IoT-Enhanced Vehicular Networks: Simulation Frameworks for Energy Efficiency and Cyber-Security in Smart Cities

Reham Mutlaq Almutairi

Supervisors:

Prof. Graham Morgan

Dr. Giacomo Bergami

Submitted for the degree of Doctor of Philosophy

in the School of Computing Science, Newcastle University

April 2025

Abstract

The Internet of Things (IoT) has rapidly evolved over the past two decades, transforming the way we interact with the environment through a network of interconnected devices. The purpose of this thesis is to explore the integration of IoT with Vehicular Ad-Hoc Networks (VANETs) in order to enhance intelligent transportation systems (ITS) and smart city infrastructure through the use of IoT. VANETs, characterized by high mobility and dynamic topology, play a crucial role in enhancing traffic safety, efficiency, and vehicular services. They improve traffic safety by enabling real-time communication between vehicles and roadside infrastructure, allowing the sharing of critical information such as accident warnings and road conditions to prevent collisions and enhance emergency response times. VANETs boost traffic efficiency through intelligent traffic management, optimizing signal timings and route planning based on real-time data to reduce congestion and travel times. Additionally, they provide enhanced vehicular services such as infotainment, navigation assistance, and maintenance alerts, thereby improving the overall driving experience and vehicle performance monitoring.

This research addresses the significant challenges of simulating VANET environments, particularly the high mobility of vehicles and the need for realistic traffic scenarios. Existing VANET simulators, while advanced, often lack support for new technologies and comprehensive security systems, highlighting the necessity for more comprehensive simulation frameworks. The primary aim of this PhD thesis is to integrate IoT and traffic simulations to accurately evaluate vehicular energy efficiency and overall network performance. Therefore, this thesis presents multilateral research towards optimization, modeling, and simulation of VANET and IoT environments. Several tools and algorithms have been proposed, implemented, and evaluated, considering various environments and applications. The main contributions of this thesis are as follows:

- Conducting a review of current IoT simulators highlights their strengths and limitations, particularly their inability to address energy depletion security concerns. The survey identified a lack of support for renewable energy sources or VANET integration, which are essential for modern IoT applications. The absence of a ver-

satile, generic IoT simulator is noted, as existing tools often specialize in specific applications and lack flexibility.

- Conducting an in-depth performance evaluation of emerging VANET technologies, this survey addresses the urgent need for updated reviews considering electric vehicles, self-driving cars, SDN, edge computing, and 5G. The survey identifies critical gaps, including the lack of support for renewable energy, dynamic battery recharging, and encryption impact.

- Conducting a feasibility study on coupling IoT simulators with traffic simulators to enhance VANET simulations, this toward study introduces the novel *SUMOtoOsmosis* framework. Investigating the integration of IoTSim-Osmosis for IoT simulations with SUMO for traffic simulations, *SUMOtoOsmosis* marks a first in the literature. The proposed system, tested with the Hamburg dataset, focuses on communication time, this framework enables the simulation of traffic environments based on IoT infrastructure.

- Proposing and modeling a new holistic framework that simulates real-world traffic scenarios for electric vehicles, *SimulatorBridger*. Its flexible architecture allows for integration with any traffic simulator. Preliminary results validate its accuracy in simulating vehicular battery consumption and network performance, highlighting the need for efficient communication policies. This platform supports policymakers in optimizing VANET performance and developing energy-efficient transportation networks.

- Introducing *SimulatorBridgerDfT*, a novel simulator platform extending *SimulatorBridger* to support different formats of real traffic data, enhances the flexibility and applicability of urban traffic simulations by integrating IoTSim-OsmosisRES with DfT traffic data. Evaluating the impact of SUMO car traces versus static DfT data on communication delays in IoT simulations provides valuable insights for designing efficient and effective traffic simulation tools, aiding researchers and practitioners in traffic management and urban planning.

- Introducing *IoTSimSecure*, a novel simulation framework designed to detect the security attacks, particularly battery draining attacks. IoTSimSecure supports a

range of detection algorithms, including threshold-based detection and Exponential Weighted Moving Average (EWMA) techniques. This flexibility allows for comprehensive analysis and testing of various security strategies, thus enhancing the simulator's ability to develop effective countermeasures against battery-draining attacks.

As a result of addressing the key challenges in IoT and VANET simulation, the results of this thesis will contribute to the development of flexible, efficient, and secure intelligent transportation systems and smart city infrastructures.

Declaration

I declare that this thesis is my own work unless otherwise stated. No part of this thesis has previously been submitted for a degree or any other qualification at Newcastle University.

Reham Almutairi

September 2024

Publications

Published

1. [1] Platform for energy efficiency monitoring electrical vehicle in real world traffic simulation.
2. [2] Advancements and Challenges in IoT Simulators: A Comprehensive Review.
3. [3] IoTSimSecure: Towards an IoT Simulator Supporting Cyber-Threat Detection Algorithms.
4. [4] Systematic Literature Review of VANET Simulators: Comparative Analysis, Technological Advancements, and Research Challenges
5. [5] SimulatorBridger: System for Monitoring Energy Efficiency of Electric Vehicles in Real-World Traffic Simulations
6. [6] Approximating Real-Time IoT Interaction through Connection Counting: A QoS Perspective
7. [7] SimulatorbridgerDfT: A Real-Data Simulator for IoT-Osmotic Interactions
8. [8] SimulatorOrchestrator: A 6G-Ready Simulator for the Cell-Free/Osmotic Infrastructure

Acknowledgements

Above all, I am deeply thankful to Almighty God for granting me the wisdom, perseverance, and well-being needed to navigate this journey. His guidance has been my constant source of strength, helping me overcome every challenge and bringing this work to completion.

I would like to extend my deepest thankfulness, warmth, and appreciation to the following individuals who have supported and contributed to the success of this research:

I would like to begin by expressing my sincere gratitude to my primary supervisor, **Prof. Graham Morgan**, for his guidance and support throughout this project. His input and feedback were helpful in shaping the direction of my research, and I am grateful for his role in overseeing my work.

I would also like to extend my heartfelt thanks to **Dr. Giacomo Bergami**. Your immense help, thoughtful advice, and encouragement at crucial moments in my research journey have been invaluable. You were always available, even for matters outside your research area, and consistently offered guidance and support. I deeply appreciate your willingness to explain concepts and clarify anything I found unclear. Working under your supervision has been an incredibly enriching experience, and I have learned a great deal from your expertise and dedication. Your unwavering support has been instrumental in helping me reach this milestone, and for that, I am truly grateful.

A special thanks goes to my beloved parents, **Mutlaq** and **Mashael**, who have always been, and will always be, the foundation of my success. Your unfailing support and constant encouragement throughout my years of study have given me the strength to pursue my dreams and reach for the stars. This accomplishment would not have been possible without you both. I am deeply grateful for your love and the strength your phone calls provided, especially during the hardest moments.

To my first daughter, **Sumou**, the one who first made me a mother. My heart has been with you every moment, even from afar, and words cannot capture how much I have missed your presence. The distance has been a difficult sacrifice, but your love has been

my strength. I dedicate this thesis to you, with the hope that one day you will understand that every step of this journey was for you, and that my love for you knows no boundaries, not even distance. I also dedicate it to my only sister, **Atheer**, and my brothers, whose love and support have meant the world to me.

I would also like to express my deepest gratitude to my family here in Newcastle, my husband and children, for their unwavering support and for standing by me throughout this journey. To my dear husband, **Abdulrahman**, your strength and sacrifice have been incredible. I know it was not easy for you to face challenges with your job in order to be here with me, and your commitment to our family and my dreams has been a constant source of inspiration. To my boys, **Sattam** and **Fisal**, thank you for your patience and understanding during the my busy times. This achievement is not mine alone, but ours as a family, to celebrate together.

Finally, to my youngest daughter, **Kinda**, who was born during this PhD journey. From the moment you came into this world, you have been a source of pure joy and love. While I was pursuing my PhD, there were many moments when I couldn't spend as much time with you as I wanted to. Even though I was there, much of our time was spent apart as I worked and you were at nursery. I wish I could have been with you more during those early years, but every smile, every moment we shared, filled my heart and gave me the strength to keep going. As you grow and read this one day, I want you to know how much you mean to me. You were my light during this journey, and this achievement is as much yours as it is mine. I am so proud to be your mother, and I look forward to all the moments we will share together.

Contents

List of Figures

List of Tables

List of Acronyms

VANET Vehicular Ad-hoc Network

IoT Internet of Things

RSU Road-Side Unit

MEL MicroElements

SDN Software-Defined Networking

SD-WAN Software-Defined Wide Area Network

KPI Key Performance Indicator

SUMO Simulation of Urban Mobility

DfT Department for Transport

LTE Long-Term Evolution

WiFi Wireless Fidelity

List of Mathematical Notations

Symbol	**Meaning**
τ_b, τ_e	Begin and end simulation times
δ	Temporal granularity for sampling rate
$\vec{\varpi}_\nu$	Geographical location of vehicle ν
ρ_r	Communication radius of RSU r
$G = (V, E)$	Graph with vertices V and edges E
$\|\vec{\varpi}_r - \vec{\varpi}_{r'}\|$	Distance between RSUs r and r'
KPIs	Key performance indicators like energy efficiency and latency
SDN Policy	Shortest path with maximum bandwidth or similar routing policies

1

Introduction

1.1 Introduction

In the past two decades, the Internet of Things has rapidly evolved, promising to transform the way we live, work, and interact with our environment. As a whole, the Internet of Things refers to a network of physical objects that are connected and exchange data via the internet with other devices and systems through the use of sensors, software, and other technologies. Items such as household items and industrial tools can be included in this category. The primary goal of IoT is to create a smarter, more efficient world by seamlessly integrating the physical and digital worlds. The idea of interconnected devices is not new. However, the term "Internet of Things" was coined in 1999 by Kevin Ashton, co-founder of the Auto-ID Center at MIT [17]. Ashton's vision was rooted in the belief that when objects can sense the environment and communicate, they could bring about unprecedented efficiency and automation. This vision was propelled by the rapid advancements in wireless technologies, decreasing computing costs, and the proliferation of mobile devices. Today, with billions of devices connected globally, his vision is increasingly becoming a reality, reshaping industries and daily life.

This interconnected paradigm can be seen in the development of smart cities, which integrate IoT technologies in order to optimize infrastructure and services. Within these smart cities, Vehicular Ad-Hoc Networks (VANETs) represent a significant subset of Mobile Ad-Hoc Networks (MANETs). MANET is built with an infrastructure-independent network of mobile devices such as PDAs, Laptops, or any other device. This kind of network is created for communication of devices where infrastructure deployment is either not feasible or impossible to achieve. On the other hand, VANET comprises vehicles that act as independent network nodes that can join or leave the network freely. VANET is an infrastructure-independent network for safe driving, infotainment, intelligent navigation, and emergency applications. The freedom of nodes to enter or leave the network in VANET calls for different routing protocols than MANET [18]. The significant differences between MANET and VANET are exhibited in Table 1.1. The table shows that Vehicles in VANETs usually acquire their position from GPS and Radar and depend on their lifetimes. Also, vehicles move at high speeds and random mobility, which requires Multi-hop routing. This can cause rapid changes in network topology and potentially more frequent disruptions in communication channels, leading to more scalability and high reliability.

On the other hand, nodes in MANETs earn their position from Ultrasonic and their lifetime depends on the power source. Also, the nodes may have lower speeds than vehicles in VANET, and their movement is often regular and frequent, so Multi-hop routing is not required here. The relatively lower speeds and diverse mobility patterns might offer more stable connections, a low scalable network, and a lower reliability than VANET. Nodes in MANET are addressed based on certain attributes or characteristics rather than a fixed address, as in VANET where vehicles are addressed based on their geographical location, typically determined through GPS or other localization mechanisms.

VANET has been widely recognized for their pivotal role in intelligent transportation systems (ITS). Characterized by high mobility and rapidly changing network topology, VANETs indeed offer promising prospects in augmenting traffic safety and efficiency and enhancing vehicular services. The inherent high mobility of vehicles necessitates a network that can swiftly adapt to the dynamic changes in topology, ensuring continuous and reliable communication among vehicles and between vehicles and infrastructure. This enables vehicles to share vital real-time information, such as speed, direction, and potential hazards, thereby significantly enhancing traffic safety by preventing collisions and facilitating smoother traffic flow. Moreover, the ability of VANETs to disseminate real-time traffic and road condition information allows for more efficient route planning and congestion management, thereby improving overall traffic efficiency. Smart traffic management, enabled by VANETs, utilizes real-time vehicular data to optimize traffic light timings and suggest alternative routes to drivers, mitigating traffic congestion and reducing travel times. Furthermore, VANETs enhance vehicular services by providing drivers with timely and relevant location-based services, such as navigation assistance, nearby service station information, and parking solutions, thereby enriching the overall driving experience [19]. One of the critical challenges in integrating IoT with vehicular networks is managing battery consumption effectively. The increasing number of IoT devices in vehicular networks necessitates efficient energy management strategies to ensure long-term functionality and system reliability. This thesis addresses battery consumption as a central aspect, exploring its patterns and impact on communication efficiency, which is vital for the development of sustainable smart city infrastructures.

VANET architecture predominantly comprises three essential components: vehicles, roadside units (RSUs), and a central server known as the rusted authority (TA). Vehicles

Table 1.1: MANET VS VANET

Parameters	MANET	VANET
Cost	Less	costly Expensive
Bandwidth	100 Kps	1000 Kps
Mobility	Low	High
Topology Change	Slow	Frequent and Fast
Node density	Spares	Dens and Frequent
Node's Lifetime	Depends on power source	Depends on Vehicle's lifetime
Reliability	Medium	High
Node Movement	Random	Regular and Frequent
Addressing Scheme	Attribute Based	Location Based
Position acquisition	Ultrasonic	GPS, Radar
Multi-hop routing	Available	Weakly available
Power Constraint	Medium	Low
Scalability	Medium	Low

equipped with on-board units (OBUs) function as the primary nodes, enabling direct or indirect communication. Due to their mobility, vehicles create a dynamic network topology, contributing to the ad-hoc nature of VANETs. Roadside Units (RSUs), strategically positioned infrastructure, facilitate the indirect communication mode in VANETs. They work as relay nodes between the vehicle nodes and the central server, providing essential services such as internet connectivity, information dissemination, and management of network parameters. The trusted authority (TA) is a central server that administers crucial functions such as network initialization, vehicle registration, key distribution, and overall network security management [20].

When it comes to modelling a system, the difficulties of real-time implementation raise the need to use simulators, which work in the virtual mode. VANET simulation is instrumental in developing and validating network models, protocols, and applications tailored for vehicular networks. Given the inherent complexity and high-cost implications of real-world deployments, simulators provide a cost-effective and scalable alternative for VANET research, allowing researchers to create repetitive scenarios and run rapid tests that would typically take months to complete. They are well-educated about complex or unfair road problems such as driving habits, vehicle technology and drunk driving. VANET simulators have distinct features that are absent in other types of MANETs. VANET nodes must include different sets of sensors and other computational resources;

hence, it can be costly.

