have been developed for the IoT environment.

3.1.1 IoT Simulation Tools

The designers of IoTSim [61] sought to reduce the time and cost of simulating how a large amount of data generated in batch-oriented IoT-based applications is processed and analyzed. Built atop a cloud computing simulation software - CloudSim, IoTSim supports researchers and organizations alike with the ability to study how IoT-based applications perform, and

the impact they make in the industry, allowing the efficient simulation of the scalability of resources for computing in IoT applications. In [62], a simulation tool for the OASIS, a standard Devices Profile for Web Services (DPWS) was introduced, named DPWSim. This simulation toolkit supports the development of service-oriented and event-driven IoT applications on IoT devices for web services. DPWSim also allows for collaboration among designers, developers, and manufacturers of web service IoT applications built on the DPWS technology, even in the absence of physical IoT devices. Written in Java with a user-friendly GUI, DPWSim is platform independent, provides flexibility in defining new DPWS devices, can generate virtual devices from physical DPWS devices, and simulates environments where DPWS devices reside [62]. To provide an evaluation platform to quantify the performance of policies to tackle critical problems in IoT and resource management, the authors in [63] propose iFogSim, an IoT simulator. iFogSim models IoT and Fog environments, measuring latency, network congestion, energy consumption, and cost, to determine the impacts of resource management techniques on these parameters. Introduced as a cross-level wireless sensor network simulator [64], COOJA is designed to enable simultaneous simulation at network, operating system, or machine code instruction set system development levels. Before COOJA, which was built on the Contiki sensor node operating system, developers were unable to use the same simulator for more than one system level at a time. presented the initial attempt to simulate a Wireless Sensor Network (WSN) citeCupCarbon. Their tool - CupCarbon - is a multi-agent and discrete event Wireless Sensor Network (WSN) simulator, designed to study the behavior of a network and its cost.

3.1.2 Lightweight Cryptography Protocols

The challenge of providing dynamic security with resource-constrained IoT devices has resulted in the creation of lightweight cryptography protocols. The term "lightweight" has been used too broadly, making it essential to distinguish them based on their different properties. The FELICS ("Fair Evaluation of Lightweight Cryptographic Systems") framework [65] extracts RAM consumption, code size, and throughput of lightweight block and stream ciphers across three widely used micro-controllers: 8-bit AVR, 16-bit MSP, and 32-bit ARM. According to [65], the framework aims to increase transparency and trust when it comes to benchmarking lightweight algorithms, given the lack of fair, comprehensive, and consistent testing frameworks currently available.

Similar to the FELICS framework, in 2017, the Japanese Cryptography Research and Evaluation Committees (CRYPTREC) published a comprehensive report on lightweight cryptographic protocols in an attempt to standardize the evaluation of such algorithms [66]. The report provides a comprehensive overview of lightweight cryptography, describing its different applications, typical use cases, and parameter selection. Chapter 3 of the report presents a performance review, thorough benchmarking, and evaluation of various network security protocols, including block ciphers and authenticated encryption schemes. The authors define four different performance metrics for lightweight cryptography [66]: circuit size measures power consumption, energy is gauged to measure the energy-saving capabilities of devices, especially battery-powered devices, latency is used as a metric for real-time performance, and finally, memory size to determine the size of the software implementation. Furthermore, the authors of [66] present many lightweight protocols based on three main criteria: (1) the protocol has been presented at a major academic conference, (2) no severe weaknesses of the protocol have been identified, (3) it can function efficiently or has proved helpful in resource-constrained environments. As a result, several

of the protocols we surveyed within our research were first observed in this paper, which gave thorough evaluations of the various popular and reliable lightweight network security protocols, such as Ascon, SPONGENT, Grain v1, and SipHash.

Authors in [67] evaluated more than 100 lightweight algorithms extracting information such as internal state size, key size, number of rounds, and block size. Their table on lightweight block ciphers provided information on the ciphers and a list of best-known attacks against each cipher. [67] also discusses high-level trade-offs in lightweight cryptography, specifically concerning *Performance vs. Security* and *Specialization vs. Versatility*. The former discusses how strong security and high performance are often at odds regarding lightweight network security protocols, especially in resource-constrained environments like IoT. For example, KTANTAN and QUARK are optimized for low gate count in hardware implementation. However, their implementation is strict in design, such that the protocols can only be used on particular platforms [67]. This is in contrast to the GIMLI sponge, which uses simpler operations, allowing its usability on virtually any platform. However, given its general nature, the protocol has a larger internal state to help it adjust to various platforms and perform different functions.

3.2 IoT Resource Management

The existing literature is categorized and discussed through the following domains: resource optimization, service prediction, and service recommendation in IoT. The following sub-sections review research efforts in these domains that are important for designing and developing the proposed optimized resource management framework for IoT applications.

3.2.1 Resource Optimization in IoT

Various researchers have addressed resource optimization challenges in IoT environments, focusing on resource management, energy consumption, service composition, and allocation. [68] avoids the waste of service resources and reduces energy consumption during service composition in dynamic IoT environments by proposing a particle swarm optimization algorithm to maintain the load balance of smart devices. The authors consider the energy consumption of IoT devices during service composition and the impact of energy consumption on mass services during dynamic provisioning. In [69], optimized resource composition for IoT service is done by adopting the concepts of Service-Oriented Architecture (SOA) and Service-Oriented Composition to tackle chain composition and deployment. Approaching the IoT Resource Allocation Problem (IRAP) from a scheduling perspective, a heuristic function-based solution, the Whale Optimization Algorithm (WOA), is implemented in [70] to reduce communication costs of IoT nodes and increase system efficiency. [71] minimized computing and energy costs by determining the optimal location of service functions at the edge and cloud layers. To improve resource sharing and allocation for application deployment, [72] proposed a decentralized resource allocation framework for deploying applications at the edge network. The framework focuses on IoT resource-sharing based on the application requirements, and factors in the resource preference of the participating nodes.

Shortcomings: These works deviate from considering QoS evaluation indicators that affect service composition optimization. By not considering the service function being performed, or user functional and non-functional preferences when calculating cost and determining optimal resource allocation, they miss out on critical factors and lack a comprehensive optimization approach for maximizing resource utilization, system performance, and user satisfaction.

3.2.2 Service Prediction in IoT

Efforts have gone toward progress in QoS prediction accuracy, privacy protection, user-centric connectivity, and time-sensitive service prioritization. This demonstrates the ongoing optimization of IoT service performance and user satisfaction. Using an alternative similarity computation mechanism that reduces the error in the predicted QoS value, [73] introduced IoTPredict, a neighborhood-based prediction approach for IoT services. Benefits include allowing users to receive QoS predictions of services that have not been invoked based on QoS values from other users and gathering QoS information without invoking services, thereby protecting the performance of the infrastructure to retrieve QoS values. [25] achieved Collaborative filtering-based QoS prediction and privacy preservation with a distributed edge QoS prediction model with privacy preservation.

In [23], contextual information from the user and service was used to improve QoS prediction accuracy based on a Fuzzy C-Means (FCM) approach. User and service contextual information are clustered. Neural Collaborative Filtering (NCF) is then used to learn cluster features from contextual information and deep latent features from historical QoS records for QoS prediction. [74] used data from the widely used IFTTT (if this, then that) platform, known for the self-automation of IoT services, to construct a network of IoT applications to implement graph embedding-based link prediction to identify possible connections of IoT applications, thereby highlighting the importance of user-driven design and data-driven insights in shaping the future of IoT applications. [75] introduces a Time-aware Service Ranking Prediction approach. The approach employs a pairwise comparison model to describe relationships between different services by applying time series forecasting on QoS values for Quality of Service-aware service selection in IoT systems.

Shortcomings: Collaborative Filtering techniques are applied in most of the works mentioned, making them prone to issues such as data sparsity and scalability.

3.2.3 Service Recommendation in IoT

Extensive research to advance the area of recommendation systems in IoT has leveraged graph-based models, security-driven approaches, time-awareness, and user-profile-based methods to enhance service recommendations and user experiences in IoT environments. Finding relevant services users are interested in based on objects they already own was the focus in [76]. An efficient IoT-based Recommender System (RS) requires information like service profiles and user device ownership, which were new challenges to Recommender Systems. [76] investigated and presented a tripartite graph-based model with weight spreading for IoT RSs that employ information of services and ambient resources for services and things recommendation. [77] proposed a security-driven hybrid collaborative recommendation method that deals with large-scale, cloud-based IoT services in a scalable and secure manner. This is in response to the security issues raised in the cloud-based IoT service RS that affect the accuracy of service recommendations.

The contribution in [26] uses time information to learn user preference behavior over time to develop a personalized recommendation system that considers the pattern of the user's preference with changing user interest over time. The model is based on a time correlation coefficient and an improved K-means with cuckoo search (CSK-means). [78] discusses challenges users face in selecting appropriate IoT services due to the increasing variety of services available. RSs tailored to individual user profiles and service interests are proposed. In [78], the authors present RSs for an IoT environment, utilizing user profiles and previous activities to create a binary decision tree for user classification. This work builds upon the time-awareness approach presented by [26], incorporating additional

contexts, such as location and historical behavior, along with time-based user patterns. A more comprehensive solution is designed by integrating the prediction and recommendation processes into a single framework. This comprehensive approach enhances resource management and allocation in IoT environments.

3.3 Enhanced Energy Management with Energy Harvesting

This section includes discussions about existing literature on energy harvesting in IoT, RF energy harvesting with wireless power transfer to support IoT operations, and the development of adaptive and intelligent energy management strategies in IoT. The efforts towards developing and efficiently managing a more reliable energy source, such as RF, to support sustainable operations in IoT, are a response to the energy issues identified as drawbacks to service continuity with resource-constrained, battery-dependent IoT devices.

3.3.1 RF Energy Harvesting In IoT

In [79], RF energy is harvested from any direction within a smart home by using a high-gain transmission antenna array alongside a circularly polarized array rectenna system.RF energy harvested is used to power the batteries used within the IoT system, thereby increasing the battery lifetime. The energy consumption is optimized by adopting the Zigbee communication protocol for the sensor nodes and using a low-power microcontroller to operate the sensors in lower-power mode. A significant factor affecting RF EH is the signal strength. [80] proposed a transmitter positioning scheme, using an RF-wave propagation model for signal strength prediction to improve modeling accuracy of the RF-waves and energy efficiency in IoT networks. A multiband ambient RF energy harvesting rectenna with high energy conversion efficiency for autonomous IoT devices is designed in [81]. The RF harvester is designed for use in various wireless sensor applications.

An RF energy harvesting scheme using multiple dedicated RF sources to avert energy holes is proposed in [82]. The study aimed to place energy transmitters in optimal positions and determine the optimal number of energy transmitters needed in multi-hop WSNs to avoid energy holes and enhance network lifetime. To prevent an IoT sensing device from completely depleting its energy when not attached to a battery charger, the work done in [83] introduces a mobile charging technique with RF energy harvesting. The mobile charger can determine an optimal location for charging the sensing device. The collection of research presented in [84] [33] [85] [86] all point towards the different approaches and techniques that have been implemented to improve RF energy harvesting. This work leverages the existing studies and developments that have been made in the area of RF energy harvesting for low-power IoT devices.

3.3.2 RF Energy Harvesting with Wireless Power Transfers (WPT) in IoT

Another factor impacting RF-EH is the distance of the harvesting device from the RF energy source. In the past, wireless technology has been used only for communication. To improve the harvesting capabilities of harvesting devices irrespective of distance from the source, some aspects of RF-EH research have been dedicated to developing methods for RF energy harvesting with wireless power transfer. This was the objective in [87], where the performance of an IoT network was improved using cooperative communication techniques with relay devices that harvest energy from RF signals. A closed-form expression for the average and variance of harvested RF energy over an arbitrary period is formulated in [88]. A power transfer plan to simultaneously distribute energy for the continuous operation of multiple low-power devices is derived. A similar concept is introduced to maximize network lifetime in [89], jointly implementing computation offloading operations with Simultaneous Wireless Information and Power Transfer (SWIPT).

The work done in [90] explores a more advanced solution to the efficiency issues encountered with WPT. The paper proposes a concept based on signal amplification for multi-hop(MH) wireless power transfer. Furthermore, [91] proposes an event-driven sensor network to recharge device batteries with RF-EH. The work employs a method that enables energy spent on data transmission to be used for battery charging and data decoding. The relationships between event signaling and detection and the system behavior were mainly considered. In addition, [92] [93] [94] have used RF EH combined with a Wireless Power Transfer (WPT) System to capture and convert RF energy into DC signals that are then further used to feed low-power digital devices.

Shortcomings: Existing WPT methods have approached energy transfer as stand-alone solutions apart from other aspects of energy harvesting that affect transmitters and receivers used in various harvesting techniques.

3.3.3 Adaptive and Intelligent RF Energy Management In IoT

Maintaining the efficiency of harvested energy and preventing energy loss by providing effective energy management solutions has proven to be the most challenging aspect of energy harvesting [95]. The authors in [96] proposed a combination of power extraction with an adaptive analog power management module (PMM) to increase piezoelectric energy harvesting. The proposed PMM powers a wireless sensor node (WSN) by being adaptive to variations in amplitude and vibration frequencies. The research in [97] presents a power management system to generate a power supply from RF energy harvesting in WSNs. The system features a DC-DC boost converter to recharge an energy reservoir and a linear regulator to scale down the output voltage for an integrated temperature sensor. To determine the optimum number of energy transmitters needed for RF energy harvesting in a multihop

WSN, [82] approached the challenge as an optimization problem that satisfies constraints on minimum energy charged by each sensor node based on Wireless Energy Transfer.

An IoT-based home energy management system model is presented in [98]. The study uses batteries and renewable energy sources to access an intelligent system by proposing an improved multi-objective optimization scheduling strategy- butterfly optimization algorithm, that considers satisfaction functions like energy consumption cost, and user satisfaction to improve the system efficiency. Other IoT-based frameworks for intelligent energy management applications that support intelligent analysis for monitoring, controlling, and enhancing system efficiency in smart buildings are analyzed in [99]. The study reveals that IoT-based frameworks have the potential to reduce energy consumption and environmental impacts in smart buildings.

Shortcomings: The adaptive and intelligent energy management systems discussed above have only approached RF energy management from the infrastructure point of view, focusing on boosting the harvesting abilities of the harvesting devices. Most do not pay attention to RF energy management based on varying user habits and QoS demands except in [98] where user satisfaction was a function of the optimization problem. Even in that instance, the user's satisfaction was only based on the energy harvesting cost to the user rather than on the user's stated service preferences.

3.4 Integrating Energy Harvesting Management with Energy Security Techniques

We present the existing literature on Energy Depletion Attacks (EDA) in Energy Harvested (EH) networks and adaptive energy security solutions in this section. Adaptive energy security solutions are solutions that aim to establish an optimal tradeoff between maintaining robust security and optimizing energy consumption to extend the lifespan of EH-powered devices and ensure continuous operation.

3.4.1 Energy Depletion Attacks in EH Networks

The authors in [59] presented the different security attacks and threats that affect the EH networks. EDA, a form of Denial of Service (DoS) attacks, occurs in EH networks through the following attack scenarios - flooding, spoofing, and jamming [59]. The goal of this attack is to drain the energy provision of the EH-powered devices quickly. Different security countermeasures, such as energy-aware security optimization algorithms and offloading to energy-supplied devices, have been proposed to mitigate these security attacks. However, these security countermeasures often overlook user-centric behavior.

A flooding attack is an attack that severely affects the performance of EH-Wireless Sensors Nodes (EH-WSN). To show how quickly bogus traffic depletes the energy of a device, an investigative study on the survivability of EH-WSN nodes was conducted by the authors in [58]. Through the use of an attack-effect model, the authors were able to estimate the intensity and direction of the attack on the sensor nodes in the EH-WSN. Additionally, the different energy attacks that occur in battery-less IoT devices due to the exploitation of limited control of energy sources underscore the growing threat landscape of EH-powered devices [100].

In [101], a Power-Positive Network (PPN) was proposed to prevent the possibility of an EDA occurring in an EH-powered device. Communication channels using wireless charging signals were used to build the PPN, which was used to offload the energy requirement to the energy source. While this solution proved to be effective in mitigating the occurrence of an EDA in EH-powered devices, it is dependent on the used communication protocol, and offloading the task makes it susceptible to man-in-the-middle attacks.

A lightweight security framework based on dynamic Bayesian game theory was proposed to protect the energy level of EH-powered devices against barrage attacks [102]. The proposed framework was modeled as a two-player game that uses Bayesian games to project

a future outcome based on past observations. This helped provide insights that were used to facilitate informed decision-making. However, the utilization of game theory assumes every node in the IoT network is rational, making it difficult to adapt to unpredictable behavior from a malicious node in a real-world deployment.

3.4.2 Adaptive Energy Security Solutions

Several adaptive approaches have been proposed to address EDAs in EH-enabled wireless sensor networks (WSNs). A comprehensive survey on energy-aware security solutions that highlight the tradeoff between energy consumption and security levels was discussed in [103]. A federated learning approach that analyzes aggregated energy profiles to detect anomalies was proposed to ensure energy efficiency [104]. However, user request patterns were not incorporated, thereby reducing the accuracy of the model to detect anomalies.

An adaptive security configuration strategy is proposed in [105] to dynamically adjust the security level of the EH-powered devices in function of the threat level and available energy. The authors in [106] proposed a lightweight scheduling mechanism that enhances data security and energy efficiency for applications on RF-based EH platforms. An adaptive framework that uses the Hungarian algorithm to adaptively map different AES methods to the IoT-based system based on their resources and throughput, for efficient energy in resource-constrained devices [107]. These solutions adjust security protocols based on harvested energy to ensure energy efficiency. However, they ignore malicious request behavior, making it susceptible to EDA.

In [108], the authors designed an optimization strategy that improves energy utilization by considering the energy cost required to transmit data in an IoT sensor. Likewise, in [109], the design space of an EH-powered device's secure authentication was investigated

to identify the varying energy cost to the layers in the cryptographic protocol design. These methods minimize the cryptographic costs required by the EH-powered devices, but outsourcing critical operations weakens the entire IoT network security, making it more susceptible to other forms of attacks.

Shortcomings: the current state-of-the-art solutions are primarily reactive and do not prevent external adversaries from launching continuous energy depletion attacks [110]. These solutions mainly focus on the energy profiles of the devices, without considering the request behavior of the user, leading to reduced accuracy and energy inefficiency. These device-centric approaches or detection schemes cannot entirely prevent persistent attacks. Therefore, there is a need for a security framework that integrates the request behavior of the user, alongside the profile of the device.

CHAPTER FOUR

RO1: IOT SIMULATOR TOOL FOR PROVISIONING SECURE AND DYNAMIC USER-CENTRIC SERVICES

IoT is a growing domain of interconnected devices, with about 12.2 Billion actively connected IoT as of August 2022 [1]. [111]. IoT devices are resource-constrained (memory, energy, computation power, and bandwidth). So, to improve user experiences and ensure secure interactions with applications on IoT devices, it is vital to understand how interconnected devices work. In addition, knowing that the preferences of users who subscribe to services on applications supported by IoT devices change over time, the interactions between IoT devices become complex. The need to satisfy a combination of personalized user preferences, based on service parameters like latency, bandwidth, and security, is defined as user-centric service provisioning.

Optimally providing user-centric services by resource-constrained IoT devices makes service provisioning even more challenging. Hence, IoT simulation tools enable the modeling and testing of IoT systems and applications, with use case scenarios, before deployment in target environments. IoT simulation tools help test the performance and deployment of IoT applications by determining and comparing key performance indices. A bottleneck in developing IoT simulation tools is that the proposed frameworks treat service security as a static parameter during IoT service provisioning. This negatively impacts performance, as the strength of service security is inversely related to service QoS parameters. Additionally, for user-centric service provisioning, one cannot assume that each user's security requirements will remain static. However, to address this challenge,

[1] iot-analytics.com/number-connected-iot-devices/

traditional security protocols cannot simply be incorporated into existing IoT simulation tools.

The goal of secure IoT research is to develop lightweight encryption and decryption protocols while maintaining efficiency and reliability [7]. Much work has gone into creating tools in the IoT research space to simulate the implementation of said lightweight protocols and depict their scalability in the IoT environment before deployment. Tools like IoTSim [61], DPWSim [62], ifogSim [63], Cooja [64], and CupCarbon [112] all serve this purpose [7]. However, of all the lightweight protocols addressed by existing simulators, there is limited discussion on how simulation tools address the issue of lightweight security affecting service performance in the use case for a user-centric IoT service provisioning. For RO1, we present an IoT simulation framework where user QoS experience can be improved by including lightweight security as a dynamic parameter. The framework is represented in Fig. 4.1.

Our previously proposed user-centric framework for IoT service provisioning [6] is implemented by modifying an existing IoT simulator tool, CupCarbon. Before modifying the simulator tool, we conducted an extensive survey and performance analysis on some existing lightweight security protocols for IoT applications, such as Trivium, Ascon, and NtruEncrypt, using metrics like key generation time, encryption and decryption speeds, and latency values. Based on performance outcome, NtruEncrypt lightweight security protocol is consequently integrated with CupCarbon [112]. A performance comparison is then carried out between the selected security protocol and CupCarbon's in-built security protocol - Blowfish. The result of the comparison is presented with a detailed discussion of the results. The contributions are as follows:

- Literature survey and performance analysis of lightweight IoT encryption protocols,

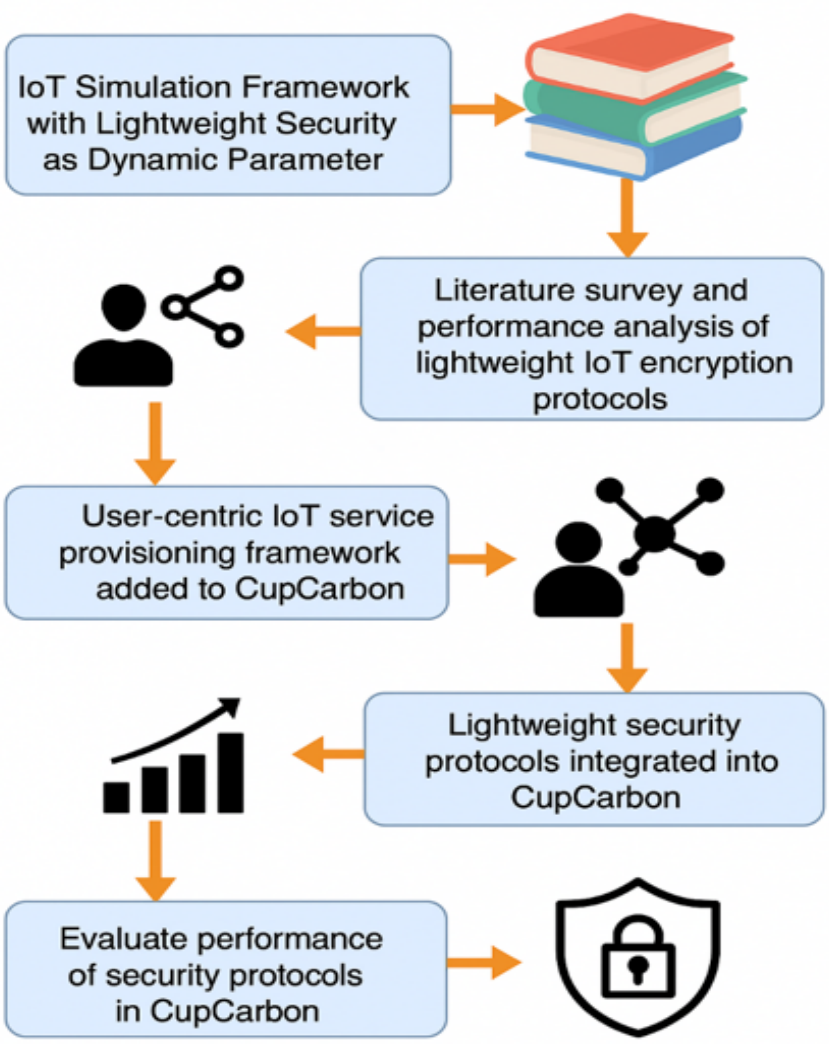

Figure 4.1: Framework of IoT Simulator Tool for Provisioning Secure and Dynamic User-centric Services

- Implement previously proposed user-centric IoT service provisioning framework [6] onto the CupCarbon simulator tool,
- Implement integration of IoT lightweight security protocols onto the CupCarbon simulator tool, and

- Evaluate the performance of the IoT lightweight security protocol implemented in CupCarbon against its default security protocol.

4.1 Performance Analysis of Cryptography Protocols

After surveying and qualitatively analyzing about 100 lightweight protocols, we shortlisted a set of candidate protocols on which quantitative analysis was carried out and schemes were compared based on performance metrics such as encryption-decryption time, ease of use, and key generation time.

4.1.1 Protocol Selection Criteria:

To qualitatively assess the surveyed protocols, three criteria were used. First, protocols that consider the inherent resource constraints of IoT devices.

Second, protocols that will offer an expected level of baseline security strength. Third, the flexibility of security protocols such that service providers can either add or remove the schemes in a modular way. Taking these factors into consideration, the final list of five protocols that were quantitatively tested is given in Table 4.1.

4.1.2 Performance Analysis

Symmetric Protocol Testing: Ascon, Trivium, and PRESENT were given three different-sized text files to encrypt and decrypt. The file format that represented the data packets generated by IoT devices was of sizes 32 bytes, 150 bytes, and 600 bytes, respectively. Each test was executed three times, and then the average time was logged for each algorithm. These results are shown in Fig. 4.2, Fig. 4.3 and Fig. 4.4.

Public Key Testing: Although symmetric protocols tend to be significantly faster than asymmetric ones, an issue arises concerning key distribution as corresponding parties require identical keys. To alleviate this issue, we also implement a public key protocol

Table 4.1: Shortlisted Lightweight Cryptography Protocols

Protocol	Description
Ascon [8]	An encryption and hashing algorithm with different modes - Ascon-128, -128a, -80pq, and Ascon-hash. Versatile in satisfying a user's basic needs for confidentiality and integrity of data.
Trivium [9]	A lightweight stream cipher, balancing security and performance, making it a good choice for resource-constrained devices. It uses an initialization vector of 80 bits and a symmetric key of 80 bits. It has an internal state size of 288 bits and requires at least 2600 gate equivalents (GE) for implementation.
PRESENT [113]	A lightweight block cipher that is comparable to AES but only requires 1570 GEs for implementation. It has a block size of 128 bits, utilizing a symmetric key of either 80 or 128 bits.
ECDH [114]	A lightweight public key protocol that uses elliptic curve cryptography (ECC). It can provide equivalent security compared to traditional PKI protocols like RSA, which is resource-intensive.
NtruEncrypt [10]	A lightweight public key protocol. It consists of 10 modes of operation, making it very versatile. Each mode of operation is a tradeoff between performance and security.

that safely and securely distributes the necessary symmetric keys. Table 4.2 benchmarks popular public key protocols against one another.

Table 4.2: Benchmarking of public key protocols

Encryption Protocol	Key Generation	Encryption	Decryption
Ntru-439 (128 bit key)	561.46ms	629.24ms	791.75ms
Ntru-743 (256 bit key)	410.53ms	331.89ms	591.34ms
RSA-3072 (128 bit Key)	1879.99ms	280.00ms	258.39ms
ECC (curve25519) (128 bit key)	637.43ms	657.41ms	660.22ms
ECDH-256 (128 bit key)	1193.79ms	1226.52ms	1144.04ms
ECDH-521 (256 bit Key)	1110.50ms	1089.91ms	1082.70ms

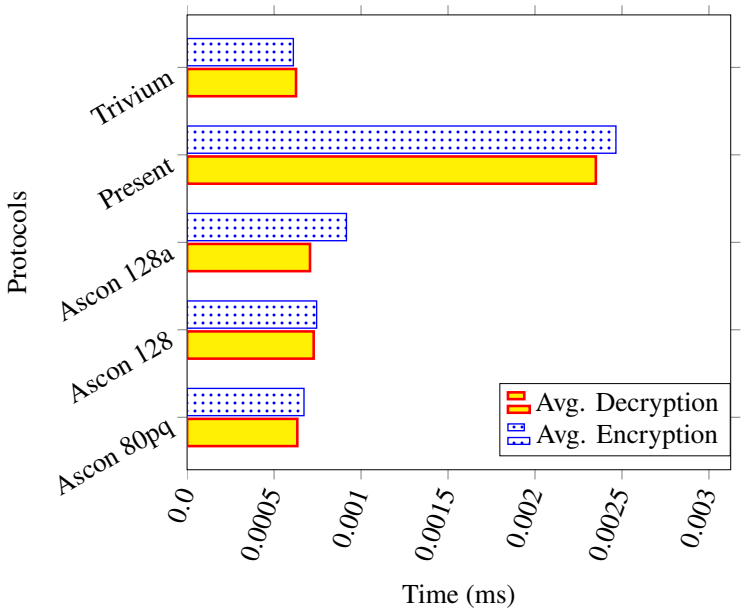

Figure 4.2: Avg. encryption and decryption time for 32 bytes

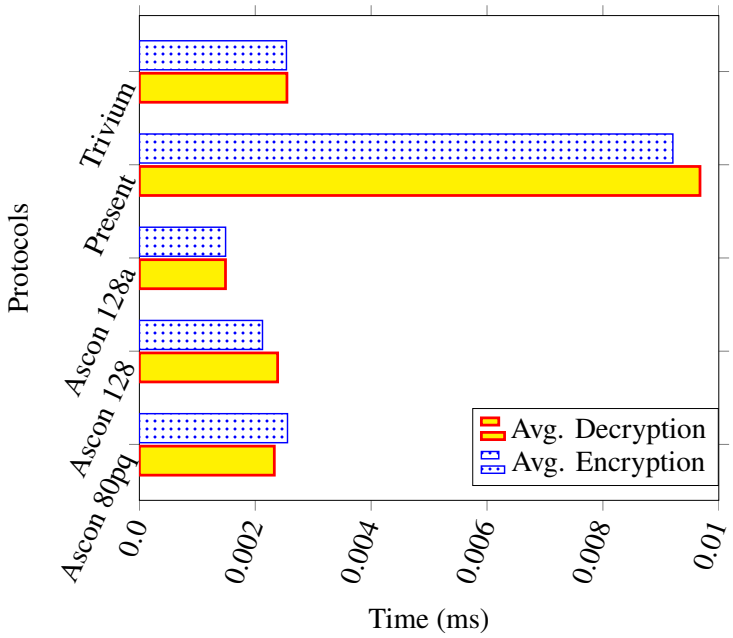

Figure 4.3: Avg. encryption and decryption time for 150 bytes

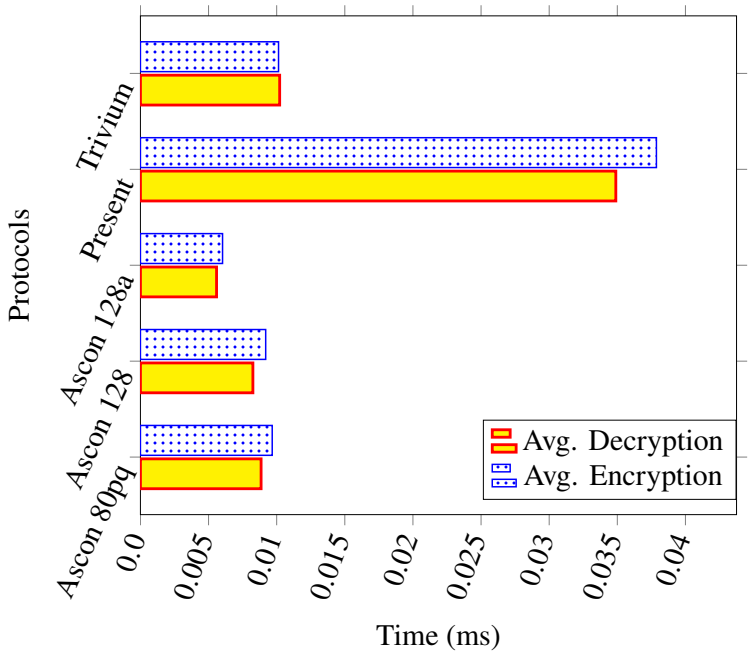

Figure 4.4: Avg. encryption and decryption time for 600 bytes

The two public key protocols that we could reasonably implement into our simulator were ECDH and NtruEncrypt. However, through benchmarking, we deduced NtruEncrypt had the fastest speeds overall as seen in Table 4.2.

Furthermore, the benefits of NtruEncrypt highlighted in [10] heavily outweigh those of ECDH, making it the ideal choice for our simulator. Given that our implementation of NtruEncrypt contained 10 different modes of operation, we tested the encryption/decryption speeds of each mode 3 times. Fig. 4.5 and Fig. 4.6 display the average of those speeds. Fig. 4.7 displays the average key generation speeds of each of the modes.

For our public key protocol, we decided to go with NtruEncrypt mode EES1499EP1-FAST as it provides users with stronger security, given its larger key size, 256 bits to be exact. Furthermore, this mode is optimized for encryption/decryption speeds,

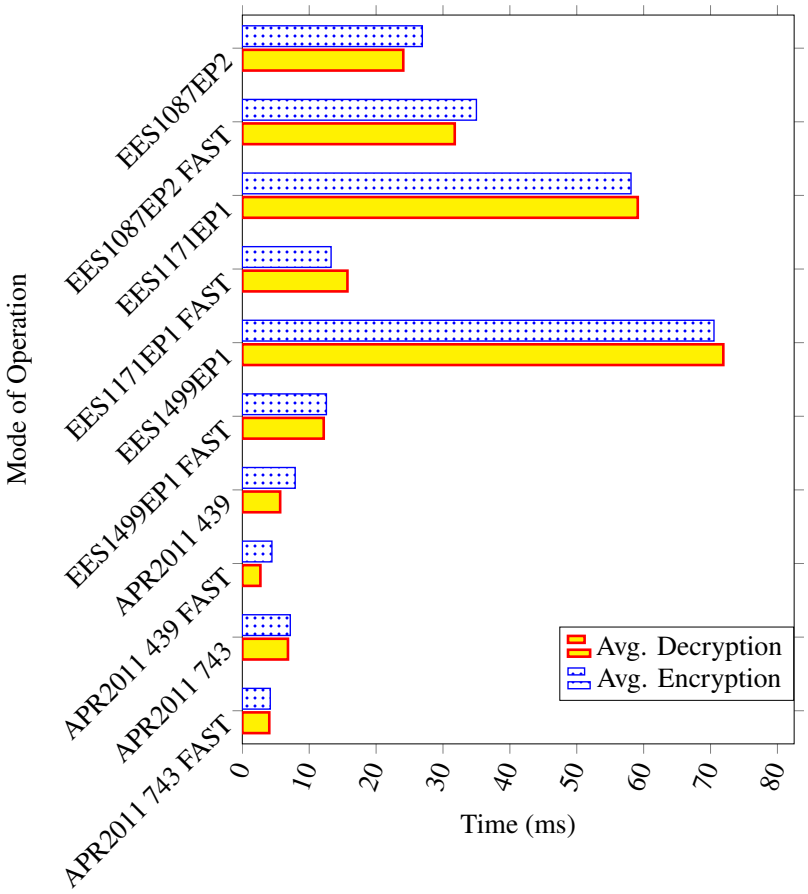

Figure 4.5: NTRU Avg. encryption and decryption speed for 16 bytes

making it ideal for this level, as it provides users with stronger security while still maintaining relatively fast speeds.

4.2 CupCarbon Tool Setup

In this section, we provide an in-depth discussion on CupCarbon as a simulation tool, the installation process for the Java-based client application, the connection to MongoDB, its project directory, and lastly, the functional workflow implemented.

CupCarbon Background: CupCarbon is a functional tool developed to aid the accurate simulation of signal propagation and interference in a 3-dimensional environment [112].