Simulators for VANETs need to consider two main factors: mobility modelling and network communication. Mobility modeling replicates the movement pattern of vehicles based on traffic rules and road structures. It's essential for creating realistic scenarios where vehicles' speed, direction, and routes change over time. On the other hand, network communication models focus on the networking aspect of the simulator, replicating the underlying communication protocols to evaluate network performance under various scenarios[21]. This dual focus on mobility modeling and network communication underscores the comprehensive design of VANET simulators. Specifically, a VANET simulator consists of two components: Traffic Simulators and network simulators. Traffic simulators are software environments that generate vehicle movement in trace files. On the other hand, network simulators are used to build communication topologies, evaluate network protocols, and exchange routing information between the nodes after importing the traces of mobility models generated by traffic simulators. Several simulators have been designed explicitly for VANET research, each with unique features and capabilities, such as NS-3, Veins, and VANETMobiSim. These tools aid researchers in understanding and predicting the behaviour of VANETs in a controlled and reproducible environment, thus playing a critical role in the progression of VANET-based technologies. These insights are particularly valuable given the practical challenges faced in real-world VANET deployments. In real-world scenarios, VANETs are typically employed over shorter distances, such as when vehicles require interaction in less populated areas [22]. However, vehicle movements make the VANET link's life limited, leading to frequent changes in the network architecture. This imposes severe latency and capacity constraints on these technologies [23]. Additionally, road topology, which represents road links, controls the motions of VANET nodes. Accurate positioning of these nodes is crucial, as any error can be fatal for humans. Fortunately, with long-life batteries providing a constant power source, VANETs have no power limits [24]. These extraordinary qualities make VANETs suitable for a wide range of applications in various vehicles.

1.2 Research Motivations and Challenges

Despite the potential benefits of VANETs, their simulation remains challenging due to the high mobility of vehicles and the need for realistic traffic scenarios. Current tools predominantly couple traffic simulators with network simulators to model VANET environments. Recent VANET simulators can perform multiple functions but have major limitations in supporting new technologies and their security systems. These concerns open many opportunities for researchers to promote VANET simulators. This section discusses issues in detail, highlighting the need for more research to design advanced VANET simulation tools. The main motivation behind this research is to enhance VANET (Vehicular Ad-hoc Network) simulation by integrating advanced IoT and traffic simulators, addressing various limitations and expanding their capabilities. This involves creating a comprehensive and flexible simulation framework that supports moving IoT devices, handles diverse traffic data formats, and ensures the security and functionality of IoT devices. The following sections detail the specific motivations for each aspect of this research, focusing on improving simulation accuracy, versatility, and security in dynamic and heterogeneous environments.

Integration IoT with Traffic Simulators: Modeling real use case scenarios where each IoT device is embedded in a VANET vehicle allows for a complete simulation of edge computing in an efficient environment, providing a single data storage and analysis point where the network edge is closer to the IoT devices. This synergy between IoT and traffic simulators enables the orchestration of cloud, edge, and/or SDN-based networks and the abstraction of complex IoT applications generated by vehicular traffic, ensuring each component of the IoT applications is covered holistically. Through the use of osmotic computing, researchers can evaluate the performance of these applications from end to end, enhancing the accuracy and reliability of simulations and providing a robust framework for developing and testing complex IoT applications that operate in dynamic and varied conditions. All existing VANET simulators combine a traffic simulator and a network simulator; however, in the context of VANET simulation, current research does not allow simulation based on IoT infrastructure. Using IoT simulators in conjunction with traffic simulation allows more accurate evaluation of the energy efficiency of vehicles.

An examination of different battery consumption patterns may assist in determining the most effective strategy for minimizing the amount of energy consumed by network communications. To investigate the idea of integrating IoT simulators with traffic simulators, we need to conduct a feasibility study. Also, how this could be accomplished and what is required to achieve it will be examined.

Overcoming Mobility Limitations of Existing IoT Simulators: Various frameworks have been introduced for modelling and simulating IoT environments and evaluating resource management. However, supporting mobility, an essential feature of IoT, is not comprehensively represented in many IoT simulators. Most tools offer limited mobility modelling capabilities, underscoring the urgent need for advanced simulators. These advanced simulators should be capable of accurately depicting the dynamic nature of IoT topologies, especially for complex simulation environments such as smart vehicles, wearables, and individuals on the move. Given the rising prevalence of mobile IoT devices, future simulators are anticipated to incorporate advanced mobility models. These models might leverage real-time data and predictive analytics to emulate complex mobility patterns and scenarios. The development of advanced simulators that can more accurately and comprehensively represent IoT deployments' dynamic and mobile nature is urgently required, perhaps integrating real-world mobility data or leveraging machine learning to predict mobility patterns. The current implementation of the state-of-the-art IoT simulator, *IoTSim-OsmosisRES* [25] assumes IoT devices to be in fixed locations. So, IoT device mobility is not supported. Motivated by these limitations, it is necessary to enhance IoTSim-OsmosisRES capabilities to be able to simulate moving IoT devices.

Optimizing Electric Vehicle Energy Efficiency in Real-World Traffic Simulations: Modern vehicles are equipped with navigation systems and video cameras that enable them to record and geo-tag video and images of their surroundings. For municipal authorities dedicated to road maintenance and safety, this capability is essential, as it allows early detection of road conditions before they become worse [26]. To make this possible, data must be collected from vehicles and transmitted to central data centers, where AI models assess road conditions [27]. Road-Side Units (RSUs) play a crucial role by gathering data through 5G antennas and streaming it to data centers using low-latency optical fiber com-

munication. Data collection will result in a massive amount of data streaming from the cars every second towards the data centres, which is likely to increase significantly during peak times when traffic congestion might be more likely to occur [28]. However, the massive data volume, especially during rush hours, can create communication bottlenecks and negatively impact performance. To manage this, traffic load balancing and real-time traffic flow redirection are essential, underscoring the importance of dynamic communication flow management through Osmotic Computation. This situation highlights the critical need to enhance the evaluation of energy consumption and communication efficiency in Vehicular Ad-hoc Networks (VANETs). Developing energy efficiency monitoring simulator platforms is crucial to address these challenges effectively, ensuring that VANETs operate optimally under varying traffic conditions.

Enhancing Flexibility with Multiple Traffic Dataset: Simulating traffic accurately while minimizing computational overhead and privacy concerns is one of the greatest challenges in the VANET simulation field. Current simulators often require detailed tracking of individual IoT devices to mimic real-time traffic behavior accurately, which involves high computational resources and raises privacy issues. So, there is a need to determine whether it is necessary to track each IoT device precisely, or if considering only the number of communicating devices at specific times is sufficient to provide accurate simulations. By addressing this challenge, we may be able to develop more efficient, privacy-conscious traffic simulation methods that will reduce computational demands while maintaining the integrity of simulation outcomes. Further, the differences between various real-world traffic datasets, such as those provided by the Department for Transport (DfT) and those generated by SUMO simulator suggest a need to investigate the effects of different types of data on simulation performance. A VANET simulator platform should be enhanced to allow the processing of a wide variety of traffic data formats in order to compare the detailed car traces obtained through SUMO with static data from the Department for Transport in terms of their impact on communication delays. An understanding of this concept is crucial for improving the design and application of simulation tools in urban traffic field.

Ensuring Security and Functionality in IoT Devices: The growth of IoT devices in various sectors, including smart cities, requires a focus on their security and functionality, particularly concerning battery life. Malicious activities, such as battery-draining attacks that flood devices with unnecessary tasks, can rapidly deplete battery power and interrupt essential services. For instance, in a smart city infrastructure, such attacks on IoT devices managing traffic lights and environmental monitoring can cause significant operational failures. To counter this, a system that detects unusual activity patterns indicative of battery-draining attacks is essential. This involves creating and updating security policies and detection models to ensure continuous protection. Despite the critical need, existing IoT simulators lack comprehensive support for addressing battery-draining attacks, highlighting the importance of developing tools to ensure the operational integrity and longevity of IoT devices across all smart scenarios. The current simulation capabilities in existing IoT simulators are limited, particularly in terms of simulating battery-draining attacks. There is a limitation on supporting direct communications between IoT devices, as most of them only facilitate interactions between IoT devices and cloud nodes via edge nodes. This architectural limitation is significant given that many current battery-draining attacks involve direct communication between IoT devices in IoT environments. By engaging in overly extensive, unnecessary communication requests or other energy-consuming activities, attackers target another IoT device to drain the battery. A fundamental understanding of the security of IoT networks can be obtained only by significantly revising or extending IoT simulators with new agents, such as attackers and cyber-security diagnostic tools. Security attacks, especially those involving direct communication between IoT devices, which are increasingly common in modern IoT networks, require the development of a simulator platform that is capable of effectively modeling complex attack scenarios or providing a thorough understanding of security attacks.

1.3 Research Aims

To address these challenges, this PhD thesis aims to integrate IoT and traffic simulations to assess vehicular energy efficiency in VANET environments. It aims to accurately evaluate battery consumption patterns and overall energy efficiency by simulating real-world

traffic and communication scenarios. Additionally, the thesis develops and introduces novel techniques for modeling various VANET environments, along with optimization solutions. The main objective of this PhD thesis is to provide an efficient simulation platform that evaluates energy efficiency, accuracy, computational efficiency, and security, enabling researchers to analyze their solutions and hypotheses more effectively, saving time and cost.

1.4 Research Questions

This PhD research aims to conduct an efficient simulation platform while considering the previously mentioned challenges. In particular, the following is the primary research question that guides this thesis:

Is it possible to couple IoT simulators with traffic simulators in order to improve simulation realism and utility?

In this study, the following research sub-questions will be addressed:

- What are the potential challenges for integrating IoT simulators with traffic simulators in VANET environments?
- What are the factors that impact the battery consumption of IoT devices?
- Can *SimulatorBridger* process different formats of real traffic data, such as those from the Department for Transport (DfT), without relying on SUMO for data generation?
- Does counting IoT device connections suffice to estimate real-time communication times rather than tracking specific locations?
- How can simulation platforms be enhanced to model complex security attack scenarios involving direct communication between IoT devices?

1.5 Research Contributions

A number of contributions are made to the field of VANET simulation and Internet of Things by this PhD thesis:

1. **Review of IoT simulators: tackling energy, security, and integration challenges:** Conducting a thorough review of current IoT simulators and identifying their strengths and limitations. It has been demonstrated in this review that existing tools are unable to effectively address security concerns related to the depletion of energy in IoT devices. This survey evaluates updated, open-source IoT simulators to ensure accessibility and customization for researchers, highlighting the need for advanced features like device mobility, comprehensive energy models, SDN support, and scalability. The review also observes that most simulators are unable to fully support renewable energy sources or integrate VANET simulations, which are crucial for modern IoT applications. Despite extensive research, there is a notable absence of comprehensive reviews covering all these aspects, especially the integration of IoT simulations with VANET simulations. Our findings indicate the lack of a generic IoT simulator capable of handling diverse scenarios, which is a critical shortcoming in the current landscape of IoT simulation tools. This deficiency means that existing simulators often specialize in particular aspects of IoT applications, such as smart cities, smart homes, smart health, and smart agriculture, but fail to offer the flexibility needed to simulate a wide range of scenarios comprehensively. Furthermore, our findings highlight the significant challenges associated with supporting mobility, SDN, and scalability, as well as emphasizing the need for more adaptable tools to meet the rapidly evolving demands of the Internet of Things.

2. **Survey of insights and gaps in current VANET simulators:** Conducting an in-depth performance evaluation of emerging technologies in VANET such as electric vehicles, self-driving cars, SDN, edge computing, and 5th-generation technology. This survey addresses the urgent need for an updated review of VANET simulators considering these advancements. IT provides valuable insights into the capabilities and limitations of current simulators, highlighting essential requirements for modeling urban mobility in smart cities. These requirements include scalability, node mobility, battery management, recharging, and renewable energy integration. Moreover, it identifies critical gaps such as the lack of support for renewable energy sources, dynamic battery recharging, and the impact of encryption algorithms. Our survey paves the way for developing more robust and comprehensive VANET simulators, which will advance the field and contribute to creating safer, more efficient,

and smarter transportation networks.

3. **SUMOtoOsmosis:** Conducting a feasibility study on coupling IoT simulators with traffic simulators to enhance the simulation of VANET environments. It introduces *SUMOtoOsmosis* a novel framework that integrates IoT simulations using IoTSim-Osmosis with traffic simulations using SUMO (in chapter 4), marking a first in the literature. *SUMOtoOsmosis* extends the capabilities of IoTSim-Osmosis to simulate moving IoT devices within VANET environments and provides a detailed system architecture. In the proposed architecture, the IoTSim-Osmosis simulator is invoked at each time frame to simulate the scenario. The proposed system have been tested using Humburg dataset, and provided a result only for the communication time. It is a preliminary experiment for simulating traffic environments based on the IoT infrastructure.

4. **SimulatorBridger:** Developing a novel simulator platform *SimulatorBridger* to address critical limitations of our previous platform *SUMOtoOsmosis*, which is that it fails to continuously simulate real-world traffic scenarios for electric vehicles, as well as not taking battery information into consideration. *SimulatorBridger* provides a simple extension for a mobility scenario to address the limitation of fixed-location IoT devices in IoTSim-OsmosisRES, allowing accurate simulation of mobile IoT devices and supporting energy consumption. This represents a transformative advance in the simulation of real-world traffic scenarios for electric vehicles. The proposed simulator *SimulatorBridger* incorporates Osmotic Computation to provide dynamic communication flow management, which is extremely important when dealing with the massive amounts of data and communication bottlenecks that occur during peak traffic periods. With SimulatorBridger's flexible architecture, you can integrate any traffic simulator using a plug-and-play approach, enhancing its adaptability to a variety of traffic environments. Based on preliminary results, the simulator validates its accuracy, demonstrating an aligned relationship between vehicular battery consumption trends and packet distribution patterns, along with potential bottlenecks in the network with an increase in the volume of packets. This validation highlights the necessity of efficient communication policies. By providing a digital twin for realistic urban mobility scenarios, SimulatorBridger allows policymakers to test

various network configurations and traffic distributions, minimizing network communication overload and optimizing VANET performance under a wide range of conditions. The platform offers a comprehensive and robust tool for developing more efficient and sustainable vehicular networks, which will be a significant step forward in the field of energy-efficient transportation.

5. **SimulatorBridgerDfT:** A new simulator platform , *SimulatorBridgerDfT*, is proposed as an extension of *SimulatorBridger* with support for multiple traffic data formats, including real-world CSV data from the Department for Transport (DfT), significantly enhancing its functionality, which is unprecedented in current VANET simulator literature. The development of SimulatorBridgerDfT, which integrates IoTSim-OsmosisRES with DfT data, makes the platform more versatile for various urban scenarios. The study evaluates and confirms SimulatorBridgerDfT's ability to process different traffic data formats independently of SUMO, advancing the flexibility and applicability of VANET simulation tools in traffic management and urban planning. The experimental analysis confirms that *SimulatorBridgerDfT* accurately simulate real-world traffic patterns; this simulator proves also the effectiveness of Osmotic architectures for dealing with considerable network traffic initiated by a realistic number of devices. The results highlight the significant impact of IoT device load on network performance, especially during peak traffic periods, without severely affecting communication times in non-rush hour fragments. Furthermore, this study examines whether the detailed movement patterns generated by SUMO provide a significant advantage over static data provided by the Department for Transport in terms of communication delays over the static data. Based on our findings, precise node locations and movements may not significantly impact communication patterns from the simulator's perspective, as the interaction with stationary Edge devices remains consistent regardless of vehicle movement. Based on this insight, the DfT datasets have the potential to be used for effective and scalable communication analysis without requiring granular movement information.