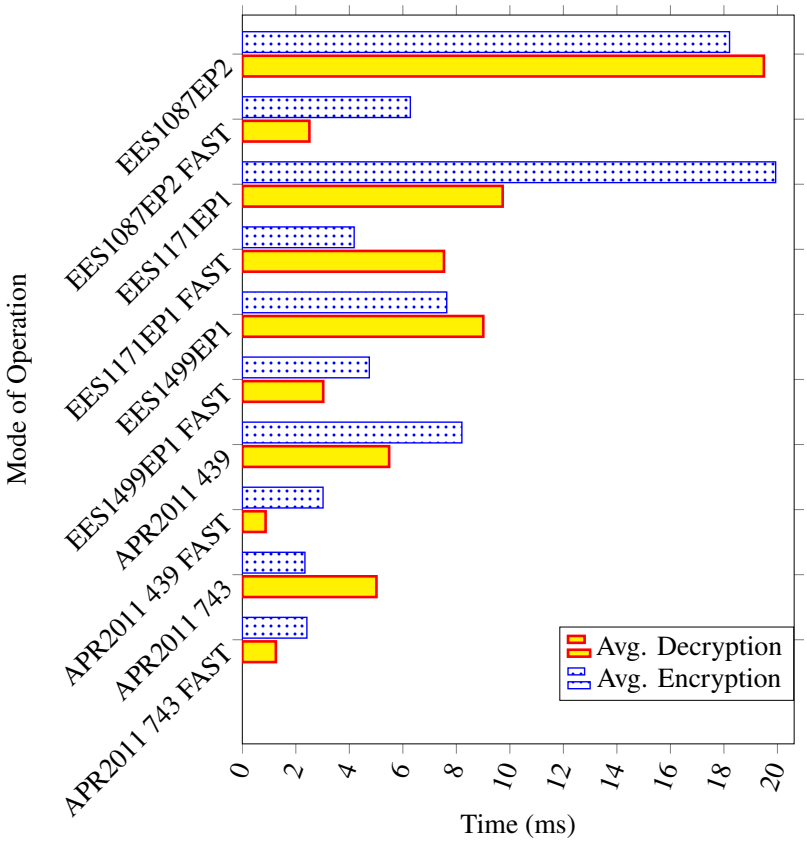

Figure 4.6: NTRU Avg. encryption and decryption speed for 32 bytes

Such an environment includes a Smart City and the network of Internet of Things (IoT) sensors deployed within it. In essence, CupCarbon simulates the interactions that take place within the Smart City and Internet of Things Wireless Sensor Network (SCI-WSN). CupCarbon provides an effective sandbox to design, visualize, debug, and validate distributed algorithms to monitor environmental data, and create environmental scenarios like fires and gas.

An intuitive interface uses the OpenStreetMap (OSM) framework to deploy sensors onto the map, making it easy to design and prototype the networks. It makes use of a script, known as SenScript, which allows users to program and configure each sensor

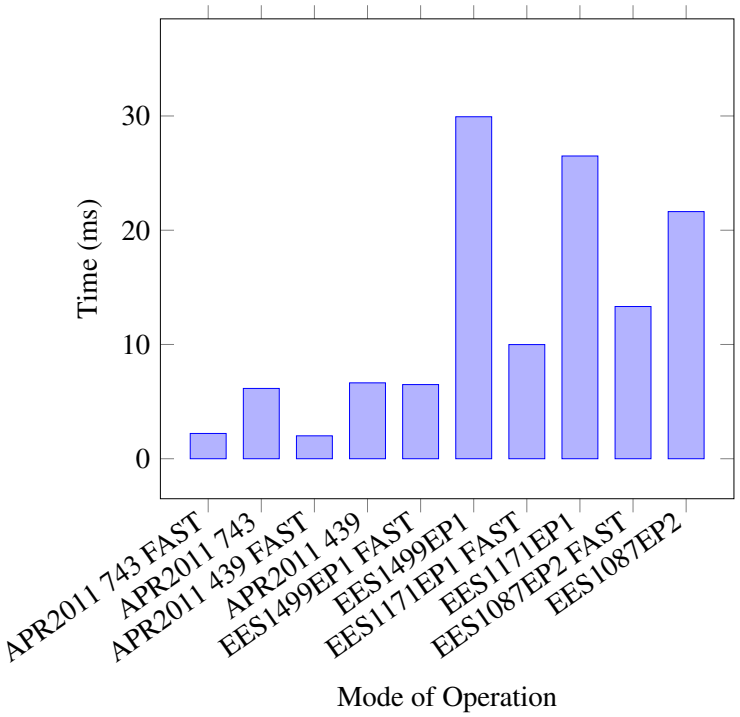

Figure 4.7: Average key generation of NtruEncrypt modes

node. In addition to selecting environment parameters from the interface, data to set up the simulation environment, and SenScripts to configure the sensor nodes can be preloaded from a connected database. For this project, we used a NoSQL database, MongoDB, as the connecting database. CupCarbon includes various protocols like Zigbee, LoRa, and WiFi protocols, which help to visualize and explain the basic concepts of sensor networks and how they work by providing a realistic implementation of sensor networks before they are deployed.

Installation: The entire simulation package named IoT-Simulation-Platform-master houses the folders needed for both Java and MongoDB installations. For the Java installation, source code from an existing version of CupCarbon (CupCarbon-master), with database connections and SenScripts already implemented, was downloaded. The downloaded

source code was imported into the Eclipse IDE workspace. It is important to note that JavaFX and Java compiler version 1.8 are required to ensure client files and packages run smoothly.

On the Database side, the MongoDB Community server with the default settings is installed. The `pythonGenerateNetworkForMongoDB` folder within the simulation package is made up of the `generator.py` and `importToMongoDB.py` Python files used to generate realistic values for users, devices, and project preferences to be exported into the client from the database and import results from the simulation back to the database. These values are written into and stored in `.csv` files, making it easy to load data when a simulation is started from the database.

Initializing CupCarbon Client Application: The project folder, CupCarbon-master, consists of several packages that perform different functions, ensuring the application runs effectively. The client application is initialized from the Eclipse IDE by running it as a Java application. The main class package `cupcarbon`, containing the main class file `cupcarbon.java`, is called. Other noteworthy java files in this package includes `CupCarbonClient`, `ConsoleWindow`, `ConsoleController`, `NaturalEventGenerator`, `NaturalEventGeneratorController`, `ResultWindow`, and `ResultController`. All of which take care of the client application's user interface window, console, controller, and natural events generator, including the result window.

Loading Data from Database: Once the GUI for the client application is running, we reset/reload the simulation from the database. This calls the database package using the `ConnectToDB, DBMethods, ImportFromDB, and ExportToDB` Java files to create a connection to the database, methods for database operations, import data to set up the environment in the client, and export results from the simulation back to the database. From the interface, we then select the connection button in the user section to load data

from the database. By selecting the Mobility/Events checkbox in the simulation parameters section, we indicate that the events and mobility of sensors should be considered during the simulation. The sensor button on the state bar at the bottom of the window displays the sensor nodes.

Running Simulation: The simulation is started by clicking the "Run Simulation" button in the Simulation Parameter section. This action calls the `Wisen-Simulation.java` file in the simulation package, which runs the entire simulation. `Simulation-Inputs.Java` file takes the simulation parameters, and SimLog.java stores the simulation logs. During simulation, the senscript package holding a set of Java files, each defining a senscript command like AND, FUNCTION, GETPOS, is also called. Any meaningful combination of these commands can be used to pass the instruction to the sensor nodes. The `senscript_functions` package holds the `Functions.java` file, where functions like encrypt, sum, and factor are defined, and the `ScriptFunctions.java` file, where the defined functions are called.

Security Implementation: For this project, to integrate lightweight encryption protocols with the CupCarbon client, library packages for Blowfish and Ntruencrypt, lightweight encryption protocols were imported into the security package of the simulation tool. The files in the `senscript_functions` package were also modified to implement the Blowfish and NTRUencrypt lightweight protocol methods in the class Functions of the `Functions.java` file. Function calls to the encryption protocols in the method function of the `ScriptFunctions.java` file were also implemented. The code snippets to implement and call the NtruEncrypt function are shown in Algorithm 1 below. To pass the encryption commands now included in the `senscript_functions` package to the sensor nodes during simulation, the `SensorWithRouterFunction`.csc file within the `files_for_database` folder was updated with the instruction: "function v1 NTRUencrypt $v1". Here, "$v1"

is the event value to be encrypted and decrypted, "v1" is assigned the newly encrypted or decrypted value, and "NTRUencrypt" is the function name for the encryption protocol. Detailed code on how to integrate Ntruencrypt with CupCarbon is available on GitHub[2]

Algorithm 1 Integration and Calling of NtruEncrypt Function in CupCarbon

```
catch
Function NtruEncrypt(msg, kp)
    if msg ≠ null then
        enc_start_time ← getCurrentTime() encoded_data ← ntru.encrypt(msg.getBytes(),
        kp.getPublic()) enc_end_time ← getCurrentTime() encryption_time ←
        enc_end_time - enc_start_time
        dec_start_time ← getCurrentTime() decoded_data ← ntru.decrypt(encoded_data,
        kp) dec_end_time ← getCurrentTime() decryption_time ← dec_end_time -
        dec_start_time
        return decoded_data
    end
End
// Logic to call the function from CupCarbon
if function.equals("ntruencrypt") then
    try result ← NtruEncrypt(args) return {result, "0.0", "0.0"} Exception e
     e.printStackTrace()
end
```

4.3 Simulation Parameters

In this section, we discuss how our simulations were set up. Simulation parameters are presented, followed by short descriptions of the simulation setup for different experiments. To run a successful simulation, the input parameters are loaded from the MongoDB database collection. Each collection consists of the attributes for each of the input parameters, which include:

[2] github.com/alaodamilola/CupCarbon-Simulation-Tool-with-NtruEncrypt

a. users: the user name, desired location given as geographic coordinates (longitude and latitude -71.044, 42.319, -71.036, 42.310), event sensing (temperature, humidity, gas, lighting, wind, and water) set to true, preferred-Latency = 10 milliseconds, and preferred-Frequency = 10 seconds.

b. devices: A total of 280 devices are deployed. These consist of sensors and routers. Device type, with routers and sensors with router functions, are denoted as 1, base stations with router functions denoted as 4, while 2, 13, 14, 15,16, and 17 represent gas, temperature, humidity, wind, water, and lighting event sensing devices. We also have the Device ID, with device location given as geographic coordinates. The device SenScript file name could either be Router.csc, BaseStationWithRouterFunction.csc, SensorWithRouterFunction.csc or event.evt file for routers, base stations, sensors, and event sensing devices, respectively.

c. result: Once the simulation is complete, the output is stored as a document in MongoDB named "result". The results document is a collection of the user ID for the requesting user, the simulation time at the point the sensing service was requested, and the ID and location of the sensor that fulfilled the sensing request. The latency in communication between the devices in milliseconds is recorded, and lastly, the levels of the sensed natural events are also recorded.

We performed incremental sets of experiments to show how user security preferences impact the resulting non-functional requirement of the user, such as latency, based on the user-selected lightweight security encryption protocol. With the `generator.py` file, a maximum of 11 users, 460 devices (100 sensors, 28 base stations, 52 routers, 180 radio modules, and 100 nature events), including the project preferences, were generated. The `importToMongoDB.py` file imports the generated values into the MongoDB database.

4.4 Simulation Results and Observations

In this section, we discuss the results and analysis of our simulation experiments on CupCarbon.

Three different sets of IoT use-case scenarios are carried out using the aforementioned simulation parameters: (1) simulation with no encryption, (2) simulation with Blowfish encryption, and (3) simulation with Ntruencrypt encryption. We compared latency results from all three use cases to observe how the encryption protocol implemented affects the overall outcome of the simulations. The number of simulation cycles is depicted on the X axis, while the Y axis represents the latency value in milliseconds (ms) per simulation cycle.

4.4.1 No Encryption vs. Blowfish Encryption

In Fig. 4.8 we compare the latency of the BlowFish encryption simulation to the latency of the simulation with no encryption. There are 88 cycles for each simulation run. During the first couple of cycles (Simulation Cycles 1-4), the event values transmitted have a value of zero; hence, we see very low latency levels for both simulations. Latency values rise as event values begin to be transmitted.

For Blowfish encryption, we observe high values similar to values like 3.47E-03(ms) and 3.66E-03(ms) recorded at simulation cycles 12 and 24, respectively. Points with such high values depict periods during the simulation where at least one event value has just been transmitted. Increased latency at these points can be attributed to the increased computational power required to perform encryption and decryption for secured data transmission.

Blowfish encryption algorithm uses a 64-bit block size and key size between 32 bits and 448 bits. During encryption, key expansion occurs, causing an increased key size, thereby

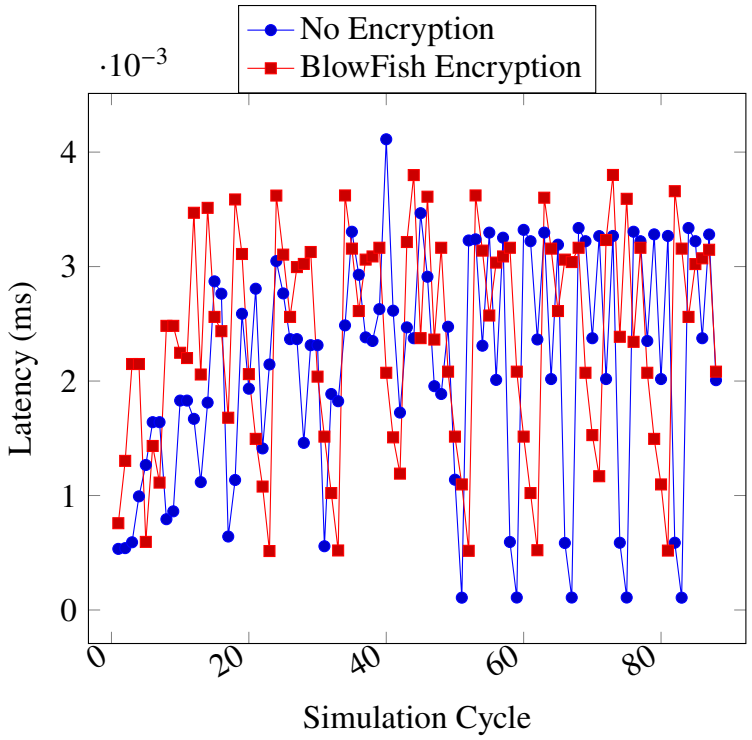

Figure 4.8: Latency: No Encryption vs. BlowFish

requiring more computation for decryption as well. Alternatively, we see the lowest latency values similar to values 5.15E-04(ms) and 5.22E-04(ms) recorded at simulation cycles 23 and 62, respectively. Such reduced latency occurs right after the transmission of more than 2 event values in the two preceding cycles. At these simulation cycle points, no event data transmission is taking place; hence, no encryption or decryption is happening, bringing latency to the lowest.

4.4.2 No Encryption vs. NtruEncrypt Encryption

In Fig. 4.9, the latency of the NtruEncrypt encryption simulation is compared to the simulation latency of no encryption. We see high latency values similar to values like 3.08E-03ms, 3.26E-03ms, and 3.16E-03ms recorded at simulation cycles 14, 45, and 84.

High latency values are observed during these cycles as they are preceded by cycles where two or more event values have been transmitted. It implies higher computation resource usage due to higher encryption demands.

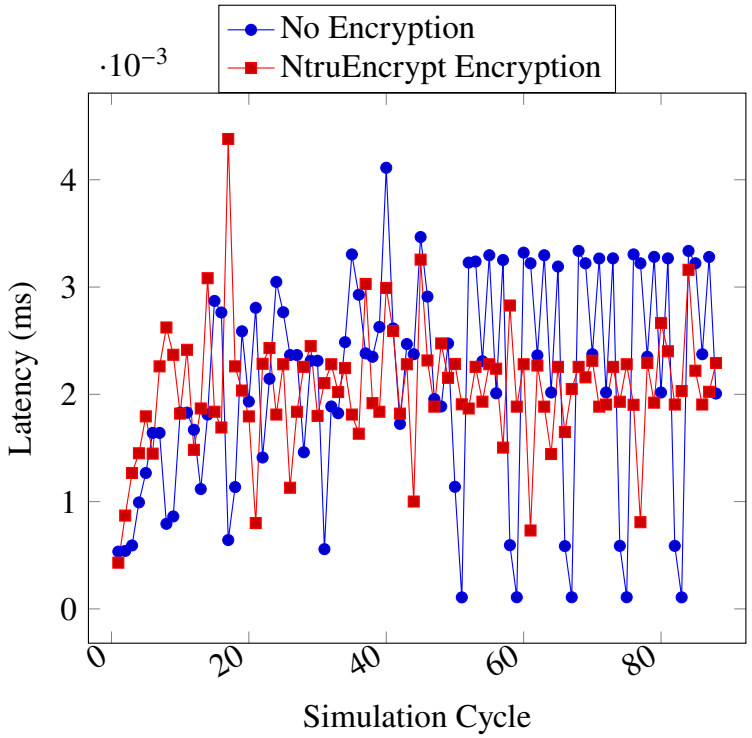

Figure 4.9: Latency: No Encryption vs. NtruEncrypt

In contrast to no encryption, comparison with Blowfish encryption, we make an interesting observation in the latency values. Here, latency for NtruEncryption seems to perform better than without encryption. Ideally, the presence of an encryption protocol should cause a bit of delay in throughput, thereby impacting latency. More computational resources are required to perform encryption and decryption as opposed to non-encrypted processing. However, NtruEncrypt is optimized explicitly for resource-constrained devices like IoT devices and sensors; hence, it is designed to have very low memory usage,

which contributes to an improved latency performance. In addition, NtruEncrypt uses only simple polynomial multiplications, making it a high-speed, lightweight encryption protocol. As such, we realize that NtruEncrypt takes generally below 2.50E-03ms as opposed to 3.00E-03ms for the simulation with no encryption.

4.4.3 Blowfish vs NtruEncrypt Encryption

Fig. 4.10 presents the comparison in latency of Blowfish encryption simulation to the latency of NtruEncrypt encryption. It is observed that the NtruEncrypt performs better than the Blowfish encryption. Latency for NtruEncrypt generally performs well below 2.50E-03ms when compared to Blowfish with a latency of about 3.50E-03ms, making it 1.00E-03ms slower than NtruEncrypt.

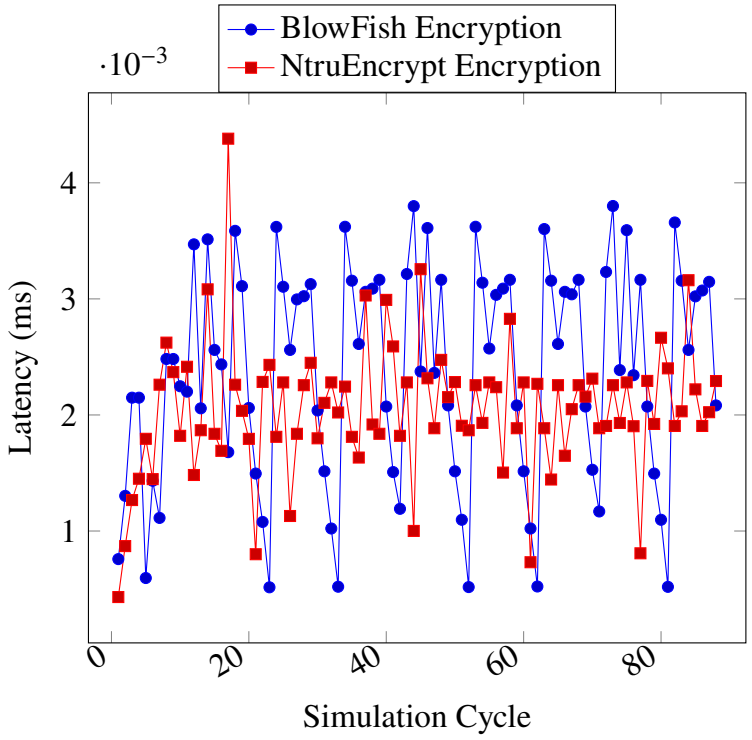

Figure 4.10: Latency: BlowFish vs. NtruEncrypt

The better performance by NtruEncrypt can be attributed to the cipher block size and the encryption key setup process. With a 64-bit block size cipher, Blowfish is only able to perform about 8 table look-ups during encryption. In comparison, NtruEncrypt can perform an average of 10 table look-ups, giving NtruEncrypt an advantage where large file sizes are being encrypted.

In addition, the Blowfish encryption key setup process is more cumbersome than that of NtruEncrypt, making it slower than NtruEncrypt, which can easily and efficiently perform encryption key generation. Although for NtruEncrypt, the decryption process requires a little more computation, it doesn't affect its performance as such. Spikes in simulation latency occur where event data has been transmitted in the simulation cycle immediately before and after, like around cycles 14, 37, or 40. These spikes are more pronounced when multiple event data are transmitted, as observed in cycle 61. The low latency values could indicate a brief downtime between encryption and decryption, as the values right before and after these points fall within normal ranges for both NtruEncrypt and Blowfish encryption simulations.

4.5 Conclusion

This work surveyed several symmetric and asymmetric lightweight cryptography protocols used in an IoT environment. We then analyzed the performance of the protocols. We learned about suitable protocols like NtruEncrypt, appropriate for IoT applications based on users' desired security levels. We looked into CupCaborn, a commonly used simulator, to properly recreate real-world scenarios of interactions between heterogeneous IoT sensors and devices, which supports effective design and testing of IoT applications. Integrating the NtruEncrypt lightweight cryptography protocol with CupCaborn enabled us to extend the implementation of a previous work, where security is implemented as a

non-functional parameter. We then demonstrated the viability of this work by comparing the performance of the integrated NtruEncrypt protocol and CupCarbon's inbuilt protocol, Blowfish. NtruEncrypt presented better results than Blowfish, due to larger cipher blocks processing larger file sizes, and easier encryption setup processes for encryption key generation. Future work would extend the functionalities of the integrated encryption protocol to provide prospective users with varying degrees of security options based on findings on the performance of the different NtruEncrypt protocol models.

CHAPTER FIVE

RO2: RESOURCE MANAGEMENT WITH SERVICE PREDICTION AND RECOMMENDATION

In simulating real-world applications with dynamic user-centric service provisioning in an IoT environment, RO1 implements the framework in [6] by integrating a lightweight encryption protocol with an IoT simulation tool [115]. Data generated during simulation to monitor device performance reveals the need for improved IoT resource management using users' service preferences to improve their experience. A motivating example is an IoT environment where individuals can improve their quality of life through access to various environmental sensing services provided by devices within the environment. As more users adopt IoT-enabled applications and demand real-time services with dynamic QoS preferences, the need for the available IoT devices to support those applications and provide such services increases. The challenge is maximizing the utilization of resource-constrained devices while ensuring a high QoS for each user, typically referred to as resource optimization. Existing methods often cannot accommodate the dynamic and user-specific QoS demands, and simultaneously prioritize resource allocation for real-time user requests.

The second research objective is driven by the need to anticipate such demands and dynamically reallocate IoT resources, thereby minimizing service delays and optimizing device efficiency. We introduce a system that predicts future user service needs by analyzing historical data and current user requests. The system also ensures resource optimization by recommending and dynamically allocating IoT resources based on real-time data and predicted user needs, aligning with explicitly stated preferences of the user for each service they request. The approach addresses two significant issues- first, the data sparsity problem,

where available historical data might be insufficient for accurate IoT resource allocation. Second, the cold start problem, where no historical data exists for new users, emphasizes the need for combined prediction and recommendation processes for optimized resource management.

For IoT service providers to efficiently manage and allocate devices suitable for users' service requests, we introduce an IoT resource management framework using service prediction and recommendation. Knowing the future service demands of a user aids the service provider in making informed recommendations about devices capable of providing those services and ensuring their availability. ML-based prediction and recommendation techniques are leveraged for an optimized resource management framework to increase the efficiency of constrained IoT resources. Without the ability to predict and make recommendations, service providers may struggle to meet user needs efficiently. The contributions in this area are as follows:

- Allow the optimization of IoT devices' resource management and utilization through users' service request prediction.
- Extend IoT devices' network lifetime by recommending adequate devices while considering user context and dynamic QoS requirements for device and service matching.
- Improve resource allocation efficiency using Vector Autoregression (VAR), a semi-supervised ML multivariate time-series predictive analysis technique to anticipate service demand with up to 95% accuracy.
- Design a prediction and recommendation framework to tackle the cold-start and data sparsity problems by incorporating default or generalized profiles for new users.

5.1 Proposed Framework

Traditional approaches to resource management struggle with the vast deployment and heterogeneous nature of interconnected devices across geographically dispersed locations, requiring innovative solutions to optimize resource allocation. Prior work in [115] offers insights into the complexity of deploying IoT applications and underscores the significance of optimizing resources. The work in [6] also emphasized user-centric service provisioning by identifying how user preferences can influence resource allocation decisions. Understanding user-centric resource management can lead to more tailored and efficient resource optimization strategies. Building upon these insights, this paper introduces a novel framework to enhance IoT resource management by leveraging predictive analytics and ML techniques.

The proposed framework predicts user sensing services and QoS requirements within specific Regions of Interest (RoIs) and recommends the devices that meet those needs. Analyzing new and historical user interactions and preferences, the framework predicts future service demands, facilitating optimized resource allocation and operational efficiency in IoT deployments. The following subsections provide technical details of the design, including the predictive analysis modeling approaches and the recommendation engine. These components form a comprehensive approach to advancing IoT resource management, addressing critical challenges, and paving the way for more effective deployment and utilization of IoT devices.

5.1.1 Conceptual Framework Design

The proposed framework is designed as a two-phase system: a prediction model and a recommendation engine. The prediction model uses a semi-supervised multivariate time series analysis ML model to forecast future service requests of users based on their

previous service preferences within a specified region over some time. In contrast, the recommendation engine is designed to suggest optimal resource allocations based on the predicted future user requests. The aggregate of predicted service requests for the user and the suggested device-supported services offers the service provider a more accurate representation of the users' needs and the resources available to support those needs in the specified region. As a result, recommendations on a set of devices that would serve those service requests can be made, thereby optimizing the management of resources.

As depicted in Fig. 5.1, the framework comprises several interconnected components, each playing a distinct role in optimizing IoT resource management.

User Profiles: For new users, initial profiles are created by capturing their preferences during their first interaction with the system. To address the cold-start problem, recommendations for new users are based on default or generalized profiles until sufficient data has been collected. The dotted lines depict the flow of initial recommendations for new users. Historical data, including QoS parameters such as RoI, throughput, and response time of existing users, is captured and leveraged for more personalized service recommendations.

Predictive Analysis Function: Using Vector AutoRegression (VAR), a multivariate time-series ML model, the framework forecasts future user service requests and device service capabilities based on historical data and user preferences captured in real-time. This proactive approach ensures that resources are allocated optimally to meet anticipated demands.

Device Recommendation Engine: This component matches predicted user service requests with the most suitable IoT devices based on user QoS preferences (e.g., response time, throughput, security). It can reserve devices for future requests, enhancing the system's ability to handle real-time and predicted demands for a seamless QoS experience.

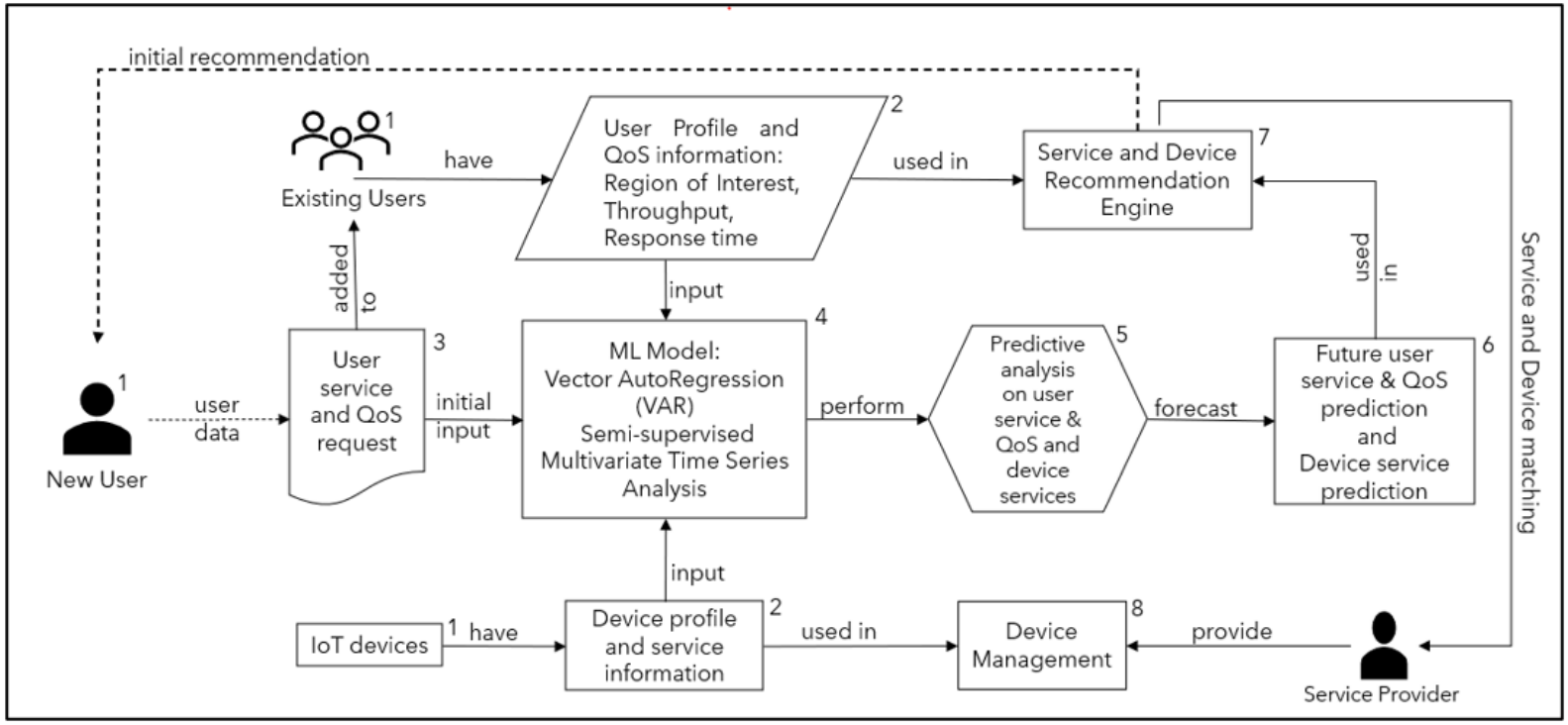

Figure 5.1: Machine Learning-driven IoT Service Prediction and Recommendation Framework for Optimized Resource Management

Service Provider Integration: The outputs of the predictive and recommendation engine guide service providers in ensuring resource availability and optimizing resource management. This helps improve network efficiency and user satisfaction. The overall framework operates as a multivariate time-series resource management system, combining predictive modeling and recommendations for better efficiency. The methodology involves data collection and pre-processing, the implementation of the predictive model, the simulation of the recommendation engine, the execution of the resource optimization function, and the evaluation of the framework's performance. Details of each step are elaborated in subsequent sections.

5.1.2 Predictive Model

The analytical process for predicting the behavior of IoT services over time employs a predictive modeling technique known as Vector Autoregression (VAR). This ML model, which is used for time series analysis and forecasting, has proven effective in IoT service prediction. Unlike neural networks, which are computationally intensive, VAR is computationally efficient and well-suited for the constrained environments of IoT systems. In this work, the VAR model is designed to discern relationships and dependencies among multiple variables across time intervals. When predicting IoT service performance, the model leverages historical user data, encompassing sensor readings, device statuses, network conditions, and other pertinent variables organized in a time series.

The VAR model forecasts the IoT service behavior of users based on past observations. This section details the selection of pivotal variables for predicting IoT service behavior using statistical tests such as Granger causality and multicollinearity (eq. (5.1), (5.2), (5.3), and (5.4)). Johansen's cointegration test (eq. (5.5)) is also employed to assess stationarity among the variables. The goal is to identify significant predictors and determine the optimal

lag order for the VAR model to properly evaluate its performance in forecasting IoT service dynamics based on historical data. This is crucial for the accuracy of the predictive model.

5.1.2.1 Variable Selection: This step is performed to determine the variables that are considered essential for predicting the IoT service behavior. These variables include inputs that affect the service performance, such as sensing service requests, and factors that influence the underlying system, like throughput, response time, and device energy level. To ensure that only the essential variables are selected, we perform a Granger causality test. Observations in the dataset are recorded sequentially (Section 5.2.1); therefore, the Granger causality test is used to assess which time series variable within the dataset can be used to predict the other. The formula for the Granger causality test is shown in eq. (5.1).

$$Y_t = \alpha + \sum_{i=1}^{p} \beta_i Y_{t-i} + \epsilon_t \tag{5.1}$$

As expanded in eq. (5.2), the Granger causality test involves comparing two regression models:

$$Y_t = \alpha + \sum_{i=1}^{p} \beta_i Y_{t-i} + \epsilon_t Y_t = \alpha + \sum_{i=1}^{p} \beta_i Y_{t-i} + \sum_{j=1}^{q} \gamma_j X_{t-j} + \epsilon_t \tag{5.2}$$

$$Y_t = \alpha + \sum_{i=1}^{p} \beta_i Y_{t-i} + \epsilon_t \tag{5.3}$$

In equations (5.1) and (5.2) above, Y denotes the dependent variable, such as sensor readings or service performance metrics. α represents the intercept term, β_i are coefficients corresponding to lagged values of Y (historical data), t denotes the current time period, and i indicates the lag or number of time periods before t. The summation symbol $\sum_{i=1}^{p}$ signifies summing over values of i from 1 to p, with Y_{t-i} indicating the value of Y at time $t-i$, representing Y at i periods before t. ϵ_t denotes the error term t, while X_t represents the independent variable (network conditions, device statuses) being evaluated for its predictive

impact on Y. This analysis assesses whether past values of X contain pertinent information for predicting future values of Y, as depicted in eq. (5.3). The proposed framework addresses a multivariate time-series problem, requiring multiple linear regression analysis. Hence, we perform the multicollinearity test using the Variance Inflation Factor (VIF) on the dataset described in section 6.4. The Variance Inflation Factor (VIF) is calculated using the formula in eq. (5.4).

$$\mathrm{VIF}_j = \frac{1}{1 - R_j^2} \tag{5.4}$$

Where R_j^2 is the R^2 value obtained by regressing the j-th predictor variable, such as sensing service request and QoS requests (throughput and response time), against all the other predictor variables. The VIF quantifies how much the variance of an estimated regression coefficient is inflated due to multicollinearity, indicating the extent to which other independent variables explain a given independent variable. VIF values range from 1 to 10, with values exceeding 5 suggesting strong multicollinearity. Variables with high VIF values (> 5) are highly correlated with other predictors, leading to multicollinearity, which can compromise the stability and reliability of regression coefficients.

5.1.2.2 Stationarity Check: Stationarity is an essential assumption in VAR models, which means that the statistical properties of the variables should remain constant over time. Before fitting a VAR model, it is crucial to check the stationarity of each variable and, if necessary, apply transformations to achieve stationarity. In the proposed framework, we performed the stationarity test using Johansen's cointegration test. Johansen's Cointegration Test involves computing the Trace Statistic given in eq. (5.5).

$$\text{Trace Statistic} = -T \sum_{i=1}^{r} \ln(1 - \hat{\lambda}_i) \tag{5.5}$$

Where T is the sample size for this project, T = 10,080. r is the number of cointegrating vectors which in our case $\bar{5}$ vectors, $\hat{\lambda}_i$ are the estimated eigenvalues obtained from the Johansen procedure. This test helps to determine the presence of cointegration among a set of time series variables. This means it is designed to assess the existence of long-term relationships among multiple variables. By performing Johansen's cointegration test, we can identify if a linear combination of the various variables is stationary and has a stable relationship over time.