6. **IoTSimSecure:** Introducing *IoTSimSecure*, a novel and robust simulation framework designed specifically to detect security challenges especially battery draining attacks. This platform offers advanced features for detecting and preventing such

attacks, incorporating a variety of detection algorithms, including threshold-based detection and Exponential Weighted Moving Average (EWMA) techniques. A number of scenarios for use in various settings are presented, including smart buildings, industrial complexes, and healthcare systems, all of which illustrate practical applications as well as the potential consequences of battery draining attacks. By offering comprehensive analysis and practical solutions, this proposed platform not only highlights the urgent need for effective mitigation strategies but also enhances the resilience and security of IoT systems. The development of *IoTSimSecure* as a modular and extensible simulator establishes a foundation for future advancements in IoT security, ensuring the sustained operation and protection of IoT devices in our increasingly connected world. This contribution addresses a significant gap in existing IoT simulation tools, making it a valuable tool for researchers and practitioners concerned with securing IoT ecosystems against evolving cyber attacks.

1.6 Thesis Structure

The rest of the thesis is organized as follows:

- **Chapter 2** provides an overview of VANET simulation area as well as an in-depth evaluation of VANET simulators, focusing on emerging technologies like electric vehicles, self-driving cars, and 5G. It identifies critical gaps, such as the lack of support for renewable energy sources and dynamic battery recharging. This chapter is derived from [4].
- **Chapter 3** provides an overview of IoT simulators, highlighting their strengths and limitations, especially in regards to energy depletion. It identifies the need for support of renewable energy sources and VANET integration. This chapter is derived from [2].
- **Chapter 4** presents a novel framework integrating IoT simulations with traffic simulations. It is a preliminary experiment for simulating traffic environments based on the IoT infrastructure.
- **Chapter 5** introduces a framework that models and simulates the integration of

VANETs with IoT infrastructures and supporting battery consumption for electric vehicles. This chapter is derived from [1, 5]

- **Chapter 6** proposes a flexible simulation platform that supports various traffic dataset formats, enhancing flexibility and applicability to urban traffic scenarios [6, 7].
- **Chapter 7** introduces a robust simulation framework designed to detect and prevent battery draining attacks in IoT systems, addressing a significant gap in existing IoT simulation tools. This chapter is derived from [3]
- **Chapter 8** summarizes the thesis and illustrates directions for future work.

2

Systematic Literature Review of VANET Simulators

2.1 Introduction

Vehicular Ad-Hoc Networks (VANETs) have emerged as a critical component in the development of intelligent transportation systems, aiming to enhance road safety, traffic efficiency, and overall driving experience. Since the early 2010s, numerous surveys have been conducted to analyze the efficacy of various VANET simulation tools. These surveys have mainly focused on assessing mobility simulators and network simulators, providing insights into their individual functionalities and their interaction within VANET environments [29, 30].

Despite substantial advancements in the field, several important factors essential for the success of VANETs remain unresolved. In particular, there is a significant gap in the performance evaluation of recent technological innovations such as electric vehicles, self-driving cars, Software-Defined Networking (SDN), edge computing, and fifth-generation (5G) technology. The lack of in-depth analysis of these technologies within VANET simulators poses a challenge for researchers and practitioners in their efforts to implement these innovations in real-world scenarios.

Motivation: This survey is motivated by the need to fill these gaps and provide a thorough evaluation of existing VANET simulators in the context of these emerging technologies. The integration of electric vehicles and self-driving cars into urban mobility scenarios demands VANET simulators that are not only scalable but also capable of supporting large-scale experiments, node mobility, battery management, and the recharging of vehicles' batteries while moving. Additionally, the ability to model renewable energy sources within the simulation framework is becoming increasingly important as we move towards more sustainable smart cities. By identifying and addressing these deficiencies, we aim to guide future research and development, enhancing the realism and applicability of VANET simulations in real-world scenarios.

Contribution: Although significant research has already been done, many key factors for the success of VANETs are still open. Also, there is a lack of in-depth performance evaluation of new technologies such as electric vehicles, self-driving cars, SDN, edge computing, and 5th-generation technology. So, researchers need an updated review that

evaluates the current VANET simulators based on these new technologies. These technologies are essential in improving bandwidth, potential, and applicability, increasing their implementation in a production environment. The advancement of technology has revolutionized every aspect of our lives, making everything around us more intelligent. Therefore, in modelling an urban mobility scenario for a smart city that is heavily populated with electric vehicles, there is a need for a VANET simulator that meets specific requirements: it should be highly scalable and capable of supporting large-scale experiments. In addition, it should support the mobility of nodes. Furthermore, the system should support battery management and recharging of battery power while moving from one point to another in the city. As well as supporting the modelling of renewable energy sources.

Therefore, our objective in this survey is to evaluate VANET simulators' ability to simulate this scenario and support these technologies and to ensure that they meet the specific requirements including support mobility, autonomous vehicles, renewable energy sources, edge computing, and battery management and recharging of battery power while moving in the city. And aiming to identify the open challenges in VANET simulators that still need to be addressed.

The remainder of this chapter is organized as follows: section (§2.2 provides a background in the VANET simulation area. Section (§2.3) explains the research protocol used in this survey, and Section (§2.4) reviews the most up-to-date VANET simulators. A deep comparison of the existing VANET simulators is carried out in section (§2.5). Section (§2.6) analyses the support of selected simulators for novel technologies. The current field challenges are discussed in the last section (§2.7).

2.2 Background:

2.2.1 VANET Protocols

VANET protocols are a crucial part of the network that regulates the data transfer and communication within the network. They are generally classified into two broad categories: Routing Protocols and MAC (Medium Access Control) Protocols [31].

a) Routing Protocols in Communication Networks: The primary goal of the routing protocol is to discover and determine the paths data packets should take to reach their destination in the highly dynamic VANET environment. They are often classified into Proactive, Reactive, and Hybrid protocols, each with unique strategies for route discovery and maintenance:

- Proactive Routing Protocols: These protocols maintain fresh lists of destinations and their routes by periodically distributing routing tables throughout the network. They aim to maintain up-to-date routing information for all nodes, ensuring that routes are readily available when needed.
- Reactive Routing Protocols: These protocols find a route on-demand by flooding the network with route request packets. They are generally considered more efficient in highly dynamic networks like VANETs since they only seek routes when needed, reducing the overhead of maintaining up-to-date routing tables.
- Geographic Routing Protocols: These protocols utilize the geographical position of nodes to route data packets. Given that vehicles in VANETs are typically equipped with GPS, geographic routing protocols can efficiently manage routing by forwarding packets to nodes that are geographically closer to the destination.
- Hybrid Routing Protocols: These protocols combine the advantages of proactive and reactive routing protocols, aiming to optimize the routing process by utilizing the best aspects of both approaches.

b) Medium Access Control Protocols: MAC protocols manage access to the communication medium, ensuring that data packets can be transmitted without collision on the network. Given the importance of timely and reliable communication in VANETs, especially for safety applications, MAC protocols must efficiently manage access to the medium under varying network densities and traffic conditions.

- Contention-Based MAC Protocols: These protocols allow nodes to contend for access to the medium, with mechanisms in place to resolve conflicts (e.g., collisions) when multiple nodes attempt to transmit simultaneously. Carrier Sense Multiple Access (CSMA) and its variants are common examples. IEEE 802.11p is a

contention-based MAC protocol that utilizes a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism.

- Time-Division Multiple Access (TDMA) Based Protocols: These protocols divide access to the medium into time slots and assign slots to nodes, ensuring they can transmit without interference from neighbouring nodes. TDMA protocols can provide guaranteed access to the medium but must manage the assignment of time slots efficiently, especially in the dynamic VANET environment.

- Frequency Division Multiple Access (FDMA) Protocols: These protocols divide the available bandwidth into frequency bands and assign each band to a node to avoid collision and ensure smooth communication. FDMA can be helpful in scenarios where spectrum resources are scarce and must be utilized efficiently.

- Code Division Multiple Access (CDMA) Protocols: CDMA allows multiple nodes to transmit simultaneously on the same frequency band, utilizing coding schemes to separate and decode the overlapping transmissions at the receiver. CDMA can provide robust communication in scenarios with high node densities.

2.2.2 Communication Models

The communication in VANETs is generally divided into three types: Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), and Vehicle-to-Everything (V2X) [32]. Figure (§2.1) shows these different communication modes:

- **V2V Communication:** This type of architecture consists of nodes communicating without involving infrastructure components such as RSU, WIFI access points, or any other central access point. The nodes independently work as routers and pass the message from the sender to the receiver by the self-deciding shortest path. It is a less expensive mode of communication and helps give emergency alerts directly to the corresponding node with less delay.

- **V2I Communication:** This mode involves infrastructure to exchange messages between the vehicles, which makes it a pretty expensive and delay-prone method of communication.

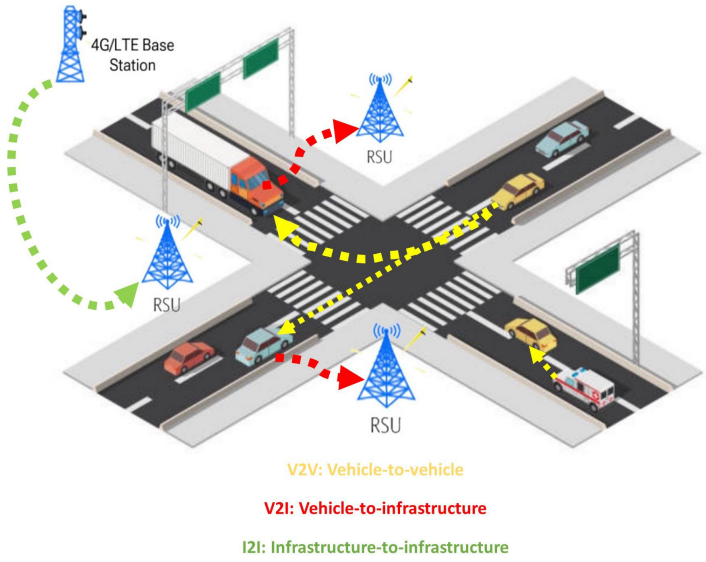


Figure 2.1: VANET Component and Communications

- **I2I Communication:** this communication mode takes place when an RSU (roadside unit) communicates with its counterpart or with the BS (base station) to pass critical information such as the picture of overspeeding vehicles, handing over the connected vehicle information to the next RSU, sharing the routing information and many more.

- **V2X Communication:** Vehicle-to-Everything including Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrian (V2P), and Vehicle-to-Network (V2N) communications, each serving different purposes and scenarios in the smart transportation ecosystem. On other words, this is a particular mode of communication in which all three modes take place simultaneously. It can be used to exchange information from one node connected with an RSU with another node or base station associated with a different RSU; in this way, all three types of communication take place at once. encompasses various types of communication between a vehicle and any entity that may affect or be affected by the vehicle [33, 34].

2.2.3 Potential Applications

The potential applications of VANETs extend well beyond the traditional realms of traffic safety and efficiency, promising transformative changes in diverse areas of transportation and mobility.

Safety Applications: VANETs can significantly enhance road safety by disseminating warnings about road hazards, accidents, and other emergencies. Examples of such applications include collision warning, lane change assistance, and cooperative forward collision warning [35].

An attack on these applications could occur in various forms, such as disrupting vehicular communication to create unsafe traffic conditions, disrupting safety warnings, or enabling unauthorized control over vehicular functions. Recent years have seen the development of several collision warning systems utilizing VANET [35]. A majority of the systems are designed to warn of forward collisions.

Traffic Efficiency Applications: By enabling real-time traffic monitoring and management, VANETs can contribute to improved traffic flow, reduced congestion, and optimized fuel consumption. Examples include intelligent traffic signal control, real-time traffic routing, and cooperative adaptive cruise control. The application of congestion control in vehicular networks can be divided into identifying, minimizing, and preventing congestion. Traffic congestion is effectively identified by identifying the patterns of congestion in the traffic flow. The minimization process is intended to reduce detected congestion. An effective congestion avoidance strategy requires the management of vehicular flows on roads to prevent the occurrence of congestion. A minimum-cost flow problem can be used to solve the problem of preventing traffic congestion. CARTIM in [36] is a collaborative identification and minimization approach to vehicular congestion.

Infotainment Applications: VANETs can also support various infotainment applications that enhance the driving experience, providing services like internet access, multimedia content sharing, and location-based services. Using these applications, travellers can enjoy comfort and entertainment. For multimedia streaming services in VANETs, wireless technologies, including IEEE 802.11p, DSRC/WAVE, can be utilized to establish vehicular networks (V2V), vehicle-to-vehicle (V2I), vehicle-to-roadside unit (V2R) and vehicle-to-universe (V2U), which are able to communicate via satellite, GPS, pedestrians, public safety, etc. In addition, peer-to-peer (P2P) systems have been extensively investigated as a possible solution for multimedia streaming services in VANETs [37].

Environmental Applications: Worldwide, there is a significant problem caused by the increase in vehicle numbers, congestion in vehicular traffic, and carbon dioxide emissions.

Several problems are associated with this, including direct effects on people's health, adverse economic effects, negative social impacts, local environmental damage, and the possibility of catastrophic global climate change. Through efficient traffic management and routing, VANETs can contribute to reducing vehicle emissions and environmental impact. Many existing approaches and techniques exist, such as EcoTrec [38], a novel eco-friendly routing algorithm for vehicular traffic that balances the requirements of both reducing travel time and emissions along the vehicle's route.

Automated Driving: With the rise of autonomous vehicles, VANETs can play a crucial role in facilitating vehicle automation, enabling cooperative driving, platooning, and other advanced functionalities. Many existing works exist on platoon-based vehicular cyber-physical systems, control of connected and automated vehicles, and parallel driving in cyber-physical systems. For example, in [39], the paper proposed a novel approach to distributed cooperative longitudinal platooning control that accounts for various challenges and is scalable and independent of platoon scale or global knowledge of communication topology, making it implementable on each vehicle in a scalable manner.

2.2.4 Security Considerations

Security is a paramount concern in VANETs due to the sensitive nature of the information being exchanged and the potential impacts on road safety and traffic management. Key security challenges in VANETs include maintaining user privacy, protecting against malicious attacks, and ensuring the authenticity and integrity of data. Various cryptographic techniques, trust models, and privacy-preserving methods are employed to address these issues [40]. Ensuring robust security in VANETs is a challenge due to the intrinsic characteristics and operational paradigms. The high mobility of vehicles results in a dynamically altering network topology, requiring security mechanisms that can efficiently deal with frequent changes in network configurations and neighbourhood relationships.

Moreover, the requirement for real-time communication, especially in safety-critical applications, imposes significant requirements on the degree of latency introduced by security protocols, such as cryptographic operations. Scalability arises as an important concern, given that security solutions must be adept at managing a potentially large number of communicating nodes. This ensures consistent performance and reliability across varied

network sizes and densities. Privacy preservation, particularly when protecting user-specific information such as location and preferences while still enabling effective communication and safety applications, further challenges the security paradigm.

Furthermore, managing cryptographic keys, which includes their distribution, revocation, and renewal, in a dynamic and distributed environment like VANETs, demands the most incredible attention to ensure that network participation is restricted to authorized vehicles and that compromised nodes are promptly identified and isolated. Also, it is essential to ensure the integrity of the data during the communication process to ensure that the message content is not altered. The public critical infrastructure and cryptography revocation process can be used to ensure data in these networks in VANETs [41].

Moreover, the role of authentication in VANET is crucial. It prevents the VANET from being attacked by suspected entities. User identification and sender address are essential information regarding the transmission mode. Authentication is responsible for controlling authorization levels for vehicles and preventing Sybil attacks by assigning an authorization level.

Consequently, developing security solutions for VANETs requires an approach capable of addressing several challenges resulting from their unique operational and application-specific characteristics [40].

2.2.5 Factors Affecting VANET Simulator Efficiency

VANET simulator efficiency is governed by an array of factors, ranging from the underlying models to computational resources and the scale of the simulation. Considering these factors to maximize simulator efficiency and achieve accurate, reliable results is imperative.

Scalability: As the size of the simulated network grows, the computational requirements also increase significantly. Therefore, the simulator's ability to efficiently handle large-scale simulations is a crucial determinant of its efficiency. This often necessitates the use of optimization techniques and parallel processing capabilities.

Realism of Models: The accuracy of the mobility, communication, traffic, and environmental models has a direct impact on the simulator's efficiency. More realistic models tend to be more computationally intensive, thus affecting the simulation speed. Striking a

balance between realism and computational efficiency is often a key challenge in VANET simulations.

Computational Resources: The hardware capabilities of the system running the simulation, including processor speed, memory, and storage, can significantly influence the simulator's efficiency. Advanced hardware can facilitate faster and more efficient simulations, especially for large-scale networks and complex models.

Software Optimization: The design and implementation of the simulator software can also affect its efficiency. This includes aspects like data structure design, algorithm efficiency, and software architecture. Optimizing the software can significantly improve simulation speed and resource usage.

Real-Time vs. Non-Real-Time Simulation: Real-time simulations, which mimic the network behavior in real-time, are generally more computationally demanding than non-real-time simulations. Depending on the specific use case, it might be more efficient, especially for large-scale networks and long simulation duration.

Complexity of Applications: The complexity of the modeled applications can also influence the simulator's efficiency. Applications that involve intensive data exchange or complex conducting non-real-time simulations can increase the computational load, thus affecting the simulation speed. These factors collectively shape the efficiency of VANET simulators, underscoring the need for careful planning, design, and optimization in VANET simulation studies.

2.2.6 VANET Simulator Components:

VANET simulators are sophisticated tools designed to accurately replicate the dynamic nature of vehicular networks. Typically, a VANET simulator consists of two primary components: a network simulator and a traffic simulator. Both components play a crucial role in representing the diverse aspects of VANETs, facilitating a holistic understanding of these complex systems [30].

- **Network Simulator:** The network simulator component is responsible for modelling the communication protocols and network behaviours in a VANET environment. It simulates the detailed interactions of the network layers, from physical layer phenomena like signal propagation and interference up to the transport layer dynamics

such as congestion control and data transfer. Furthermore, the network simulator also models the specific communication patterns within VANETs, such as Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) interactions, characterized by high mobility and rapidly changing network topologies. Network simulators that are commonly used in VANETs include NS-3[42] and OMNeT++ [43]. These tools provide comprehensive frameworks for the simulation and analysis of diverse networking protocols and scenarios, contributing significantly to the advancement of VANET research.

- **Traffic Simulator:** Complementing the network simulator is the traffic simulator component, which is responsible for modelling vehicular mobility patterns and traffic dynamics within the network. This includes the representation of individual vehicle behaviours, traffic flows, road network characteristics, and other aspects that dictate the movement of vehicles within the network. Traffic simulators are tasked with generating realistic mobility models that reflect various driving behaviours, traffic rules, and environmental factors, thereby contributing to the overall authenticity of the VANET simulation. Commonly used traffic simulators include SUMO (Simulation of Urban Mobility) and VISSIM. These tools enable researchers to create intricate traffic scenarios with varying traffic densities, diverse road layouts, and a range of vehicular movements.

The synergy between network and traffic simulators allows for a comprehensive representation of VANETs in simulated environments. Through their combined capabilities, these tools enable the creation of realistic scenarios and the exploration of a wide array of research questions, thereby driving the continuous evolution of VANET technologies.

2.3 Systematic literature review (SLR)

This research uses the systemic review and meta-analysis by PRISMA standards. A successful PRISMA schematic diagram is generated in Figure 5.5. The following points are important to interpret the PRISMA flow diagram:

Search process: The literature review considers the paper being published since 2006 and is available at the time of the writing (2023). This literature review was conducted by

initially searching the selected databases: Google Scholar, Science Direct, Sage journals, Wiley online library, ACM Digital Library, IEEE Xplore, Springer Link, Scopus, and Elsevier.

Search Keywords: Specific keywords and titles like vehicular ad hoc networks (VANET), VANET simulator, and VANET novel technologies are applied in the selected databases.

Inclusion and Exclusion Criteria: The criteria outlined below have been set to include the relevant publications identified for inclusion in this review: Articles that focus on the VANET simulation area, all articles should be online to ensure the paper accessibility, articles should be written in the English language, and peer-reviewed papers. On the other hand, we exclude all articles that do not meet the inclusion criteria, for example, articles related to traffic simulation or network simulation only, duplicate records, and papers before 2006.

Quality Assessment: The assurance of search result quality is achieved through the extraction of information from carefully chosen digital libraries. We have read the abstracts of the papers listed in the search results and made decisions regarding the inclusion or exclusion of each paper in order to proceed. The selected databases (Google Scholar, Science Direct, Sage journals, PubMed and Wiley online library) aimed at the VANET simulator and its novel technologies and produced a total number of 723 papers till May 5th, 2023. Google Scholar led with 470 results. Sage produced 104 results, Science Direct produced 82 results, PubMed produced 32 results and Wiley Online generated 35 results. After finalizing the list, the papers are filtered and we only consider the papers published since 2006. At the end of identifying the studies and the filtering process, we have a total of 128 papers that are included in this review.

2.4 VANET Simulators

This section shows an overview of current VANET simulators and a detailed description. This section focuses only on simulators that have been published since 2006.

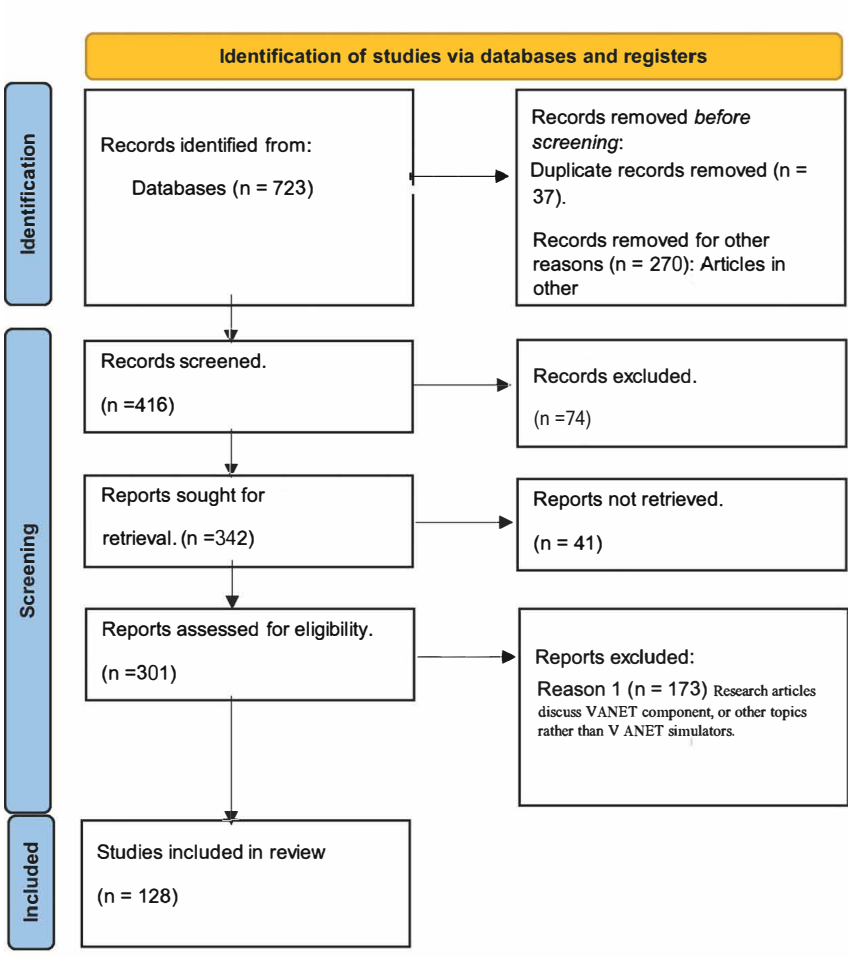


Figure 2.2: PRISMA flow diagram of current study

Veins: is an open software which allows simulations of vehicular networks. Veins is based on the incorporation of OMNeT++ with SUMO, which is a well-established simulator [22]. In its working, the OMNeT++ nodule for every vehicle is instantiated by the simulator. Movements of nodes and vehicles are then paired in the SUMO simulator. Mobility and network simulations can run parallel during this case. This is possible because its communication protocol and the traffic control interface i.e. TraCI allow bidirectional coupling [44]. During simulation, the exchange of messages (e.g., mobility traces) between OMNeT++ and SUMO is enabled by TraCI. Currently, the simulator includes more than 17 extensions [22] which allows diverse protocol modelling including the IEEE 802.11p standard that gives wireless access in vehicular communication [45], and ETSI ITS-G5 provides safe and secure communication in vehicular environments [46]. These extensions of simulators show diverse applications e.g., vehicle platooning [47]. However, Veins has some limitations, such as a lack of support for more realistic

models. For instance, it does not provide a full-stack implementation of the main security standards for VANETs (e.g., IEEE 1609.2) or offer any mechanism for systematically modelling faulty nodes (e.g., an unreliable RSU). The simulator was applied in a simple Manhattan Grid topology, which is a common model for urban traffic simulation, so there is no consideration for an urban mobility scenario.

VENTOS: Vehicular network applications can be analyzed using an open-source VENTOS [48] simulator. Important applications include vehicle platooning, participative driving, and automatic speed control. It incorporates SUMO for mobility and OMNET++ for modelling networks like Veins. In contrast, prebuilt modules in VENTOS allow simulation in difficult situations. E.g., Implementations for traffic control signal procedures, as well as managing platoon processes are provided by the VENTOS simulator. The two unique modules of VENTOS add Node and Traffic Control, making generating traffic demand an uncomplicated process. In addition, simulations can incorporate mobile and fixed nodes on users' choice, while traffic Control allows regulation of car traffic via speed change or identifying platooning plans. Expansion of simulators allows for interaction with onboard Unis and Road Side Units in HIL or the hardware in the loop situation. In this respect, there is a matching virtual node for every separate device linked with a computer, so any work done on a connected device affects the alias and vice-versa. The Ethernet port connects the hardware and machine running the VENTOS device that can communicate through SSH connections. Simulators need the support of software (that controls data management) to run on the device. As a result, the integration with some boards may get interrupted. To get correct results, VENTOS depends on the smooth working of both SUMO and OMNeT++. Windows, Linux, and Mac Operating systems support the VENTOS simulator. However, VENTOS has been used to study the security of connected vehicles, and it is designed to simulate small and specific scenarios, and it may not be suitable for large-scale simulations.

NetSim: is a discrete and broad-range commercial simulator which covers sensor, mobile, wired, and wireless networks [49]. Three different licenses present in the NetSim include standard, pro and academic. Pro and standard licenses are compatible with VANET models. The NetSim and SUMO interface is created to simulate VANET. Vehicle wire-

less communication is handled by NetSim while SUMO models road traffic conditions. NetSim delivers several networks evaluating measures, relation and application output plans. The type of network simulated generates varied metrics. By using packet and event traces, users can access details of all the packets in the network flow. Netsim makes the connection of simulations with live hardware applications possible and provides a sound number of components. RF Propagation Model is one of the most significant NetSim VANET Modules that incorporates different models: Path-Loss Model indicates the weakening of a signal in the wireless network. The Shadowing Model shows variation in signals as a result of obstacles. The fading Model shows changes in the strength of signals travelling in multiple paths. This is important for effectively predicting encountering signals and the loss of signals within a building or heavily occupied zones. These modules help create realistic simulations, as obstacles like buildings and heavy traffic are most likely encountered during signal communication in a real scenario. The design of Netsim allows the addition of requests-based components and flexibility for innovative technologies. However, NetSim is a commercial simulator that requires a license to use, and only the pro and standard licenses provide support for VANET simulations.

ezCar2X: This simulator incorporates an integrated system framework that allows fast modelling of applications within ITS and innovative networking procedures [50]. ezCar2X simulator is not an open-source software yet, but it is expected to be free software in a short time. It was developed to form a communication network called Car2X [51], allowing automobile creators, distributors, and road framework workers to evaluate its uses in a model simulation setting. C++ effectively implements ezCar2X with specified amendments, allowing proficient resource utilisation. Besides this, ezCar2X contains SUMO that can connect with different simulators via its TraCI API [44]. ETSI structural designs for ITS stations [52] formed the basis of ezCar2X architecture defining its access, network setup, capacity, and control and security layers. The core module of ezCar2X offers a logging system and schedules events that allow easy management of asynchronous tasks in real-time. Other modules to Access and Network that support WAVE, 3rd / 4th Generation cellular networks, Geonet working protocol, and Basic Transport Protocol. Another module named the security module regulates network security entities, allowing encryption of transmitted messages and decryption of received ones. Linux and Windows

both support the ezCar2X simulator. However, ezCar2X is a proprietary simulator, which may limit its accessibility to researchers. Additionally, there are no papers using ezCar2X to test SDN scenarios.