5.1.2.3 Model Specification and Estimation: The VAR model is specified by determining the appropriate lag order following the variables selection and preparation. The lag order represents the number of past time steps (lags) to predict future values. Statistical techniques such as Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), or sequential testing are used to select lag orders. To accommodate the unique challenges in addressing the cold start problem, a sequential testing method was adopted to select the lag order, ensuring the predictive model can still perform well with limited historical data and minimize prediction errors. In addition, an incremental approach was adopted, starting with a conservative lag order. A variable, `optimal_lag`, was initialized to systematically evaluate lag orders, starting from 35, where noticeable enhancements in predictive accuracy began, continuing up to 50 in increments of 5. The evaluation determined that a lag order of 40 provided optimal performance for the proposed model, showing no significant improvements beyond this point. This allows the model to adapt and enhance predictions gradually as more historical data becomes available, thereby improving its ability to capture complex relationships and enhance forecasting accuracy over time.

The VAR model parameters are estimated using maximum likelihood estimation or ordinary least squares regression. This involves fitting the model to historical data and

calculating coefficients that best represent the relationship between the variables. To accommodate the variability in user interests, the dataset is partitioned into training and testing sets using a 90:10 ratio. A larger training set provides more examples for the model to learn from, thereby capturing more complex patterns in the data. An instance of the VAR model is initialized with the training dataset and then fitted using the selected `optimal_lag = 40`.

5.1.2.4 Model Forecasting and Evaluation: Once the VAR model's predictive accuracy is satisfactory, it can be used to predict future IoT service behavior. Given the historical values of the selected variables, the model can generate forecasts for the next time steps based on the estimated coefficients and the lagged values of the variables. The model is set up to forecast future user requests for the next 120 minutes to balance real-time decision-making with computational feasibility. The steps carried out for predictive analysis in the proposed framework are summarized below.

Variable selection

#1: Perform Granger causality test using eq. (5.1)

#2: Perform Multicollinearity test using eq. (5.4)

#3: Perform Johansen's Cointegration Test using eq. (5.5)

Model Specification

#4: Perform sequential testing to determine optimal lag order

#5: Select lag order with most suitable order

Fit Vector AutoRegression (VAR) Model

#6: Train-test split: train = data * 0.90; test = data * 0.10

#7: Model = VAR(train)

#8: Fitted model = model.fit(optimal_lag = 40)

#9: Forecast = fitted_model.forecast(train_values)

Model Evaluation

#10: Calculate evaluation measures and accuracy

5.1.3 Recommendation Engine

Following the predictive analysis, the recommendation engine suggests appropriate IoT devices according to user-defined criteria. The recommendation engine is designed as a function that receives two sets of information as input. The first set of information is about the user's service request pulled from `service_request_filepath` including sensing service requested, desired response time and throughput, and required security protocol received in addition to information from `device_filepath` about the IoT devices, such as current energy level, response time, throughput, and security protocol. To determine a set of devices that can fulfill each user request, criteria such as requested sensing service, preferred response time, throughput, and security protocol preferences from the user are matched with the device information. To support device management, the energy level of the devices is compared with the computed amount of energy required to complete service requests.

This filters devices that either meet or surpass the user's specified service criteria. The filtered devices that align with the user's criteria are subsequently compiled into a list of recommended devices, tailored to fulfill the user's service requests. The list of recommended devices is returned as the result of the function. Otherwise, if no exact match is found, a fallback ensures a best-effort match. If processing a predicted service, a device that best fits the service request is selected and assigned to the service request from the recommended devices. The assigned device is then added to a list of reserved devices to fulfill specific predicted user requests. When all the devices have been compared, the function returns a set

Algorithm 2 Device Recommendation Function with QoS-Aware Fallback

Input: device_filepath, service_request
Output: reserved_device
$service\_chance \leftarrow$ random.uniform$(0,1)$ **if** $service\_chance \geq 0.6$ **then**
| Use grid search to select service
else
| Select randomly or fallback option
end
if *sensing_service* == *predicted_service* **then**
| $service\_type \leftarrow$ `predicted_service`
else
| $service\_type \leftarrow$ `non_predicted_service`
end
$recommended\_devices \leftarrow$ `recommend_devices`$(device\_filepath, service)$
foreach *device in recommended_devices* **do**
| **if** *device_energy* > *required_energy* ***and***
| *device_service* ∈ *sensing_service* ***and***
| *device_RT* ≤ *requested_RT* ***and***
| *device_TP* ≥ *requested_TP* ***and***
| *device_security* ∈ *requested_security* **then**
| | **return** recommended_device
| **end**
| **else**
| | **return** fallback_device
| **end**
end
$recommended\_devices \leftarrow$ filtered_device_list
if *sensing_service* == *predicted_service* **then**
| $reserved\_devices \leftarrow$ `assigned_device` = max(`matching_devices`,
| key=lambda x: $(x.current\_energy, -x.response\_time, x.throughput)$)
| $reserved\_device\_list \leftarrow reserved\_devices$
return reserved_device

of recommended devices for the service request. The recommendation engine enhances the user experience by providing personalized suggestions based on historical data and specified preferences. The algorithm for the recommendation function is shown in Algorithm 2.

5.1.4 Resource Optimization

This section discusses the technical concepts and detailed steps of the service recommendation process within the proposed framework for resource optimization. The process involves a joint simulation of predicted and non-predicted service requests to evaluate device performance during service requests. The resource optimization problem is formulated in eq. (5.6):

$$\text{Maximize: } L_{\text{network}} = \frac{\sum_{i=1}^{N} R_i}{T_{\text{total}}} \quad \text{subject to: } Q_{i,j}(t) \geq Q_{\min} \tag{5.6}$$

The network lifetime, L_{network} represents the ratio of total fulfilled user requests ($\sum_{i=1}^{N} R_i$) to the total operational time (T_{total}). R_i is the number of fulfilled user requests handled by device i, and T_{total} is the total operational time of the network. $Q_{i,j}(t)$ represents the QoS parameters such as response time and throughput for device i at time t, and $Q_{\min}$ is the minimum required QoS for a given service request. This formulation prioritizes extending the network lifetime by fulfilling more user requests while maintaining acceptable QoS levels ($Q_{i,j}(t) \geq Q_{\min}$).

The inputs for this process are predicted service requests, non-predicted service requests, and device information. The resource optimization process involves executing the recommendation function described in Algorithm 2. The devices that meet the users' service requests are filtered from the list of available devices. Filtered devices are then assigned to services identified as `predicted service` or `non-predicted services`. The `simpy` Python library is used to model the execution of predicted and non-predicted

services. Devices are initialized within this environment to simulate their behavior during service requests. Device metrics such as `network lifetime`, `number of requests fulfilled`, `number of unfulfilled requests`, and `device request frequency` are collected during the execution of the function. `Network lifetime` refers to the duration for which a network or a set of devices can operate before its components, in our case, device energy, are depleted and the network becomes non-functional. The `number of requests fulfilled` counts the number of service requests completed before the network becomes non-functional; conversely, `number of unfulfilled requests` are also computed. The `device request frequency` tracks how frequently each device handles service requests. To determine if predicting user service requests supports resource optimization, the result of the device metrics from the function execution is collected for both predicted and non-predicted services for comparison (discussed in section 6.4).

The algorithm for the recommendation process is shown in Algorithm 3. The simulation for IoT device optimization is initiated by importing the necessary libraries for data processing and manipulation. It then receives the inputs for the simulation from three sources (file paths of device information, predicted service, and non-predicted service information). Thereafter, the constructor class sets up the simulation environment with a 7200-second duration (2 hours) and initializes devices with uniform battery lives of 140(mAh), 180(mAh), 220(mAh), and 240(mAh) for different simulations. This reflects realistic energy budgets of ultra-low-power IoT sensors and helps analyze network lifetime under constrained conditions. The recommendation function is called to simulate device activities while fulfilling service requests, with metrics like network lifetime and service requests collected. Battery depletion is modeled per service cycle using a fixed-rate energy model based on real-world IoT sensor specifications. Each reading (including sensing, optional encryption, and transmission) consumes approximately 2–3 mAh, depending on the

Algorithm 3 Process for IoT Device Optimization

```
Class IoTDevice
    __init__(env, battery_lifetime) self.env ← env  self.battery ← battery_lifetime
    env.process(self.run())  run() while self.battery > 0 do
        self.battery ← self.battery - 1  self.env.timeout(1)
    end
End

Function read_csv_and_get_service(path)
    return service_data from CSV at path
End
Function recommend_devices(service_request, all_devices)
    filtered_devices ← devices matching service_request criteria  return filtered_devices
End

// 1. Initialization
env ← simpy.Environment()  predicted_env ← simpy.Environment()

// 2. Load Data and Initialize Devices
normal_services ← read_csv_and_get_service("non_predicted_path")  predicted_services
 ← read_csv_and_get_service("predicted_path")  normal_devices ← create IoTDevice
 instances for env  predicted_devices ← create IoTDevice instances for predicted_env

// 3. Run Simulations
for each request in normal_services do
    recommended ← recommend_devices(request, normal_devices) // Assign
        request and update metrics...
end
env.run()  for each request in predicted_services do
    recommended ← recommend_devices(request, predicted_devices) // Assign
        request and update metrics...
end
predicted_env.run()

// 4. Collect and Analyze Metrics
normal_metrics ← collect_metrics(normal_devices)  predicted_metrics ←
 collect_metrics(predicted_devices)  print("Compare average metrics:", normal_metrics,
 "vs", predicted_metrics)
```

selected throughput and security protocol [116]. Higher throughput or stronger encryption modes (e.g., AES 256 encryption) lead to faster battery depletion. The simulation updates each device's battery level after every service execution and separately analyzes the metrics for predicted and non-predicted services.

5.2 Results and Analysis

The outcome of the proposed resource optimization framework is presented and analyzed in this section. The focus is on evaluating the performance of our predictive model, the effectiveness of our recommendation function, and the impact of the overall efficiency of the resource allocation process on resource management. Assessing the accuracy of our Vector AutoRegression (VAR) model's predictions and examining the impact of the recommendation engine using key performance metrics, such as network lifetime, the number of fulfilled and unfulfilled service requests, and device request frequency, to determine the benefits of incorporating predictive analytics recommendation functions. The results are compared between predicted and non-predicted service scenarios to highlight the improvements achieved, demonstrating the combined impact of predictive analytics and the recommendation function on enhancing IoT resource management.

5.2.1 Data Collection and Description

This subsection briefly describes how the data used in this research were collected. The original dataset used in this work originates from the Intel Collaborative Research Institute (ICRI) for Urban IoT. Designed mainly for research purposes to support operational management in an IoT environment, the dataset is made available by ICRI for experimentation in data visualization, data modeling, simulation, IoT data pipeline development, or machine learning. Since this work aims to improve device management

operations in the IoT environment using machine learning, this makes the dataset a valuable starting point. The original dataset is available at IEEE DataPort [117].

Envirosensor Data: The dataset includes real-time telemetry data, collected over 22 weeks, harnessed from 19 custom sensing devices, each crafted by ICRI for Urban IoT applications. The ICRI envirosensor data only provides raw sensor readings for Ambient Light (OPT/OPT3001), Ambient Temperature (TMP), Temperature (BAT/BMP280), Ambient Temperature (HDT), Barometric Pressure (BAR/BMP280), and Humidity (HDH/HDC1000), with one reading recorded per minute for each sensor. The ICRI data consists of real environmental sensor readings, but lacks user-service metadata and device information, a requirement to evaluate the proposed framework. Given the unavailability of real datasets consisting of user-service requests with dynamic QoS and security requirements, synthesized data was generated based on empirically derived distributions of previously validated deployments in energy-aware sensing and industrial IoT systems, and appended to the ICRI dataset. The resulting semi-synthetic dataset retains the real-world environmental diversity of the ICRI dataset and also enables evaluation of the proposed user-service request forecasting and device allocation. The framework's prediction component relies on temporal patterns in user demands from the synthetic profiles, while the recommendation component matches devices to these demands based on their QoS compatibility. This integration ensures the framework's evaluation remains valid and reflects scalable, real-world deployments. The user profile parameters include user preferred response time, throughput, security levels, and expected service duration and frequency with distributions based on realistic IoT environments [118]. The synthesized device data is generated using uniform distributions based on smart sensor benchmark specifications observed in previous IoT studies [118] [119]. Device capabilities like response time, throughput, and security protocol are based on industry standards. The synthesized

values, while not captured in a real-world activity, are sampled to depict realistic operational ranges in validated real-life deployments. Providing the ability to evaluate the framework under controlled but realistic IoT deployment conditions.

5.2.2 Data Pre-processing

The data preprocessing subsection briefly describes how the data used in this research was prepared for developing the prediction and recommendation models described in the proposed framework discussed in Section 5.1. In alignment with the research objectives, a preprocessing stage is executed to refine the raw sensor data. The raw sensor data, initially stored in extensive JSON files, undergoes extraction to isolate relevant information. This data is then mapped to user and device profiles, including key attributes like RoI, QoS, and Security preferences. To optimize the dataset, only the values related to four key sensor readings are retained, notably, Light, Temperature, Pressure, and Humidity. Illustrative samples of synthetically generated user interests and device capability profiles for low-powered IoT devices are presented in Table 5.1 and Table 5.2, respectively. Depicting realistic service demands aligned with service timing behavior and operational limitations within an IoT sensing environment. To represent realistic patterns of usage as discussed in [120], the profiles were uniformly sampled within a 5-day window for user requests, and a 7-day window for device capabilities. Some user requests fall within early morning hours (2:02 am - 5:22 am) as shown in Table 5.1. User interests include requested sensing service type (e.g Light, Pressure), Response Time or delay a user can tolerate before receiving feedback (in seconds), throughput or required data frequency in seconds, Security Protocol (Standard or High) for encryption strength, Timestamp for the start time the service was requested, and Service Duration (in minutes), indicating how long the service is requested for. For example, User45 requesting light sensing at 2:02 am with low response time, high

throughput, and standard security for 62 minutes may be an industrial plant monitoring environmental conditions during a night shift.

Table 5.1: Illustrative samples from synthesized user interest dataset (sampled uniformly over five days)

Timestamp	User ID	Response Time (s)	Throughput (s)	Service Requested	Security Protocol	Service Duration (min)
2:02AM	User46	5	5	Hu	H	80.72
2:02AM	User45	2	10	L	S	62.70
3:17AM	User71	8	1	P	S	98.20
4:03AM	User51	5	4	T	S	64.53
5:22AM	User39	3	6	P	H	76.97

Legend: RT = Response Time, TP = Throughput, L = Light, T = Temperature, P = Pressure, Hu = Humidity, H = High, S = Standard

Table 5.2: Illustrative samples from synthesized device capability dataset (sampled uniformly over seven days)

Device ID	Current Energy (mAh)	Response Time (s)	Throughput (s)	Sensing Service	Security Protocol
Device001	189	9	9	Hu	H
Device002	154	7	5	P	S
Device003	153	7	4	T	H
Device008	197	6	5	T	S
Device010	144	4	4	L	S

Legend: RT = Response Time, TP = Throughput, L = Light, T = Temperature, P = Pressure, Hu = Humidity, H = High, S = Standard

The synthesized device capability profiles include Response Time, the time allowed for valid readings after a sensing request, distinct from sensing duration. Throughput is the

time interval allowed between consecutive readings. The Security Protocol specifies the supported encryption level, balancing data security with computational load. These features mimic the expected behavior of constrained devices where their responsiveness and energy consumption are affected by operational demands, such as cryptographic operations and wake-up cycles [118]. For example, Device003 may have longer response times and lower throughput when using high-security protocols. To further prepare the dataset for use in the proposed framework, encoding of non-numeric attributes like `User ID` and `Device ID` into numeric values was performed using the `LabelEncoder` Python class function. As standard practice, the `Timestamp` attribute was also converted to numerical values to provide a uniform representation of time across the dataset. Thereafter, one-hot encoding was performed for categorical attributes like `Sensing Service` and `Security` to represent them as binary vectors. Non-numeric or categorical attributes were encoded and converted to improve the predictive capabilities of the Vector AutoRegression algorithm, prevent misinterpretation of ordinal data, and avoid bias in the model during training. Performing data preprocessing enhances the interpretability, consistency, data compatibility, and overall performance of the model.

5.2.3 Datasets

The User dataset encompasses meticulously curated data collection representing 90 distinct users. These users actively engage with IoT applications by requesting diverse sensing services, including Light, Temperature, Pressure, and Humidity measurements. The collection shows user interactions over 5 days, during which users operate within a defined geographic coordinate range (42.6657,-83.201; 42.6763,-83.217). To gauge the efficacy of the provided sensing services in meeting user expectations, this dataset is enriched with user-defined QoS specifications. These QoS parameters encompass crucial attributes like

desired throughput and response time, ranging from 1 to 10 seconds. In addition, Users can decide to request either a standard security or a high-security protocol for each service request.

The Device dataset The original ICRI dataset included environmental sensor readings from 19 heterogeneous sensing devices. To enhance scalability, 31 additional synthetic sensing devices were created, increasing the number of devices from 19 to 50. Additional device profiles were generated using statistical sampling to replicate the sensing abilities and QoS capabilities with similar distributions as the original devices. Hence, preserving the functional diversity within the dataset and ensuring its validity. The devices provided measurements for Light, Temperature, Pressure, and Humidity over 7 days, with at least two devices per square foot within the geographic range (42.6657,-83.217: 42.6667,-83.2306). Each device has varying QoS capabilities, with throughput and response times between 1 and 10 seconds, and has the ability to support user-defined security levels, either standard or high, for the sensing services.

5.2.4 Data Analysis

For the predictive model, we employ a semi-simulated dataset encompassing both service requests and sensor readings, spanning 22 weeks, with readings recorded every minute. To enhance efficiency and optimize computational resources, the temporal scope of our dataset was strategically aggregated by narrowing it down to a 7-day timeframe for device-related aspects and a 5-day timeframe for user-related aspects. The variable selection process encompasses key parameters critical for analysis and modeling. Using the Granger causality test, it was determined that the following variables within the data set: the ROI expressed as longitude and latitude, response time, throughput measurements, device energy level, and security preferences, can be used to predict the other variables,

like service request, for the time series prediction step. Furthermore, the Variable Inflation Factor (VIF) analysis determined the important predictor variables shortlisted for the IoT service behavior model, including sensing service requests, throughput, response time, and device energy level. These variables were found to be less correlated with each other ($VIF < 5$), making them suitable for inclusion in the predictive model without significant multicollinearity issues.

A stationarity test was executed to ensure the suitability of the chosen variables for modeling. Initially, Johansen's Cointegration test was performed to confirm the constancy of these variables over time. The test determines whether there is a statistically significant relationship between multiple time series variables, indicating the presence of cointegration. Cointegration suggests a long-term equilibrium relationship among the variables. With a 95% confidence level, we find that already selected variables have significant relationships. The Augmented Dickey-Fuller test is applied to affirm the stationarity of the variables. An initial examination of the dataset unveiled the presence of non-stationary variables, and using the first-order differencing technique effectively transformed these variables into stationary entities, rendering them suitable for integration into our modeling framework. After the stationary transformation, model estimation to validate the integrity and robustness was performed.

5.2.5 Evaluation Measures

To assess the accuracy of the predictive model, two distinct metric categories are evaluated. First is the Categorical Prediction Accuracy for User ID and Service ID (ID Forecasting), which are categorical labels. Classification accuracy (percentage of correctly predicted labels) is reported rather than continuous-error metrics. This ensures that a misclassification (e.g., predicting user 42) is counted appropriately rather than

producing an arbitrary numerical error. Second is the Continuous QoS Forecast Error: For continuous variables, namely throughput and response time, we retain MAE and RMSE [121]. These metrics serve as quantitative indicators of the model's predictive performance and recommendation capabilities for optimized resource management.

MAE eq. (5.7) measures the average of the absolute differences between predicted and actual values. It treats all errors equally, regardless of their magnitude. MAE is easy to interpret, with a measure of the average absolute prediction error.

RMSE is the square root of the average of the squared differences between predicted and actual values. It is in the same unit as the dependent variable, making it more interpretable. The RMSE is calculated as shown in eq. (5.8).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{5.7}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{5.8}$$

5.2.6 Results

5.2.6.1 User Variables Forecasting To determine the user's future service request, QoS, and security requirements, a forecast was performed on each of the required parameters, and the accuracy of the forecast was measured. The measured forecast accuracy values for future user requests are presented in Table 5.3. The accuracy of the user service forecasts is quite promising. The model excels at predicting which users will request services in the future and the geographic location where the request will likely be requested, as indicated by the consistently low error values across the accuracy, MAE, and RMSE metrics. This suggests that the forecasting model is precise in anticipating users' service needs and their preferred geographic coordinates.

Table 5.3: User Service Prediction Performance Metrics

Variable	Type	Metric(s)	Value(s)
User_Id	Categorical	Accuracy	97.8%
Service_Id	Categorical	Accuracy	92.4%
Secure_Id	Categorical	Accuracy	99.6%
Latitude	Continuous	MAE / RMSE	0.0227 / 0.0236
Longitude	Continuous	MAE / RMSE	0.0268 / 0.0275
Throughput	Continuous	MAE / RMSE	2.5833 / 3.3773
Response_Time	Continuous	MAE / RMSE	1.2465 / 1.7182

In terms of throughput and response time predictions, a slight underestimation in throughput on average is observed. The forecasts for response time are reasonably accurate. Nevertheless, both throughput and response time exhibit relatively high RMSE values, implying some variability in prediction accuracy across different data points. When predicting `Secure ID`, which reflects users' security preferences, the model forecasts values with high accuracy. This suggests the model excels in anticipating users' security preferences with high precision. The `Service ID` forecasts, indicating the type of service users might request in the future, show moderate errors; this is slightly expected as the variable was initially non-stationary and was transformed using the first-order differencing technique to make it stationary. However, while errors exist, they are not significantly large in absolute terms. For new users, the forecast accuracy and request fulfillment rates are measured and compared during the first ten interactions and after the transition to personalized profiles has taken place. In testing 30 new users with default profiles, QoS profile forecast produced a response time with MAE of 3.12s and a 61% fulfillment rate. After just ten interactions and retraining, the MAE fell to 1.24s, and the fulfillment rate jumped to 93%, aligning with the system's long-term performance.

5.2.6.2 Device Variable Forecasting The measured forecast accuracy values for future services to be completed by a device are presented in Table 5.4. The accuracy metrics provide valuable insights into the quality of the forecasts for various variables. For `Device ID`, `Latitude`, and `Longitude`, the forecasts exhibit high accuracy, with minimal errors, indicating precise predictions. However, there is a slight underestimation in `Longitude` that warrants consideration. `Throughput` and `Service ID` forecasts, while reasonably accurate, show signs of overestimation and underestimation, respectively, with moderate errors. `Response Time` forecasts demonstrate good accuracy overall, despite a positive bias. In contrast, `Secure ID` forecasts are the most accurate.

Table 5.4: Device Service Prediction Accuracy

Variable	Type	Metric(s)	Value(s)
Device_ID	Categorical	Accuracy	99.97%
Service_ID	Categorical	Accuracy	98.5%
Secure_ID	Categorical	Accuracy	100.0%
Latitude	Continuous	MAE / RMSE	0.0005 / 0.0005
Longitude	Continuous	MAE / RMSE	0.0012 / 0.0016
Throughput	Continuous	MAE / RMSE	1.8336 / 2.1633
Response_Time	Continuous	MAE / RMSE	4.2801 / 4.4382

5.2.6.3 Device Optimization Results Results of the IoT Device Optimization simulation process outlined in Algorithm 3, including outcomes for network lifetime, unfulfilled user requests, completed user requests, and device performance, are presented below. **Network Lifetime:** Each data point corresponds to a combination of an energy level and its corresponding network lifetime representing the time point during the simulation when any of the devices reached less than 10% of the initial energy levels. Network lifetimes

for energy levels at 140 mAh and 180 mAh are 110 minutes and 114 minutes, respectively. Indicating that at these energy levels, devices were available for 92% and 95% of the service duration (120 minutes), respectively. Reported network lifetimes for energy levels at 220 mAh and 260 mAh are both 0 minutes, indicating that devices were available the entire service duration, with none falling below the 10% threshold. This suggests a critical limitation in the network's ability to sustain operation beyond a certain amount of time with lower energy levels. It implies that higher energy levels result in longer network lifetimes, limiting the rapid depletion of resources or inefficiencies in network management.

Unfulfilled User Requests: as shown in Fig. 5.2 present the simulation results at each energy level for both predicted and non-predicted services: 140 mAh (4,32), 180 mAh (0,29), 220 mAh (0, 24), and 260 mAh (2,55), respectively. The data shows a clear trend where the number of unfulfilled non-predicted user requests increases as the energy levels increase. Comparing predicted unfulfilled requests with non-predicted requests helps assess the accuracy and reliability of the simulation in representing real-world scenarios. Understanding the distribution of unfulfilled user requests across different energy levels is essential for operational planning and resource management. It identifies bottlenecks or limitations in fulfilling user demands, allowing for targeted improvements or resource reallocation strategies.

Completed User Requests: are presented in Fig. 5.3 comparing the number of predicted and non-predicted user requests completed across different energy levels. At each energy level, the completed requests for predicted and non-predicted services are 140 mAh (123,95), 180 mAh (112,83), 220 mAh (140,99), and 260 mAh (155,102), respectively. The chart suggests that the system's predictions are generally aligned with the actual fulfillment of service requests. Predicted services report a higher number of fulfilled requests compared

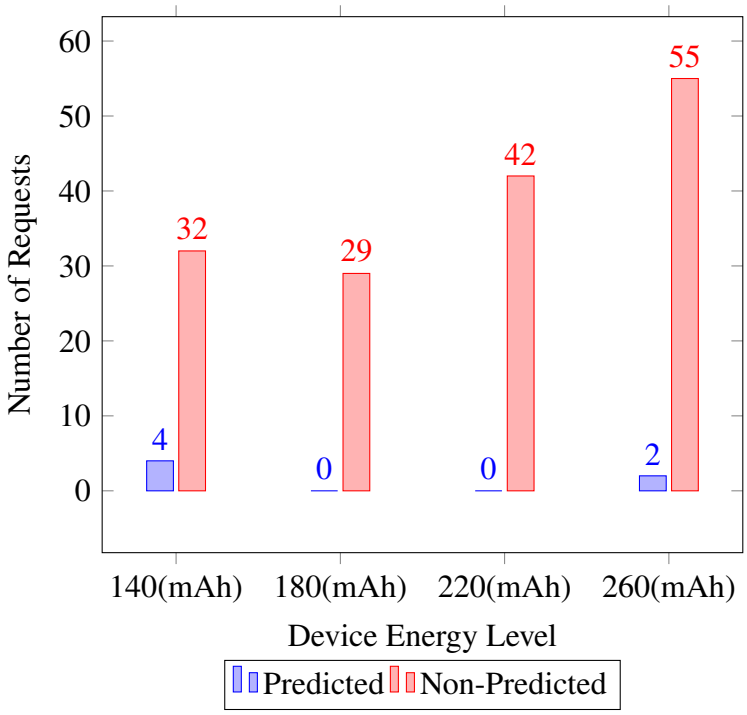

Figure 5.2: User Requests Not completed

to non-predicted services across different energy levels. In addition, it indicates that the system is effective in fulfilling predicted services.

Device Service Frequency is represented as a bar chart in Fig. 5.4 showing the number of requests and the number of devices for each service type (Humidity, Light, Pressure, Temperature). Throughout the simulation, the following number of sensing services were requested and fulfilled by the specified number of devices: Humidity:(142 requests, 14 devices), Light: (252 requests, 13 devices), Pressure: (225 requests, 16 devices), Temperature: (101 requests, 7 devices). The chart indicates varying demand levels for different service types, highlighting the importance of resource allocation and management based on service popularity. The analysis of user behavior and service usage patterns provides insights into service demand trends over time, helping in long-term resource planning and optimization.

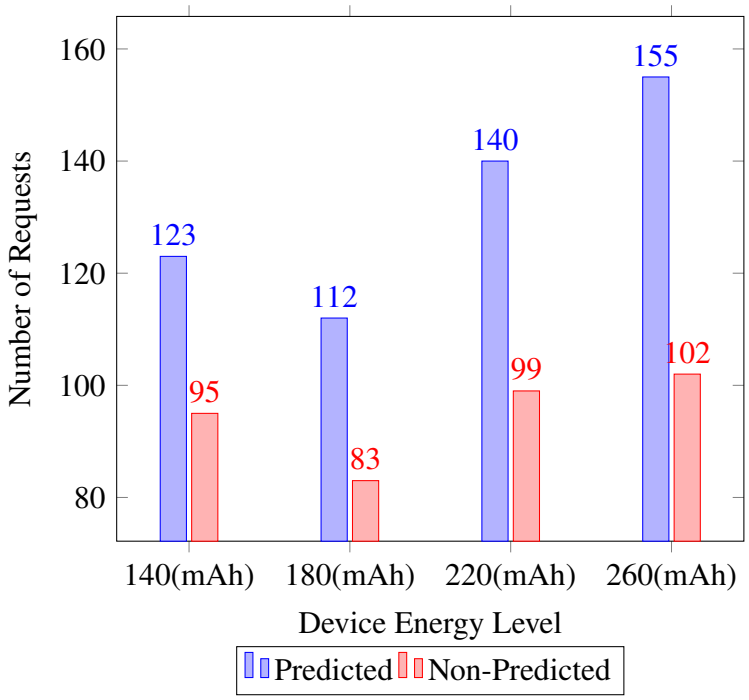

Figure 5.3: Completed User Requests

Device Performance is presented in Fig.5.5, depicting the number of services completed by each device during the simulation. Through the results, we see the efficiency of each device. Note devices D023, D025, and D046 provide up to 28 Light services, and D014 provides up to 24 Pressure services. This indicates these devices consistently demonstrate high efficiency by completing more services for the sensing type they provide, showing reliability and effective utilization of resources.

5.3 Discussions and Comparative Analysis

High Precision in User Service Forecasting: The accuracy metrics indicate that the forecasting model was good at predicting which users will request services `Service ID` and where they are likely to request them `Latitude and Longitude`. The precision is reflected in the consistently low error values across the MAE and Root Mean Square Error

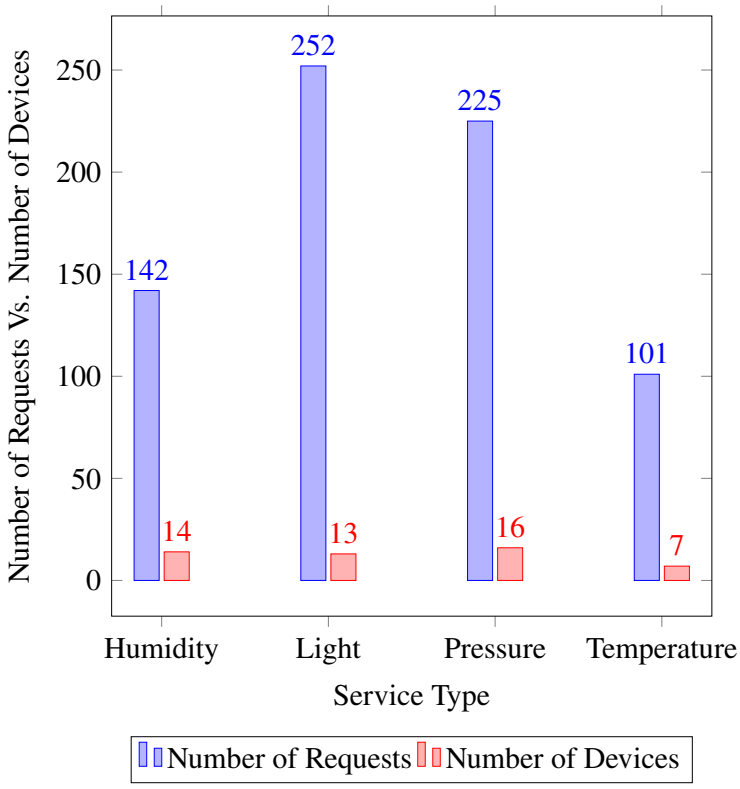

Figure 5.4: Number of Requests and Devices by Service Type

(RMSE) metrics. Essentially, the model is very good at anticipating users' service needs with a high level of accuracy.

Addressing Minor Underestimation Trends: While the model performs well, a minor trend of underestimation is observed in the forecasts for `Longitude`. Implying the model tends to predict slightly lower values for texttttLongitude than the actual values. Although minor, it highlights the need for ongoing monitoring and fine-tuning of specific aspects of the model to ensure it captures all trends accurately. It is crucial to investigate the root causes behind such trends and make adjustments.

Throughput and Response Time Predictions: The forecasts for `Throughput` and `Response Time` are reasonably accurate, even though there is a slight underestimation

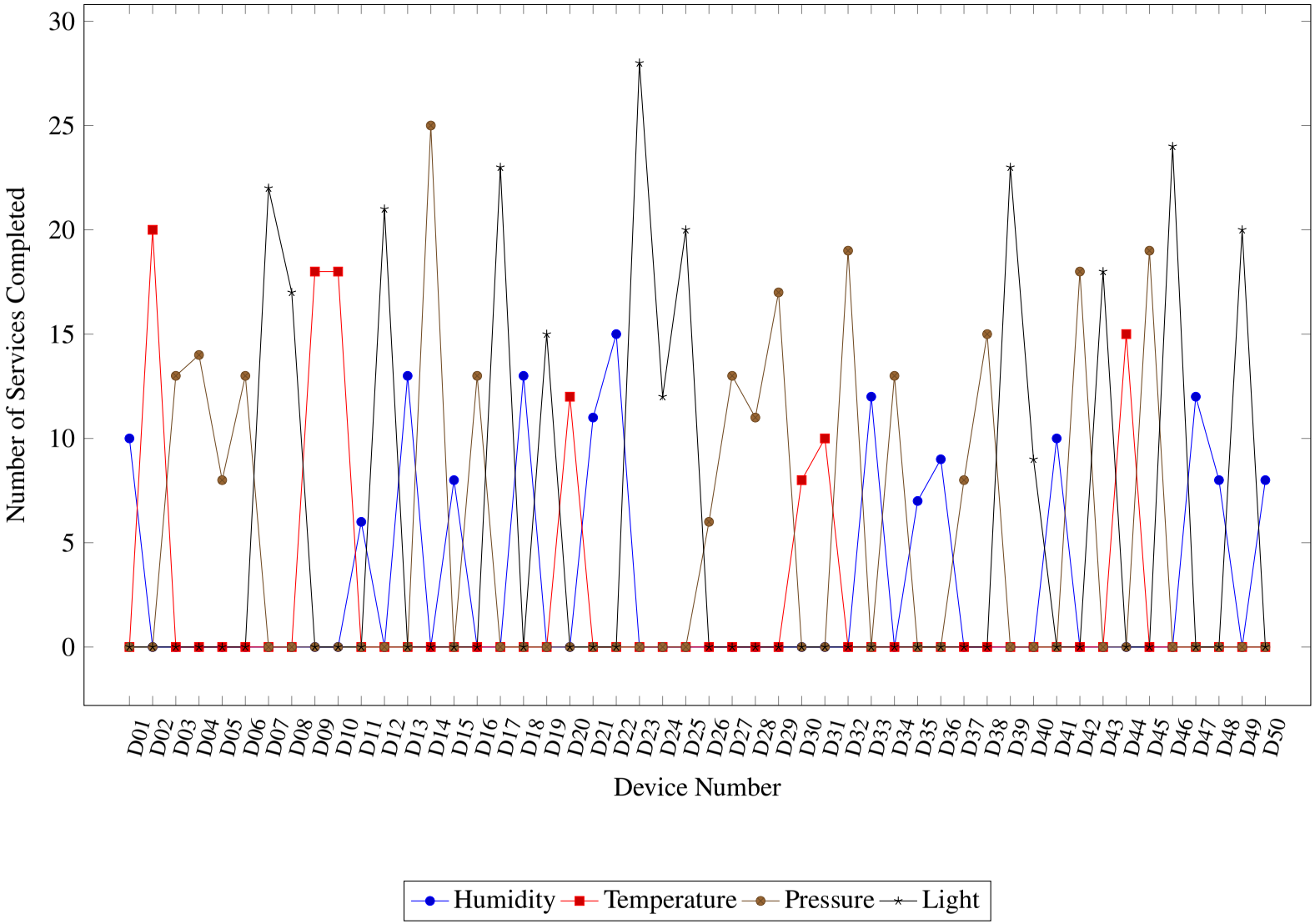

Figure 5.5: Stacked area chart showing the number of services completed by each device, categorized by service type.

trend observed on average. The relatively high RMSE values suggest that there is some variability in prediction accuracy across different data points. This variability might be due to external factors or complex relationships that the model should consider.

Accuracy in Predicting Secure ID: One of the significant successes of the model is in predicting `Secure ID`, which reflects users' security preferences. The model's forecasts for Secure ID are quite accurate, with low error values across all metrics. This suggests that the model excels in understanding and predicting users' security-related choices with some precision.

Service ID Predictions and Overestimation: Forecasts for `Service ID`, the type of service users might request in the future, show moderate errors. These errors are primarily characterized by a positive Mean Percentage Error (MPE), indicating a tendency for overestimation. However, the relatively low RMSE suggests that while errors exist, they might not be significantly large in absolute terms. Implying the model generally provides reasonably accurate predictions for `Service ID`, but it may need adjustments to mitigate the overestimation tendency. Table 5.5 with a snapshot of the predicted values,

Table 5.5: Sample of Predicted Values

User ID	Device ID	RT (s)	TP (s)	Service	Protocol
User 55	8	4.90	4.41	L	H
User 64	7	6.92	4.30	T	S
User 61	7	5.12	4.75	T	S
User 44	8	5.99	7.28	L	S
User 42	14	7.27	5.92	L	H
User 43	11	5.49	3.72	L	S

Legend: RT = Response Time, TP = Throughput, L = Light, T = Temperature, P = Pressure, Hu = Humidity, H = High, S = Standard

shows the predicted `UserID` at each service point, the predicted QoS preferences of the user for that service time (e.g., Response time, throughput, sensing service, and security protocol), and the recommended device that can fulfill that service given the predicted user QoS preferences. For instance, for the first service period, the model predicts that User 55 will request Light sensing service with Response Time and Throughput preferences at 4.90 seconds and 4.41 seconds, respectively, with a high-level security protocol. Based on the predicted user QoS preferences, Device 8, a light sensor, is recommended to fulfill a predicted user request involving ambient light monitoring with specific latency and security constraints. A practical deployment scenario is within a smart city block or campus where a user often requests ambient light data in the early hours of the day (e.g., during sunrise). The framework predicts this demand and recommends a nearby light sensor (e.g., Device 8) that meets the user's response time and security needs while conserving device energy. This ensures timely service and extends network lifetime in busy urban environments where sensor resources are shared.

Our forecasting model significantly enhances device management in IoT environments by accurately predicting user service needs and locations, allowing for efficient resource allocation and proactive IoT device maintenance, by reducing downtime and optimizing device management. Overall, our approach achieved a 95% fulfillment rate of user requests, surpassing the 85% of existing schemes such as [44]. Additionally, our framework increased network lifetime by 20% compared to [22]. These results demonstrate clear improvements that the predictive and recommendation-based framework offers in both efficiency and energy usage over traditional resource management methods. Furthermore, the reasonable predictions of network throughput and response times facilitate efficient data traffic management within IoT networks. Its accuracy in predicting `SecureID` reflects robust security management capabilities, vital for safeguarding IoT devices and data.

Addressing overestimation tendencies in `Service ID` predictions can further optimize resource allocation.

5.4 Conclusion and Future Work

The main technical contribution of this paper is a QoS-aware predictive framework that provides a novel integration of VAR time-series forecasting with a QoS-constrained recommender engine. It offers a proactive, learning-based allocation strategy that surpasses conventional approaches like task offloading and heuristic allocation to extend network lifetime and improve service availability. Other technical contributions include proactive device reservation and addressing the cold-start issue by onboarding new users and devices using generalized population-based QoS profiles. Ensuring there are no delays in the operability and effectiveness of the framework without sacrificing personalization of user services or device performance.

Demonstrated improvements in key performance indicators include 20% extension in network operational lifetime, and a 95% increase in fulfilled user requests compared to baseline approaches; efficient resource utilization, accurate service demand prediction with up to 95% accuracy, and consistent device performance. The use of hybrid datasets and realistic QoS simulations offers a robust validation ground for future IoT frameworks, especially where real-world user-service traces are unavailable. While the framework allows for controlled experimentation, it does not capture all real-world effects. Unpredictable RF interference, hardware-to-hardware performance variability, and dynamic user mobility are factors that introduce packet loss, uneven energy consumption, and intermittent connectivity that may affect the application of the framework in real-world scenarios.

The framework uses a shard-based architecture for horizontal scaling. By partitioning the IoT deployment into geographic RoIs, each RoI can run its instance of the VAR predictor

and recommendation engine. As devices and users increase, additional RoIs can be deployed on parallel edge servers or cloud nodes, maintaining near-linear computational scaling. This design supports millions of devices without overloading any single node. While currently focused on classical IoT architectures with lightweight predictive models and heuristics, future enhancements may benefit from emerging paradigms in quantum-aware computations [122] [123].

Future work will focus on enhanced IoT resource allocation and management to improve service availability and prolong device network lifetime using sustainable ambient energy sources like Radio Frequency (RF) Energy Harvesting (EH). An extension of the future work includes continuous model improvement, which aligns with the dynamic nature of IoT ecosystems, especially as it concerns security. Service prediction and recommendation in IoT resource management can be used to enhance security by incorporating encryption and data protection measures. This involves predicting services and recommending devices that comply with requested service encryption standards, preserving user privacy, and adapting recommendations based on the security context. By integrating these aspects, IoT systems can optimize resource management while ensuring data security and user privacy.

CHAPTER SIX

RO3: ENHANCED ENERGY MANAGEMENT WITH ENERGY HARVESTING

A significant challenge to service availability and network sustainability in IoT service provisioning is the energy constraints of IoT devices. With the ever-increasing number of IoT devices, EH has been leveraged to provide a continuous power supply to maximize IoT network lifetime and increase the number of provided services [124] [84] [33]. With the knowledge that the availability of harvested energy is affected by environmental interference and intermittent supply, several energy management solutions have been developed [35] [99] [125]. However, these solutions fulfill energy management needs from the viewpoint of the harvesting infrastructures, neglecting the equal relevance of changing user preferences to efficient energy management. These solutions also struggle with inefficient resource allocation and suboptimal energy utilization [126] [127]. By considering the impact of changing user preferences, this work proposes to improve existing energy management schemes using an optimized energy management framework that implements an NSGA-III multi-objective algorithm and a user-centric energy allocation algorithm. Leveraging RF energy harvesting to improve IoT service provisioning, the proposed framework will ensure long-term network sustainability, higher service request fulfillment, and extended device lifetime.

Acknowledging the existence of other genetic algorithms applicable for multi-objective optimization, such as MOEAD and NSGA-II [6], this work will adopt the use of NSGA-III because it is more suited for situations with substantial amounts of data, as is the case with IoT. Considering the variations in IoT user requests, unlike NSGA-II, NSGA-III is also able to effectively identify the diversity in the data provided, thereby providing a wider spread of optimal solutions for the objective functions [128] [129].

The contributions are as follows:

- To design an IoT network powered by RF energy harvesting for energy efficiency and sustainable operation.
- To maximize energy utilization and service request fulfillment by implementing a multi-objective optimization approach using NSGA-III, thereby reducing energy wastage.
- To dynamically allocate harvested energy based on real-time user service requests and network conditions using the optimal solutions from the NSGA-III optimization algorithm.
- To validate and evaluate the network performance based on energy efficiency, network lifetime, and service request fulfillment.

Implementing a multi-objective optimization framework using NSGA-III enables the simultaneous optimization of two main objectives: to maximize the usage of harvested energy and the number of service requests fulfilled. An increase in the number of services requested will naturally lead to higher consumption of energy by the devices fulfilling those requests. Multi-objective optimization balances the trade-off between these conflicting objectives based on specified constraints. In addition, NSGA-III is better suited to manage complexities such as high data dimension and dynamic solution spaces that arise within the IoT network [115]. In contrast to other multi-objective optimization methods like MOEAD and NSGA-II, NSGA-III maintains a uniform distribution of solutions by preserving the diversity in the high-dimensional IoT space for effective decision-making.

The proposed framework will consider real-time user requests, network constraints, and energy harvesting dynamics to optimize power distribution, ensuring that IoT devices remain operational even under varying user preferences and energy conditions. Details

about the framework design, the optimization algorithm, and the energy allocation strategy to be implemented are provided in the following sections.

6.1 Network Scenario

The network scenario depicts an IoT-based sensing environment spanning across different locations, including urban, semi-urban, and rural areas. Users desiring environment sensing services initiate service requests that have to be fulfilled by IoT sensing devices active within the environment.

These IoT devices typically run on a battery and require a consistent energy supply to continue to fulfill user requests over time. EH IoT devices within the environment can provide a constant energy supply to the sensing devices to sustain their operations by harvesting RF energy from nearby sources like telecommunication towers and WiFi stations. Fig. 6.1 describes the interactions between the different components within the environment.

Users provide their service requests, specifying their desired sensing services, service duration, and frequency alongside required QoS parameters like response time, throughput, and security level preferences. The requests are managed by a centralized request management system that assigns the requests to the IoT device capable of performing the sensing request. An optimization algorithm performs the assignment by matching QoS parameters and computing the required energy usage to perform the request. If the assigned sensing device lacks sufficient energy to complete the request, the system attempts to allocate additional energy resources to the device from an active EH device. The energy allocation algorithm is used to determine how much energy to assign to the sensing device, and from what EH device, based on available harvested energy and proximity.

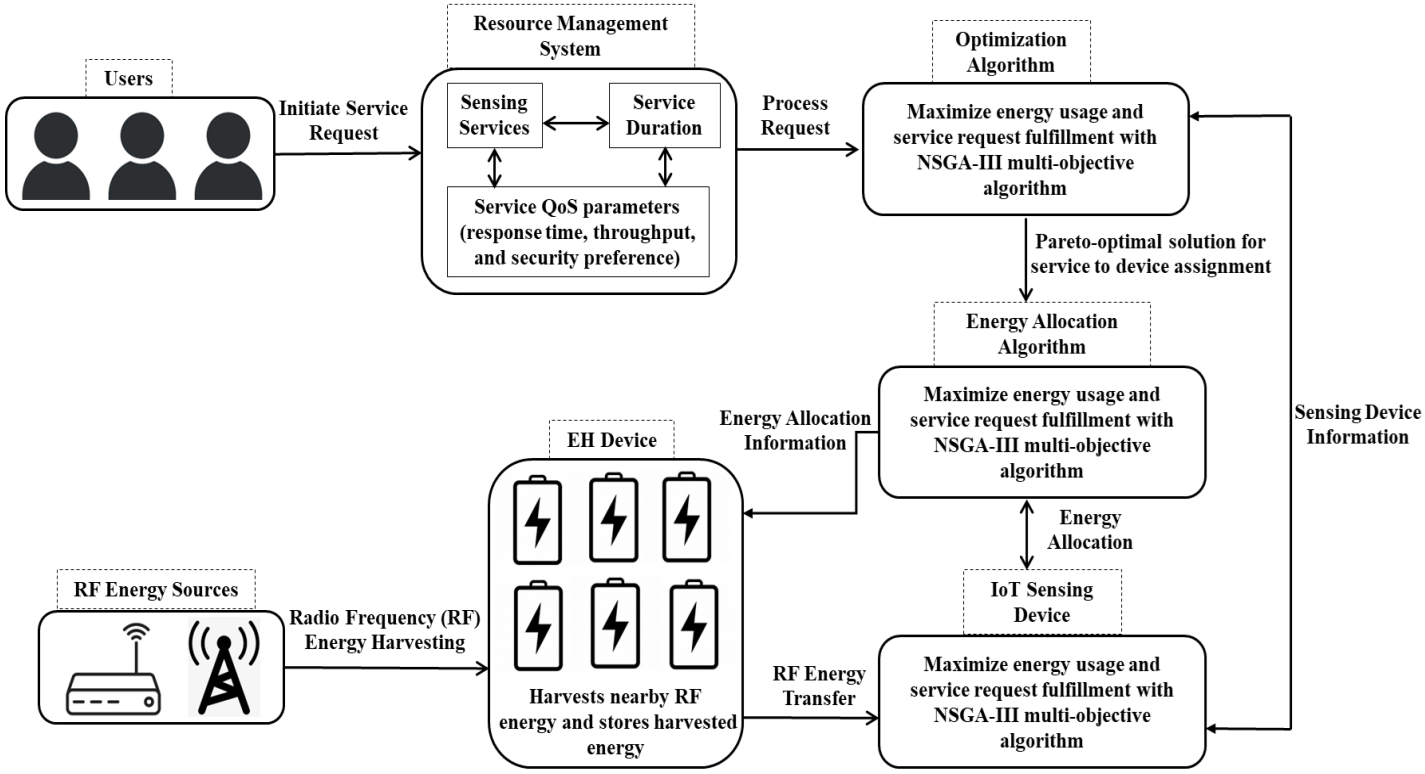

Figure 6.1: Network Scenario with RF Energy Harvesting from Telecom Towers and WiFi Stations for Service Provisioning in IoT

The framework will integrate an optimization algorithm that efficiently assigns user service requests to available sensing devices, with an energy allocation algorithm that efficiently manages the distribution of harvested energy to sensing devices to maximize service request fulfillment and extend network lifetime. The optimization and energy allocation algorithms are described next.

6.2 Optimization Algorithm

This work aims to achieve multiple objectives: to maximize harvested energy usage and maximize service request fulfillment within an IoT environment. Given that as users submit more service requests, energy usage invariably increases due to high processing demands, the objectives are potentially conflicting. To avoid prioritizing one over the other, effectively managing these objectives within a complex IoT network requires a robust multi-objective optimization strategy. Therefore, NSGA-III, a non-dominated multi-objective optimization algorithm, is considered appropriate for this work. NSGA-III is designed to handle multiple-objective optimization problems by exploring the trade-offs and finding a set of solutions across multiple objectives. The set of solutions is referred to as the Pareto front, which provides the best results known as Pareto-optimal solutions.

NSGA-III implementation runs iteratively by evolving through a population of candidate solutions through selection, crossover, mutation, and replacement to identify optimization solutions to conflicting objectives. The optimization problem is set up to maximize energy usage and increase the number of service requests completed while satisfying specific constraints. The goal is to find optimal ways to balance efficient energy usage with service fulfillment. The main steps in the implementation include initializing the solution population set, where each solution is the possible assignment of service requests to the sensing devices.

Each solution S can be represented as in eq. 6.1:

$$S = \{(R_1, D_1), (R_2, D_3), ..., (R_n, D_k)\} \tag{6.1}$$

. Where R_i denotes a user request, and D_j the sensing device it is assigned to. The solutions are then ranked based on non-dominant sorting, to ensure solutions providing the best trade-offs between efficient energy usage and service fulfillment are prioritized. To guarantee solution diversity, the crowding distance for each solution is determined, and the solutions that are more dispersed over the different objective functions are prioritized. Thereby ensuring a uniform distribution on the Pareto front, providing a range of trade-off options for decision-making. Two solutions could be combined to create a new service-device assignment; this step is known as crossover. On the other hand, a process known as mutation randomly modifies already existing solutions to generate new solutions that provide more balanced service-device assignment configurations to meet the objectives.

For crossover and mutation, solution rankings and their computed crowding distances are taken into account. The best solutions from the current set of solutions are selected for the next iteration based on the non-dominant sorting and crowding distance. By replacing the old solution population set with a new solution set, the algorithm ensures only high-quality service-to-device assignment solutions are considered throughout the process, while allowing the exploration of new assignment strategies. The algorithm continues to iterate until it produces a final set of optimal solutions, representing the different possible trade-offs between efficient energy usage and service fulfillment. This shows the capability of NSGA-III for dynamic optimization of the multi-objective problem posed in this work. The algorithm can ensure efficient allocation of sensing devices to user requests and maintain a balance between service sustainability and QoS.

The NSGA-III algorithm is presented in Algorithm 4. The formulation of the optimization problem, including the objective functions to be achieved, the constraints that need to be satisfied, and the decision variables to be considered by the multi-objective optimization algorithm in optimizing the conflicting objectives, is described next.

6.2.1 Objective Functions

Prioritizing solutions that lead to the efficient usage of energy by devices within the IoT network ensures less energy wastage. In addition, configurations that optimize the fulfillment of more user requests during the network lifetime also help ensure continuity in the IoT services and the overall network efficiency. Therefore, the first goal is to maximize the use of the harvested RF energy (U) as represented in eq. 6.2. The second goal, given in eq. 6.3, is to maximize the total number of user service requests fulfilled (R_f) by IoT sensing devices. The objective function is said to be a multi-objective optimization problem when both goals are to be achieved simultaneously.

Optimization Objectives

- **Maximize Usage of Harvested Energy (U):**

$$U = \sum_{d=1}^{N} E_u[d] \tag{6.2}$$

  where d represents each sensing device, $E_u[d]$ is the energy utilized by each device d, and N the total number of devices providing sensing services.

- **Maximize Requests Fulfilled (R_f):**

$$R_f = \frac{\sum_{d=1}^{N} R_{fulfilled}[d]}{\sum_{d=1}^{N} R_{total}[d]} \tag{6.3}$$

  where $R_{fulfilled}[d]$ is the number of requests fulfilled by device d and $R_{total}[d]$ are the total number of requests received by device d.

These optimization objectives are subject to constraints such as energy usage and energy storage. To guide resource and workload distribution to the devices, decision variables are also introduced.

6.2.2 Constraints and Variables of Objective Functions

The constraints enforced on the optimization functions are as follows:

Constraints on energy usage: To ensure device operations are sustained, a constraint is placed on the energy usage of devices $E_u[d] \leq E_h[d]$, such that E_u, energy consumed by a device d during service fulfillment cannot exceed the energy harvested E_h.

Constraints on energy storage: Efficiency in energy harvesting is maintained and energy wastage is prevented by placing the constraint $EA_D \leq$ Capacity, to ensure energy accumulated EA by a harvesting device D does not exceed its storage capacity.

Decision variables: For proper distribution of harvested energy based on availability and as needed by the devices, we use the variable $A[d]$ to depict the amount of energy allocated to a device d. The battery capacity and service demand of the device are evaluated for energy allocation. The variable $R_{assign}[d]$ is also used to keep track of the number of user service requests assigned to a device to prevent bottlenecks. Assigning user service requests to a device depends on the device's sensing capability, battery level, and device QoS properties. This avoids overloading the device while user request fulfillment is being maximized. The complete optimization problem formulation for NSGA-III is shown below.

Optimization Problem Formulation for NSGA-III

$$\text{Maximize:} \quad \begin{aligned} U &= \sum_{d=1}^{N} E_u[d], \\ R_f &= \frac{\sum_{d=1}^{N} R_{fulfilled}[d]}{\sum_{d=1}^{N} R_{total}[d]}, \end{aligned} \tag{6.4}$$

Subject to these constraints :

$$\begin{aligned} E_u[d] &\leq E_h[d], \quad \forall d \\ R_{fulfilled}[d] &\leq R_{total}[d], \quad \forall d \\ S_d &\leq \text{Capacity}, \quad \forall d \end{aligned} \tag{6.5}$$

Based on Decision Variables:

$$\begin{aligned} &A[d] \\ &R_{assign}[d] \end{aligned} \tag{6.6}$$

The formulation of our multi-objective optimization balances efficient energy usage and service request fulfillment in an IoT environment while considering real-world constraints and variables using the NSGA-III algorithm for sustainable IoT network management.

Here we summarize the steps in the NSGA-III multi-objective optimization presented in Algorithm 4. The algorithm receives as input the service request and sensing device information. In steps 1 and 2, data pre-processing is performed by encoding categorical variables for efficient processing and normalizing values of features critical for the optimization process to prevent bias (lines 1 to 6). Then the linear regression model is trained to predict energy consumption and service response time based on service request parameters (lines 7 to 9). In step 3, the optimization problems formulated in equations (6.2) to (6.6) are defined alongside the constraints and decision variables (lines 10 to 13). The NSGA-III algorithm is then configured and initialized with a population of solutions. Crossover and mutation strategies, and reference directions to guide the search for the Pareto-optimal solutions are set in step 4 (lines 14 to 19). In step 5, the optimization algorithm is run by iterating over multiple generations, evaluating the objective functions, and fine-tuning the solutions (lines 20 to 25). The Pareto-optimal solutions are extracted

Algorithm 4 Multi-Objective Optimization Algorithm using NSGA-III

Input: IoT service request and sensing device dataset
Output: Pareto-optimal solutions for energy usage and user request maximization
Step 1: Data Preprocessing
Load dataset and drop unnecessary columns
Encode categorical variables using Label Encoding
Normalize Energy Consumed (joules) and Response Time (seconds)
Step 2: Train Regression Models
Train the regression model to predict energy consumption
Train linear regression model to predict response time
Step 3: Define Optimization Problem
Define decision variables X including energy, throughput, service duration, and service frequency
Define objective functions:

- Maximize f_1: Equation 6.2
- Maximize f_2: Equation 6.3

Set variable bounds from dataset min-max values
Step 4: Configure NSGA-III Algorithm
Set population size to 100
Use Latin Hypercube Sampling (LHS) for initialization
Apply Simulated Binary Crossover (SBX) with $p_c = 0.9, \eta = 15$
Apply Polynomial Mutation (PM) with $\eta = 20$
Generate reference directions for NSGA-III
Step 5: Run Optimization
for each generation in 200 generations **do**
    Generate offspring using crossover and mutation
    Evaluate objective functions for each individual
    Perform non-dominated sorting and update population
end for
Step 6: Extract Pareto-Optimal Solutions
Retrieve non-dominated solutions
Visualize Pareto Front
Compare optimized solutions with dataset averages
Identify best solutions:
Lowest energy consumption, Fastest response time, Best trade-off
Return: Optimized decision variable values and objective function results.

for comparison with the dataset averages to identify the best solutions in Step 6 (lines 26 to 30). The output of the algorithm is returned as results of the objective function, which are the Pareto-optimal solutions for allocating sensing devices to user requests (line 31).

With the NSGA-III multi-objective optimization algorithm, the framework can maximize energy usage and fulfill user requests with specified constraints. While NSGA-III provides an optimal set of decision variables for energy usage and request fulfillment, the framework still requires an effective energy allocation strategy for the efficient distribution of harvested energy to sensing devices in real-time. The energy allocation algorithm designed to dynamically allocate harvested energy to devices based on energy consumption demands of the devices and network constraints is discussed in the following subsection.

6.3 Energy Allocation Algorithm

A novel energy allocation algorithm is proposed in this work to build upon the optimization algorithm by using the Pareto-optimal solutions to determine how it will allocate harvested energy to improve usage efficiency. The energy allocation algorithm is described in Algorithm 5. Allocation of harvested energy is calculated based on the QoS requirements specified with the user's service request, such as the type of sensing request, service duration, service frequency, response time, throughput, and security preference, as well as the calculated distance between the harvested energy source and the sensing device. The appropriate device is matched to a user request based on the user's QoS parameters. Then the total amount of energy required to fulfill the request E_{req} is determined based on the data transmission energy requirement of the device E_{tx}, in addition to the energy required per total packet for the specific sensing request $N_{\text{packets}} \cdot E_{\text{sense}}$ as shown in eq. 6.7.

The energy required E_{req} for a sensing device to fulfill a user request is the sum of the transmission energy and the energy consumed during data sensing, given by:

$$E_{\text{req}} = E_{\text{tx}} + N_{\text{packets}} \cdot E_{\text{sense}} \tag{6.7}$$

where E_{tx} is the transmission energy, E_{sense} is the sensing energy per packet. N_{packets} is the total number of data packets transmitted, calculated as:

$$N_{\text{packets}} = \frac{T_{\text{dur}}}{T_{\text{freq}}} \tag{6.8}$$

where T_{dur} is the total service duration and T_{freq} is the service frequency. Additional energy costs are incurred based on response time R and throughput T, formulated as:

$$E_{\text{extra}} = \alpha R + \beta T \tag{6.9}$$

where α and β are weighting factors dependent on the security level requested by the user. These weights represent how much additional energy is required for every second of response time R, and the energy consumption per unit of data throughput T, respectively. the values for α and β range from 0.01 J/s; 0.05 J/Mbps for low security with minimal energy overhead, to 0.05 J/s; 0.1 J/Mbps for standard security with moderate energy overhead, and 0.1 J/s; 0.2 J/Mbps for high security with high energy overhead. The values for α are lower compared to β as energy consumption is mainly based on the amount of data transmitted and not on the delay experienced during transmission.

If the sensing device lacks sufficient energy to complete the request, energy transfer is attempted from nearby RF energy harvesting devices, as in Eq. 6.10. The amount of energy transferred E_{trans} is determined by:

$$E_{\text{trans}} = \min(E_{\text{stored}}, \gamma \cdot E_{\text{req}} \cdot \eta) \tag{6.10}$$

where E_{stored} is the available energy in the harvesting device storage, γ is a fraction of the requested energy that can be transferred, and η is the transfer efficiency.

6.3.1 Energy Allocation Key Decision Variables

In Algorithm 5, energy allocation is a critical component, especially when considering the impact of available energy in devices and the need to optimize energy-harvesting resources. The optimization decision variables returned in Algorithm 4 guide the energy allocation process, ensuring that energy requirements for fulfilling user requests are efficiently managed. Specifically, the optimization process will help adjust energy allocation based on device energy levels and the available resources, ensuring that devices are used optimally and energy is allocated efficiently.

The key decision variables for energy allocation include the device energy level E_d, which represents the current energy level of the device d, the energy required e to fulfill the request r, and the energy harvested E_h from RF energy sources such as Wi-Fi stations or telecom towers. The optimization variable, A_d, is the allocation ratio for energy transfer, defined as:

$$A_d = \frac{E_h}{E_d + E_h} \tag{6.11}$$

where A_d indicates the proportion of harvested energy allocated to the device based on its current energy state and the available harvested energy.

The optimization model works to minimize energy consumption and maximize request fulfillment efficiency. The energy required for a device d to fulfill a request r is adjusted by a factor of A_d, as follows:

$$E_{\text{adjusted}} = e \times (1 - A_d) \tag{6.12}$$

where E_{adjusted} is the adjusted energy requirement after considering the optimization variable. This ensures that energy is allocated based on the available resources, maximizing the overall performance of the network and prolonging its lifetime. The optimization ensures that devices with lower energy levels receive higher priority for energy allocation, thereby maximizing the fulfillment of requests while optimizing network efficiency. The following paragraph describes the process followed by the energy allocation algorithm.

Algorithm 5 presents the energy allocation algorithm simulation that begins with a system initialization with the user service requests, sensing, and harvesting devices datasets. The expected output of the algorithm is the evaluation metrics for network lifetime and energy usage efficiency. Pre-processing to normalize the geographic positions of sensing and harvesting devices is performed (lines 1 to 3). Each service request is then matched with a suitable sensing device by computing the energy required by the device for the service in eq. 6.7. If $E_{\text{req}} \leq E_d$, the device is assigned to the request. Otherwise, if energy harvesting is enabled, energy transfer to the sensing device is attempted using eq. 6.10 (lines 4 to 14). Based on its proximity to the nearest energy harvesting device. eq. 6.11 determines if the sensing device now has sufficient energy to proceed, and tracks the battery levels and energy consumption of the sensing device (lines 15 to 23). If no energy has been harvested to transfer to the sensing device, it is marked inactive (lines 24 to 25). Otherwise, if energy harvesting is disabled, the current energy level of the device is used to determine the match, and the number of fulfilled or unfulfilled requests for the device is recorded(lines 26 to 32). Key metrics like network lifetime, service request fulfillment, and energy usage efficiency are tracked throughout the process. The simulation terminates when all user requests have been processed or once the simulation time is completed (line 34). The results of the evaluation metrics are compared for when energy harvesting was performed during the simulation and when it was not (line 35).

Algorithm 5 Energy Allocation Algorithm with or without Energy Harvesting

Input: User requests, sensing devices, and RF energy harvesting devices
Output: Evaluation Metrics for Network Lifetime and Energy Usage Efficiency

Normalize positions of sensing and RF devices
for *each user request r* **do**
    Find matching sensing devices D_s
    Compute required energy: $E_{\text{req}} = E_{\text{tx}} + N_{\text{packets}} \cdot E_{\text{sense}}$
    if $E_{req} \leq E_d$ *for any* $d \in D_s$ **then**
        Assign device d to request r
    end
    else
        if *Energy harvesting is enabled* **then**
            Attempt energy transfer: $E_{\text{trans}} = \min(E_{\text{stored}}, \gamma \cdot E_{\text{req}} \cdot \eta)$
        end
    end
end
if *Harvesting simulation is enabled* **then**
    for *each time step t* **do**
        Harvest and transfer energy to nearby sensing devices ($dist \leq max\_range$)
        Update battery levels, check for request fulfillment
    end
    for *each device d lacking energy* **do**
        Transfer from nearby RF devices if available: $A_d = \frac{E_h}{E_d + E_h}$
        if *No energy available* **then**
            Mark device as inactive
        else
            Skip harvesting, use initial energy
            Track request fulfillment
        end
    end
end
Track metrics: fulfilled/unfulfilled requests, energy data, battery levels, lifetime
End simulation: on a defined condition
Compare results: with vs without energy harvesting

The two algorithms presented above are used to efficiently optimize the objectives of the proposed framework to provide an enhanced energy management and distribution system in IoT networks. Integration of decision optimization variables with dynamic energy allocation allows the system to easily adapt to the complex dynamics within the IoT environment, such as varying user requests and availability of harvested energy. Thereby ensuring optimal and continuous operation of devices for a longer time frame. The performance and efficiency of the network, based on the number of completed user requests, energy usage efficiency, and device network lifetime, are assessed throughout the simulation. The results from the framework proposed under this research objective, RO3, are discussed in the following section.

6.4 Results and Analysis

This section discusses the simulation process and results. First, a description of the dataset used is provided, then the simulation setup is described, followed by a discussion of the key metrics for evaluating the system's performance. Thereafter, the results of the simulation for three different scenarios are analyzed. The results compare the number of user service requests fulfilled and not fulfilled by the proposed framework, the energy usage efficiency of the proposed framework, and the network lifetime. The section concludes with a comparative analysis of the results with existing solutions for energy management in an IoT environment.

6.4.1 Dataset

User data: A minimum of 500 and a maximum of 1000 distinct user data points were curated for the simulation. To depict a real sensing environment setting with diverse environmental conditions, each user is associated with a geographic location within the

coordinate range (42.6657,-83.201; 42.6763,-83.217). The number of user service requests is assumed to be between 1 and 10 requests per user to mimic realistic variations in service demands. In addition to their sensing requests, users provide input on their QoS requirements, such as response time and throughput, ranging between 1 and 10 seconds, replicating typical QoS requirements in high-frequency environmental monitoring applications. To avoid allocating resources to a single request for too long, the expected duration of service is assumed to be no more than 60 minutes per service request. In addition, users' specified service frequency is considered to be between 1 and 15 minutes intervals, which is practical for periodic data collection. The user dataset also includes user security preferences of either low, standard, or high. The security preference is considered a constraint when assigning service requests to devices.

Sensing device data: The sensing device dataset is assumed to contain a minimum of 100 and a maximum of 1000 devices representative of a scalable deployment. Each of these devices provides a specific sensing service measurement, such as Light, Temperature, Pressure, and Humidity. In addition to delivering sensing services, devices also support user QoS requirements. To evaluate the energy needs of each device, the energy consumed by each device for data transmission is computed based on its QoS capabilities. This value is assumed to vary between 5 and 20 joules per transmission, accounting for the varying energy consumption patterns in IoT devices [130] [131]. The sensing devices are also distributed in a similar geographic location as the users, defined by the geographic coordinate range (42.6657, -83.217; 42.6667, -83.2306). Thereby ensuring the geo-spatial alignment between service demands and the sensing devices is realistic.

RF energy harvesting device data: A minimum of 10 and a maximum of 100 harvesting devices are considered in the simulation. The harvesting devices are deployed within the same region as the users and sensing devices. However, they are placed in

different locations with different RF energy sources to simulate various levels of RF energy availability. For instance, harvesting devices deployed in rural areas will predominantly harvest energy from telecoms towers. In contrast, harvesting devices deployed in urban and semi-urban areas will rely primarily on harvesting from Wi-Fi stations around them. This distribution reflects realistic variations in RF energy density and helps to assess the impact of energy harvesting across different environments on system performance.

6.4.2 Simulation Setup

The simulation models the interaction between 10 to 100 energy harvesting devices and 100 to 1000 IoT sensing devices within a 1000 square meter area over 24 hours, with 500 to 1000 users making service requests. The simulation was performed using three different scenarios. First, the number of sensing devices and user requests were varied, keeping the number of harvesting devices fixed. In the second scenario, the number of sensing devices varied, and the number of user requests and harvesting devices was constant. Finally, the number of active harvesting devices was varied, and other parameters remained fixed in the third scenario. The various scenarios aim to capture the dynamics of energy harvesting, energy consumption, and service request fulfillment in various environmental settings.

6.4.2.1 Factors Impacting Energy Harvesting and Distribution

Each of the EH devices is initialized with a simulated battery capacity of 100 Watt-hours (Wh) to store harvested energy and to simplify the comparison within the system. These devices are meant to collect energy from Radio Frequency (RF) signals emitted by nearby telecom towers and WiFi transmitters. During energy harvesting, the harvesting efficiency of the device is the percentage of the harvested energy that can be converted to electrical energy for use by IoT devices. For RF energy, the harvesting efficiency is typically 50% in commonly used harvesters, and at best 70% for optimized devices [132] [126].

Moreover, the type of environment in which the IoT network is deployed also introduces unique factors that cause signal degradation, thereby influencing the energy harvesting efficiency. For example, *urban environments* have high signal degradation or loss of signal strength due to more physical infrastructure, taller buildings, and other obstructions like interference from other RF devices. This causes the energy harvesting efficiency to drop; as such, the harvesting efficiency is set to 50% in this environment to reflect the impact of lost signal strength. However, the availability of RF energy in urban areas is more due to the increased use of cellular and WiFi devices, which makes up for the low harvesting efficiency.

Whereas *semi-urban environments* consist of more open spaces and fewer physical infrastructures, causing low to medium interference for RF energy harvesting. RF signals undergo moderate signal obstruction, and the energy harvesting slightly improves compared to an urban environment; hence, harvesting efficiency is set at 60%. Finally, *rural environments* have the least obstructions, and RF signals travel longer distances without significant loss. This results in minimal signal loss, and energy harvesting efficiency is at the best level of 70%. These environmental factors will be used to adjust the rate at which the EH devices harvest energy throughout the simulation.

6.4.2.2 Factors Impacting Service Fulfillment The simulation involves IoT sensing devices responsible for fulfilling user service requests. Each sensing device has a specific energy consumption profile, which includes two factors. First, the *energy consumption for initial connection and each user request*. The energy consumed by each device to establish a connection with the network is set at 0.1 Wh, based on typical power consumption rates for low-power IoT devices [133]. Each service request requires energy to be processed, and this consumption varies depending on the complexity of the request and the level of

service needed. The total energy consumption for each request is computed using eq. 6.7 as described in section 5.1. The second factor is *user preferences, QoS, and security levels*. Users in the simulation make service requests by specifying their preferences for service frequency and duration, along with their required QoS Security levels, which influence the energy consumption of IoT devices. Low QoS requests may require minimal computational resources, standard QoS requests keep energy demand at reasonable levels, while high QoS requests may demand more energy for processing and response time. These preferences can impact the ability of IoT devices to fulfill requests if energy availability is limited.