VANETsim: [53] is a particular type of event-simulator that is designed to investigate safety and confidentiality in the communication of vehicles. VANETsim permits the investigation of attacks and the study of recovery measures from an application point of view. For example, Impact was created through proposed strikes on different types of vehicles [54]. There are four main parts of VANETsim design. The First is the Graphical User Interface or GUI, where users are provided with a graphical map editor to create and operate road maps. OpenStreetMap [55] allows you to import or develop the maps from the beginning. Imported maps can be changed and stored as XML files, allowing information exchange with other tools. For example, TraNS which simulates realistic vehicular communication and mobility scenarios [56], and VanetMobiSim, which models the mobility of vehicles in VANET [57]. The interface of VANETsim is user-friendly indicating the vehicle's transmission range and activating functionality on demand. Users can propose a set of tests and save the configurations in XML libraries using the second core part, Scenario Creator. These configurations can be shared online which facilitates reproducibility of experiments. The third core part carries out actual simulation called the Simulation core which manages the map displaying road traffic, organization of the network, as well as specified safety units. Lastly, the Post Processing Engine procedures the files generated for displaying related actions.

Few pre-defined safety and privacy tools in VANETsim implement silent periods (vehicles refrain from transmitting their personal information) [58] and Mix Zones (intermingled communication between vehicles) [59]. Algorithm [60] makes ways of determining destinations which are individually navigated by vehicles. Two types of messages are present in communication between vehicles: beacons, which broadcast information. This includes speed and position; transmission of relevant and particular designed messages like in case of a vehicle suddenly drawing dangerously near. In April of 2017, the VANETsim project was closed. Even though there is no update in the tool, documentation and downloadable content are still featured by the simulator site [61], which guides how the simulator works. Windows supports the VANETsim simulator. However, VANETsim focuses on analyzing

security and privacy attacks and countermeasures on the application layer, so it doesn't support energy management. Also, VANETsim lacks support for 3D environments, which may limit its ability to model complex urban scenarios.

EstiNet: is a commercially successful emulator and simulator which uses kernel re-entering [62], an advanced methodology with combined benefits of both simulators and emulators. In kernel reentering methodology, these applications exchange packets, which are interrupted automatically by the tunnel network interface and redirected into the EstiNet simulation engine. EstiNet simulator is a VANET simulator and emulator that originates from NCTUns, a high-fidelity network simulator and emulator [63]. NCTUns used novel kernel reentering simulation and has been widely used since it was first developed in 2002. Due to kernel reentering implementation, NCTUns can implement a real-life TCP/IP protocol stack in the Linux kernel. This gives us high fidelity in simulation outcomes [63]. EstiNet can optionally have a VANET module as an add-on. The VANET module enables wireless communication between vehicles and their infrastructure. These allow the exchange of necessary information such as location and speed etc. EstiNet supports a function that assists in road building to simulate vehicular traffic by building a road network, starting from the beginning or by bringing in a guideline strategy. Mobility simulator features help in basic vehicle driving performance like following a car, changing direction, overhauling, and complying with traffic rules. However, this simulator is a commercial tool.

Eclipse MOSAIC (VSimRTI): was previously called V2X Simulation Runtime Infrastructure (VSim-RTI) [64]. The free simulation software proposes new solutions for automated and connected mobility in a multi-scale range. The Eclipse MOSAIC offers flexibility to the users in performing various V2X simulations by choosing their simulators. Eclipse MOSAIC achieves this by coupling different simulators, generating a realistic presentation of wireless communication between vehicles, vehicle traffic, and emissions. Examples of the simulators that support Eclipse MOSAIC include SUMO, which is used to simulate traffic scenarios. SNS and NS3 for simulation of a communication network; and Eclipse MOSAIC Application used for simulating applications. Runtime infrastructure requires three core elements that join the simulators with Eclipse MOSAIC. Runtime

Table 2.1: VANET Simulators Features

Name	ReleasedDate	OpenSource	NetworkSimulator	MobilitySimulator	Topology	Language	Use
GrooveNet	2006	✓	NS-2	own	Grid	C,C++	Moderate
CAVENET	2007	✗	NS-2	own	Straight line and circle	Matlab	Hard
TraNS	2008	✓	NS-2	SUMO	Urban	Java,C++	Moderate
NCTUns	2010	✗	NS-2	own	User-Defined	Java,C++	Hard
Veins	2010	✓	OMNeT++	SUMO	Urban	C++	Easy
VSimRTI	2011	✓	NS-3, OMNeT++	SUMO, VISSIM	Urban	C++	Moderate
EstiNet	2013	✗	own	own	Custom	Java	Moderate
NetSim	2014	✗	own	SUMO	Urban	C	Moderate
ezCar2X	2014	✗	NS-3	SUMO	Highway	C++	Moderate
VANETsim	2014	✓	own	own	Urban	Java 6	Moderate
VENTOS	2019	✓	OMNeT++	SUMO	Urban	C++	Easy

⋆NS-2: [65]

infrastructure is connected with each contributing simulator through the first core element called Federation Management. This consists of a simulator and connectors. Runtime infrastructure sends data that is received by one connector, while another connector sends data towards runtime infrastructure. Simulation and coordination of participating federate require crucial time management, ensuring the events of federate processes are in the correct order. Interaction Management allows messages to be exchanged between federates through publish-subscribe communication. Eclipse design offers a unique possibility of visualizing information in various ways. Diverse visualisation tools coupled with simulation can be used to evaluate this setup. ITEF, PHABMap, and WebSocket Visualization are examples of these visualization tools. However, Eclipse MOSAIC lacks support for fault injection, which may limit its ability to evaluate the robustness and resilience of VANET protocols and applications. Also, as the focus is on the simulation of V2X communication scenarios and its applications in traffic efficiency, this simulator doesn't support energy management.

2.5 Comparison of VANET simulators

This section compares the most recent VANET simulators based on their availability, functionality, complexity, resource consumption, accuracy, cost, and technical support. The most commonly used simulators and information are included in Tables (§2.1) and (§2.2).

Table (§2.1) presents a summary of the most recent VANET simulators focusing on their main characteristics. The scalability is supported by the most of simulators without any additional requirements. It is illustrated that Groove Net simulator [66] is created using

Table 2.2: Simulators and Recommendation Levels

Simulators	Routing	Clustering	Platooning	Security	Privacy
GrooveNet	V1,V3,V4	V1,V3,V4	-	V1,V3,V4	V1,V3,V4,V5
CAVENET	V1,V3	-	-	-	-
NCTUns	A1-A5	V1,V3,V5	-	V1	-
VEINS	V1-V6	V1,V3,V4,V5	V1,V2,V3,V4,V5,V6	V1,V2,V3,V4,V5	V1,V3,V4,V5
VENTOS	V1,V5	V1,V4	V1,V2,V3,V4,V5,V6	V1,V2,V3,V4,V6	V1,V3,V4,V5
EstiNET	V1-V5	V1,V5,V6	-	V1,V2,V6	-
NetSim	V1,V2,V5,V6	V1,V2,V5,V6	-	V1,V5	-
VANETSim	-	-	-	V1,V2,V3,V4	V1,V3,V4

C and C++ programming languages. As it uses a built-in mobility simulator it doesn't rely on a specific framework, thus can be used for multitasking as its moderate difficulty level. It can run on different operating systems. However, there aren't many resources and in-depth information regarding Groove Net.

TraNS [56] is one of the versatile simulators and is also compatible with many operating systems. CAVENET is a MATLAB-based VANET simulator, ideal for simulating and analyzing situations where people are driving together and communicating among vehicles. It works best with the Windows operating system, but data availability is limited. NCTUns [67] is developed using Java and C++. Its mobility generator allows users to customize parameters to better fit their needs. It runs on both Windows and Linux. On the other hand, VEINS is a simulator that is built using C++ and is more challenging to configure. VEINS runs on multiple operating systems and has comprehensive documentation, making it easier for users to understand and utilize the program effectively.

Key Considerations in Choosing VANET Simulator:

There are several simulators available for testing VANET. Since the performance or support of the simulators varies, choosing the appropriate one from the list is essential. When selecting a simulator, several variables are taken into consideration. These include the simulator's capacity to design and evaluate various application scenarios (V1), the availability of full instructions and resources (V2), the existence of published works and citations in the relevant field (V3), the number of active users (V4), the presence of active user groups and support (V5), as well as the ongoing development efforts (V6).

Researchers who have already begun their work may have difficulties if a new simulator is switched. As a result, these variables will aid in selecting the proper simulator which is given in Table (§2.2).

The routing column shows the simulator's ability to support routing protocol and analysis. Another column is clustering which shows the simulator's capabilities to handle

its algorithms. It refers to the process of grouping vehicles or nodes into smaller clusters based on certain criteria. The platooning column represents the capabilities of simulating platooning vehicles. In this, various vehicles travel together to improve traffic flow efficiently. The security column represents the security features that the simulator supports. The simulators listed under the privacy column provides the features as a solution to privacy-related issue or aspects within the vehicle.

Groove Net [66] is a universal simulator capable of running simulations of routing algorithms and clustering for various applications in the same domain, with citations demonstrating effectiveness for any number of active users (V1, V3, V4). However, it does not support platoon simulation. Groove Net can also handle security and privacy-related aspects for multiple applications in the same work domain, and links support functionality for active user numbers or active user groups (V1, V3, V4, V5).

CAVENET [68], on the other hand, has more limited features. It can handle routing algorithms for multiple applications in the same domain, with references (V1, V3) demonstrating its efficiency. It doesn't support clustering simulations or platoon simulations. Moreover, CAVENET does not provide any security or privacy-related aspects.

NCTUns is a simulator that supports routing algorithms (V1, V2, V3, V4, V5) and cluster modelling for various applications in the same domain with active user and group numbers (V1, V3, V5). Like previous simulators, it does not support platoon simulation. NCTUn can handle security-related aspects of many applications (V1) but does not support privacy-related features.

VEINS stands out as a powerful simulator dealing with routing algorithms and clustering modelling for many applications in the same domain, with citations confirming its effectiveness and active development (V1-V6). It also offers platoon simulation with active user count and group support (V1-V6). VEINS also excels in terms of security and privacy for various applications with active user and group numbers (V1, V2, V3, V4, V5).

Another simulator, VENTOS, handles routing algorithms and can simulate clustering for multiple applications with estimates for active user groups and support (V1, V5). We also support platoon simulation for various applications, with estimates for active user groups and support (V1, V2, V3, V4, V5, V6). VENTOS also provides security and

privacy-related features with active user and group numbers (V1, V2, V3, V4, V6).

EstiNET is a comprehensive simulator that supports the modelling of routing algorithms and clustering for various applications, quotes support functions for active users and group numbers (V1, V2, V4, V5, V6). It doesn't support platoon simulation. EstiNET also provides features on security-related aspects with links to active development (V1, V2, V6). However, it lacks support for privacy-related features.

NetSim is a simulator that can handle routing algorithms and simulate clustering for many applications with active user groups and resource availability for support and active development (V1, V2, V5, V6). However, it does not support platoon simulation. Provides capabilities for security-related aspects of various applications, along with the availability of resources and links to active user groups (V1, V5). Unfortunately, NetSim does not support privacy-related features.

VANETSim is limited because it does not support routing algorithms, clustering modelling, or platoon simulation. However, the availability of resources in the same working domain with active user numbers (V1, V2, V3, V4) allows you to handle security aspects for other applications. It also supports privacy-related features for multiple applications in the same work domain with active user numbers (V1, V3, V4).

2.6 Alignment with Recent Technological Developments

The last few years introduced up-to-date technologies in VANET design research as shown in Table (§2.3) [69, 70]. From self-driving cars to 5th Generation technology, new advancements have a critical role in refining bandwidth, potential, and applicability, which increases their implementation in a production environment. This section discusses the compatibility of various novel technologies with VANET simulators. We exclude VANETsim from this section as it was closed in 2017.

2.6.1 Software-Defined Networking (SDN)

In active network settings that connect lots of devices with diverse uses, SDN is the best suitable option [71]. Vehicular communication networks interestingly fit this case. In SDN, the centralized controller coordinates the entire network process [72] and control

and data planes are separated. More recently, researchers have been focusing on Software-Defined Vehicular Networks development with the combination of SDN and VANETs. This opened the door for interesting initiatives in the area, and VANET simulators now must support (SDN) Software-Defined Network, as it provides improved coordination of different network operations working in dynamic vehicles [73]. There are currently very few simulators that specifically support SDN or are used in SDN and VANET research.

NetSim version 11 supports SDN and is compatible with OpenFlow, a communication protocol that differentiates between the control and data planes in SDN. Therefore, similar technology can be implemented easily using varying networks like LTE, MANET and VANETs. Roadside Units has SDN controller in VANETs case. In [74], the objective is to study OMNeT ++ network equipment models and develop alternate samples incorporating all aspects of various software-defined tools applications. In this study, simulation accuracy remains improved regarding packet processing delay parameters using NetSim as one of the simulators. Many SDN-related papers are present when considering the Veins simulator, for example, [75, 76]. In [75], SDN creates a framework that decreases the load of storage, communication and response time, thus providing a secure platform for VANET communication. Veins and OpenFlow [77] configurations allow the support of SDN controllers. These controllers cooperate with Roadside Units to take control of the whole network. Alternately, in [78], SDN offers a highly proficient routing procedure. In this respect, the simulation of SDN controllers is done by POX which is incorporated in Veins. Data is sent to nodes by the source in diverse places with the help of SDN controllers. For example, a vehicle leaves one VANET range and enters another.

In [76], Within a low Road Side Unit that is simply a gadget or any infrastructure piece erected beside roadways to allow for vehicle communication, density zones, the Combination of Veins and OpenFlow allows rapid and reliable message circulation in an emergency. Novel solutions can be modelled for Cooperative ITS Systems with a powerful Eclipse MOSAIC simulator. It combines several simulators for modelling different application environments. The combination of SDN with the MOSAIC simulator was not used in any notable research studies. OpenFlow unit in EstiNet is a pre-defined component of the simulator. This SDN module has an Ethernet switch, which supports in-band and out-of-band network communications regulated by controllers. VANET module is a module in the simulator for simulating or modelling the behaviour of VANET simulators [79]. It

was combined with a simulator in EstiNet version 9. EstiNet provides good potential to create accurate setups using SDN, as shown in [80]. The main components to rapidly generate model applications for the in-vehicle networks are provided by ezCar2x. In-vehicle networks are the system protocols used within vehicles to provide many services such as sensors and control units depending on the effective communication and data transfer within the vehicle. This commercial tool is not easily accessible, which makes it hard to identify compatibility between ezCar2x and Software-Defined Networking (SDN). Papers incorporating ezCar2x to test SDN scenarios were not found in our study.

Among practical applications, VENTOS is used in studies of managing platoon [81] to inspect safety weaknesses of VANET-supported combined driving; on the other hand, no study related to SDN was found using this simulator. Nonetheless, as OMNeT++ supports SDN, compatibility between VENTOS and OpenFlow is necessary. There is a lack of evidence about SDN compatibility in VANETsim documentation could be due to the limited documentation and focus on specific VANET functionalities, outdated information, and limited integration in the system. In addition, components are entirely built, making it difficult to incorporate new novel technologies in the simulator because the present technologies don't support upcoming technologies. For example, if the VANETsim is designed with preset models and protocols, it lacks capabilities to support the integration of new technology or features as the present models are built and cannot easily be modified or replaced. It would require huge changes in the simulator's architecture to integrate new features, which is not easy and feasible practically [53].