The overall goal of the simulation is to evaluate how effectively energy harvested by EH devices can be distributed to IoT sensing devices, enabling them to meet the varying demands of user service requests across different environmental settings, while trying to maintain a high user service request fulfillment rate.

6.4.3 Key Performance Metrics

The performance of the IoT network is evaluated using several key metrics that provide insights into the system's efficiency, sustainability, and ability to fulfill user requests. These metrics include the network lifetime, user requests fulfilled and unfulfilled, and energy usage efficiency, and are defined as follows.

Network Lifetime as expressed in eq. 6.13 is calculated by identifying the time it takes for the first sensing device to reach a battery level of zero, taking into account the energy harvested over time and the energy required to process and fulfill user requests. This is crucial for understanding the sustainability of the IoT network and depends on factors such as energy consumption, energy harvesting, and the number of active devices.

$$\text{Network Lifetime} = \min\left(\frac{E_{\text{battery, initial}}}{P_{\text{device}}(t)}\right) \tag{6.13}$$

where $E_{\text{battery, initial}}$ is the initial energy stored in the battery of the device. $P_{\text{device}}(t)$ is the energy consumption rate of the device over time t.

User Requests Fulfilled refers to the number of user requests that are successfully addressed by sensing devices. It is an important measure of how well the network is meeting the service demands in eq. 6.14, the calculation of user requests fulfilled involves counting the requests that are processed and completed within the requested service parameters, including service duration, throughput, response time, and energy constraints. The following equation gives this:

$$\text{User Requests Fulfilled} = \sum_{i=1}^{n} \alpha_{\text{fulfilled}}(r_i) \tag{6.14}$$

where $\alpha_{\text{fulfilled}}(r_i)$ is an indicator function that equals 1 if the i-th user request r_i is successfully fulfilled and 0 otherwise. n is the total number of user requests.

User Requests Unfulfilled is the number of requests that cannot be serviced due to insufficient energy, lack of suitable devices, or other network limitations. This metric is essential for understanding the gaps in network performance and resource allocation. The number of unfulfilled requests is calculated in eq. 6.15 as follows:

$$\text{User Requests Unfulfilled} = \sum_{i=1}^{n} (1 - \alpha_{\text{fulfilled}}(r_i)) \tag{6.15}$$

where $1 - \alpha_{\text{fulfilled}}(r_i)$ represents an unfulfilled request, indicating that the request could not be completed.

Energy Usage Efficiency evaluates how effectively the energy is utilized in the network. It compares the total energy harvested from RF energy sources with the energy consumed by the sensing devices to process user requests. The energy efficiency is calculated as a percentage of the ratio of energy harvested to energy consumed, expressed as:

$$\text{Energy Usage Efficiency} = \left(\frac{E_{\text{harvested}}}{E_{\text{consumed}}} \right) \times 100 \tag{6.16}$$

where $E_{\text{harvested}}$ is the total energy harvested by RF energy harvesting devices. E_{consumed} is the total energy consumed by the sensing devices to fulfill user requests. eq. 6.16 quantifies the efficiency of energy usage, with higher values indicating better utilization of available energy sources.

6.4.4 Results and Discussions

The range of possible outcomes and efficiency of the proposed framework in situations with varying numbers of user requests, sensing devices, and harvesting devices are determined by conducting simulations for three scenarios. The results of the simulations are presented in the following sections.

6.4.4.1 Scenario 1: Varying Sensing Devices and User Requests

In this scenario, the number of available harvesting devices is set to 10. The number of sensing devices varied from 100 to 900, in increments of 200. The number of users also varied from 100 to 900 in increments of 200, with each iteration having a different number of user requests ranging from 4 to 10 per user, with increments of 1. As a result, the total number of user requests for each sensing device iteration is as follows: 100 sensing devices - 4900 user requests, 300 sensing devices - 14700 user requests, 500 sensing devices - 24500 user requests, 700 sensing devices - 34300 user requests, and 900 sensing devices - 44100 user requests. The results for the total number of user requests fulfilled and unfulfilled, both with energy harvesting (EH) and with no energy harvesting (NEH), are presented in Tables 6.1 and 6.2.

Given in Fig. 6.2, as the number of sensing devices increases, with energy harvesting, the number of user requests fulfilled increases significantly because the available devices can serve more users simultaneously. This results in a more efficient network that can handle higher loads without missing too many requests. The EH scenario consistently outperforms that of NEH presented in Fig. 6.3, meaning that energy harvesting improves

Table 6.1: Fulfilled and Unfulfilled User Requests With Energy Harvesting (EH)

Sensing Devices (SD)	**100**	**300**	**500**	**700**	**900**
Total User Requests (TUR)	**49000**	**14700**	**24500**	**34300**	**44100**
Fulfilled (EH)	4853	14581	24278	33978	43732
Unfulfilled (EH)	47	119	222	322	368

Table 6.2: Fulfilled and Unfulfilled User Requests With No Energy Harvesting (NEH)

Sensing Devices (SD)	**100**	**300**	**500**	**700**	**900**
Total User Requests (TUR)	**49000**	**14700**	**24500**	**34300**	**44100**
Fulfilled (NEH)	4663	13969	23310	32692	42022
Unfulfilled (NEH)	237	731	1190	1608	2078

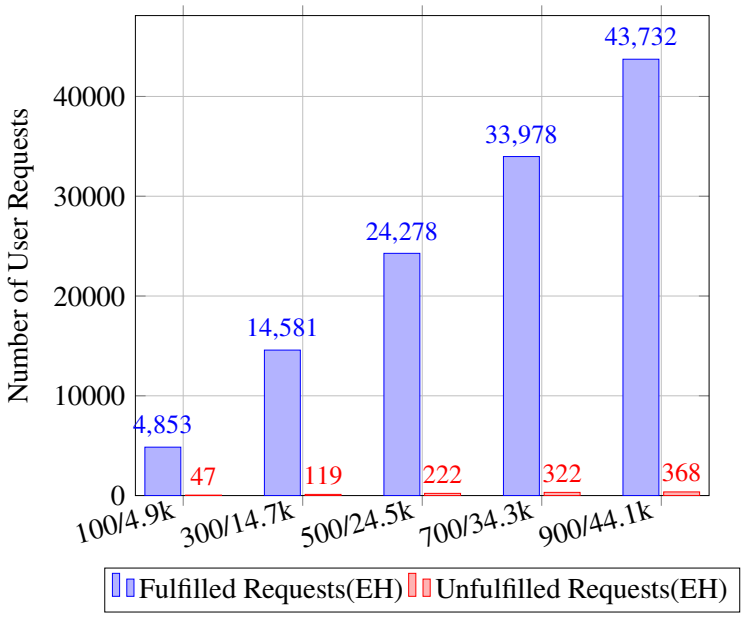

Figure 6.2: Number of Fulfilled and Unfulfilled User Requests With Energy Harvesting

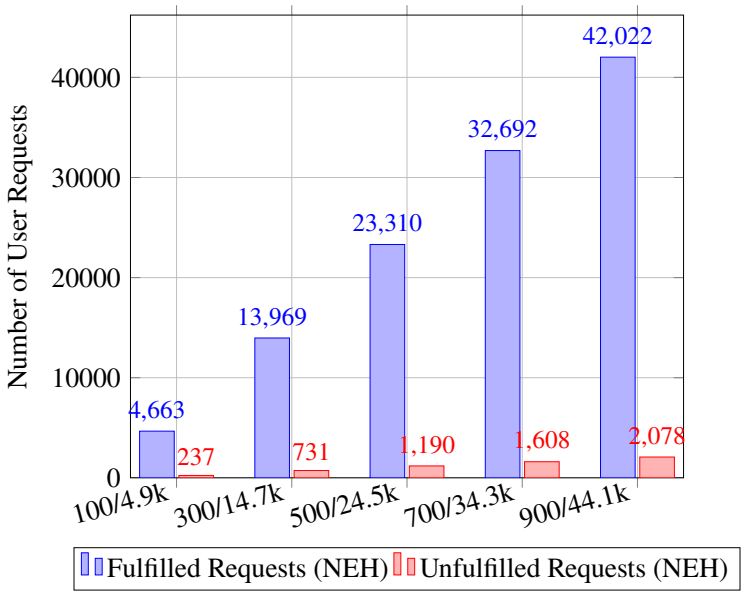

Figure 6.3: Number of Fulfilled and Unfulfilled user requests without Energy Harvesting

the system's ability to fulfill user requests. It shows that EH significantly improves request fulfillment up to 99%. Unfulfilled requests remained below 1% across all ratios. Conversely, No Energy Harvesting (NEH) results in higher unfulfilled request percentages, ranging from 4.69% to 4.97%. As the system scales (more devices and users), the number of fulfilled requests increases in both scenarios. Still, the EH system better manages the energy demands, reducing unfulfilled requests even at larger ratios. This underscores the importance of energy harvesting in large-scale IoT systems. Energy harvesting devices contribute additional power to the sensing devices, ensuring that the devices can continue operating even when their internal energy levels are depleted. The gap between EH and NEH increases with more users. This is likely because, as the demand for services grows (i.e., as more users submit requests), the reliance on harvested energy becomes more critical to sustaining operations. Without energy harvesting, the network's available energy source is insufficient to meet the demand, resulting in a higher number of unfulfilled requests. Energy

efficiency is higher in the EH scenario, as the system can utilize harvested energy to reduce the dependence on internal energy sources, thus conserving battery power and reducing energy wastage. The increasing number of unfulfilled requests in the NEH scenario also suggests that, without energy harvesting, the devices are running out of power, leading to more energy wastage and inability to fulfill requests.

6.4.4.2 Scenario 2: Varying the Number of Sensing Devices with Fixed User Requests This scenario examined the impact of varying the number of sensing devices from 100 to 500 while keeping the number of harvesting devices and users fixed at 10 and 500, respectively. Each user requesting up to 10 services created a total of 5000 user requests. Table 6.3 below summarizes the performance results.

Table 6.3: Performance Metrics for Varying Number of Sensing Devices

ND	**100**	**200**	**300**	**400**	**500**
URF	4170	4782	4800	5000	5000
URU	830	218	200	0	0
NL (hrs)	11	19	24	24	24
EUE (%)	9.9	5.73	3.92	2.94	2.28

Legend: ND - Number of Devices, URF - User Requests Fulfilled, URU - User Requests Unfulfilled, NL - Network Lifetime, EUE - Energy Usage Efficiency.

In Fig. 6.4, the percentage of the 5000 user requests fulfilled increased from 83% to 95% and then 96% with 100, 200, and 300 sensing devices, respectively. 100% of the requests were completed with 400 and 500 devices. The results indicate a significant 12% increase in the number of fulfilled requests from 100 to 200 devices, while the percentage increase in fulfilled requests between 200 and 300 devices was only 1%. This is also

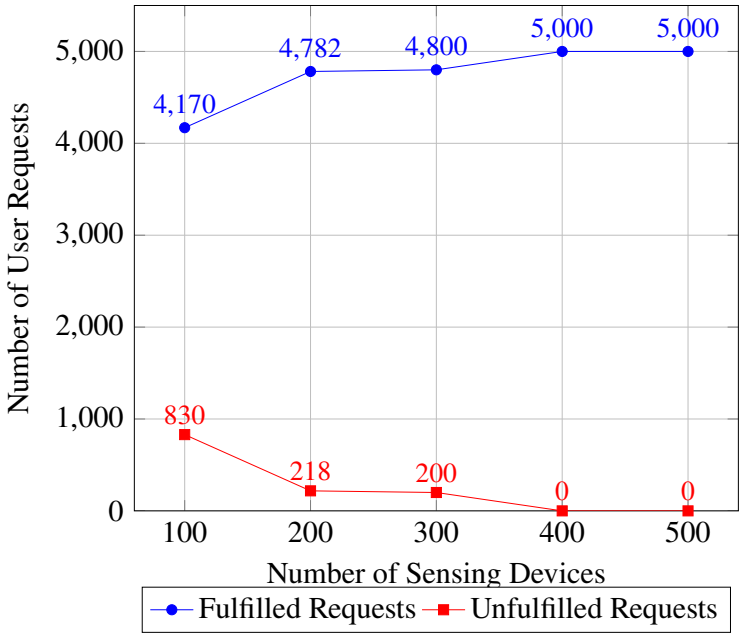

Figure 6.4: Fulfilled and Unfulfilled user requests based on the number of sensing devices

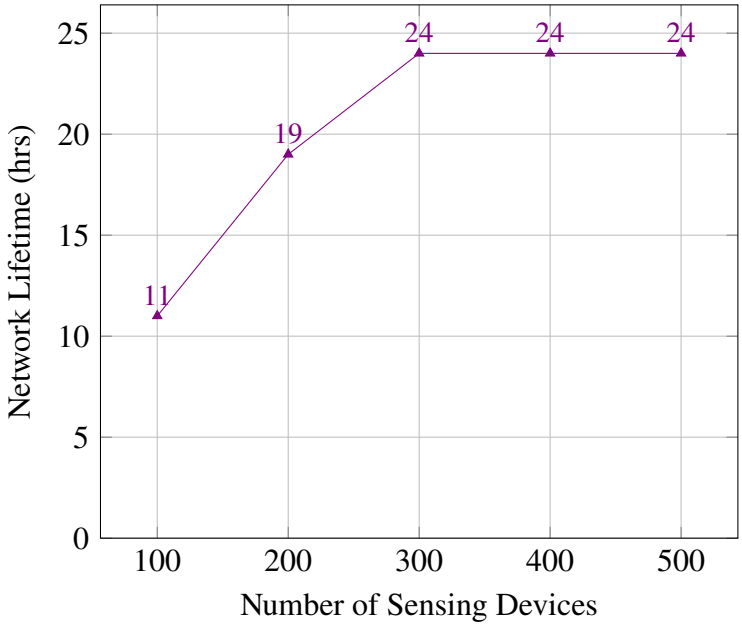

Figure 6.5: Network Lifetime (hrs) based on the number of sensing devices

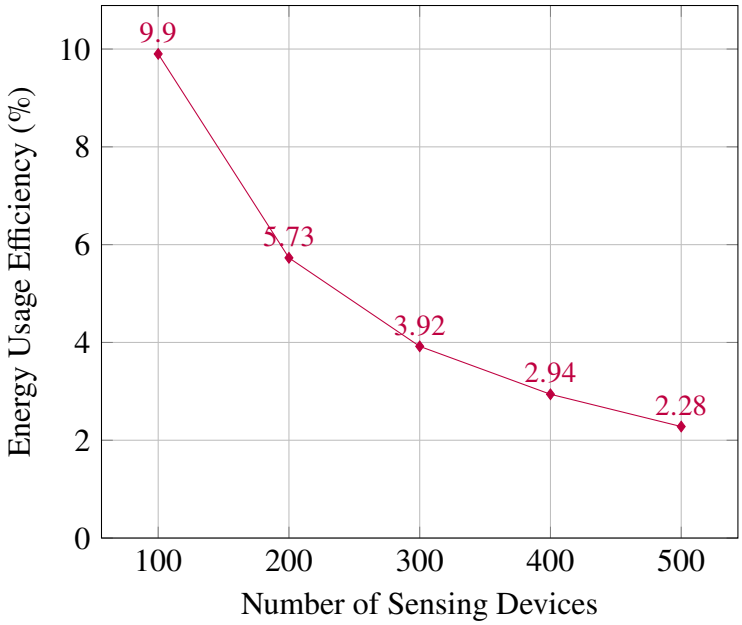

Figure 6.6: Energy Usage Efficiency (%) of the network with varying numbers of sensing devices

reflected in the number of unfulfilled requests. Network lifetime increased from 46% to 79%, and then 100% with 100, 200, and 300 sensing devices, indicating a 33% increase in network lifetime from 100 to 200 devices and a 21% increase for 200 to 300 devices. At 400 and 500 devices, the network lifetime remained 100%. As observed in Fig. 6.5, network lifetime increases with more sensing devices, but stabilizes at 300 devices. This indicates that with 300 devices, the network becomes more resilient to failure due to power depletion in our use-case scenario. The additional devices no longer significantly extend the network's lifetime. The Energy Usage Efficiency(EUE) shown in Fig. 6.16 is at 9.9% with 100 devices and falls to 2.28% with 500 devices, showing 4%, 2%, 1%, and 0.6% decrease in EUE for every additional 100 devices. This suggests that, while the system can handle more requests with the addition of devices, the energy overhead increases. More devices consume more energy, and not all devices are utilized efficiently. The energy required to

support the additional devices may exceed the demand, leading to energy wastage and a decrease in overall efficiency.

6.4.4.3 Scenario 3: Varying the Percentage of Active Energy Harvesting Devices In this scenario, the number of available energy harvesting devices was set at 50 and 100, respectively. The number of sensing devices was set at 500. 500 users with numbers of requests varying from 4 to 14, made a total of 7200 requests. This scenario tests how the framework responds to a higher density of EH devices and user requests. The percentage of active energy harvesting devices also varied from 10% to 70% with an increment of 15% per iteration to simulate situations with low energy availability to higher energy availability. The performance results are presented in Figures 6.7, 6.8, 6.9, and 6.10.

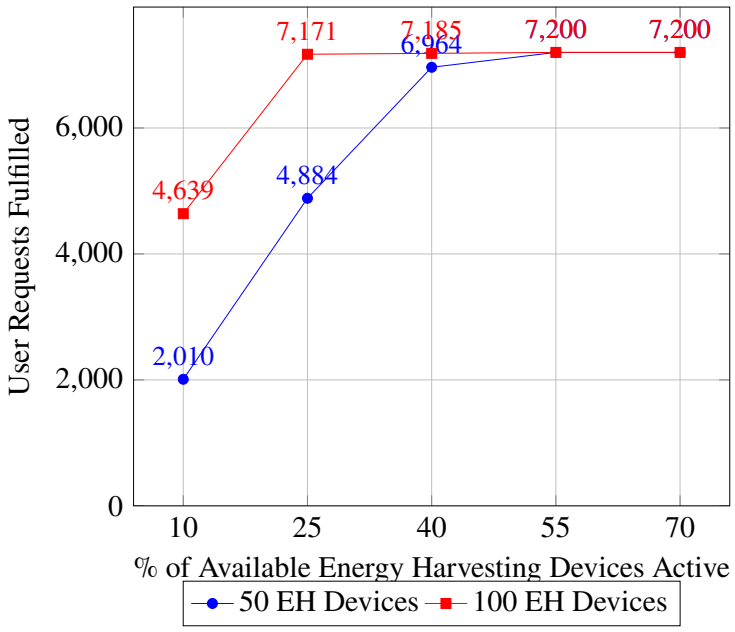

Figure 6.7: User Requests Fulfilled based on % of available EH Devices

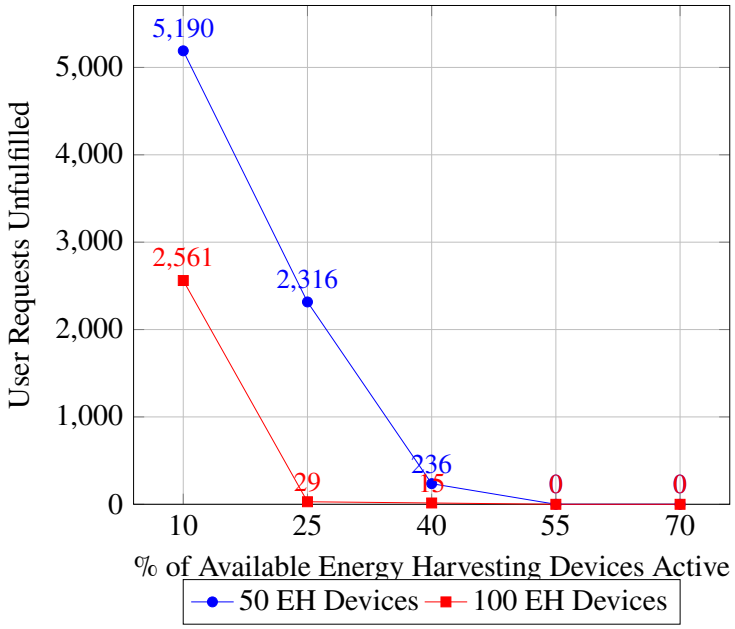

Figure 6.8: User Requests Unfulfilled based on % of available EH Devices

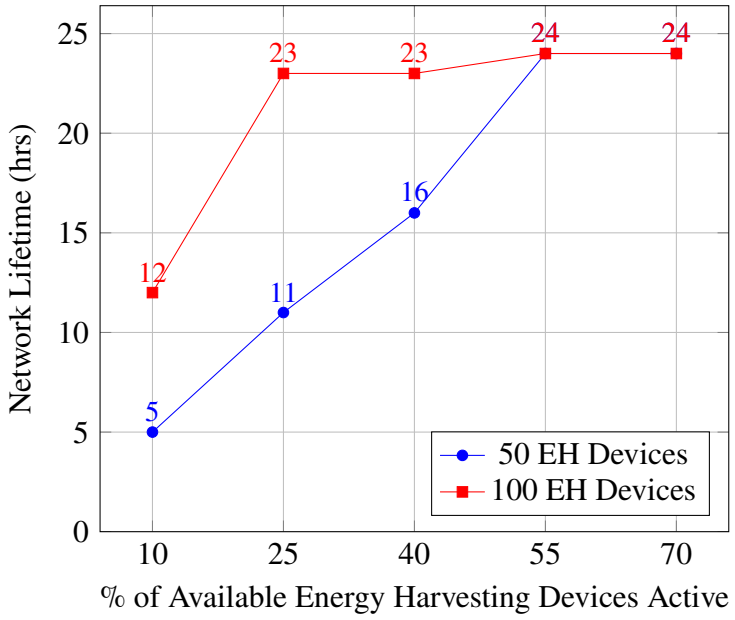

Figure 6.9: Network Lifetime based on % of available EH Devices

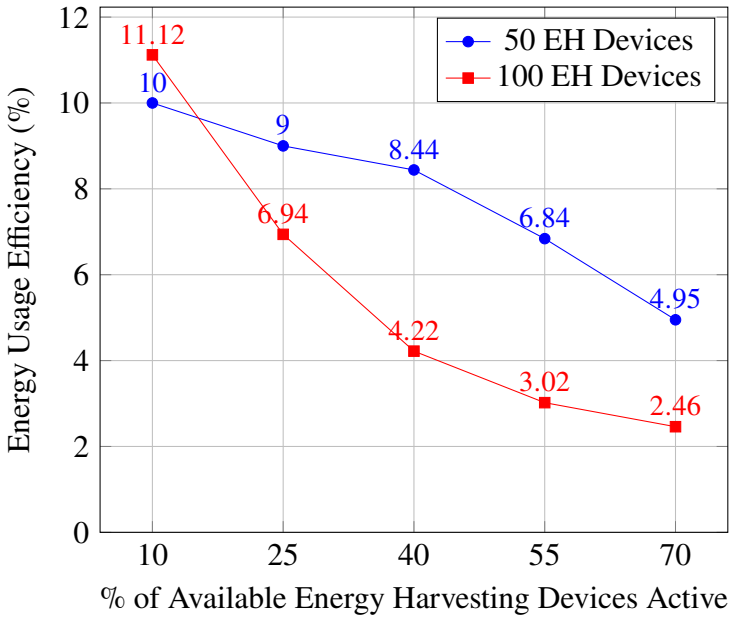

Figure 6.10: Energy Usage Efficiency based on % of available EH Devices

A comparison of the number of user requests fulfilled shows that 37% more requests were fulfilled when only 10% of the available 100 EH devices were active, compared to 50 EH devices. This value drops to 32% with 25% active devices and then to 3% with 40% active devices. With 55% and 70% active devices, the number of fulfilled requests peaks at 100% in both instances. The reverse is the case for unfulfilled user requests. The network lifetime lasted 7 hours longer when only 10% of the available 100 EH devices were active compared to 50 EH devices. With 25% and 40% active devices, the network lifetime lasted 12 hours and seven more hours, respectively, when 100 EH devices were available compared to 50 EH devices. When the percentage of active devices was 55%-70%, the network lifetime lasted throughout the simulation period. EUE did not show much improvement when only 10% of the available 100 EH devices were active compared to 50 EH devices. However, as the percentage of active EH devices increases, EUE is cut in half, with 50 available devices

compared to 100 EH devices. It is important to note that the EUE values are lower when more EH devices are available.

It is observed that having more EH devices only proves effective during an initial heavy inflow of user requests. When the system reaches 55%-70% active harvesting devices, all user requests are fulfilled, and the system operates for a full 24-hour cycle, indicating that the energy harvesting devices can meet the demand without depletion. As more energy-harvesting devices become active, the energy efficiency decreases. Initially, when fewer devices are active, the system is more efficient because the available energy is used more conservatively. However, as more devices are activated, the supply of energy exceeds immediate demand, leading to energy wastage. While the system fulfills more requests and lasts longer, the marginal utility of additional energy harvesting devices diminishes, leading to a lower energy efficiency percentage.

6.4.5 Comparative Analysis

This work demonstrates that energy harvesting incorporated with joint optimization and allocation methods enhances energy management strategies, leading to significant improvement in network lifetime, energy usage efficiency in IoT systems, and user request fulfillment when compared to existing EH-based solutions. Discussions on the comparison of this work to existing works in terms of key performance metrics are presented in this subsection.

Network Lifetime: A significant gain in network lifetime was achieved using the joint optimization and energy allocation strategy implemented in the proposed framework. For instance, in scenario 3 of the simulation, with 50 EH devices available and only 10% of those actively harvesting energy, the network lifetime was sustained up to 5 hours, as compared to [134], where the network lifetime was sustained a little over 1 hour using a

simple RF-EH protocol. In addition, when 25% of the devices are actively harvesting, the network lifetime is extended to 11 hours, leading to a 120% gain in network lifetime. When compared to [135], a threshold-based EH scheduling System with a 100% network lifetime gain, our framework outperforms by 20%. When 40% of the available EH devices are actively harvesting, the network lifetime is sustained for up to 16 hours, and through the entire simulation period, when 55% or more devices are actively harvesting, showing almost a 50% increase in network lifetime. This is a significant improvement from the findings in [89], where the network lifetime in an energy-harvesting-enabled IoT network was found to only increase by 15-30%.

Efficiency of Harvested Energy Usage: Energy Usage Efficiency (EUE) is another key performance metric, and the results from scenario 2 show how our framework improves on this metric as compared to existing work. In Table 6.3, with a maximum of 5000 user service requests, at 100 sensing devices, the EUE of harvested energy is at 9.9%, which is about 40% improvement in comparison to 7% EUE reported in [136], which uses a transient computing approach for an RF EH-based 100 node testbed. At 200 devices, a 5.73% EUE is achieved, giving about a 43% gain against roughly 4% EUE achieved in [137] with a similar number of devices deployed. The framework achieves an EUE of 3.92% with 300 devices outperforming [136], reporting about 2.7% EUE for a similar scenario with 300 devices. Although the EUE is shown to decrease as number of sensing devices increases in 6.3, and also in 6.10 where with increasing percentage of energy harvesting devices, the EUE values decrease from 10% when only 10% of EH devices were active, to 2.46% when 70% of EH devices were active, the decrease in EUE is attributed to the overhead in managing energy storage and distribution. Despite the overhead, the framework proposed in this work still completes a higher number (at least 95%) of user requests and achieves a longer network lifetime, without falling below the best-reported EUE in [137] and [136].

User Request Fulfillment: A significant aspect of the study is the high number of user requests fulfilled (URF) in the EH scenarios. Table 6.1 shows that with energy harvesting, and a relative increase in sensing devices under increasingly heavy traffic, the URF percentage increased significantly. For example, with 100 sensing devices and 49 000 requests, the percentage of URF was only 9.9% $4\,853/49\,000 = 9.9\%$. However, with 300 devices and 14 700 requests, the URF increased to 99.2% $14\,581/14\,700 = 99.2\%$. In contrast, the simple RF-EH protocol in [134] only achieved 70% URF with a similar heavy request load. Although the threshold-based EH scheme presented in [135] achieves 100% URF, that was achieved with 25 devices and 75% less request load than the load handled by our framework. In addition, [137] experienced packet drops as the load requirements increased. Our framework, therefore, outperforms existing works by maintaining URF of up to 99% even with considerable request demands.

Overall, the comparative analysis highlights the improvements achieved by combining resource optimization with strategic energy allocation as proposed in the framework when compared to existing solutions concerning network lifetime, energy usage efficiency, and user request fulfillment.

6.5 Conclusion and Future Work

This paper introduces a joint optimization and energy allocation framework that enables the management and effective allocation of RF energy harvesting for improved user service provisioning in IoT networks with varying QoS and security preferences. By dynamically adapting to a varying number of user requests, sensing devices, and EH device activation levels, the framework ensures network sustainability and service reliability. When compared to recent works, the proposed framework exhibits considerable improvements in extending

network lifetime, increasing energy usage efficiency, and yielding a higher rate of user requests fulfilled, highlighting its benefits for large-scale EH-enabled IoT deployments.

Energy harvesting for IoT has mainly focused on the optimization of energy efficiency, the improvement of service quality, and extending network lifetime. Neglecting the aspect of energy security opens up energy management to potential threats and vulnerabilities. In the process of energy harvesting and energy allocation management in IoT, it is essential to pay attention to energy security, and safeguard the energy harvesting and distribution process from attackers who could intercept the process either by eavesdropping on the control signals being used, wiretapping and tampering with energy allocation commands, or by launching a denial of energy attack. Future work will focus on designing light-weight security countermeasures to protect energy harvesting processes and their allocation strategies to ensure the efficient use and distribution of harvested energy by preventing or resisting unauthorized manipulations.

CHAPTER SEVEN

RO4: INTEGRATING ENERGY HARVESTING MANAGEMENT WITH ENERGY SECURITY TECHNIQUES

The need for real-time data, efficient resource management, remote monitoring, and task automation has led to the adoption of the Internet of Things (IoT). IoT is the interconnection of devices that are equipped with sensors to collect data, actuators to perform actions, and communication modules to interact autonomously with other devices in the network. Devices used in IoT are physical objects, such as wristwatches and mobile phones, that utilize communication protocols like Wi-Fi and Sigfox, to provide services to users [138]. These devices are also referred to as IoT devices. Through the utilization of IoT devices, services such as personalized healthcare [139], predictive maintenance of industrial machines [140], and real-time security surveillance [141] are offered to users. Likewise, IoT has led to significant advances in diverse domains, such as vehicle-to-everything (V2X) communication in connected and autonomous vehicles [142], intelligent energy management in smart cities [143], and disaster detection through remote environmental monitoring [144].

Many IoT devices are often battery-powered, and the increased utilization of these devices has led to higher energy consumption. Moreover, these devices are usually deployed in open environments with limited energy resources and no continuous power sources, requiring frequent battery replacements and maintenance [103]. This has introduced several challenges, such as limited operational lifetime due to their finite battery life [145], intermittent availability [146], and increased response time for real-time applications. Additionally, environmental factors such as humidity and pressure adversely affect the battery performance of IoT devices [147], and battery maintenance is costly

and time-consuming [148]. Overall, these challenges affect the performance, reliability, scalability, and lifespan of the IoT devices. Therefore, there is a need for a sustainable alternative energy source for powering IoT devices to ensure a reliable service delivery. This need is met through Energy Harvesting (EH) technology.

EH is the process through which energy is captured and stored from external sources such as solar power, Radio Frequency (RF), vibrations, wind, and thermal energy, to power resource-constrained devices such as IoT devices [149]. The integration of EH mechanisms with IoT networks provides a complementary power source for IoT devices, thereby reducing their dependencies on batteries and increasing their lifespan. Furthermore, it is environmentally friendly and supports autonomous operations, which improves network reliability. This has led to the adoption of a sustainable IoT network that is easily scalable without increasing the energy cost and optimized energy usage through continuous monitoring of its environmental parameters, particularly in remote environments. However, the integration of EH with IoT devices has introduced additional attack vectors that an attacker can exploit for energy-based attacks, such as pilot signal [150], which compromises the quality of service (QoS), reliability, and lifetime of the IoT devices.

A significant form of these attacks is the Energy Depletion Attack (EDA), also referred to as sleep deprivation or vampire attack [57]. EDA occurs when an attacker sends bogus packets to EH-powered devices, preventing legitimate operations and causing rapid energy drainage [59]. Examples of attacks that can be categorized as EDA attacks include flooding [58], jamming [59], and spoofing [60] attacks. For example, solar-powered, energy-harvested IoT sensor nodes are deployed to monitor soil moisture, temperature, and humidity. An attacker, through a compromised node, infiltrates the IoT network to send malformed data packets continuously. These packets prevent the IoT devices from entering sleep mode, thus causing increased power consumption, greater than the harvested power.

This will result in rapid battery drain, causing device shutdown, and subsequently disrupting the performance of the IoT network as the device fails to collect and transmit the required environmental data. Therefore, there is a need for a robust energy security framework that protects EH-powered IoT devices from EDA, while ensuring the user's QoS is not compromised.

In existing literature, different security countermeasures have been proposed to mitigate the occurrence of EDA in IoT devices. The authors in [104] proposed a joint protection framework that provides data privacy and protects the energy of the EH-powered devices using an incentive-based federated learning approach. A dynamic optimization mechanism that improves the energy efficiency based on the recharge profile of the devices was proposed in [151]. Additionally, by determining the energy cost used by EH-powered devices, the authors in [108, 109] were able to outsource resource-intensive operations and provide a threshold for identifying the energy-efficiency level of EH-powered devices.

However, these proposed solutions primarily focus on the energy profiles of the devices, without considering the request behavior of the user, leading to reduced accuracy and energy inefficiency. These device-centric approaches or detection schemes cannot entirely prevent persistent attacks. Therefore, there is a need for a security framework that integrates the request behavior of the user, alongside the profile of the device. In [105, 106], different security configuration adaptation strategies were proposed to improve the energy efficiency of EH-powered devices. These security configuration strategies used energy indicators to integrate authentication and confidentiality of EH-powered devices. When there is abundant energy, the security of the device is strengthened, but security becomes weakened when the energy level is low. This creates a predictable window of vulnerability that an attacker can exploit to create an EDA. Therefore, there is a need for a deterministic security framework that prevents EDA regardless of the energy levels of the devices. Overall, the current

state-of-the-art solutions are mostly reactive and do not prevent external adversaries from launching continuous energy depletion attacks [110].

To address these highlighted shortcomings, this work presents a sensor-cloud-based, user-centric energy scheduling framework that incorporates the service request behavior of a user to provide a holistic defense. This user-centric approach builds upon prior work while addressing energy depletion threats in EH-powered sensor-cloud environments. In our proposed framework, each user's request, coupled with the target device profile, passes through a predefined safe threshold that prevents EH-powered devices from losing energy to malicious users' requests, so there is sufficient energy to service the request of legitimate users. This ensures energy efficiency, uncompromised QoS, and a robust security framework against malicious users. The major contributions of this paper are as follows:

- We propose a dynamic admission control mechanism that blocks expected users' requests that deplete a node before the safe energy thresholds.
- We design an anomaly detection technique to detect unusual request patterns and a rate-limiting technique that prevents flooding and rapid request bursts.
- We evaluate the effectiveness and resiliency of the proposed framework by simulating multiple scenarios with varying values for comparative analysis

7.1 Network Scenario and Attack Model

This section presents the network deployment scenario considered for implementing the proposed user-centric energy-aware scheduling framework. The scenario outlines the various devices and communication patterns that exist in a typical Energy Harvesting (EH)-powered IoT network. Additionally, we describe the three security attacks - flooding, spoofing, and jamming- that were considered for our deployment scenario. These attacks

represent Energy Depletion Attacks (EDAs), which target resource-constrained EH-powered devices.

7.1.1 Secure EH-IoT Architecture

The proposed user-centric energy-aware scheduling framework is built on three distinct layers for the overall secure EH-IoT framework, as shown in Fig. 7.1. These layers are *user-centric, sensor cloud, and EH-IoT device layer with EH sources.*

- **User-centric Layer** enables effective interaction between the end-users and the EH-IoT framework. In our proposed framework, this is where the information of various IoT users resides. In this layer, users' sensing service requests are generated. These requests are characterized by the user's specific Quality of Service (QoS) preferences, such as allowable response time, preferred service delivery throughput, and security preferences. These requests are then forwarded to the sensor cloud control layer for processing and other control activities. The number of requests from this layer is dependent on the number of users at a given time. This layer is decentralized.

- **Sensor Cloud Control Layer** is the middleware component of the EH-IoT framework that connects the user-centric layer to the IoT device layer. This layer provides data processing and service provisioning, enabling real-time decision making. In our proposed framework, this layer is responsible for managing incoming user service requests and device profiles. The dynamic admission control of requests module, responsible for making intelligent decisions about the requests, tracking the energy profiles of the devices, and estimating the energy impact of each user request, is implemented in this layer. Furthermore, functions such as rate limiting of suspicious traffic, lightweight anomaly detection in request patterns, and adaptive lightweight

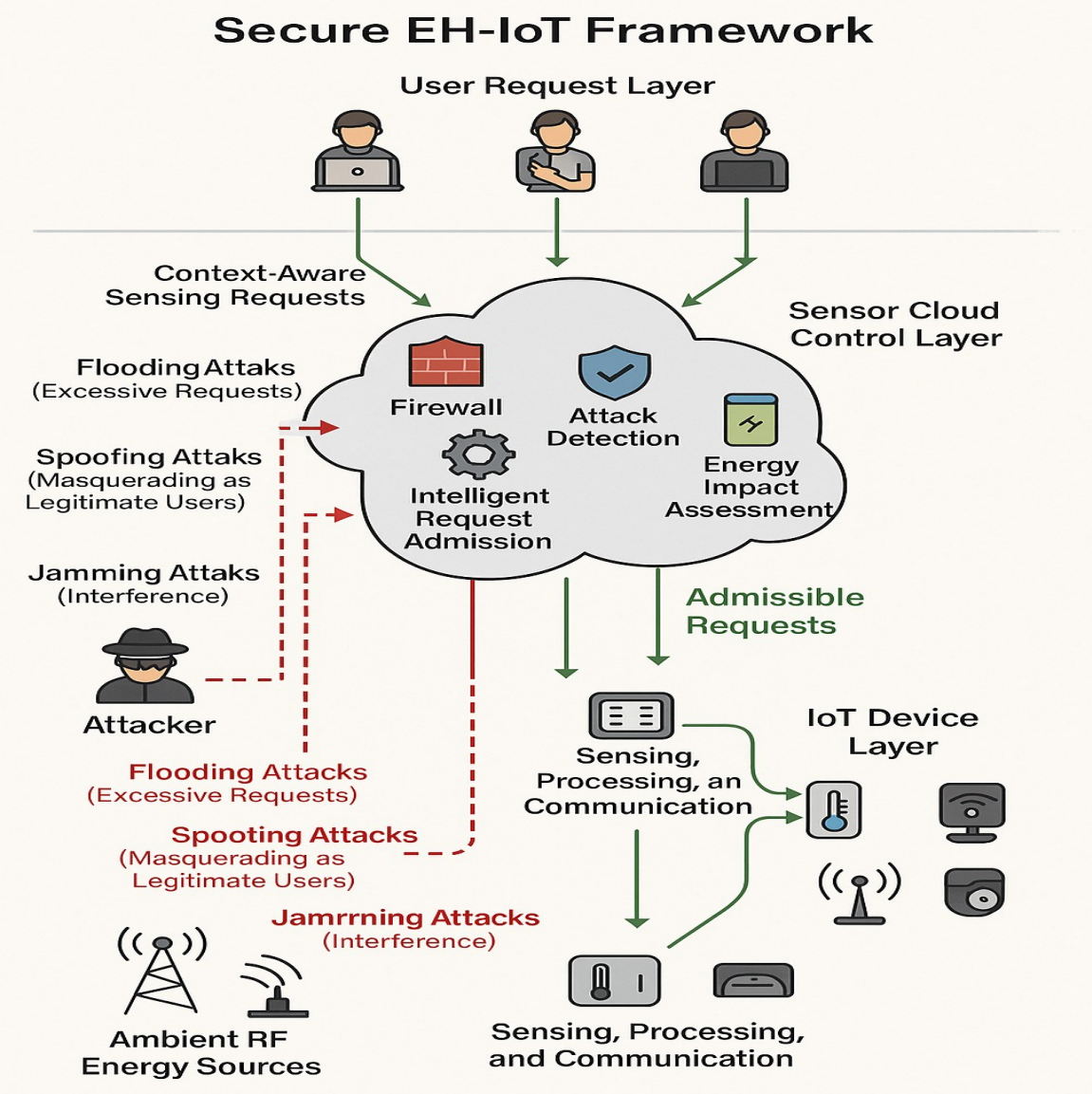

Figure 7.1: Proposed Secure EH-IoT Architecture
This figure shows the different entities interconnected in the secure EH-IoT architecture. Additionally, three different forms of EDA attacks are listed.

cryptography security implementation are carried out in this layer. These functions detect and prevent potential malicious requests or attacks that may cause severe energy drainage for the devices. Upon validating the requests, only requests that are considered legitimate are forwarded to the target device for service fulfillment. This is a centralized layer, and it is not resource-constrained, like the IoT device layer.

- **IoT Device Layer** is the sensing layer. It consists of IoT devices that are powered by RF energy harvesting techniques to prolong operational life. These devices collect data from the physical environment, such as temperature, light, and humidity. In our proposed framework, the service requests of users are processed, and secure data communication between the devices takes place. The harvested energy is sourced from nearby RF energy sources, like Wi-Fi modules or telecommunication towers, as shown in Fig. 7.1. Only legitimate service requests approved by the sensor-cloud control layer are processed by the EH-powered IoT devices. This is to ensure the devices are not directly exposed to energy-draining malicious requests, providing an extra layer of security to enable resource-constrained IoT devices to focus on more energy-aware operations while achieving longevity and self-sustainability. This layer is decentralized and resource-constrained.

7.1.2 Attack Model

For deployment of our proposed framework, we assume a small to medium-sized IoT deployment within a hostile energy harvesting environment, laden with malicious attackers, as shown in Figure 7.2

A typical adversary in such an environment does not necessarily steal data or directly compromise the sensing functionalities of a device. Rather, the attacker tries to exploit the energy-constrained IoT devices by stealing or impersonating legitimate user identities

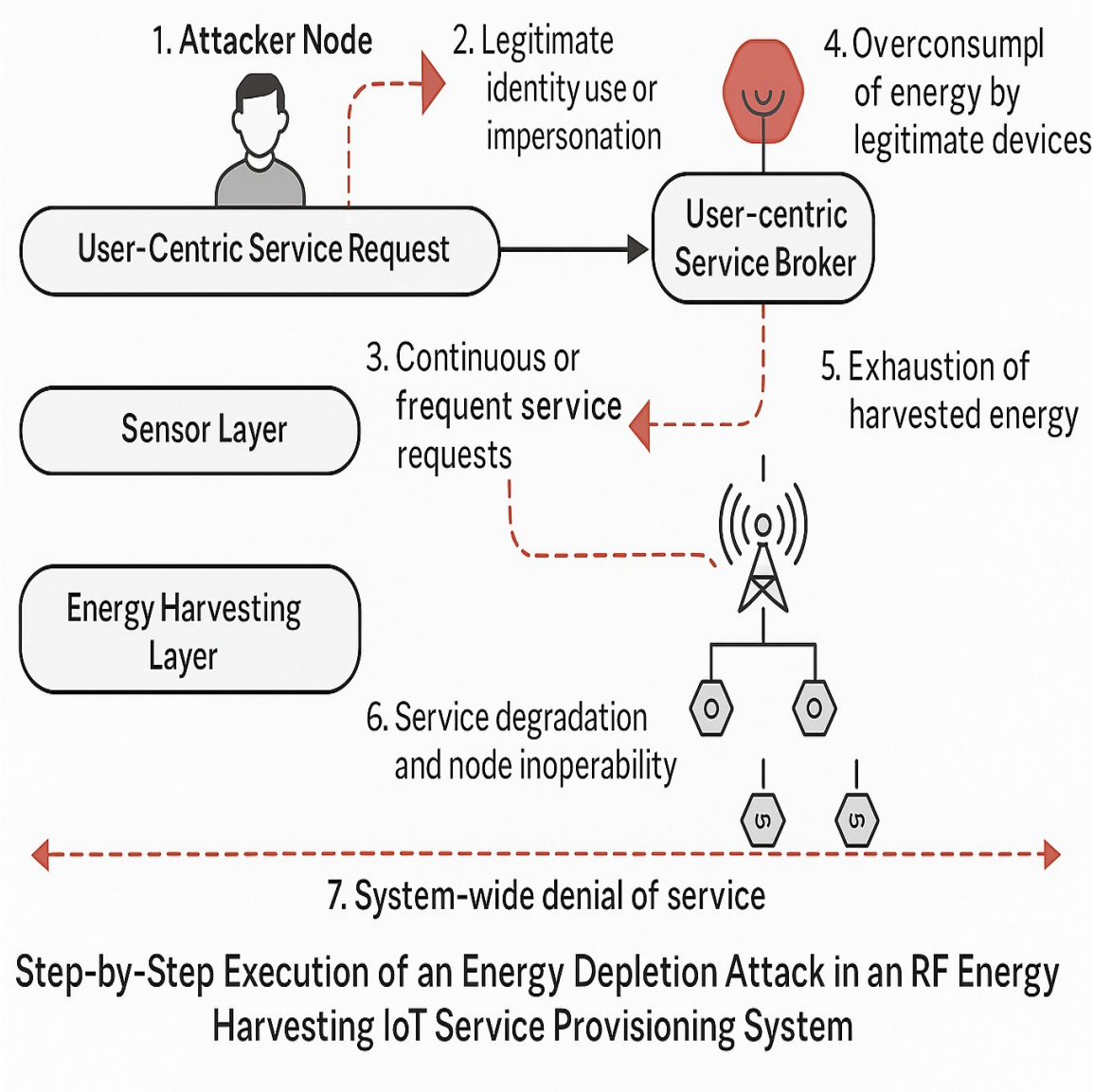

Figure 7.2: Data flow of an EDA in an EH-powered device
This figure shows a sequential flow in which an EDA attack is carried out by an attacker who exploits an existing security vulnerability for malicious intent.

from the user-centric layer to send fake, high-frequency, high-energy cost requests to the sensor cloud control layer. Thereby causing a rapid depletion of the energy reserves of the EH-powered devices, which leads to a Denial of Service (DoS), or Denial of Energy (DoE) in the case of EH-powered systems, due to service degradation and node inoperability. Such attacks are referred to as Energy Depletion Attacks (EDA). The three EDAs (flooding, spoofing, and jamming attacks) considered in our proposed framework are discussed next.

- **Flooding Attacks** are quantity-based service-level attacks where an adversary continuously creates large volumes of illegitimate or redundant requests that seem valid but are aimed at consuming EH-powered IoT device energy resources. Devices expend energy in receiving and processing such requests. Moreover, due to the high frequency of the transmitted requests, the device is forced to remain active for the period during which the flooding attack is happening. This overwhelming amount of request rapidly drain the conserved harvested energy, since the device loses energy faster than it can harvest. This causes the EH-powered device to go offline once all its conserved energy is depleted.

- **Spoofing Attacks**, in contrast to flooding attacks, are more serious, quality-based service-level attacks where an adversary steals or impersonates a legitimate user's credentials to disguise themselves as a trusted user. With the stolen credentials, the attacker creates energy-intensive requests, which put high energy demands on the EH-powered devices. An example is a user sending a computationally intensive request, such as a temperature sensing request, with a short response time, high throughput, security, and service frequency preferences. As long as the request is believed to be authentic, energy resources of the EH-powered device will continue to

be dedicated to fulfilling such a request, regardless of the high energy demand. This eventually causes the device to exhaust its energy.

- **Jamming Attacks** disrupt legitimate data and energy transmission between devices when an attacker introduces interference during the transmission of energy-carrying signals. For instance, an RF energy-powered device might become starved of its power source if an interference hinders the EH process through which it replenishes its energy. Furthermore, when a data transmission process is interrupted, the devices experience a higher packet error rate that requires data retransmission. Each attempt at retransmission consumes additional energy that leads to the EH-powered device rapidly losing energy. The jamming attack on the energy replenishing and data transmission activities ultimately limits the operational lifespan of the EH-powered IoT devices.

The current state of the art for adaptive energy security solutions described in Section 3.4.2 shows that when EH-powered IoT systems are designed to adjust security levels dynamically based on the observed energy levels of the device, the EH-powered devices are more susceptible to potential EDAs when the security measures weaken in proportion to the drop in device energy level. For instance, as a device's energy level becomes critically low, in a bid to remain operational and conserve energy, it might temporarily reduce its threshold or use less computationally expensive cryptographic algorithms. It is assumed that an attacker can become aware of this adaptive behavior (such as a basic authentication process or slower service responses) by observation and proceed to exploit it by launching any of the attacks mentioned above and succeed due to lower security measures at that time.

To effectively address consequent EDAs that arise from an adversary exploiting the existing adaptive energy security measures in EH-IoT deployments, there is a need for a more

centralized security mechanism. In our proposed framework, the approach is to pre-process user service requests through a dynamic admission control system at the sensor cloud layer. This process will be implemented at the sensor cloud control layer, where there are sufficient computational resources required to process and store the data. This energy-aware process takes the burden away from individual resource-constrained IoT devices that only need to perform basic tasks like environment sensing and data transmission. The dynamic admission control approach analyzes the user service request through a function further described in Section 7.2. This analysis is used to determine the energy impact of receiving, processing, and securely transmitting the requested service before it is forwarded to the target device. Through this analysis, malicious requests such as excessive flooding or suspicious spoofing can be identified and blocked before reaching the resource-constrained IoT devices. Additionally, this comprehensive approach helps prevent jammed devices from receiving legitimate requests, which might cause unnecessary energy depletion.

Essentially, a unique challenge is presented by the attack model where an adversary can observe and exploit fresh energy vulnerabilities that EH-powered IoT systems become susceptible to, as they try to balance energy limitations and security requirements. The proposed solution is a user-centric, energy-aware, centralized sensor cloud-based approach that intelligently implements a dynamic admission control mechanism that preserves the operational lifespan of constrained IoT devices by proactively scheduling legitimate user requests and blocking EDAs.

7.2 Proposed Framework

The methodology used by our proposed framework to prevent against EDAs for EH-powered devices is presented in this section. The proposed framework consists of different modules - dynamic admission control with adaptive lightweight cryptography, rate

limiting of suspicious traffic, and lightweight anomaly detection, executed cohesively to achieve a secure, user-centric, energy-aware admission control and scheduling framework. Additionally, we presented the two distinct scenarios we implemented to validate the effectiveness of our proposed framework.

The proposed framework considers a sensor-cloud-based architecture where user service requests are sent and managed over the cloud computing paradigm, and the low-power IoT devices that service these requests are powered by harvesting ambient energy. These IoT devices intermittently harvest and store ambient energy, such as Radio Frequency (RF) energy, for use in processing sensor readings and secure communications. These devices are responsible for fulfilling user service requests. However, before a service request is routed to an EH-powered IoT device, it is routed through different modules to validate the legitimacy of the request. The various modules involved in validating the legitimacy of a service request are implemented on the sensor cloud control layer of the secure EH-IoT architecture described in Section 7.1.1. The following subsections describe the user service request validation and verification process, as well as the framework execution environment.

7.2.1 User Service Request Validation

In this subsection, we discuss the components for the user service request validation process and the components of the verification process, including the workings of the centralized dynamic admission control, the request rate limiting module, and the anomaly detection modules.

7.2.1.1 EH-powered IoT device

An EH-powered IoT device, d, is a device that has the ability for energy harvesting with a defined energy capacity. It resides in the IoT device layer. This device, d, performs a specific environmental sensing activity, which is represented as a state

tuple $S_d(t)$ at a given time, t. $S_d(t)$ is depicted in eq. 7.1.

$$S_d(t) = (E_d(t), B_{\max}) \tag{7.1}$$

where $B_{\max}$ represents the maximum battery capacity and $E_d(t)$ represents the energy level of the device. $E_d(t)$ is a crucial variable that is represented as depicted in eq. 7.2

$$E_d(t) = \min\left(B_{\max}, E_d(t-\Delta t) + E_h(\Delta t) - E_c(\Delta t)\right) \tag{7.2}$$

where $E_h(\Delta t)$ is the energy harvested, and $E_c(\Delta t)$ is the energy consumed during interval, Δt. Once a request is approved and admitted by the security modules in the sensor cloud control layer, the d fulfills the sensing service request of the user.

7.2.1.2 User Service Request A user-sensing service request, R, represents a user's request to receive service from an EH-powered IoT device. It is generated by an end-user at the user-centric layer and consists of the user's QoS preferences, such as response time, throughput, and security preferences. In our proposed framework, R is represented as a tuple and is depicted in eq. 7.3.

$$R = (id_s, id_d, qos, t_a) \tag{7.3}$$

where id_s represents the user ID, id_d represents the target device ID, and qos represents a composite structure representing user response time (RT_p), throughput (TT_p), and security preferences (S_p). The qos is depicted in eq. 7.4,

$$qos = (RT_{\mathrm{p}}, TP_{\mathrm{p}}, S_{\mathrm{p}}) \tag{7.4}$$

and t_a is the timestamp recording the exact time the request was made. In our proposed framework, there are two user groups: legitimate and blacklisted users. Every user that sends a sensing service request is assumed to be legitimate until proven otherwise. A user

is classified as a blacklisted user when they fail at least three security checks carried out by the modules in the sensor cloud control layer. Once a user is blacklisted, they are prohibited from sending R in the proposed framework. This is done to ensure that malicious users do not infiltrate the secure EH-IoT system. The user-sensing service request, R, serves as input to the different security modules in the sensor-cloud layer in our proposed framework.

7.2.1.3 Request Verification At the sensor cloud control layer, the following modules - dynamic admission control, rate limiting of suspicious traffic, and lightweight anomaly detection - are implemented to verify the legitimacy of a user sensing service request. A detailed description of each module is presented below.

- *Dynamic Admission Control Module*: The incoming user service requests, R, are evaluated in this module based on the level of harvested energy to prevent spoofing requests. It acts as the first point of defense in our proposed framework. In this module, the expected energy cost for each received R is computed using a function. The function calculates the expected energy cost, $C(R)$, by evaluating response time, throughput, and the given security level of the R. Additionally, the energy cost to the device that will fulfill the request is calculated by deducting $C(R)$ from its current energy level, which is denoted as $E_{\text{curr}}(id_d)$. The dynamic admission control module, $AC(R)$, is depicted as eq. 7.5.

$$AC(R) = \begin{cases} \text{Accept}, & \text{if } E_{\text{curr}}(id_d) - C(R) \geq E_{\text{thresh}} \\ \text{Reject}, & \text{otherwise} \end{cases} \tag{7.5}$$

  The energy threshold, E_{thresh}, is the minimum energy level a device is required to have to process a request. Only requests that leave $E_{\text{curr}}(id_d) \geq E_{\text{thresh}}$ are

accepted; requests failing this check are rejected before transmission to preserve device longevity.

Adaptive Lightweight Cryptography Function performs the final security check within the Admission control module. This function selects a security level $S \in \{\mathrm{l}, \mathrm{m}, \mathrm{h}\}$, representing low, medium, and high, respectively, based on user QoS requests and device residual energy, as shown in Algorithm 6. It acts as the security energy broker by checking the desired security level of the request and the current energy level of the device. Since each security level incurs some energy cost, to prevent high energy costs when a device is running low on energy, it dynamically selects the strongest security level a device can support at its current energy level, or automatically downgrades the security level to be implemented for the predefined energy threshold. Let $T_{\text{high}}, T_{\text{medium}}$ be energy thresholds such that:

$$E_{\text{thresh}} < T_{\text{medium}} < T_{\text{high}}.$$

Algorithm 6 Adaptive Security Level Selection

Input: qos, E_{curr}
Output: Selected Security Level
if $qos == \mathtt{h}$ ***and*** $E_{curr} \geq T_{high}$ **then**
  return S_{high}
end
else if $(qos == \mathtt{h}$ ***or*** $qos == \mathtt{m})$ ***and*** $E_{curr} \geq T_m$ **then**
  return S_{medium}
end
else
  return S_{low}
end

The selected level determines the energy cost $E_{\text{cost}}(S_j)$, used as $C(R)$ in the Energy-Aware Admission Control module, ensuring integrated security and energy management.

- *Request Rate Limiting Module*: In this module, the received user sensing service request, R, for each unique user id_s is tracked and logged with a timestamp, L_s, to mitigate flooding attacks.

$$L_s = \{t_1, t_2, \ldots, t_k\} \tag{7.6}$$

For each received request, R, the module queries the log, L_s, to count the number of requests already received from the same user within a predefined time frame. The count of recent requests within the time window T_w is given as:

$$N_{\text{recent}}(id_s, t_a) = |\{t_i \in L_s \mid (t_a - t_i) \leq T_w\}| \tag{7.7}$$

where t_a represents the exact time the current request was made and t_i represents the timestamp of a previous request already logged in the system. If the count shows that the specified request rate limit has been exceeded within a given time, subsequent requests from that user are flagged and temporarily blocked to limit abnormally high request rates, which can be indicative of a potential attack. Given a specified request rate limit $N_{\max}$, the request rate limit, $RL(R)$, is depicted in eq. 7.8.

$$RL(R) = \begin{cases} \text{Flagged}, & \text{if } N_{\text{recent}} + 1 > N_{\max} \\ \text{Permit}, & \text{otherwise} \end{cases} \tag{7.8}$$

Older timestamps ($< t_a - T_w$) are pruned periodically from the log, L_s, to reduce congestion.

- *Anomaly Detection Module*: The behavior patterns of legitimate users and devices are analyzed over time in this module to detect unusual changes in request patterns. The analysis is done using the sliding window technique to compare current and historical request volumes. For the historical request volumes, a record of request counts from various users over a series of N periods is maintained. A baseline for normal activities is established based on the average of the last time series window $N-1$. Let $W_c(id_s)$ be the count in the current window, and $\mu_{\text{hist}}(id_s)$ the historical average:

$$\mu_{\text{hist}}(id_s) = \frac{1}{N-1} \sum_{i=c-N+1}^{c-1} W_i(id_s) \tag{7.9}$$

An anomaly is flagged if:

$$F_{\text{anomaly}} = \begin{cases} 1 \text{ (Alert)}, & \text{if } W_c(id_s) > \alpha \cdot \mu_{\text{hist}}(id_s) \\ 0 \text{ (Normal)}, & \text{otherwise} \end{cases} \tag{7.10}$$

Where α is a weighted sensitivity value, and any slight deviation of the request count in the current window from the computed baseline (i.e., current_count $\geq \alpha *$ average) is flagged as suspicious and triggers stricter rate limits or administrative alerts. The average smooths out the baseline for normal behavior over time. In our proposed framework, we set the value of α to 1.5 to detect even the slightest deviation from the baseline. This module enables the detection of stealthy attacks and aggressive flooding activities.

Once the request, R, has passed through all three modules, a final decision is made. The final decision made by the modules is represented as F_{cloud}, and depicted in eq. 7.11.

$$D = F_{\text{cloud}}(R, \{S_d(t)\}, AC(R), L, W) \tag{7.11}$$

where $D \in \{\text{Accept}, \text{Reject}\}$, R, is the user sensing service request, $\{S_d(t)\}$ is the set of device states, $AC(R)$ is the admission control decision, L is the rate-limit log, and W is the anomaly window history.

7.2.2 Execution Environment

The proposed framework was implemented using a SimPy-based discrete event simulator model. This was done to create a realistic and reproducible testing environment, where the security modules in the proposed framework can be evaluated.

SimPy is a discrete-event process-based simulation framework in Python, with high capacity for highly detailed, component-level representation required to model complex energy dynamics in energy-harvesting IoT environments accurately [152]. Its discrete event capabilities help to effectively model energy state with time variations by integrating realistic energy traces [153]. The Python foundation of SimPy provides the flexibility required to implement the dynamic admission control module by facilitating easy data integration, analysis, and visualization. This makes it suitable for modeling the concurrent and asynchronous nature of our proposed framework.

General-purpose tools (such as Simulink and MATLAB) and other simulators like NS-3 and OMNeT++ were potential options considered for the simulation; however, each comes with some trade-offs. Network simulators like NS-3 lack the flexibility to customize the dynamic admission control policies without the need for extensive modification. In addition, general-purpose tools like Simulink require complex scheduling and are less intuitive when used to model logic for discrete events. The SimPy-based simulator, on the other hand, provides the flexibility and precision control necessary for detailed modeling of the different security modules. Additionally, it provides ease of real-world data integration within a powerful programming environment.

Two distinct scenarios were simulated on the SimPy execution environment. This was done to highlight the effectiveness of the different security modules in our proposed framework. Each scenario was subjected to identical parameters, such as the number of users and devices, user sensing service requests, and the Energy Depletion Attacks (EDAs). A more detailed description of these parameters is given in Section 7.3.2. The following are the different scenarios considered.

7.2.2.1 Scenario 1: Energy-Aware Scheduling (EAS) In this scenario, the effectiveness of the proposed framework is evaluated when only the energy level of the EH-powered devices is considered. The user sensing service request is not included, thereby isolating the effectiveness of the dynamic admission control module. This was done to emphasize the need for a robust user-centric approach in a security framework against EDAs. We refer to this scenario as the *baseline control model.*

Operational Logic: This model activates the dynamic admission control module while deactivating the request rate-limiting and anomaly detection modules. The requests are filtered based only on their projected impact on the device's energy reserves, as calculated by the module.

Formal Representation: The decision for a user sensing service request, R, is governed solely by the dynamic admission control function, $AC(R)$.

$$\text{Decision}(R) = AC(R)$$

where $AC(R)$ returns 'Accept' only if $E_{\text{curr}}(id_d) - C(R) \geq E_{\text{thresh}}$. The energy cost, $C(R)$, is dynamically calculated by the adaptive cryptography function based on the user's security preference (S_{pref}) and the device's current energy (E_{curr}) in the $AC(R)$ module.

$$C(R) = E_{\text{base}} + E_{\text{sec}}(S_{\text{actual}}) + \ldots$$

This scenario helps to establish a benchmark for comparative analysis with our proposed user-centric security framework, as it only allows us to quantify the improvement in the device lifespan as a basis for energy efficiency.

7.2.2.2 Scenario 2: Proposed Secure Admission Control Framework This scenario represents the full implementation of our proposed framework, that is, all the security modules - dynamic admission control, request rate limiting, and anomaly detection - are integrated to create a holistic, user-centric context-aware security countermeasure. We refer to this scenario as the *secure model.*

Operational Logic: This model evaluates an incoming user sensing service request, R, sequentially by all security modules - dynamic admission control, request rate limiting, and anomaly detection. Furthermore, requesting processing is managed by a priority-based scheduler, which is executed by the response time preference, RT_{pref}.

Formal Representation: The final decision for R is a logical conjunction of the outputs from the security modules. A R is accepted *if and only if* it is permitted by the rate limiter, not flagged as an anomaly, and the energy cost is within the energy threshold.

$$\text{Decision}(R) = (\text{RL}(R) = \text{Permit}) \wedge (\neg F_{\text{anomaly}}) \wedge (AC(R) = \text{Accept})$$

Here:

- $RL(R)$: The rate-limiting function is active.
- F_{anomaly}: The anomaly detection flag for the request's source is checked.
- $AC(R)$: The final energy check is performed using the comprehensive, multi-dimensional QoS cost function for $C(R)$.

This scenario is designed to demonstrate the combined effect of the integrated modules in providing robust resilience against a wide range of EDAs while maintaining a high QoS for legitimate users and extending the operational life of devices.

7.3 Simulation

In this section, we present the simulations that were carried out to validate the effectiveness of the proposed framework. The different variables and value ranges considered in our simulations are presented. Two distinct scenarios were simulated and used for comparative analysis. To evaluate the performance of these simulations, we used various performance metrics, including the Legitimate Request Fulfillment Rate (LFR), False Negatives (FN), and network energy efficiency, to assess the Quality of Service (QoS), resource efficiency, security, and resilience of the proposed framework.

7.3.1 Simulation Setup

This simulation was carried out on a platform with 16 GB RAM and a 512 GB NVMe SSD; an 11th Gen Intel(R) Core(TM) i7-1165G7 CPU at 2.80 GHz, running a 64-bit Windows operating system. The core simulation process was developed leveraging the SimPy 4.0.1 library with Python 3.11. The proposed framework is implemented in a discrete-event simulation environment using `SimPy` because it is efficient, asynchronous, and shows the different states triggered by the simulation events. Our simulation was designed to model the energy dynamics, request processing, and adversarial conditions typical of a sensor-cloud IoT ecosystem.

System Model: The simulated secure EH-IoT system is composed of IoT devices, user sensing service requests, and a server that hosts the central processing logic for the different

security modules in our proposed framework. The simulation run represents a 24-hour operational day, corresponding to 1440 minutes.

Device Generation: The EH-powered IoT devices were auto-generated using an algorithm for the simulation run. The following attributes define each device:

- Sensor Type: defines the type of data a device is sensing in the physical environment. This value is randomly assigned from a set of {'Temp', 'Humidity', 'Light', 'Voltage'}. A device can only have one sensor type.

- Energy Capacity: defines the amount of energy a device can have. The initial energy capacity of each device is determined from a uniform distribution, $U(80, 120)$ mJ. The maximum energy buffer is set to 150% of this initial capacity to allow for surplus storage.

- Energy Harvesting: defines the rate at which the devices harvest energy from ambient energy sources like Radio Frequency (RF) energy. A device continuously harvests energy at a constant rate between 1.5mJ and 2.5mJ per minute, replenishing its buffer up to the maximum capacity set in 7.3.1 (Energy Capacity).

- Concurrency: determines the service request capacity of a device per time. Each device is modeled as a `simpy.Resource` that only processes a single service request at any given time.

User Service Request Generation: The user service requests were randomly generated by selecting a user from the list of specified users and assigning a sensing request type, response time, throughput, and security level. Each request generated is labeled as either malicious or legitimate based on the attack ratio defined in the scenario. These user service requests were consistently generated to ensure there is an optimal workload during the

simulation. The `inter-request interval` is the time taken between the generation of consecutive requests. It is modeled as a random variable derived from a uniform distribution ranging from $U(1,2)$ to $U(5,10)$ requests per minute. This was done to model a high-load scenario with a steady but non-deterministic stream of requests. Each request is randomly assigned a desired security level $\{Low, Medium, High\}$, response time, and throughput requirement.

Energy Cost: It is the amount of energy a device requires to fulfill the service request of a user. It represents a composite value assigned based on the security level and data throughput of the service request. The energy costs, defined in milliJoules (mJ), are summarized in Table 7.1. The maximum legitimate energy cost, denoted as C_{max_legit}, is the sum of the highest possible security, transmission (TX), and reception (RX) costs.

Table 7.1: Energy Cost Configuration

Parameter	**Level/Throughput**	**Cost (mJ)**
Security	Low	2
	Medium	4
	High	6
Data TX (kbps)	10	5
	20	7
	50	9
Data RX (kbps)	10	3
	20	4
	50	5

Adversarial Model: To evaluate the resilience of the proposed framework, a pre-configured, non-overlapping attack schedule was randomly generated for each simulation run. This ensured the defense mechanisms were not tailored to a predictable

threat pattern. Four types of Energy Depletion Attacks (EDAs) were considered, and they are as follows: flooding, spoofing, jamming, and stealth attacks. The stealth attack was conducted to allow for the detection of false positives. Furthermore, a stress test was conducted to assess the load capacity of the proposed framework. To conduct the stress test, the proposed framework was inundated with 1000 requests. The following are the characteristics of the different attacks implemented in our adversarial model.

- Flooding: A high-frequency burst of requests lasting 30 time units.
- Spoofing: A 30-time-unit period of requests with illegitimate identities.
- Jamming: A 60-time-unit period where device communication is impaired, forcing costly retransmissions.
- Stealth Attack: A 40-time-unit period of legitimate-looking requests that incur a subtle, extra energy cost.

7.3.2 Values Used in our Simulations

To simulate diverse real-world deployment scenarios, a range of values was used to evaluate the effectiveness of our proposed framework under various conditions. These parameters were systematically varied to model different network scales, user loads, and threat intensities. Table 7.2 presents the values that were implemented in our simulation.

Our choice of this range of values in Table 7.2 was informed by a review of the existing literature on Wireless Sensor Networks (WSNs) and IoT deployments [154] [155] [156]. This helped establish a challenging and realistic simulated EH-IoT environment. A detailed justification for these values is as follows.

- *Network Scale:* The number of devices (5-50) and users (10-50) was chosen to represent a small-to-medium-scale IoT deployment, such as an IoT network for a

Table 7.2: Simulation Parameter Values

Parameter	Value / Range
Simulation Duration (minutes)	1440 minutes
Number of Legitimate Requests	{400, 500, 600}
Inter-Request Interval (requests/minute)	{(1, 2), (3,5), (5,10)}
Number of Users	{10, 20, 30, 40, 50}
Number of Devices	{5, 10, 20, 35, 50}
Attack Ratio (% of malicious request)	{00, 10, 30, 50}
Energy Threshold (mJ)	{10, 20, 30, 40}
Energy Harvest Rate (mJ/minute)	{1.0, 1.5, 2.0, 2.5}

home automation system. This scale is representative of many practical, bounded scenarios and allows for a clear analysis of network dynamics without prohibitive computational cost [156].

- *Traffic Load:* The inter-request interval is the duration of time between the generation of two consecutive user service requests. The values (1,2) requests/minute model a high-traffic environment. This is representative of event-driven applications, such as motion detection, that generate bursts of data, or critical monitoring systems, like industrial control, where high-frequency reporting is required. This choice deliberately puts pressure on the system's resources, making the impact of energy depletion and processing delays more pronounced. Higher intervals like (5,10) requests/minute mean less frequent requests [154]

- *Threat Model:* The attack ratio, representing the percentage of malicious users in the network, ranges from 10% to 50%. This range is consistent with threat models used in a significant body of security research for evaluating the robustness of network protocols. A ratio of 10% can represent an incidental threat or a small number of compromised nodes, while ratios up to 50% are used to stress-test the framework

against large-scale, coordinated threats like botnets, which is a common practice for demonstrating protocol resilience [157].

- *Energy Dynamics:* The values for the energy threshold and harvest rate are derived from the power budgets of real-world, low-power IoT hardware. A typical request in our simulation costs approximately 15-25 mJ. If a time unit is one second, this corresponds to a power consumption of 15-25 milliwatts (mW) for an active transaction, which aligns with the operational power consumption of common IoT nodes using protocols like Zigbee or BLE. The simulated energy harvesting rate of 1.5mJ - 2.5 mJ per unit (i.e., 1.5mW - 2.5mW) is a realistic value for devices powered by small indoor photovoltaic cells or efficient RF energy harvesters. The values of the energy threshold (10-40 mJ) represent conservative power management policies designed to ensure a device retains enough energy for critical functions, a common strategy in energy-aware WSNs [158].

7.3.3 Scenarios Considered

Two distinct scenarios were considered in our simulation. These scenarios, namely, the baseline control model and secure model, have been described in Section 7.2.2. In the baseline control model, a device-centric approach is simulated. This scenario does not consider the user service request behavior; hence, it has minimal security features. In the secure control model, the user service request behavior is simulated with the device-centric approach to provide a robust, secure framework for EH-powered devices. **The secure model is the full implementation of our proposed framework**, as described in Section 7.2.2.2. These scenarios were implemented concurrently. A comparative analysis to validate the effectiveness of our proposed framework was conducted between the baseline control model and the secure model.