2.6.2 The edge computing

This is considered the finest way to handle large data in VANET because it increases dependability by dividing up computational work across several edge nodes, optimizes bandwidth by allowing the relevant information to reach the data center, and improves the overall system of the network. It also defines the limitations of solutions like networks such as RSUs, and mobile cloud computing. Edge computing platforms successfully replace RSUs supporting services efficiently [82]. VANET research revolves around edge computing, which moves computation and data closer to the network's edge. In this architecture, edge devices process data locally instead of transmitting it to a central data centre, enhancing efficiency and reducing latency. Integration with other technologies,

such as 3G, 4G, and 5G networks, supports this distributed computing model, known as Multi-Access Edge Computing (MEC). MEC facilitates the implementation of cloud computing platforms closer to end users, improving response times and bandwidth usage. Despite its potential, no publicly documented projects in the research utilize NetSim, Eclipse MOSAIC, or EstiNet for evaluating edge computing in VANETs [83–85].

Veins models are designed with both pathways, such as direct and reflected [86], that are further applied in [83]. In [86], leveraging automobiles is shown as supporting infrastructures in improving bandwidth clogging. This approach highlights the issue of bandwidth limitations [84]. In [85], the proposed solution in this paper provides an idea of how vehicles learn unloading interruption acts of other surrounding vehicles. Veins provide V2V and V2I dumping using the autonomous vehicular edge (AVE) [87] technology, and this project is known as Car2MEC and uses ezCar2X. This project highlights the importance of connectivity for critical-time traffic safety applications. MEC Multi Edge Computing is considered to achieve this aim as it helps reduce the communication potential for short-distance information exchange. [88] introduces a new system named the MEC-enabled cooperative collision Avoidance system. This system is vigilant as it extends vehicles' range and navigation properties and helps detect any hazardous objects around them. The Connected and Autonomous Vehicles (CAV) service is designed to utilize ETSI ITS-G5 standard signals that vehicles transmit. It includes messages like cooperative awareness of the vehicle and surrounding environment signals notification.

Moreover, VENTOS emphasizes dynamic car models and traffic routing [89]. VENTOS is designed to separate workstations by creating diverse road conditions and a centralized server. In this, edge servers serve as a specific region and analyze traffic events. The edge layer can provide information by platoon by gaining information from other connected entities as in [90], attachment of edge server with RSUs and base stations enables the gathering of all relevant information. As the VANETsim is specifically oriented towards communication within the networks, it is difficult for them to simulate newer technologies [91]. 5G technology offers high-range bandwidth and connects to networks as they have millimetre waves to perform these functions for them [92]. When we Compare it with 4G, 5G offers better analysis, convenience, higher network density, presents a new approach to technology, and fusing stage. Delivery service is not guaranteed in large-scale networking systems with these communication standards as along (IEEE802.11p) [93]. The 4G

LTE [94] supports the incorporation of wireless internet only whereas 5G supports the incorporation of other networks. These wonderful features allow passengers or drivers to connect their vehicles in multiple ways to maintain safety levels [92]. NetSim is the source of 5G New Radio (NR) standard simulations that enable higher data rates. Veins framework allows the simulation of multiple devices that are designed on the principles of a project named 3GPP38 (3rd Generation Partnership Project) series [95]. Using NetSim and the Veins framework, researchers set an example to analyze the route protocols for warning signals in the mining sites. Veins uses an Internet Network Simulation framework (INET) that provides sets of protocols to support and connect to 5G server [96]. The discrete event simulation system known as OMNeT++ is also available as an open source [43, 97].

Noteworthy work done by Zhang et al. [98], Huang et al. [99] and Checkered et al. [100] are considered good references for 5G coupling. MOSAIC is the best example of 5G coupling because it offers a complete integration system, provides a complete analysis of realistic road conditions, and enhances platooning with 5G connectivity in VENTOS. A system combines VANET, mesh, and 5G infrastructures known as VsimRTI (predecessor of Eclipse MOSAIC). In the area of transportation, the integration of VANET, mesh networks, and 5G infrastructures offers several benefits. To prevent accidents, this integration provides realistic periodic information flow between cars and helps them comprehend the present condition of the road. This ensures road safety by delivering pertinent information, such as the state of the traffic [101]. EstiNet also supports 5G simulations through the free Graphical Console (freeGC) alliance, a built-in feature that provides support for 5G simulations within the EstiNet Simulation program, without any extra cost and simulation tools, including different modules for simulating Radio Access Network (RAN), that is a component of mobile network architecture, enables the communication between User Equipment's (UEs) behaviours and the network through antennas or maybe stations. UEs are the devices or gadgets individuals use to connect to the network including mobile phones, tablets, etc. In [102] cars can now serve as mobile cell towers because the VANET technology can simulate 5G servers. EstiNet analyses the road conditions, whereas ezCar2X [50], was designed to simulate networks including 3G and 4G. VENTOS uses 5G technology. The study [103] describes the implementation of 5G to VENTOS to show the connection in platooning. Simu5G technology is made

especially for simulating and analyzing 5G networks and highlighting their functions and features and serves to use 5G technology in VENTOS simulators.

2.6.3 Autonomous vehicles

Autonomous vehicles are specially designed for their smart navigation system. Six Levels of automation are described (0-5). At level 0, it is totally manual as the driver is in charge because there is no automation at this level. At level 1, assistance systems for drivers are introduced to help them with distance maintaining and adopting control of their vehicle. Level 2 introduces limited automation, as controlling the steering and braking system regulation, but the driver can take control anytime when necessary, especially in uncertain road conditions. At level 3, most of the tasks are attained by automation; drivers can be relieved from paying constant attention on the road, but in abrupt circumstances, drivers are supposed to take control back into their hands. Level 3 automation is not frequently used as it is uncommon. At level 4, automation is high as compared to level 3, but it is under certain conditions such as for specific locations or within specific location routes as in taxis and auto shuttles that work in predetermined locations. However, the driver can still take control in his hands whenever the circumstances are. Level 5 represents complete automation where the vehicle is designed to take complete control of all the tasks. There's no need for a driver because the vehicle behaves like a driver. Passengers can move within the vehicle without restriction because the vehicle controls everything for them. However, this level gives the basic idea of complete automation. Exact definitions can vary from different organizations or related bodies. Intermediate levels aim to achieve higher levels of automation [104].

AVs have proven beneficial for VANET technology as they improve this network capacity, enable a steady flow of traffic on the road, and result in smooth and safe travel [105]. Existing simulators in modified forms are used to stimulate Avs in the VANET system. To stimulate automation in VANET, factors including gap acceptance, deceleration, acceleration, etc. are considered as they play an important role in modelling the performance of autonomous vehicles (AVs) within the VANET setting. Simulator models like SUMO need to be adjusted in vehicles and should include parameters and performances that are unique to each vehicle to represent and simulate each vehicle in the system accurately. Simulators like SUMO should integrate macro and micro models

to capture automation and its network. SUMO simulator is modified in a way that it merged with an open-source simulation system known as USARsim (Urban Search and Rescue Simulation) and handles the traffic system. USARsim is responsible for robotic simulation for autonomous drivers [106]. SUMO along with other systems such as Veins, ezcar2X are useful in AV experimentation where they help them in simulators mobility as research papers [107–109] for Veins, [107, 109] for VENTOS, [110] for Eclipse MOSAIC, and [111] for ezcar2X provides evidence to support its claims.

2.6.4 Unmanned aeriform vehicles (UAVs)

Unmanned aeriform vehicles (UAVs) are auto aircraft that can operate without human pilot involvement. These are also known as drones, remotely controlled to perform various tasks including monitoring from the air, photography, and shipment delivery. They offer many advantages, such as operating in 3D space and reaching high altitudes. This makes them ideal for creating more networks and building a useful environment. These can also be used as mobile infrastructure elements to enable vehicles with their advanced services. UAVs are also very reliable and can serve as information transmitters between vehicles communicating through radio frequencies connection in data channels and extending the communication and providing relevant information about present conditions such as road hazards in hilly rural areas where direct communication with another car or vehicle is impossible [112].

UAVs have many features that make them useful for VANET applications as their position can be adjusted dynamically to provide the best signal range. Simulators are common to deal with 2D approach [33]. A 3D setup has a greater impact on VANET technology. Extending the simulation context and generating road networks is required to set up a 3D vehicular network [113]. Digital Elevation Models represent the Earth's topographic surface devoid of any surface structures, trees, or other objects and are used to calculate the elevation or height of a topographical point. DEMs enable Veins to create 3D scenarios although these do not present any current module for 3D provision [112, 114]. The digital elevation model helps block signals. That is why the strengths of the signal matter a lot as these are affected by elevation provided by DEM because the OMNeT++ figures UAV movement and accounts for the impact of elevation on signal intensity while modelling UAV movement and signal propagation. Wireless communication networks

Table 2.3: Support for novel technologies in current VANET simulators

Technology	NetsSim	Veins	VSimRTI	EstiNet	exCar2X	VENTOS
SDN	✓	✓		✓		
Edge computing		✓			✓	✓
5G	✓	✓	✓	✓		✓
Self-driving cars		✓	✓		✓	✓
UAVs	✓	✓				

*UAVs: Unmanned aerial vehicles

may be modelled using the discrete event simulation framework OMNeT++.

On the other hand, NetSim is common to simulate unmanned aerial vehicles [49]. This involves the mutual set-up of a toolbox of MATLAB equipped with NetSim and UAV, that allows co-simulation of UAV flight. VANET and UAV systems are not directly linked; still, NetSim offers source code that can be modified for potential experiments involving VANET scenarios. Both Eclipse MOSAIC and NetSim lack research despite having wonderful features. NetSim assists in the simulation of UAVs, although no research on using their combination has been found yet. Eclipse MOSAIC is based on a 3-dimensional imagining apparatus. The tool is constructed on the principles of PHABMACS, a vehicle simulator. It is very useful in using its 3D technology for visualizing objects, vehicle movements, and simulation events. Eclipse MOSAIC has made it possible to explore more opportunities in the future for visualizing UAVs along Veins and MOSAIC as they have the power to use multiple simulators. The summary of current VANET simulators based on novel technology can be seen in Table 2.3.

2.7 The Challenges Facing VANET

Recent VANET simulators can perform multiple functions but have major limitations in supporting new technologies and their security systems. These concerns open many opportunities for researchers to promote VANET simulators. This section discusses issues in detail, highlighting the need for more research to design advanced VANET simulation tools.

Insufficient network security One of the biggest challenges VANETs face is their vulnerability to network security breaches. These breaches can disrupt public safety and

cause traffic congestion in urban areas. As mobile networks that self-organize and adapt dynamically, VANETs are particularly vulnerable to cyber-attacks [115, 116]. Therefore, it is important to address these issues promptly. The most common types of attacks [117] include location deception, denial of service, data integrity attacks, and authentication and identification attacks. One instance of this phenomenon is hackers assuming the identity of internal nodes within a network. Through this deceptive tactic, they can transmit error messages to neighbouring nodes, thereby consuming network bandwidth or distributing incorrect data on busy traffic. The main goal of such actions is to hide or harm the network's structure. Malicious parties can modify data packets by introducing fabricated information to fulfil diverse malevolent objectives. The objective could be the infection of nearby vehicles that are currently running as relays or modifying a routing table to redirect a vehicle towards a region lacking GPS coverage, hence increasing the challenge of tracing a real-world criminal activity [118, 119].

Systematic fault injection Fault injections are the induction of faults into a system to check how the system will respond to these intentionally introduced faults. This helps the researchers understand the system's response to errors or faults. This fault injection is useful for the dependability and robustness testing of the system. It enables the system to show its fault tolerance capacity. This allows the system to go through various ways where the system may fail when exposed to deliberate errors or faults, including hardware faults, data faults, and transient and permanent faults [120, 121]. The best time frame to apply fault injection is during the development period of the system. Nevertheless, it is essential to note that every system in VANET relies on its own specific ad hoc injection solution, and thus far, there have been no public tools for fault injection in VANET that have been put out. A suitable fault injection tool can be employed to assess and authenticate the functioning of VANETs in the presence of potential faults, avoiding the need to await the occurrence of such problems in a real-world scenario. This has the potential to provide benchmarks for dependability, allowing the comparison of the safety of various solutions [122].

Insufficient reliability As previously indicated, integrating mobile entities and wireless communication within a self-organizing network necessarily makes the topological

configurations of Vehicular Ad Hoc Networks (VANETs) susceptible to instability and unreliability. Moreover, the rapid movement of cars requires frequent changes in network topologies, resulting in frequent interruptions. These interruptions occur when vehicles swap wireless connection points or attempt to communicate with other vehicles that have changed their positions. The limited lifetime of communication routes has a significant impact on the reliability of networks. One limitation of Vehicular Ad hoc Networks (VANETs) is the restriction of vehicle movements to the road infrastructure, hence limiting the use of conventional "re-routing" techniques to enhance network stability. To enhance the overall dependability of the system, it is imperative to consider many factors such as data exchange, transmission media, network recovery, and network administration.

Real-Time simulations The strategy of combining real-time simulation systems with event-based simulations in Hardware-in-the-loop (HIL) is a simulating technique along with a testing feature where simulated components are incorporated with hardware components to authenticate the system's behaviour in real- conditions or environments. Some new challenges exist as the existing simulators struggle to fulfil the performance requirements of hardware prototypes when they connect to other large-scale networks. This limitation comes from the restricted resources that impede the ability of a simulator to stimulate complex connections with multiple vehicles [123]. The Ego Vehicle Interface is a solution researchers propose to address performance limitations in Hardware-in-the-loop [124]. Whereas these workarounds often ignore the additional performance introduced by simulating the security mechanisms, including cryptographic processes and others. These have negative implications for the behaviour of real VANET components. That is why it is essential to check the integration system of these simulators with hardware devices to make them secure and reliable. The exploration of the integration of VANET simulators and hardware devices in the context of security primitives presents a compelling avenue for research [125, 126].

Supporting security standards In recent years, enormous progress has been seen in establishing high-security setups for transport vehicles. IEEE 1609.2 is designed as a security system for VANET. It provides secure communication, integrity, confidentiality,

and protection. It uses public key infrastructure information for authentication and describes cryptographic algorithms for limited-resourced vehicles. This system promotes the resilience of VANET and enhances the safety and efficiency of systems. ETSI (European Telecommunications Standards Institute) has also significantly developed security standards for intelligent transportation systems, including VANETs [127]. Even though there are a couple of existing works that extended the existing simulators to support IEEE 1609 [128], existing VANET simulators still do not fulfil the proposed security standards [34], which is why it prevents researchers from effectively comparing security systems with that of the VANET system. It is necessary to ensure the development of correct and efficient system components as it is a complex process to extend simulators to support the standards. It would require considerable research and advanced programming systems.