Baseline Control Model: In this simulation, only the energy levels of EH-powered devices are considered for admission in the dynamic admission control module in our proposed framework, which is implemented and evaluated. It is representative of the device-centered adaptive energy security approaches described in Section 3.4.2. The following are the components of this simulation.

- Request Generation: Legitimate and malicious requests are generated according to the specified user counts and attack ratio, as shown in Table 7.2.
- Middleware Forwarding: The middleware in this model acts as a simple 'pass-through' layer, as no security module is activated. It performs no security checks, such as no request rate limiting or anomaly detection. All incoming requests are forwarded directly to a target EH-powered device.
- Device-Level Processing: The device performs a single, reactive check. A user service request is processed only when the current energy level is sufficient to cover the request's energy cost; otherwise, it is denied.

This model is vulnerable, as it processes any request, legitimate or malicious, as long as the device has enough energy to process the request, including covering high cryptographic overhead, leading to accelerated energy depletion. The algorithm for this simulation is given in Algorithm 7.

Secure Model: For this model, all the different security modules in our proposed framework - dynamic admission control, request rate limiting, and anomaly detection modules - are implemented to provide a user-centric secure energy scheduling framework. It is representative of the proposed framework described in Section 7.2. The algorithm for this simulation is given in Algorithm **??**. The following are the components of this simulation.

Algorithm 7 Baseline Control Model Simulation

Function *run_control_simulation(env, devices, users, energy threshold, request rate, config, attack ratio, interval)*

```
    Class ControlDevice
        // Defines device behavior with energy-aware admission
    End
    Class ControlMiddleware
        generate_requests()        // Generate legitimate and malicious
           requests with cost
        forward_requests()     // Forward requests to available devices
           without admission control
        simulate_attacks()  // Simulate flooding, spoofing, and jamming
           attacks
    End
    // 1.  Initialize Components
    middleware ← ControlMiddleware(env, devices, users, energy threshold, request rate,
     config, attack ratio, interval)
    for d in devices do
        env.process(d.run())
    end
    env.process(middleware.run())
    // 2.  Run Simulation
    env.run(until=duration)
    // 3.  Calculate and Return Results
    final_metrics ← calculate_metrics(energy use, device depletion, latency)
    return event_log, final_metrics
End
```

- Request Generation: An identical stream of legitimate and malicious requests is generated as in the baseline run.
- Proactive Middleware Filtering: Upon receiving a user service request, the secure middleware executes a sequence of checks *before* forwarding it to the device:
  - *Blacklist Check:* The source user ID is checked against a dynamic blacklist. If found, the request is immediately dropped.
  - *Flood & Spoof Detection:* Behavioral and heuristic checks are applied to identify and block ongoing flooding and spoofing attacks in real-time.
- Intelligent Device Analysis: If a request passes the security modules on the middleware, it is sent to the target EH-powered device, which performs its analysis:
  - *Cost-Based Anomaly Detection:* The device calculates the projected energy cost of the user service request. If the energy cost is abnormally high (exceeding a defined energy threshold), it is flagged as an anomaly.
  - *Feedback Loop:* The device reports any detected anomaly back to the middleware, sensor cloud control layer. This report acts as a ‘strike‘ against the source user, contributing to their potential blacklisting.
- Final Energy-Aware Admission: Only requests that have passed all prior security checks are finally evaluated against the device's available energy and processed if sufficient energy reserves exist.

Through the implementation of these two scenarios for every range of values for the parameters, comparative data on key metrics were collected. This provided a quantitative approach to evaluate the effectiveness of our proposed framework in preventing Energy

Algorithm 8 Secure Simulation

Function *run_secure_simulation(env, devices, users, energy threshold, request rate, config, attack ratio, interval)*

```
    Class SecureDevice
        detect_anomaly(request)    // Implement dynamic anomaly detection
           (threshold, cost factor)...
        filter_requests(request)    // Perform energy-aware filtering and
           track false negatives...
    End
    Class SecureMiddleware
        generate_requests()        // Generate legitimate and malicious
           requests with cost...
        track_strikes(user)             // Track user strikes and manage
           blacklist...
        block_attacks(request)    // Detect and block flooding, spoofing,
           and jamming attacks...
        forward_requests(request) // Forward filtered requests to matched
           devices...
    End

    // 1.  Initialize Components
    secure_middleware ← SecureMiddleware(env, devices, users, energy threshold, request
      rate, config, attack ratio, interval)
    for d in devices do
        env.process(d.run())
    end
    env.process(secure_middleware.run())

    // 2.  Run Simulation
    env.run(until=duration)

    // 3.  Calculate and Return Results
    final_metrics ← calculate_metrics(attack_prevention, etc.)
    return event_log, final_metrics
End
```

Depletion Attacks (EDAs) in EH-powered devices, while ensuring QoS is not compromised for legitimate users.

Data Collection: A detailed event log is generated in a CSV file for each simulation run based on a specific configuration for the comparative analysis between the baseline control model and the secure model. This log captures every significant event, including request generation, energy updates, attack initiations, and final request outcomes (e.g., fulfilled, rejected due to low energy, blocked by security). These values are aggregated into a single file for further comparative analysis.

7.3.4 Performance Metrics

To evaluate the performance, resilience, and efficiency of the proposed framework, we utilized a set of evaluation metrics. These metrics are categorized into three key areas: Quality of Service (QoS) and performance, security and resilience, and resource efficiency. Together, they present a comprehensive assessment of the system's resilience - its ability to repel advanced EDAs while maintaining service for legitimate users. The following is a description of the performance metrics.

7.3.4.1 Quality of Service (QoS) and Performance Metrics

These metrics evaluate the system's ability to perform its core duties from the perspective of a legitimate user.

- *Legitimate Request Fulfillment Rate (LFR):* It measures the utility of the system. It is defined as the proportion of all legitimate user requests that are successfully processed and fulfilled by the IoT devices. A high LFR, especially under attack conditions, indicates that the security framework successfully preserves system availability for its intended users. The LFR is formally defined as:

$$LFR = \frac{R_{\text{legit\_fulfilled}}}{R_{\text{legit\_received}}} \tag{7.12}$$

where $R_{\text{legit\_fulfilled}}$ is the total count of fulfilled requests from legitimate users, and $R_{\text{legit\_received}}$ is the total count of requests received from legitimate users.

- *Legitimate Request Denial Rate (LDR):* It quantifies the collateral damage of attacks on legitimate users. It is the proportion of legitimate requests that were denied service, often due to device resource exhaustion, such as insufficient energy, caused by the strain of ongoing attacks. A low LDR is desirable, as it indicates the system can absorb the impact of an attack without significantly degrading service for good actors. The LDR is formally defined as:

$$LDR = \frac{R_{\text{legit\_denied}}}{R_{\text{legit\_received}}} \tag{7.13}$$

where $R_{\text{legit\_denied}}$ is the total count of denied requests from legitimate users and $R_{\text{legit\_received}}$ is the total count of requests received from legitimate users.

7.3.4.2 Security and Resilience Metrics These metrics directly measure the effectiveness of the security mechanisms in identifying and mitigating threats.

- *False Negatives (FN):* represents the number of malicious requests that the secure IoT framework failed to detect and passed to the IoT devices. The primary goal of any security system is to minimize the FN count. The FN count is formally represented as:

$$FN = |A_{\text{passed}}| \tag{7.14}$$

where A_{passed} is the set of all malicious attack requests that were processed by a device.

- *False Positives (FP):* It is the count of legitimate requests that were incorrectly identified as malicious and subsequently blocked by the secure IoT framework. The

FP count is formally represented as:

$$FP = |R_{\text{legit\_blocked}}| \tag{7.15}$$

where $R_{\text{legit\_blocked}}$ is the set of all legitimate requests blocked by the security middleware.

- *Attackers Blacklisted by Anomaly Detection System (ADS):* It evaluates the efficacy of the anomaly detection module in our proposed framework. It counts the number of unique malicious users identified based on their anomalous energy consumption patterns. These users are blacklisted from further interaction with the secure IoT framework. A higher count indicates a more effective and responsive ADS. The count of blacklisted attackers is formally represented as:

$$ADS_{\text{blacklisted}} = |U_{\text{blacklisted}}| \tag{7.16}$$

where $U_{\text{blacklisted}}$ is the set of unique malicious user IDs added to the blacklist.

- *Illegitimate Requests Serviced (IRS):* This metric is unique to the *baseline control model*, described in Section 7.2.2.1. It quantifies the system's vulnerability by counting the number of malicious (non-flooding) requests that the device was tricked into processing. The IRS count is formally represented as:

$$IRS = |A_{\text{serviced}}| \tag{7.17}$$

where A_{serviced} is the set of non-flooding malicious requests that were successfully processed by a device in the baseline control model.

7.3.4.3 Resource Efficiency Metrics This metric assesses the system's ability to manage its finite resources, which is critical for the longevity of energy-harvesting IoT devices.

- *Network Energy Efficiency (η_{energy}):* This metric provides a network-wide view of energy sustainability. It is defined as the ratio of the net energy (total energy harvested minus total energy consumed) to the total energy harvested. A positive η_{energy} indicates an energy surplus and sustainable operation, while a negative η_{energy} signals an energy deficit, meaning the network is on a path to systemic failure. The network energy efficiency is formally defined as:

$$\eta_{\text{energy}} = \frac{\sum_{i=1}^{N} (E_{\text{harvested}}^{(i)} - E_{\text{consumed}}^{(i)})}{\sum_{i=1}^{N} E_{\text{harvested}}^{(i)}} \tag{7.18}$$

  where $E_{\text{harvested}}^{(i)}$ and $E_{\text{consumed}}^{(i)}$ are the total energy harvested and consumed by device i, respectively, over the simulation duration, and N is the total number of devices in the network.

7.4 Results and Discussions

In this section, we present a comprehensive analysis of the simulation results, focusing on the sensitivity of key performance and security metrics to variations in critical network and attack parameters. The objective of our simulation is to quantify the impact of these parameters on the baseline control model and the proposed secure model, thereby evaluating the effectiveness and trade-offs of the security mechanisms. The sensitivity analysis examines the influence of the device and user count, attack ratio, energy threshold, request interval, and harvest rate on Legitimate Request Fulfillment Rate (LFR), Legitimate Request Denial Rate (LDR), the number of illegitimate requests serviced, and the network energy efficiency. Furthermore, for the proposed framework, security-specific metrics such as false positives, false negatives, and counts for the attacks blacklisted by the anomaly detection system, prevention counts were considered.

Table 7.3: Comparative Performance and Security Analysis Across Varying Request Loads

Metric	Mode	100R	200R	300R	400R	500R	1000R
Quality of Service Metrics							
Fulfillment Rate (LFR)	Control	86.01%	81.77%	83.35%	81.34%	81.17%	80.19%
	Secure	**89.08%**	**85.74%**	**88.46%**	**87.85%**	**86.91%**	**87.69%**
Denial Rate (LDR)	Control	9.99%	10.16%	10.64%	13.66%	12.79%	13.69%
	Secure	**6.64%**	**5.96%**	**5.19%**	**6.78%**	**6.63%**	**5.87%**
Security Effectiveness Metrics							
Illegit. Serviced (IRS)	Control	**21**	**40**	**60**	**77**	**96**	**185**
	Secure	**0**	**0**	**0**	**0**	**0**	**0**
False Negatives (FN)	Control	N/A	N/A	N/A	N/A	N/A	**N/A**[b]
	Secure	17	24	26	27	28	28
False Positives (FP)	Control	N/A	N/A	N/A	N/A	N/A	N/A
	Secure	**0**	**0**	**0**	**0**	**0**	**1**[a]
Attackers Blacklisted (ADS)	Control	N/A	N/A	N/A	N/A	N/A	N/A
	Secure	**3**	**5**	**7**	**7**	**7**	**8**
Energy Efficiency Metric							
Energy Efficiency (η_{energy})	Control	96.66%	93.67%	90.35%	87.48%	84.37%	69.47%
	Secure	**97.33%**	**95.26%**	**93.06%**	**91.18%**	**89.33%**	**79.83%**

[a] A single false positive was observed in the 1000-request batch, where a legitimate user was blacklisted.

[b] N/A values for FN, FP, and ADS in the Control mode are due to the absence of the corresponding security detection modules.

7.4.1 Results

The performance of our proposed framework based on the metrics formally defined in eq. 7.12 through eq. 7.18 and the range of values for our simulation parameters, as discussed in Section 7.3.2, are presented in this section. These results, presented in Table 7.3, represent the aggregate outcomes obtained from the comparative analysis between the baseline control model and secure model across varying request loads ranging from 100 to 500 requests in increments of 100, as simulated.

7.4.1.1 Quality of Service (QoS) and Performance The secure model, *our proposed framework*, consistently outperforms the baseline control model in serving legitimate users. The secure model achieved an average of 87.62% in LFR and 6.17% in LDR across all request loads, in comparison to the 82.64% in LFR and 11.82% in LDR achieved by the baseline control model. This shows the ability of the proposed framework to protect against resource starvation and cascading system failure by proactively filtering the service request to identify malicious traffic that can cause EDAs. The ability to preserve resources and ensure energy efficiency guarantees high availability for the EH-powered system to fulfill legitimate user service requests. This leads to increased QoS, irrespective of increased traffic in the system.

7.4.1.2 Security Effectiveness The values as shown in Table 7.3 highlight the effectiveness of a multi-layer system for the proposed framework. The baseline control model achieved an average of 79 IRS because it is highly susceptible to malicious attacks such as flooding and spoofing, as the security measures are only implemented at the device level. However, the secure model achieved 0 IRS in all scenarios, demonstrating the effectiveness of mitigating illegitimate requests at the device level before it is transmitted to the target device. This proves that integrating the user service request with the device-centric approach makes the proposed framework resilient and robust, as there is no single point of failure.

Furthermore, the secure model successfully identified and blocked malicious requests by achieving a total of 6 blacklisted malicious users using the $ADS_{blacklisted}$ metric. Though the FN is non-zero, with an average of 25 requests, it suggests the security modules in the sensor cloud control layer can be improved to identify and block high-volume attacks such as spoofing and flooding. However, it is still susceptible to advanced stealth attacks. Since

no security countermeasure was applied to the baseline control model, there are no results achieved for the `FN, FP`, and `ADS`.

7.4.1.3 Network Energy Efficiency (η_{energy}) This measures the percentage of harvested energy that is preserved by an EH-powered device. In η_{energy}, the secure model achieved 91.67%, while the baseline control model achieved 87.33%. This proves that a multi-layered approach ensures there is a high rate of energy preservation and subsequently prevents EDAs.

In the baseline control model, there is a significant increase in the energy cost used when there is a heavy attack, due to the lack of a robust security countermeasure. Additionally, the processing of malicious traffic that seeks to deplete energy was identified as the primary source of energy wastage in an EH-powered system. Our proposed framework eliminates this energy waste, thereby improving the energy sustainability, leading to an increased operational lifetime of the device.

7.4.2 Sensitivity Analysis

A comprehensive sensitivity analysis was undertaken to examine the robustness and response behavior of the proposed framework, *secure model*, against the *baseline control model* under varying operational conditions. Sensitivity analysis was performed for the device and user counts, the attack ratio, energy threshold, request intervals, and energy harvesting rates. In this section, we present and discuss the sensitivity analysis that provided the most interesting insights.

7.4.2.1 Sensitivity to Device Count This involves analyzing the impact of varying the number of devices in the simulation environment on both performance and security metrics. The values implemented in this scenario are presented in Table 7.4.

Table 7.4: Values for sensitivity analysis on device count

Parameters	Value / Range
Device Count	[5, 10, 20, 35, 50]
User Count	30
Attack Ratio	0.3
Energy Threshold	30
Request Interval	$U(3,5)$
Harvest Rate	1.5

From our analysis, we observed that as the number of devices increases, the LFR increases in both models, while the denial rate decreases, as shown in Fig. 7.3 for the control model, and Fig. 7.4 for the secure mode. This is because the large device pool provides a greater capacity to handle the request load. When the device count is at its lowest, the secure model still shows a higher fulfillment rate, indicating minimal security overhead for the devices when compared with the control model.

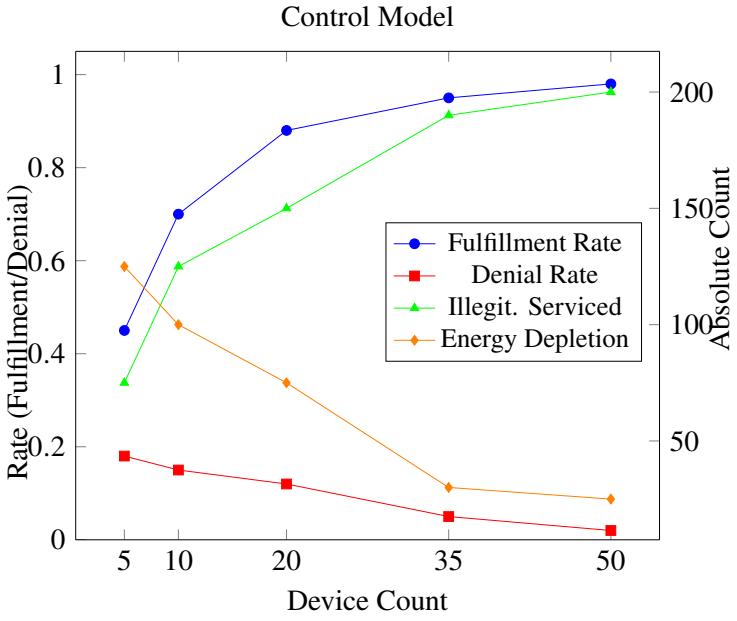

Figure 7.3: Sensitivity analysis of the control model based on the number of devices

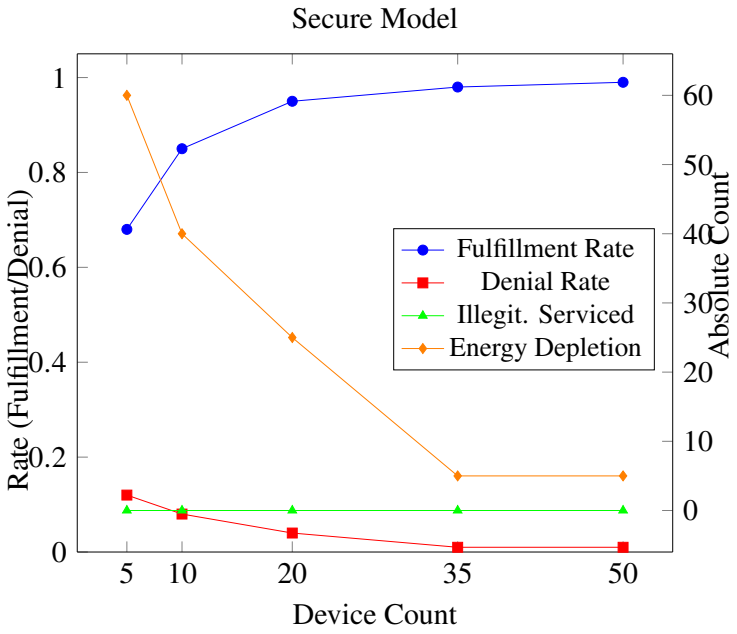

Figure 7.4: Sensitivity analysis of the secure model based on the number of devices

Additionally, the secure model does not process any illegitimate requests irrespective of the device count, and has fewer service request denials even with a lower number of devices. Compared to the control model, the number of illegitimate requests serviced increases as the number of devices increases, and they have a higher number of service denials with fewer devices. The total energy depletion rate significantly decreases with more devices, and the Secure mode exhibits substantially fewer depletion events than the Control mode, underscoring the energy-saving benefit of preventing attacks. Security-specific metrics in the Optimized Secure mode (false positives, false negatives, prevention counts) show less direct sensitivity to device count variations, primarily reflecting the effectiveness of the security logic itself rather than the number of devices.

7.4.2.2 Sensitivity to Attack Ratio This involves analyzing the proportion of malicious users in the IoT network, as a direct measure of the attack intensity. The values implemented in this scenario are presented in Table 7.5.

Table 7.5: Values for sensitivity analysis on attack ratio

Parameters	**Value / Range**
Attack Ratio	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
User Count	30
Device Count	20
Energy Threshold	30
Request Interval	$U(3,5)$
Harvest Rate	1.5

The results from the analysis are depicted in Fig. 7.5 for the control model, and Fig. 7.6 for the secure mode.

In this analysis, we observed in the control mode that the attack ratio has a direct negative impact on the LFR, number of illegitimate requests serviced, request denial rate, and energy depletion rate. As the attack ratio increases, the fulfillment rate sharply decreases while other factors increase linearly with the attack rate. Indicative of a direct relationship between successful intrusion and attack intensity. However, in the secure model, the LFR is only slightly affected when there is a higher attack rate (i.e., 50% of the requests were malicious). In addition, illegitimate requests continue to be correctly detected and only very few were serviced, even at the highest attack ratio. The secure model was able to detect an attack

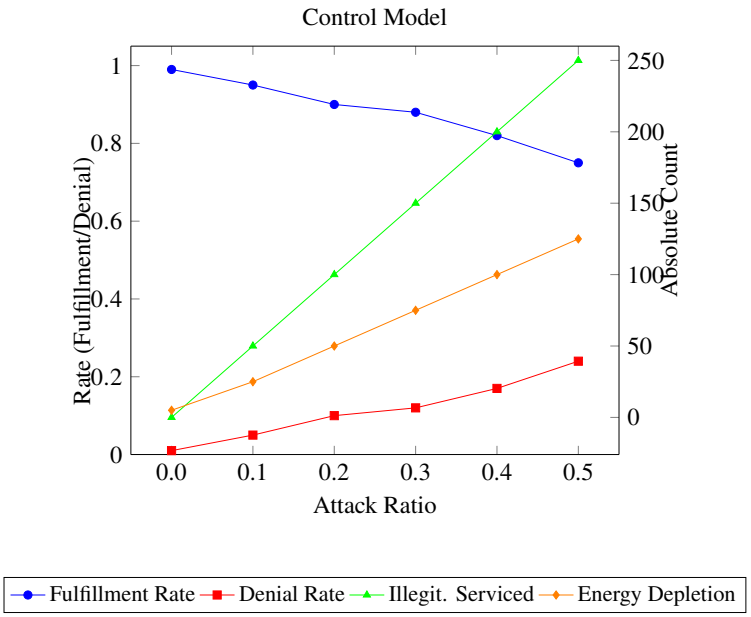

Figure 7.5: Sensitivity analysis of control model based on attack ratio

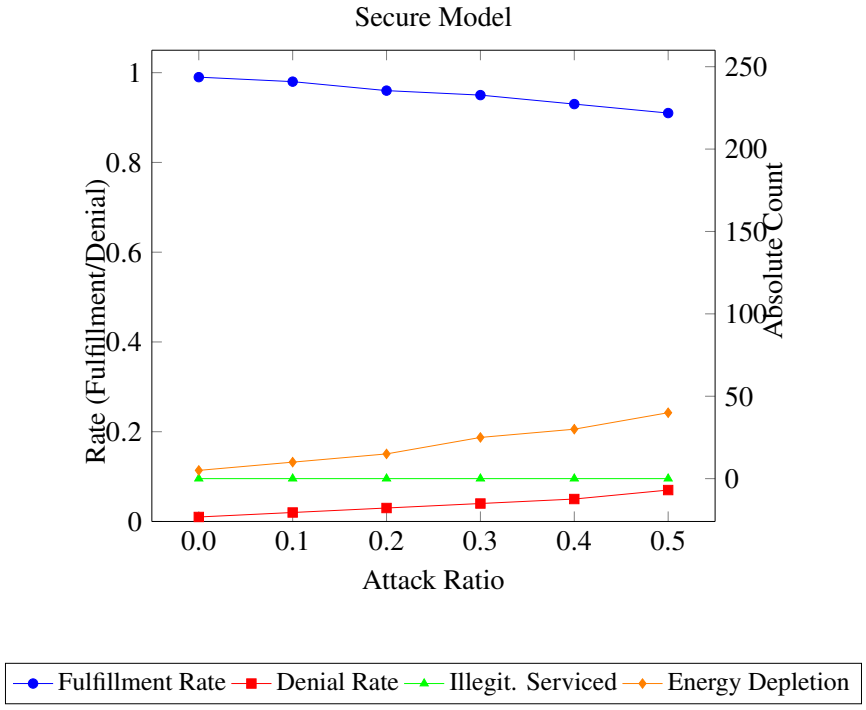

Figure 7.6: Sensitivity analysis of secure model based on attack ratio

through its security modules, hence mitigating the impact of the high attack volume, which is a core strength of our framework.

7.4.2.3 Sensitivity to Energy Threshold This involves analyzing the impact of varying the device-level energy threshold for the anomaly detection module within the baseline control and secure model. The values implemented in this scenario are presented in Table 7.6.

Table 7.6: Values for sensitivity analysis on energy threshold

Parameters	**Value / Range**
Energy Threshold (mJ)	[10, 20, 30, 40]
User Count	30
Device Count	20
Attack Ratio	0.3
Request Interval	$U(3,5)$
Harvest Rate	1.5

We observed that in the secure model, the LFR decreases as the energy threshold is increased, as shown in Fig.7.7 for the control model, and Fig.7.8 for the secure model. This pattern was consistent, even when the energy threshold was set at the highest. However, a higher energy threshold results in a more aggressive anomaly detection, potentially identifying legitimate high-energy cost user service requests as an anomaly, resulting in False Positives (FP).

Conversely, the number of IRS is significantly lower when the energy threshold is set to a higher value. Likewise, the rate of denial significantly reduces because malicious requests

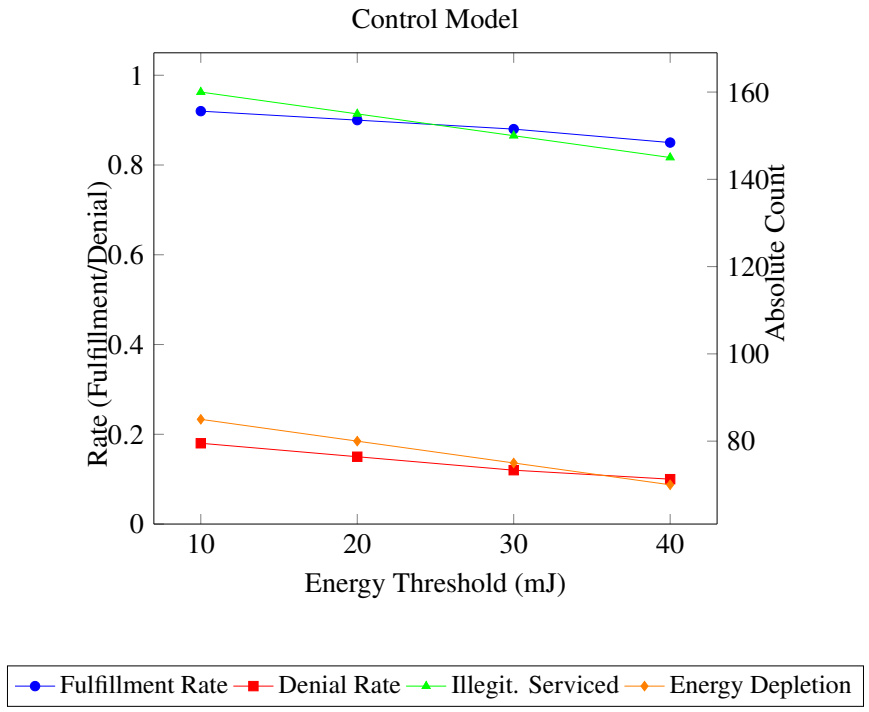

Figure 7.7: Sensitivity analysis of control model based on device energy threshold

with a higher energy cost are easier to detect. This demonstrates a clear tradeoff - a higher threshold improves security detection against costly requests, but increases the rate of false positives.

In the secure control model, the energy depletion of the EH-powered devices decreases as only a few high-cost requests are processed, leading to a low denial rate when the energy thresholds are increased. However, the amount of the `IRS` remains high (145 successful attacks) in the baseline control model, despite the increased energy threshold. This is because the utilization of a device-centric approach is reactive and insufficient to guard against security attacks, as it is unable to protect the network. This establishes the resilience of our proposed framework as it demonstrates the ability of the security modules in the

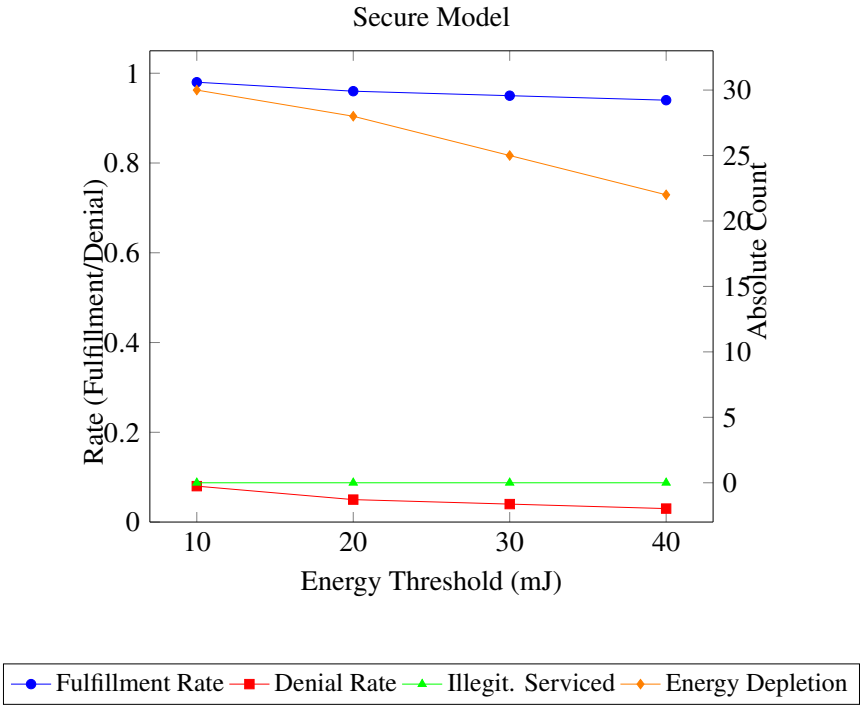

Figure 7.8: Sensitivity analysis of secure model based on device energy threshold

middleware to proactively filter out the majority of the malicious requests through flood detection, anomaly detection, and user blacklisting.

7.5 Conclusion and Future Work

Energy Depletion Attacks (EDAs) represent a critical threat to the operational viability of Energy-Harvesting Internet of Things (EH-IoT) networks, as traditional security measures can impose an unsustainable energy burden. This paper proposed and evaluated a novel user-centric sensor-cloud-based secure admission control framework designed specifically to counteract EH IoT energy depletion attacks. Through a comprehensive discrete-event simulation that modeled a wide range of operational conditions, user loads, and adversarial strategies, we demonstrated the profound effectiveness of our proposed architecture in proactively identifying and mitigating threats before they can impact device resources.

The results clearly show that our proposed secure framework outperforms the baseline control model. Key findings reveal that our framework reduces the number of successful intrusions to zero across all test scenarios under heavy request load, compared to 185 successful attacks in the control. It also significantly improves the Quality of Service for legitimate users by maintaining a higher fulfillment rate - 87.7% and lower denial rate -6% under high adversarial pressure. Our secure model achieved a network energy efficiency of 79.8% as against 69.5% in the control. From the results, we conclude that a proactive, multi-layered defense strategy is practically better than a reactive device-side defense strategy. Furthermore, by preventing the wasteful processing of malicious traffic, the framework acts as a critical energy preservation mechanism, proving that robust security and network sustainability are not mutually exclusive goals.

While this study provides strong evidence of the framework's efficacy, its validation is currently based on a simulated environment. Future work will therefore focus on deploying the framework on physical hardware to account for real-world challenges like network latency and packet loss. To enhance resilience against the subtle stealth attacks identified in our analysis, we plan to advance the system's intelligence by replacing the current heuristics with more sophisticated machine learning models for anomaly detection. Furthermore, the framework's robustness will be tested under conditions of energy uncertainty by integrating variable energy harvesting profiles such as solar and vibration energy harvesting, and we ultimately aim to evaluate the system against co-adaptive adversaries, potentially modeled using reinforcement learning, to drive the development of defenses that can respond to intelligent, evolving threats.

CHAPTER EIGHT

CONCLUSION

This chapter synthesizes the key findings and conclusions and outlines the future work stemming from the body of research presented in this dissertation. The work progressed from an initial investigation of lightweight cryptography to the development of sophisticated frameworks for Quality of Service (QoS) and energy management, culminating in a security-centric architecture for Energy-Harvesting Internet of Things (EH-IoT) networks.

8.1 Key Findings

The research yielded significant and quantifiable improvements across all stages of investigation. In the initial exploration of lightweight cryptography, the **NtruEncrypt** protocol was identified as a more efficient and viable solution for IoT environments compared to the commonly used Blowfish protocol. Its primary advantages were its ability to process larger cipher blocks, making it more suitable for encrypting larger files, and its more straightforward, more streamlined process for generating encryption keys, which is critical for resource-constrained devices.

Building on this, the development of a **QoS-aware predictive framework** demonstrated substantial gains in network performance. By integrating VAR time-series forecasting with a QoS recommender engine, the framework achieved a **20% extension in the operational lifetime** of the network and a **service demand prediction accuracy of up to 95%**. This proactive approach also led to a significant increase in the number of fulfilled user requests when compared to traditional baseline methods, showcasing its effectiveness in managing resources intelligently.

The investigation into energy management for EH-IoT networks revealed that a joint optimization and energy allocation framework can effectively manage RF energy harvesting

to improve service provisioning while accommodating varying QoS and security needs. The framework yielded **considerable improvements in extending network lifetime, increasing energy usage efficiency, and achieving a higher rate of fulfilled user requests** compared to other recent works in the field. This finding highlighted the critical link between efficient energy allocation and overall network sustainability.

Finally, the research culminated in a framework designed to counter Energy Depletion Attacks (EDAs), a critical threat to EH-IoT networks. The proactive, user-centric, sensor-cloud-based admission control framework proved to be profoundly effective, **reducing the number of successful intrusions to zero** across all tested scenarios. It significantly improved QoS for legitimate users, maintaining a high **fulfillment rate of 87.7%** even under high adversarial pressure. Crucially, the secure framework improved overall **network energy efficiency to 79.8%** (compared to 69.5% in the control model), proving that robust security can also serve as an energy preservation mechanism.

8.2 Conclusions

This dissertation has successfully demonstrated the progressive development and integration of security, QoS, and energy management to address critical challenges in IoT environments.

The initial phase of this research established a foundational understanding of lightweight cryptography in IoT. Through a survey and performance analysis of symmetric and asymmetric protocols, we identified NtruEncrypt as a suitable and efficient option for IoT applications. By integrating NtruEncrypt into the CupCarbon simulator, we demonstrated its superior performance over the built-in Blowfish protocol, particularly in handling larger data blocks and simplifying key generation. This work validated the feasibility

of implementing robust security as a non-functional parameter in IoT simulations, laying the groundwork for more complex system designs.

Building upon this, the research introduced a novel, QoS-aware predictive framework that integrates Vector Autoregression (VAR) time-series forecasting with a QoS-constrained recommender engine. This proactive, learning-based strategy proved more effective than traditional task offloading or heuristic methods, achieving a 20% extension in network operational lifetime and a significant increase in fulfilled user requests. With a prediction accuracy of up to 95%, the framework effectively addressed the cold-start problem for new users and devices and demonstrated a scalable, shard-based architecture capable of supporting millions of devices. This contribution highlighted the critical link between proactive resource allocation, QoS assurance, and network sustainability.

The subsequent research focused on the vital yet often overlooked area of energy security in EH-IoT networks. We developed a joint optimization and energy allocation framework to manage Radio Frequency (RF) energy harvesting, dynamically adapting to varying network conditions to enhance service reliability. This framework yielded considerable improvements in extending network lifetime and increasing energy usage efficiency compared to contemporary approaches. The work underscored that while optimizing energy efficiency is crucial, securing the energy harvesting and allocation process against threats like eavesdropping or denial of energy attacks is paramount for ensuring the operational integrity of sustainable IoT deployments.