2.8 Conclusion

The rapidly growing field of VANET simulators shows a great revolution in the transport sector. However, to unlock the full potential of this technology, researchers must address important issues through research and utilize new opportunities for this conduct. Implementing robust security standards such as IEEE 1609.2 and ETSI recommendations is essential to the security and effectiveness of VANET systems. The capacity of VANET simulation tools to simulate the impact of encryption on network performance metrics, such as latency and message sizes, without actually performing the encryption computations is a challenging point that needs to be investigated [129]. The ability of VANET simulation tools to simulate the effects of encryption without implementing encryption algorithms allows researchers to study how encryption, or the simulation of encryption, affects network behaviour and performance in a controlled and reproducible environment. This is a crucial open challenge that needs to be addressed in the VANET simulation field.

Moreover, developing advanced fault injection tools will enable comprehensive dependability testing and lead to more reliable VANET solutions. Integrating instantaneous simulations with hardware-in-the-loop setups offers a powerful approach to validating VANET behaviour in real-world conditions. Researchers must carefully consider coupling simulators with hardware devices, including cryptographic processes, to ensure accurate and secure simulations. Ensuring the accuracy and realism of VANET simulations is

essential, especially with the advent of 5G networks, edge computing, renewable energy sources, and autonomous vehicles. Robust simulators must adapt to new environmental conditions and technological advancements to remain effective. The tendency to extend existing simulators with new capabilities emphasizes the importance of effective automated test tools. For example, extending the existing simulators to support renewable energy sources and battery management. This is important to study all the factors that affect the simulation environment.

These context-based testing tools simplify the development process and improve the overall quality of VANET simulators. In light of the previous challenges and opportunities, researchers are faced with the critical responsibility of developing advanced modeling tools for VANETs that effectively meet the needs of today's transportation systems, such as autonomous vehicles and renewable energy sources. However, from our literature review, we can assert that no VANET simulator currently supports modeling renewable energy sources and recharging the batteries of vehicles while moving around the city. Additionally, there is a lack of VANET simulators capable of simulating the effects of encryption, as no encryption algorithms have been implemented within these simulators. Furthermore, existing simulators are unable to seamlessly integrate real-world traffic data, limiting their ability to accurately reflect urban traffic patterns and their impact on vehicular communication networks. By addressing these challenges and continuing to push the limits of VANET modeling research, we can confidently pave the way for safer, more efficient, and smarter future transportation networks.

3

Advancements and Challenges in IoT Simulators: Review

Contents

3.1 Introduction

The Internet of Things (IoT) represents a fundamental change in how devices interact, communicate, and function within a connected ecosystem. In light of the rapid development of IoT technology, the need for effective simulation tools has become increasingly important. Researchers and practitioners can use these tools to model, analyze, and optimize IoT systems before deployment in the real world. To ensure they meet the current and future demands of IoT development, existing IoT simulators must be thoroughly reviewed and updated in response to the rapid pace of technological advancement.

Contributions: The objectives of this chapter are not only to find out about existing and updated IoT simulators, analyze them, and study their uses in real-case scenarios but also to evaluate the most recent open-source IoT simulators based on specific criteria that play a crucial role in IoT development. This review emphasizes open-source simulators to ensure the accessibility of analysis and customization options for researchers and practitioners. The IoT simulation field has been subjected to substantial research, however, most of these studies focus on specific aspects of the field such as the healthcare domain, security aspects, or wireless sensor techniques. Few comprehensive reviews have been published that examine IoT simulators in a broader context, considering performance evaluation metrics critical for IoT development, such as device mobility, energy models, Software-Defined Networking (SDN), heterogeneity, and scalability. To the best of our knowledge, no review paper examines the ability of IoT simulators to support renewable energy sources or use them in VANET simulation field. So, researchers need an updated review that evaluates the current IoT simulators based on recent technological developments. This review is a valuable resource for researchers and practitioners seeking to understand the current landscape of IoT simulation tools. So, our objective in this review is to evaluate IoT simulators' ability to simulate complex and large-scale IoT scenarios and to ensure that they meet the specific requirements, including support mobility, energy model, SDN, scalability, and renewable energy sources. We also aim to identify the open challenges in IoT simulators that still need to be addressed.

The results of our literature review indicate that the IoT simulator landscape has seen a diverse range of tools, each with its capabilities and features, indicating that an appropri-

ate generic IoT simulator that can simulate any scenario across different IoT applications is lacking. Also, there are still several challenges to overcome. These challenges include security and privacy issues, the need for supporting mobility, SDN, energy models, and addressing scalability and heterogeneity. Moreover, with the recent evolution of cities and traffic, more focus should be placed on integrating IoT simulations with VANET simulations. Given these challenges, there's an urgent demand for tools that can adapt to the continuously evolving requirements of the Internet of Things. Addressing these issues will enable more comprehensive and accurate simulations and open a variety of opportunities for researchers to make significant contributions to the field. The future development of IoT simulators is likely to be dynamic and multifaceted, reflecting the complexities and opportunities of the rapidly advancing IoT landscape.

From our analysis, it is evident that there is no generic IoT simulator that able to simulate any scenario in different IoT applications. Moreover, mobility, a crucial aspect of IoT, is supported by only a few simulators, with some offering only partial modelling of mobile IoT environments. The adoption of Software-Defined Networking (SDN) is still in its early stages, with only a few simulators offering comprehensive support. Energy modelling, a critical component for sustainable IoT deployments, varies in depth and focus across simulators. While some simulators provide detailed energy modelling for specific environments, others have a broader approach. In particular, *IoTSim-OsmosisRES* stands out as the first IoT simulator supporting renewable energy sources (RES), this is the only simulator can support RES and offer a comprehensive framework for evaluating Autonomic Computing algorithms in the context of osmotic computing. Scalability remains challenging for some simulators, but tools like the NB-IoT simulator are explicitly designed for large-scale environments, showcasing the potential for simulating expansive IoT networks.

The remainder of this paper is organized as follows: Section 3.2 provides some background information on the IoT architecture and attributes. Next, Section 3.2.3 details the importance of simulations in IoT environments and their associated challenges. Then, Section 3.3 explains the research protocol used in this survey, and Section 3.4.1 presents a summary of the studied literature and reviews the most up-to-date IoT simulators. After that, the following two sections highlight the answers to some research questions by identifying the categories of IoT simulators in Section 3.4.2, and exploring the main features

of simulators in different IoT applications in Section 3.5. Based on specific evaluation criteria, we perform a deep comparison of the existing IoT simulators in Section 3.6). Finally, Section 3.7 discusses current challenges and future works in IoT simulation.

3.2 IoT Fundamentals

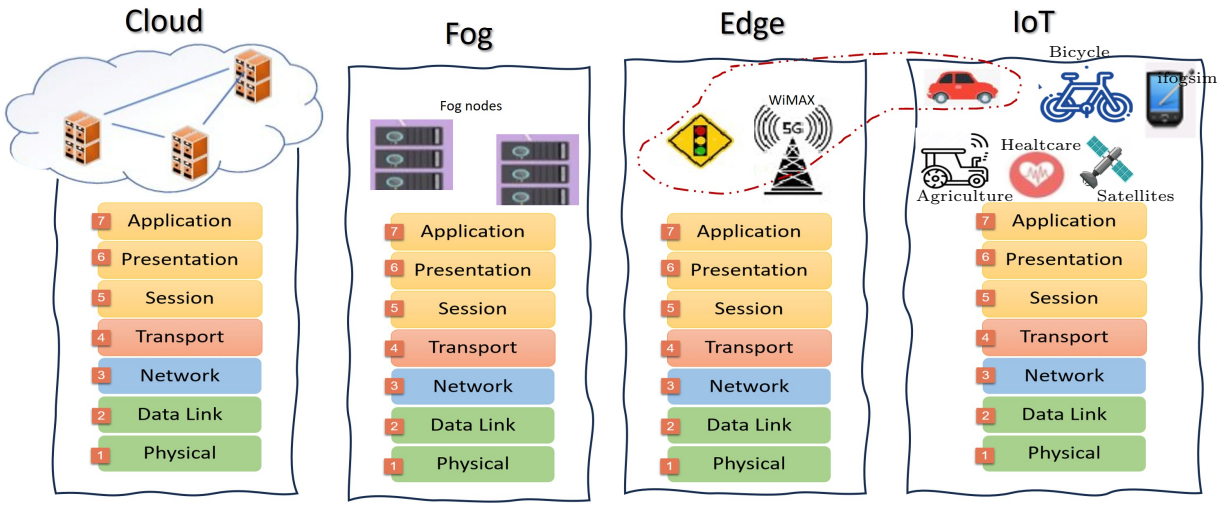


Figure 3.1: Network and IoT components [9–15]

3.2.1 IoT Architecture

In this section, we provide a brief overview of the IoT framework. It is important to note that not all applications or technologies employ a standard IoT architecture. There is a unique framework for each technology and each claims to be the best. IEEE proposed a draft of the IoT architecture framework for smart cities and a smart grid architecture standard from 2018 to 2019. Navigating the complexity and expansive nature of the Internet of Things (IoT) presents a challenge due to its inherent heterogeneity and extensive scalability. In constructing an IoT ecosystem, it's crucial to integrate a variety of elements, including devices, networks, and applications, to ensure they work in harmony to facilitate intelligent outcomes, while also prioritizing the aspect of security. IoT communication is underpinned by several layers involving a mix of technologies, protocols, and standards. These layers enable disparate technologies to interact across numerous scenarios, fostering scalability, diversity, and seamless interoperability within IoT systems as noted in the literature [130]. The architecture of the IoT typically consists of

four essential components: the service, platform, network, and device layers, with many research institutions adopting this standardized IoT framework as depicted in Figure 3.1 to maintain consistency and expertise in IoT development.

The service layer, positioned at the forefront, is designed to enhance user interaction through an interface, facilitating communication with users. This layer includes a variety of services such as autonomous driving, healthcare, smart industries, personal devices, and door security systems, all connected to a platform layer that delivers services customized for the user. Following the service layer is the platform layer, which underpins IoT applications and services. This layer is diverse, consisting of different platforms including device, data analysis, service development, and service platforms. For instance, the device platform provides an environment for the execution and development of services for users. Conversely, the data analysis platform supports context understanding and forecasting, enables entity collaboration, and bridges communication between the service layer and other layers by converting natural language into a format understandable by machines. Additionally, the service development platform offers development toolkits to simplify the creation of IoT services, while the service platform facilitates the generation and operation of various applications. Beyond these, the network layer plays a critical role in the IoT framework, responsible for the transmission of data across devices, content, services, and users. It handles the processing, control, and management of extensive network traffic. Finally, the device layer is engaged in sensing environmental conditions through various sensors, processing this data for forwarding to a sink node or gateway, and responding as necessary. The device must embody intelligence by implementing autonomous actuation and a sophisticated control algorithm. The physical layer ought to be able to obtain and manage IoT devices. Beyond the four layers, the significance of security and privacy in IoT cannot be overstated. Rather than designating security as a separate layer, each layer should embed security solutions to shield it from potential threats. Security concerns must be regarded as a crucial functional component for each layer, and any existing or future solutions should be tailored to each layer's specific characteristics and functions [131]. Every layer is important, possessing distinct roles and capabilities to facilitate IoT. While each IoT layer warrants thorough discussion, this paper will specifically delve into the IoT network. Our primary emphasis will be on exploring the challenges of the IoT network and offering foresight for its future development.

3.2.2 Attributes of the IoT infrastructure

The Internet of Things (IoT) emerges as a transformative paradigm, orchestrating an intricate web of interconnected devices spanning the spectrum from rudimentary sensors to advanced computational apparatuses throughout an expansive, distributed network. Characterized by its heterogeneous device environment, the IoT seamlessly combines low-power, resource-constrained sensors with robust computational devices, enabling both implicit and explicit interactions among them. This network, which is projected to develop into an ultra-large-scale system, navigates through the multifaceted challenges of managing real-time interactions, ensuring stable connectivity in a dynamic, frequently infrastructure-less network, and handling large amounts of event data. Moreover, the intrinsic characteristics of context awareness, intelligence, and location sensitivity in IoT devices and applications enable creating more adaptive, autonomous, and personalized smart environments. This complex infrastructure of the IoT holds the potential to reshape our digital interactions, embedding deeper intelligence and functionality into the internet-connected world that envelops us.

- **Diverse Devices:** IoT incorporates a wide array of devices, ranging from low-cost, low-power radios, which often avoid using WiFi or conventional cellular networks, to more robust computing devices necessary for tasks like routing and data processing. The diversity in devices arises not only from varying capabilities and features but also from the involvement of multiple manufacturers and distinct application needs [132].

- **Limited Resources:** The compact form factor of embedded computing and sensors restricts their processing, memory, and communication capabilities. For instance, while RFID devices might lack processing capabilities or even a power source, devices that offer more resources tend to be larger and more costly [133].

- **Unplanned Interactions:** IoT applications can facilitate unexpected interactions as devices move and enter the communication range of others, spontaneously generating events. An example would be a smartphone user unknowingly triggering events when approaching smart home appliances.

- **Extensive Network and Event Volume:** The IoT environment can involve thousands of interacting devices within a single locale, such as a building or supermarket, and on a global scale, it represents an ultra-large-scale network with potentially trillions of nodes. The vast number of interactions and events can pose challenges like event congestion and diminished event processing ability [134].

- **Dynamic, Infrastructure-less Network:** The IoT will amalgamate devices, many of which will be mobile and constrained in resources. The network will be highly dynamic due to factors like mobile nodes, unstable wireless connections, and battery limitations, making it challenging to maintain a stable network for various application scenarios [132].

- **Context Sensitivity:** The significance of context cannot be overstated in the realm of IoT applications. The vast quantities of data produced by a multitude of sensors attain their true value only through careful analysis and interpretation. The concept of context-aware computing is essential here; it involves the storage of information pertinent to sensor data, thereby enabling the interpretation that is critical for the adaptive and self-governing behaviors of IoT components [135].

- **Intelligence:** Reflecting Intel's vision for the IoT, the foundation of any IoT system is its intelligent devices and systems. Within the fluid and expansive network of the IoT, it is essential for smart entities, along with other components such as Web services and virtual objects, to work together seamlessly and possess the ability to act autonomously, making decisions based on the context and surrounding environment [132].

- **Location Consciousness:** Location information is crucial in IoT, significantly impacting context-aware computing. In a vast network, interactions are heavily influenced by entities' locations as well as the one of their surroundings.

- **Distributed Nature:** The IoT is a globally distributed network like the traditional internet. Its pronounced spatial dimension ensures that the network is distributed on various scales, both globally and locally, within a specific application area.