Finally, to directly counter the threat of Energy Depletion Attacks (EDAs), we proposed and evaluated a user-centric, sensor-cloud-based secure admission control framework. Through comprehensive discrete-event simulations, this framework demonstrated profound effectiveness, reducing successful intrusions to zero under heavy request loads and significantly improving QoS for legitimate users. The secure model achieved a network

energy efficiency of 79.8% compared to 69.5% in the control model. This concluding piece of research established that a proactive, multi-layered defense strategy is not only practically superior to reactive, device-side defenses but also proves that robust security and network sustainability can be achieved concurrently, acting as a critical energy preservation mechanism.

8.3 Future Work

The findings from this dissertation open several promising avenues for future research, aiming to enhance the robustness, intelligence, and real-world applicability of the proposed frameworks.

A primary thrust of future work will be the transition from simulation to physical implementation. While simulations provided a controlled environment for validation, deploying the frameworks on physical hardware is essential to account for real-world challenges such as network latency, packet loss, unpredictable RF interference, and hardware-to-hardware performance variability. This will involve extending the functionalities of the integrated NtruEncrypt protocol to offer users varying degrees of security based on performance trade-offs observed in real-world deployments.

Another significant direction is the enhancement of the frameworks' intelligence by integrating more sophisticated machine learning models. To counter subtle stealth attacks and co-adaptive adversaries, the current heuristics in the secure admission control framework will be replaced with advanced anomaly detection models, potentially using reinforcement learning to develop defenses that can respond to intelligent and evolving threats. This theme of continuous model improvement is critical for adapting to the dynamic nature of IoT ecosystems.

Future research will also focus on expanding the scope of energy management. This includes incorporating diverse and variable ambient energy sources, such as solar and vibration, into the energy harvesting profiles to test the frameworks' robustness under conditions of energy uncertainty. Furthermore, we will focus on designing lightweight security countermeasures specifically to protect the energy harvesting and allocation processes from unauthorized manipulation, ensuring the efficient and secure distribution of harvested energy.

Finally, the integration of security will be deepened across all aspects of resource management. Future iterations of the QoS-aware framework will use service prediction and recommendation to enhance security by incorporating encryption and data protection measures directly into the allocation process. This involves recommending devices that comply with requested service encryption standards, thereby preserving user privacy and adapting recommendations based on the security context. By pursuing these interconnected research threads, we aim to build upon the foundations laid in this dissertation to create even more secure, efficient, and resilient IoT systems.

REFERENCES

[1] P. Suresh, J. V. Daniel, V. Parthasarathy, and R. H. Aswathy, "A state of the art review on the internet of things (iot) history, technology and fields of deployment," in *2014 International Conference on Science Engineering and Management Research (ICSEMR)*, pp. 1–8, 2014.

[2] R. Yu and T. Watteyne, "Reliable, low power wireless sensor networks for the internet of things: Making wireless sensors as accessible as web servers," *Linear Technology*, vol. 12, 2013.

[3] Z. Khan, Z. Pervez, and A. G. Abbasi, "Towards a secure service provisioning framework in a smart city environment," *Future Generation Computer Systems*, vol. 77, pp. 112–135, 2017.

[4] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of things (iot) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5g-iot scenarios," *IEEE Access*, vol. 8, pp. 23022–23040, 2020.

[5] M. Asad, A. Basit, S. Qaisar, and M. Ali, "Beyond 5g: Hybrid end-to-end quality of service provisioning in heterogeneous iot networks," *IEEE Access*, vol. 8, pp. 192320–192338, 2020.

[6] A. Sen, K. Fletcher, and S. Madria, "A secure user-centric framework for dynamic service provisioning in iot environments," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 334–341, 2019.

[7] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally, "Internet of things (iot): Research, simulators, and testbeds," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1637–1647, 2018.

[8] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2: Lightweight authenticated encryption and hashing," *Journal of Cryptology*, vol. 34, no. 3, 2021.

[9] M. Philip and Vaithiyanathan, "A survey on lightweight ciphers for iot devices," in *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*, pp. 1–4, 12 2017.

[10] X. Lei and X. Liao, "Ntru-ke: A lattice-based public key exchange protocol," *IACR Cryptol. ePrint Arch.*, vol. 2013, p. 718, 2013.

[11] G. Kecskemeti, G. Casale, D. N. Jha, J. Lyon, and R. Ranjan, "Modelling and simulation challenges in internet of things," *IEEE Cloud Computing*, vol. 4, no. 1, pp. 62–69, 2017.

[12] T. Alam, "A reliable communication framework and its use in internet of things (iot)," vol. 3, 05 2018.

[13] H. K. Bharadwaj, A. Agarwal, V. Chamola, N. R. Lakkaniga, V. Hassija, M. Guizani, and B. Sikdar, "A review on the role of machine learning in enabling iot based healthcare applications," *IEEE Access*, vol. 9, pp. 38859–38890, 2021.

[14] V. P. Kour and S. Arora, "Recent developments of the internet of things in agriculture: A survey," *IEEE Access*, vol. 8, pp. 129924–129957, 2020.

[15] A. Oseni, N. Moustafa, G. Creech, N. Sohrabi, A. Strelzoff, Z. Tari, and I. Linkov, "An explainable deep learning framework for resilient intrusion detection in iot-enabled transportation networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 1000–1014, 2023.

[16] P. Radanliev, D. De Roure, R. Nicolescu, M. Huth, and O. Santos, "Artificial intelligence and the internet of things in industry 4.0," *CCF Transactions on Pervasive Computing and Interaction*, vol. 3, pp. 329–338, 2021.

[17] H. Rajab and T. Cinkelr, "Iot based smart cities," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–4, 2018.

[18] S. Zahoor and R. N. Mir, "Resource management in pervasive internet of things: A survey," *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 8, pp. 921–935, 2021.

[19] F. V. den Abeele, J. Hoebeke, I. Moerman, and P. Demeester, "Integration of heterogeneous devices and communication models via the cloud in the constrained internet of things," *International Journal of Distributed Sensor Networks*, vol. 11, no. 10, p. 683425, 2015.

[20] A. Chowdhury and S. A. Raut, "A survey study on internet of things resource management," *Journal of Network and Computer Applications*, vol. 120, pp. 42–60, 2018.

[21] F. Hussain, S. A. Hassan, R. Hussain, and E. Hossain, "Machine learning for resource management in cellular and iot networks: Potentials, current solutions, and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1251–1275, 2020.

[22] P. Ahlawat and C. Rana, "An era of recommendation technologies in iot: Categorisation by techniques, challenges and future scope.," *Pertanika Journal of Science & Technology*, vol. 29, no. 4, 2021.

[23] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware qos prediction with neural collaborative filtering for internet-of-things services," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4532–4542, 2020.

[24] L. Chen, S. Thombre, K. Järvinen, E. S. Lohan, A. Alen-Savikko, H. Leppäkoski, M. Z. H. Bhuiyan, S. Bu-Pasha, G. Ferrara, S. Honkala, J. Lindqvist, L. Ruotsalainen, P. Korpisaari, and H. Kuusniemi, "Robustness, security and privacy in location-based services for future iot: A survey," *IEEE Access*, vol. 5, pp. 8956–8977, 2017.

[25] Y. Zhang, J. Pan, L. Qi, and Q. He, "Privacy-preserving quality prediction for edge-based iot services," *Future Generation Computer Systems*, vol. 114, pp. 336–348, 2021.

[26] Z. Cui, X. Xu, F. XUE, X. Cai, Y. Cao, W. Zhang, and J. Chen, "Personalized recommendation system based on collaborative filtering for iot scenarios," *IEEE Transactions on Services Computing*, vol. 13, no. 4, pp. 685–695, 2020.

[27] D. Nawara and R. Kashef, "Iot-based recommendation systems – an overview," in *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pp. 1–7, 2020.

[28] A. Khelloufi, H. Ning, S. Dhelim, T. Qiu, J. Ma, R. Huang, and L. Atzori, "A social-relationships-based service recommendation system for siot devices," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1859–1870, 2021.

[29] K. Najmani, E. H. Benlahmar, N. Sael, and A. Zellou, "Collaborative filtering approach: A review of recent research," in *Advanced Intelligent Systems for Sustainable Development (AI2SD'2020)* (J. Kacprzyk, V. E. Balas, and M. Ezziyyani, eds.), (Cham), pp. 151–163, Springer International Publishing, 2022.

[30] S. Deng, Z. Xiang, P. Zhao, J. Taheri, H. Gao, J. Yin, and A. Y. Zomaya, "Dynamical resource allocation in edge for trustable internet-of-things systems: A reinforcement learning method," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6103–6113, 2020.

[31] F. Pereira, R. Correia, P. Pinho, S. I. Lopes, and N. B. Carvalho, "Challenges in resource-constrained iot devices: Energy and communication as critical success factors for future iot deployment," *Sensors (Basel, Switzerland)*, vol. 20, no. 22, p. 6420, 2020.

[32] M. Shirvanimoghaddam, K. Shirvanimoghaddam, M. M. Abolhasani, M. Farhangi, V. Zahiri Barsari, H. Liu, M. Dohler, and M. Naebe, "Towards a green and self-powered internet of things using piezoelectric energy harvesting," *IEEE Access*, vol. 7, pp. 94533–94556, 2019.

[33] A. Padhy, S. Joshi, S. Bitragunta, V. Chamola, and B. Sikdar, "A survey of energy and spectrum harvesting technologies and protocols for next generation wireless networks," *IEEE Access*, vol. 9, pp. 1737–1769, 2021.

[34] Q. Ren and G. Yao, "Enhancing harvested energy utilization for energy harvesting wireless sensor networks by an improved uneven clustering protocol," *IEEE Access*, vol. 9, pp. 119279–119288, 2021.

[35] C. Moser, L. Thiele, D. Brunelli, and L. Benini, "Adaptive power management in energy harvesting systems," in *2007 Design, Automation & Test in Europe Conference & Exhibition*, pp. 1–6, 2007.

[36] Q. Pan, J. Wu, A. K. Bashir, J. Li, W. Yang, and Y. D. Al-Otaibi, "Joint protection of energy security and information privacy for energy harvesting: An incentive federated learning approach," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3473–3483, 2022.

[37] L. Mottola, A. Hameed, and T. Voigt, "Energy attacks in the battery-less internet of things: Directions for the future," in *Proceedings of the 17th European Workshop on Systems Security*, EuroSec '24, (New York, NY, USA), p. 29–36, Association for Computing Machinery, 2024.

[38] C.-L. Hsu and J. C.-C. Lin, "An empirical examination of consumer adoption of internet of things services: Network externalities and concern for information privacy perspectives," *Computers in Human Behavior*, vol. 62, pp. 516–527, 2016.

[39] H. Zorgati, R. B. Djemaa, and I. A. B. Amor, "Service discovery techniques in internet of things: a survey," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 1720–1725, 2019.

[40] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad hoc networks*, vol. 7, no. 3, pp. 537–568, 2009.

[41] R. Buyya and A. V. Dastjerdi, *Internet of Things: Principles and paradigms*. Elsevier, 2016.

[42] H. S. Narman, M. S. Hossain, M. Atiquzzaman, and H. Shen, "Scheduling internet of things applications in cloud computing," *Annals of Telecommunications*, vol. 72, pp. 79–93, 2017.

[43] F. C. Delicato, P. F. Pires, and T. Batista, *The Activity of Resource Modelling*, pp. 19–32. Cham: Springer International Publishing, 2017.

[44] S. M. Oteafy and H. S. Hassanein, "Towards a global iot: Resource re-utilization in wsns," in *2012 international conference on computing, networking and communications (ICNC)*, pp. 617–622, IEEE, 2012.

[45] P. Barnaghi, W. Wang, C. Henson, and K. Taylor, "Semantics for the internet of things: early progress and back to the future," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 8, no. 1, pp. 1–21, 2012.

[46] L. Li, S. Li, and S. Zhao, "Qos-aware scheduling of services-oriented internet of things," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1497–1505, 2014.

[47] T.-C. Liu, K. Wang, C.-Y. Ku, and Y.-H. Hsu, "Qos-aware resource management for multimedia traffic report systems over lte-a," *Computer Networks*, vol. 94, pp. 375–389, 2016.

[48] A. Gharib and M. Ibnkahla, "User security-oriented information-centric iot nodes clustering with graph convolution networks," *IEEE Internet of Things Journal*, vol. 11, pp. 8311–8326, 2024.

[49] A. Rahman, M. J. Islam, A. Montieri, M. K. Nasir, M. M. Reza, S. S. Band, A. Pescape, M. Hasan, M. Sookhak, and A. Mosavi, "Smartblock-sdn: An optimized blockchain-sdn framework for resource management in iot," *IEEE Access*, vol. 9, pp. 28361–28376, 2021.

[50] O. Novo, "Blockchain meets iot: An architecture for scalable access management in iot," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184–1195, 2018.

[51] W. Zhang, D. Yang, H. Peng, W. Wu, W. Quan, H. Zhang, and X. Shen, "Deep reinforcement learning based resource management for dnn inference in industrial iot," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 7605–7618, 2021.

[52] Y. Gong, D. Yu, X. Cheng, C. Yuen, M. Bennis, and M. Debbah, "Computation offloading and quantization schemes for federated satellite-ground graph networks," *IEEE Transactions on Wireless Communications*, vol. 23, no. 10, pp. 14140–14154, 2024.

[53] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, feb 2019.

[54] S. Zhou, X. Dai, H. Chen, W. Zhang, K. Ren, R. Tang, X. He, and Y. Yu, "Interactive recommender system via knowledge graph-enhanced reinforcement learning," in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp. 179–188, 2020.

[55] L. Yao, X. Wang, Q. Z. Sheng, S. Dustdar, and S. Zhang, "Recommendations on the internet of things: Requirements, challenges, and directions," *IEEE Internet Computing*, vol. 23, no. 3, pp. 46–54, 2019.

[56] K. Haseeb, I. Ud Din, A. Almogren, and N. Islam, "An energy efficient and secure iot-based wsn framework: An application to smart agriculture," *Sensors*, vol. 20, no. 7, p. 2081, 2020.

[57] V.-L. Nguyen, P.-C. Lin, and R.-H. Hwang, "Energy depletion attacks in low power wireless networks," *IEEE Access*, vol. 7, pp. 51915–51932, 2019.

[58] V. Shakhov, S. Nam, and H. Choo, "Flooding attack in energy harvesting wireless sensor networks," in *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication (ICUIMC '13)*, pp. 49:1–49:5, 2013.

[59] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "Security in energy harvesting networks: A survey of current solutions and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2658–2693, 2020.

[60] A. H. Khan, H. Ikram, C. M. Ahmed, N. U. Hassan, and Z. A. Uzmi, "Energy level spoofing attacks and countermeasures in blockchain-enabled iot," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pp. 4322–4327, 2022.

[61] X. Zeng, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos, and R. Ranjan, "Iotsim: A simulator for analysing iot applications," *Journal of Systems Architecture*, vol. 72, pp. 93–107, 2017.

[62] S. N. Han, G. M. Lee, N. Crespi, K. Heo, N. Van Luong, M. Brut, and P. Gatellier, "Dpwsim: A simulation toolkit for iot applications using devices profile for web services," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pp. 544–547, IEEE, 2014.

[63] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[64] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pp. 641–648, 2006.

[65] D. Dinu, A. Biryukov, J. Groszschädl, D. Khovratovich, Y. L. Corre, and L. Perrin, "Felics - fair evaluation of lightweight cryptographic systems," in *NIST Workshop on Lightweight Cryptography*, 2015.

[66] N. Homma, K. Aoki, T. Iwata, K. Ogawa, H. Oguma, K. Sakiyama, K. Shibutani, D. Suzuki, Y. Nariyoshi, K. Minematsu, and et al., "Cryptographic technology guideline (lightweight cryptography)," *Cryptographic Technology Guideline*, p. 1–112, Mar 2017.

[67] A. Biryukov and L. Perrin, "State of the art in lightweight symmetric cryptography," *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 511, 2017.

[68] X. Zhu, "Energy optimization of the configurable service portfolio for iot systems," *Computer Communications*, vol. 154, pp. 491–500, 2020.

[69] L. Xian, M. Wang, and W. Jiang, "Resource optimization service chain composition and deployment in iot," in *2020 IEEE World Congress on Services (SERVICES)*, pp. 29–32, 2020.

[70] A. Kumar, A. A. Rahmani Hosseinabadi, M. Babazadeh Shareh, A. Zolfagharian, N. Chilamkurti, and S. Y. Bozorgi Rad, "Iot resource allocation and optimization based on heuristic algorithm," *Sensors*, vol. 20, p. 539, 01 2020.

[71] T.-M. Pham and T.-T.-L. Nguyen, "Optimization of resource management for nfv-enabled iot systems in edge cloud computing," *IEEE Access*, vol. 8, pp. 178217–178229, 2020.

[72] C. Avasalcai, C. Tsigkanos, and S. Dustdar, "Resource management for latency-sensitive iot applications with satisfiability," *IEEE Transactions on Services Computing*, vol. 15, no. 5, pp. 2982–2993, 2021.

[73] G. White, A. Palade, C. Cabrera, and S. Clarke, "Iotpredict: Collaborative qos prediction in iot," in *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10, 2018.

[74] S. Kim, Y. Suh, and H. Lee, "What iot devices and applications should be connected? predicting user behaviors of iot services with node2vec embedding," *Information Processing & Management*, vol. 59, no. 2, p. 102869, 2022.

[75] Y. Huang, J. Huang, B. Cheng, S. He, and J. Chen, "Time-aware service ranking prediction in the internet of things environment," *Sensors*, vol. 17, no. 5, 2017.

[76] I. Mashal, T.-Y. Chung, and O. Alsaryrah, "Toward service recommendation in internet of things," in *2015 Seventh International Conference on Ubiquitous and Future Networks*, pp. 328–331, 2015.

[77] S. Meng, Z. Gao, Q. Li, H. Wang, H.-N. Dai, and L. Qi, "Security-driven hybrid collaborative recommendation method for cloud-based iot services," *Computers and Security*, vol. 97, p. 101950, 2020.

[78] J. Pashaei Barbin, S. Yousefi, and B. Masoumi, "Efficient service recommendation using ensemble learning in the internet of things (iot)," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 1339–1350, 03 2020.

[79] N. A. Eltresy, O. M. Dardeer, A. Al-Habal, E. Elhariri, A. M. Abotaleb, D. N. Elsheakh, A. Khattab, S. A. Taie, H. Mostafa, H. A. Elsadek, *et al.*, "Smart home iot system by using rf energy harvesting," *Journal of Sensors*, vol. 2020, no. 1, p. 8828479, 2020.

[80] A. Singh, S. Redhu, B. Beferull-Lozano, and R. M. Hegde, "Network-aware rf-energy harvesting for designing energy efficient iot networks," *Internet of Things*, vol. 22, p. 100770, 2023.

[81] H. S. Vu, N. Nguyen, N. Ha-Van, C. Seo, and M. Thuy Le, "Multiband ambient rf energy harvesting for autonomous iot devices," *IEEE Microwave and Wireless Components Letters*, vol. 30, no. 12, pp. 1189–1192, 2020.

[82] P. S. Lakshmi, M. Jibukumar, and V. Neenu, "Network lifetime enhancement of multi-hop wireless sensor network by rf energy harvesting," in *2018 International Conference on Information Networking (ICOIN)*, pp. 738–743, IEEE, 2018.

[83] H. Zeb, M. Gohar, M. Ali, A. u. Rahman, W. Ahmad, A. Ghani, J.-G. Choi, and S.-J. Koh, "Zero energy iot devices in smart cities using rf energy harvesting," *Electronics*, vol. 12, no. 1, p. 148, 2022.

[84] L. Ye, Z. Wang, Y. Liu, P. Chen, H. Li, H. Zhang, M. Wu, W. He, L. Shen, Y. Zhang, Z. Tan, Y. Wang, and R. Huang, "The challenges and emerging technologies for low-power artificial intelligence iot systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 12, pp. 4821–4834, 2021.

[85] S. Jannu, S. Dara, C. Thuppari, A. Vidyarthi, and D. Gupta, "An advanced energy management and harvesting system for network lifetime for industrial iot in smart cities," *IEEE Internet of Things Journal*, vol. 10, no. 21, pp. 18663–18671, 2023.

[86] B. Reddy, "Powering the internet of things: Advances in energy harvesting technologies," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 11, pp. 1867–1876, 02 2025.

[87] Z. Behdad, M. Mahdavi, and N. Razmi, "A new relay policy in rf energy harvesting for iot networks—a cooperative network approach," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2715–2728, 2018.

[88] S. Galmés, "Statistical characterization of wireless power transfer via unmodulated emission," *Sensors*, vol. 22, no. 20, p. 7828, 2022.

[89] D. Tarchi, A. Bozorgchenani, and M. D. Gebremeskel, "Zero-energy computation offloading with simultaneous wireless information and power transfer for two-hop 6g fog networks," *Energies*, vol. 15, no. 5, p. 1632, 2022.

[90] J. Eidaks, R. Kusnins, R. Babajans, D. Cirjulina, J. Semenjako, and A. Litvinenko, "Efficient multi-hop wireless power transfer for the indoor environment," *Sensors*, vol. 23, no. 17, p. 7367, 2023.

[91] P. E. G. Silva, N. Marchetti, P. H. Nardelli, and R. A. de Souza, "Enabling semantic-functional communications for multiuser event transmissions via wireless power transfer," *Sensors*, vol. 23, no. 5, p. 2707, 2023.

[92] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall, "Wi-fi backscatter: internet connectivity for rf-powered devices," *SIGCOMM Comput. Commun. Rev.*, vol. 44, p. 607–618, Aug. 2014.

[93] V. Talla, B. Kellogg, B. Ransford, S. Naderiparizi, S. Gollakota, and J. R. Smith, "Powering the next billion devices with wi-fi," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, pp. 1–13, 2015.

[94] M. A. Halimi, T. Khan, Nasimuddin, A. A. Kishk, and Y. M. Antar, "Rectifier circuits for rf energy harvesting and wireless power transfer applications: A comprehensive review based on operating conditions," *IEEE Microwave Magazine*, vol. 24, no. 1, pp. 46–61, 2023.

[95] H. Elahi, K. Munir, M. Eugeni, S. Atek, and P. Gaudenzi, "Energy harvesting towards self-powered iot devices," *Energies*, vol. 13, no. 21, p. 5528, 2020.

[96] Z. J. Chew and M. Zhu, "Combined power extraction with adaptive power management module for increased piezoelectric energy harvesting to power wireless sensor nodes," in *2016 IEEE SENSORS*, pp. 1–3, 2016.

[97] M. Caselli, M. Ronchi, and A. Boni, "Power management circuits for low-power rf energy harvesters," *Journal of Low Power Electronics and Applications*, vol. 10, no. 3, p. 29, 2020.

[98] P. Wang, "Iot service recommendation scheme based on matter diffusion," *IEEE Access*, vol. 8, pp. 51500–51509, 2020.

[99] M. Nikpour, P. B. Yousefi, H. Jafarzadeh, K. Danesh, R. Shomali, S. Asadi, A. G. Lonbar, and M. Ahmadi, "Intelligent energy management with iot framework in smart cities using intelligent analysis: An application of machine learning methods

for complex networks and systems," *Journal of Network and Computer Applications*, vol. 235, p. 104089, 2025.

[100] L. Mottola, A. Hameed, and T. Voigt, "Energy attacks in the battery-less internet of things: Directions for the future," in *Proceedings of the 17th European Workshop on Systems Security (EuroSec '24)*, pp. 29–36, 2024.

[101] S.-Y. Chang, S. L. S. Kumar, Y.-C. Hu, and Y. Park, "Power-positive networking: Wireless-charging-based networking to protect energy against battery dos attacks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 15, no. 3, pp. 1–25, 2019.

[102] I. C. K. Sihomnou, A. Benslimane, A. H. Anwar, G. Deugoue, and C. Kamhoua, "Defending internet of things against energy depletion attack using bayesian game," *IEEE Internet of Things Journal*, 2025.

[103] M. Mahamat, G. Jaber, and A. Bouabdallah, "Achieving efficient energy-aware security in iot networks: a survey of recent solutions and research challenges," *Wireless Networks*, vol. 29, pp. 787–808, 2023.

[104] Q. Pan, J. Wu, A. K. Bashir, J. Li, W. Yang, and Y. D. Al-Otaibi, "Joint protection of energy security and information privacy for energy harvesting: An incentive federated learning approach," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3473–3483, 2022.

[105] B. Mao, Y. Kawamoto, J. Liu, and N. Kato, "Harvesting and threat aware security configuration strategy for ieee 802.15.4 based iot networks," *IEEE Communications Letters*, vol. 23, no. 11, pp. 2130–2134, 2019.

[106] C. Copello and N. Weng, "Adaptive scheduling to enhance data security and energy efficiency on energy harvesting platform," *Microprocessors and Microsystems*, vol. 60, pp. 24–37, 2018.

[107] U. Farooq, N. Ul Hasan, I. Baig, and N. Shehzad, "Efficient adaptive framework for securing the internet of things devices," *EURASIP Journal on Wireless Communications and Networking*, vol. 210, pp. 1687–1499, 2019.

[108] X. Fang, M. Yang, and W. Wu, "Security cost aware data communication in low-power iot sensors with energy harvesting," *Sensors*, vol. 18, no. 12, p. 4400, 2018.

[109] P. Schaumont, B. Yuce, K. Pabbuleti, and D. Mane, "Secure authentication with energy-harvesting: A multi-dimensional balancing act," *Sustainable Computing: Informatics and Systems*, vol. 12, pp. 83–95, 2016.

[110] S. Sciancalepore, P. Tedeschi, U. Riasat, and R. D. Pietro, "Mitigating energy depletion attacks in iot via random time-slotted channel access," in *2021 IEEE Conference on Communications and Network Security (CNS)*, pp. 10–18, 2021.

[111] I. Analytics, "State of iot 2022: Number of connected iot devices growing 18% to 14.4 billion globally." https://iot-analytics.com/number-connected-iot-devices/.

[112] K. Mehdi, M. Lounis, A. Bounceur, and T. Kechadi, "Cupcarbon: A multi-agent and discrete event wireless sensor network design and simulation tool," *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*, 2014.

[113] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007* (P. Paillier and I. Verbauwhede, eds.), (Berlin, Heidelberg), pp. 450–466, Springer Berlin Heidelberg, 2007.

[114] T. K. Goyal and V. Sahula, "Lightweight security algorithm for low power iot devices," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1725–1729, 2016.

[115] D. Alao, E. Hamrick, S. Bhat, A. Sen, and K. Fletcher, "An iot simulator tool for provisioning secure and dynamic user-centric services," in *International Conference on Internet of Things*, pp. 77–91, Springer, 2022.

[116] A. Dâmaso, D. Freitas, N. Rosa, B. Silva, and P. Maciel, "Evaluating the power consumption of wireless sensor network applications using models," *Sensors*, vol. 13, no. 3, pp. 3473–3500, 2013.

[117] O. Dawkins, "Intel collaborative research institute (icri) for urban iot envirosensor data," 2020.

[118] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, pp. 104–122, 2014.

[119] C.-W. Hung and W.-T. Hsu, "Power consumption and calculation requirement analysis of aes for wsn iot," *Sensors*, vol. 18, no. 6, 2018.

[120] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, and Y. Agarwal, "Sentinel: occupancy based hvac actuation using existing wifi infrastructure within commercial buildings," SenSys '13, (New York, NY, USA), Association for Computing Machinery, 2013.

[121] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation," *PeerJ Computer Science*, vol. 7, p. e623, 2021.

[122] S. S. Ahmadpour, M. Zaker, N. J. Navimipour, N. K. Misra, M. Zohaib, S. Kassa, A. Heidari, A. H. Navin, M. Hosseinzadeh, and M. Hakimi, "A nano-design of a quantum-based arithmetic and logic unit for enhancing the efficiency of the future iot applications," *AIP Advances*, vol. 15, no. 3, 2025.

[123] S.-S. Ahmadpour, M. Noorallahzadeh, H. M. R. Al-Khafaji, M. Darbandi, N. J. Navimipour, B. Javadi, N. U. Ain, M. Hosseinzadeh, and S. Yalcin, "A new energy-efficient design for quantum-based multiplier for nano-scale devices in internet of things," *Computers and Electrical Engineering*, vol. 117, p. 109263, 2024.

[124] T. Sanislav, G. D. Mois, S. Zeadally, and S. C. Folea, "Energy harvesting techniques for internet of things (iot)," *IEEE Access*, vol. 9, pp. 39530–39549, 2021.

[125] A. Shahini, A. Kiani, and N. Ansari, "Energy efficient resource allocation in eh-enabled cr networks for iot," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3186–3193, 2019.

[126] H. H. R. Sherazi, D. Zorbas, and B. O'flynn, "A comprehensive survey on rf energy harvesting: Applications and performance determinants," *Sensors*, vol. 22, no. 8, p. 2990, 2022.

[127] M. Pizzotti, L. Perilli, M. Del Prete, D. Fabbri, R. Canegallo, M. Dini, D. Masotti, A. Costanzo, E. Franchi Scarselli, and A. Romani, "A long-distance rf-powered sensor node with adaptive power management for iot applications," *Sensors*, vol. 17, no. 8, p. 1732, 2017.

[128] P. Chaudhari, A. K. Thakur, R. Kumar, N. Banerjee, and A. Kumar, "Comparison of nsga-iii with nsga-ii for multi objective optimization of adiabatic styrene reactor," *Materials Today: Proceedings*, vol. 57, pp. 1509–1514, 2022. International Chemical Engineering Conference 2021 (100 Glorious Years of Chemical Engineering & Technology).

[129] J. Shen, S. Tang, M. K. A. Mohd Ariffin, A. As'arry, and X. Wang, "Nsga-iii algorithm for optimizing robot collaborative task allocation in the internet of things environment," *Journal of Computational Science*, vol. 81, p. 102373, 2024.

[130] A. Bel, T. Adame, and B. Bellalta, "An energy consumption model for ieee 802.11ah wlans," *Ad Hoc Netw.*, vol. 72, p. 14–26, Apr. 2018.

[131] P. Andres-Maldonado, P. Ameigeiras, J. Prados-Garzon, J. J. Ramos-Munoz, and J. M. Lopez-Soler, "Optimized lte data transmission procedures for iot: Device side energy consumption analysis," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 540–545, 2017.

[132] M. Cansiz, D. Altinel, and G. K. Kurt, "Efficiency in rf energy harvesting systems: A comprehensive review," *Energy*, vol. 174, pp. 292–309, 2019.

[133] T. Bouguera, J.-F. Diouris, J.-J. Chaillout, R. Jaouadi, and G. Andrieux, "Energy consumption model for sensor nodes based on lora and lorawan," *Sensors*, vol. 18, no. 7, p. 2104, 2018.

[134] A. A. Thabit, M. S. Mahmoud, A. Alkhayyat, and Q. H. Abbasi, "Energy harvesting internet of things health-based paradigm: Towards outage probability reduction through inter–wireless body area network cooperation," *International Journal of Distributed Sensor Networks*, vol. 15, no. 10, p. 1550147719879870, 2019.

[135] V. Prathibha and T. Aruna, "Enhancing the network lifetime of cooperative wireless sensor networks using energy harvesting technique," in *2014 IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1–4, 2014.

[136] C. Pan, M. Xie, and J. Hu, "Enzyme: An energy-efficient transient computing paradigm for ultralow self-powered iot edge devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2440–2450, 2018.

[137] S. Khan, A. N. Alvi, M. A. Javed, Y. D. Al-Otaibi, and A. K. Bashir, "An efficient medium access control protocol for rf energy harvesting based iot devices," *Computer Communications*, vol. 171, pp. 28–38, 2021.

[138] L. Farhan, S. T. Shukur, A. E. Alissa, M. Alrweg, U. Raza, and R. Kharel, "A survey on the challenges and opportunities of the internet of things (iot)," in *2017 Eleventh International Conference on Sensing Technology (ICST)*, pp. 1–5, 2017.

[139] J. Andreu-Perez, D. R. Leff, H. M. D. Ip, and G.-Z. Yang, "From wearable sensors to smart implants-–toward pervasive and personalized healthcare," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 12, pp. 2750–2762, 2015.

[140] A. Kanawaday and A. Sane, "Machine learning for predictive maintenance of industrial machines using iot sensor data," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 87–90, 2017.

[141] S. N. Jyothi and K. V. Vardhan, "Design and implementation of real time security surveillance system using iot," in *2016 International Conference on Communication and Electronics Systems (ICCES)*, pp. 1–5, 2016.

[142] N. Sharma and R. D. Garg, "Real-time iot-based connected vehicle infrastructure for intelligent transportation safety," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 8, pp. 8339–8347, 2023.

[143] R. J. Hassan, S. Zeebaree, S. Y. Ameen, S. F. Kak, M. Sadeeq, Z. S. Ageed, A. AL-Zebari, and A. A. Salih, "State of art survey for iot effects on smart city technology: challenges, opportunities, and solutions," *Asian Journal of Research in Computer Science*, vol. 22, pp. 32–48, 2021.

[144] M. N. Hassan, M. R. Islam, F. Faisal, F. H. Semantha, A. H. Siddique, and M. Hasan, "An iot based environment monitoring system," in *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 1119–1124, IEEE, 2020.

[145] D. Ma, G. Lan, M. Hassan, W. Hu, and S. K. Das, "Sensing, computing, and communications for energy harvesting iots: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1222–1250, 2020.

[146] T. Sanislav, G. D. Mois, S. Zeadally, and S. C. Folea, "Energy harvesting techniques for internet of things (iot)," *IEEE access*, vol. 9, pp. 39530–39549, 2021.

[147] S. Ma, M. Jiang, P. Tao, C. Song, J. Wu, J. Wang, T. Deng, and W. Shang, "Temperature effect and thermal impact in lithium-ion batteries: A review," *Progress in Natural Science: Materials International*, vol. 28, no. 6, pp. 653–666, 2018.

[148] J. Hester and J. Sorber, "The future of sensing is batteryless, intermittent, and awesome," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[149] A. Harb, "Energy harvesting: State-of-the-art," *Renewable Energy*, vol. 36, no. 10, pp. 2641–2654, 2011. Renewable Energy: Generation Application.

[150] A. V. Taddeo, M. Mura, and A. Ferrante, "Qos and security in energy-harvesting wireless sensor networks," in *2010 International conference on security and cryptography (SECRYPT)*, pp. 1–10, IEEE, 2010.

[151] A. V. Taddeo, M. Mura, and A. Ferrante, "Qos and security in energy harvesting wireless sensor networks," in *International Conference on Security and Cryptography (SECRYPT)*, pp. 1–10, 2010.

[152] J. Banks, J. Carson, B. Nelson, and D. Nicol, *Discrete-Event System Simulation*. Prentice-Hall, 4 ed., 2005.

[153] D. Zinoviev, "Discrete event simulation: It's easy with simpy!," *arXiv preprint arXiv:2405.01562*, 2024.

[154] Z. A. Al-Azzawi, A. J. Al-Mousawi, and S. M. Ali, "A survey of low rate ddos detection techniques based on machine learning in software-defined networks," *Symmetry*, vol. 14, no. 8, p. 1563, 2022.

[155] A. P. Plageras, K. E. Psannis, C. Stergiou, H. Wang, and B. B. Gupta, "Efficient IoT-based sensor BIG data collection–processing and analysis in smart buildings," *Future Generation Computer Systems*, vol. 82, pp. 349–357, 2018.

[156] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[157] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the RPL-Based internet of things," *International Journal of Distributed Sensor Networks*, vol. 9, no. 8, p. 794326, 2013.

[158] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 443–461, 2011.