3.2.3 The Importance of IoT simulation:

As the Internet of Things (IoT) has become an important technology, enabling communication between devices and systems across various sectors, such as healthcare, agriculture, manufacturing, and smart cities, IoT simulators play a crucial role in the development, testing, and research within the IoT domain for several reasons [136] [137]:

- **Scalability Testing:** IoT networks often involve many devices. Simulators allow developers to test how systems perform under the stress of thousands or even millions of connected devices without needing to deploy physical devices. In addition, simulating devices distributed across various geographical locations helps understand and optimise data flow, latency, and management across different regions.
- **Cost-Effective Development:** Physical devices can be expensive and logistically challenging to manage in a testing environment. Simulators help in reducing costs related to purchasing, setting up, and maintaining physical devices. Also, simulators allow developers to utilize resources more efficiently, as they can test various scenarios and configurations without additional hardware investments.
- **Risk Mitigation:** Simulators enable developers to introduce faults and observe system responses, which is crucial for creating robust IoT systems without risking actual devices and data. Moreover, simulating cyber-attacks or security breaches on an IoT network helps identify vulnerabilities and enhance security features without exposing devices to threats.
- **Prototyping and Validation:** Developers can use simulators to create virtual prototypes of IoT systems, enabling them to test and validate functionalities before physical deployment. Also, simulators allow the testing of various use cases and scenarios to validate the feasibility and functionality of IoT solutions in a controlled environment.
- **Interoperability Testing:** IoT ecosystems often involve diverse devices with varying capabilities. Simulators enable interoperability testing among these devices without needing all of them physically. In addition, different IoT devices may use different communication protocols. Simulators facilitate testing the compatibility and interoperability of these protocols within the IoT network.

- **Real-world Scenario Testing:** Simulators can mimic different environmental conditions (like temperature, humidity, etc.) to test how devices and systems perform under various scenarios. Additionally, simulating different user behaviours and interactions with IoT devices helps understand usage patterns and optimise user experience.

- **Research and Innovation:** Researchers can use simulators to experiment with new algorithms, communication protocols, and architectures in a risk-free environment. Also, simulators can generate data that researchers can analyze to derive insights into system performance, user behaviours, and other aspects without deploying a full-scale physical setup.

- **Training and Learning:** Simulators provide a practical learning environment for students and professionals to understand IoT concepts and technologies without needing physical labs. Moreover, developers, testers, and administrators can enhance their skills by working with simulators to understand the complexities and challenges in IoT ecosystems.

3.2.4 Challenges in IoT simulation

Simulating the Internet of Things (IoT) brings about several challenges that need careful attention to ensure the simulations are accurate and can handle large-scale operations [138] [139] [140].

- **Security Challenges:** The IoT involves sharing sensitive data between various devices and sensors, which can be susceptible to cyber-attacks. Simulations need to account for these security threats and accurately mimic the actions of potential attackers. Understanding various possible attacks and being able to model these in simulations is crucial.

- **Energy Consumption Challenges:** Many IoT devices rely on battery power, which limits their available energy. It's crucial for simulations to accurately depict the energy consumption of these devices and how their behavior impacts the network's overall energy usage. Gaining insights into and modeling the energy consumption patterns of various devices and sensors is vital. Additionally, the extensive

nature of IoT systems often renders centralized management of process migration and data flow control inefficient. As such, there's a pressing need for the development of mechanisms that facilitate the use of distributed management algorithms. Moreover, with the unpredictable nature of environmental conditions—such as wind speed and solar radiation levels—simulators must be adaptable to changes in the operational environment of the devices. Modeling the diversity of green energy sources is important, moving beyond treating energy as a pre-existing factor. This includes considering all energy-related metrics, not just focusing on cost and consumption but also on the self-sufficiency provided by renewable energy sources.

- **Heterogeneity Challenges:** The IoT landscape is characterized by the integration of devices and sensors from various producers, each employing distinct communication standards. Simulations are tasked with navigating these variances, ensuring accurate representation of every network component's behavior. A thorough comprehension of the myriad communication protocols within the IoT, alongside the capability to simulate device behaviors governed by these protocols, is indispensable. For instance, in the context of supporting Software-Defined Networking (SDN), the employment of a multi-tier architecture that includes IoT, edge computing, cloud services, and SD-WAN necessitates harmonized interaction across these strata. Consequently, each layer is marked by continuous evolution, characterized by a mix of components, data formats, and protocols. This complexity may extend to varying behaviors and configuration specifics, such as power sources, computational/storage capabilities, and network bandwidth [141].

- **Scalability Challenges:** The IoT environment is characterized by a vast and ever-growing number of interconnected devices, each generating, processing, and transmitting data. As the number of devices and their interactions increase, the complexity of the simulation grows exponentially. Traditional simulation tools and methodologies designed for smaller and less dynamic networks often struggle to model the intricate behaviours and interactions of large-scale IoT networks accurately. This scalability challenge is further compounded by the heterogeneity of IoT devices, ranging from simple sensors to sophisticated computing devices, each with unique communication protocols, processing capabilities, and power constraints.

To effectively simulate such diverse and expansive environments, there is a pressing need for innovative simulation techniques that can accommodate the vastness and variability of the IoT landscape, while ensuring accuracy, efficiency, and real-time responsiveness.

3.3 Review Methodology

This section outlines the methodology employed in conducting this comprehensive literature review on Internet of Things (IoT) simulators. This research uses the systemic review and meta-analysis by PRISMA standards [142]. A successful PRISMA schematic diagram is generated in Figure 5.6.

Our literature review aimed to provide a comprehensive and up-to-date overview of the IoT simulator landscape. This involved extensively exploring academic journals, conference proceedings, technical reports, industry publications, and online repositories. The goal was to collect a wide range of scholarly and industry-specific resources that would help understand the evolving IoT simulator field comprehensively. The following points are important to interpret the PRISMA flow diagram:

Research Question The specific research questions addressed by this literature review are:

- **Q1** (Section 3.4). In which macro-categories can we ascribe each IoT simulator regardless of its specific use case of interest?
- **Q2** (Section 3.5). Do the specifics of each simulator usage scenario limits its generalizability, extension, or adaptability to other contexts?
- **Q3** (Section 3.6). What sort of evaluation criteria are applied to measure the performance of the IoT simulator?
- **Q4** (Section 3.7). Are there open challenges in IoT simulation still to be addressed?

Search Process This literature review represents the results of a search of the papers published since 2014 and is available at the time of the writing (2023). This literature

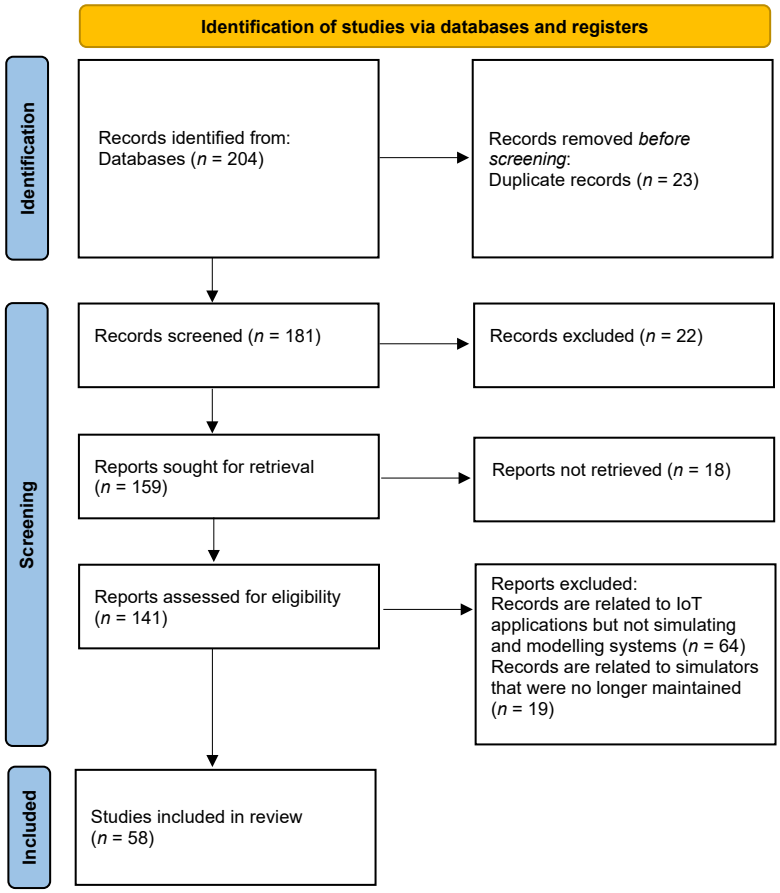


Figure 3.2: PRISMA flow diagram of current study

review was conducted by an initial search of the selected databases: Google Scholar, Science Direct, Sage journals, Wiley online library, ACM Digital Library, IEEE Xplore, Springer Link, Scopus, and Elsevier.

Search Keywords We obtained the search keywords from our research questions to find the relevant literature. The keywords are applied in this review are: IoT simulators, IoT applications, IoT challenges, IoT simulators performance, and IoT simulators types.

Inclusion and Exclusion Criteria The inclusion and exclusion criteria for selecting relevant articles in this review are as follows: Articles must be published in the specified databases to ensure and maintain quality. They should focus on simulating and modeling IoT environments. The simulators discussed in these articles must be actively maintained, with clear evidence of recent updates or releases. Additionally, the simulators should be capable of modeling IoT devices and networks. The review considered simula-

tors available in a variety of programming languages, including Java, C/C++, Python, and specialized IoT development languages. Only open-source simulators were included to provide a comprehensive view of the landscape. Furthermore, the articles must be accessible online for easy reference and must be written in English. Finally, only papers that have undergone peer review were included to ensure the credibility and reliability of the information. Papers were excluded from this review based on the following criteria: Any articles that did not align with the established inclusion criteria were excluded. Additionally, articles that focused on IoT applications without addressing simulation and modeling systems were excluded, as they fell outside the scope of this literature review. Simulators that were no longer maintained or had become obsolete were also excluded from consideration.

Quality Assessment The reliability of the search outcomes is verified through the analysis of information gathered from designated digital libraries. The authors review the abstracts listed in the search results to determine which papers should be included or excluded for further consideration. It has been observed that a majority of the studies assess synthetic datasets; however, our selection emphasizes papers based on real datasets within actual system environments to ensure practical relevance in engineering. A total of 58 papers met the selection criteria and were included in our literature review. The papers selected by the authors were collaboratively reviewed, and any discrepancies in opinions regarding the papers were resolved through discussion. Once the list was finalized, an in-depth analysis of the papers was conducted. The outcomes of our review and our contributions are detailed in the subsequent section.

3.4 Results of the systematic literature review

This section overviews current IoT simulators and provides a detailed overall description. The following section focuses only on simulators that have been published since 2014 until this survey was written. The table presents the existing IoT simulators including the focus, key features, and limitations for each simulator. We have divided the table into two Tables 3.1 and 3.2 for the space reason.

Table 3.1: Existing IoT Simulators (a)

Tool	Focus	Key Features	Limitations
iFogSim	Resource management in IoT, Edge, Fog	Evaluates policy impact on latency, energy. Models complex fog environments.	Does not support mobility.
MyiFogSim	VM migration in fog computing	Supports VM migration policies. Models mobile users and wireless access points.	Needs improvement in scalability.
EdgeCloudSim	IoT services over Edge and Cloud	Detailed analysis of service time, energy consumption. Accommodates mobile devices.	Missing nuances of diverse hardware features.
Mercury	Real-time fog computing scenarios	Focuses on low latency, high throughput, 5G. Data stream analytics and federated computation offloading.	Cloud computing not included in initial approach.
IoTSim-Edge	IoT and edge computing challenges	Models device diversity, protocols, mobility. Supports mobile IoT devices.	Does not consider energy consumption of infrastructure.
SimIoT	Cloud computing and IoT	Models users, data centers, virtual machines. Optimizes message exchanging. Supports heterogeneity.	Lacks real-world IoT implementation and explicit energy efficiency measures.
SimulateIoT-Mobile	IoT environments with mobile nodes	Extends SimulateIoT for mobile scenarios. Utilizes MQTT for mobility management.	Assumes guaranteed connectivity, which may not reflect real-world conditions.
PIoT	Network performance of IoT in cities	Front-end for simulation configuration. Models millions of IoT devices using cellular infrastructure.	Focuses on network performance, less on IoT device energy sources.
Contiki-Cooja	Network simulation for Contiki OS	Enables specification of Contiki motes. Provides crucial network data post-simulation.	Emphasizes hardware and network challenges, not IoT communication models.
ABS-SmartComAgri	Precision agriculture	Manages pesticide usage. Implements smart communication protocols.	Specifically for precision agriculture, not general IoT applications.
FS-IIoTSim	Industrial IoT systems	Supports communication protocols. Scenario modelling and performance evaluation.	Tailored for industrial environments, may not cover broader IoT applications.
IoTSim-Osmosis	Integrated edge-cloud IoT applications	Models dynamic workload transfer. Unified modelling for IoT in edge-cloud environments.	Limited wireless communication layer, fixed IoT device locations.
IoTSim-Osmosis-RES	Sustainable IoT ecosystems	Incorporates renewable energy sources. Models energy management and network infrastructure.	Doesn't support direct communication between IoT devices.

3.4.1 Current Simulation Platforms

IoTNetSim: [145], is an advanced platform for modelling and simulating end-to-end Internet of Things (IoT) services and networks. It is a valuable tool for researchers and practitioners, offering a self-contained, multi-layered architecture to model IoT systems with different structures, application models, services, and network connections. Its distinguishing feature is the detailed modelling of IoT nodes and sensors, including power sources and mobility, enabling highly accurate simulations for testing configurations and algorithms. The platform covers a wide spectrum, from IoT networking, including wired and wireless connections and protocols, to services and applications, and is designed to be modular and extendable. It supports a broad spectrum of IoT components, including mobile nodes and gateways, facilitating the simulation of diverse IoT applications like environmental monitoring and disaster response. This platform enables realistic mod-

Table 3.2: Existing IoT Simulators (b)

Tool	Focus	Key Features	Limitations
IoTNetSim	End-to-end IoT services	Detailed modelling of IoT nodes, sensors, mobility. Supports various protocols. Modular and extendable architecture.	limitations in supporting certain sensor types due to the complexity of modelling their mobility.
EdgeMiningSim	IoT data mining in edge computing	Multi-layered architecture. Supports task offloading and edge server management. High scalability.	Requires substantial computational resources.
Large-scale NB-IoT Simulator	IoT in smart cities	Integrates real geographical data. Tailored for NB-IoT and LTE devices. Discrete-event simulation approach.	Limited to NB-IoT and LTE devices.
ASSIST	Social IoT environments	Models social interactions among IoT devices. Supports common IoT protocols. Scalable for extensive networks.	Primarily supports SIoT environments.
Co-simulator for Smart Grids	Smart grids	Integrates Gridlab-D and CORE. GUI for efficiency and software emulation for fidelity.	Focus on smart grids, limited IoT applicability.
GVSoC	RISC-V-based IoT processors	Event-driven, balances accuracy and speed. Highly configurable for DSE.	Focuses on RISC-V, lacks support for other architectures.
Large-Scale IoT Simulator	IoT systems in urban settings	Simulates thousands of devices. High level of generality.	Limited to application-layer perspective, limits its suitability for testing low-level networking aspects
IoT simulator in [143]	Energy management in city districts	Integrates diverse data sources. Leverages LinkSmart Middleware.	Requires expansion for more IoT devices such as weather and traffic sensors.
LoRa-MAB	Resource allocation in LoRaWAN	Event-driven framework. Provides insights into network performance.	Focus on LoRaWAN, may not cover all IoT scenarios.
[1]Dynamic Co-simulation with Multi-Agent System	Modular IoT system simulation	Enables separate simulation of IoT components. Adaptable and modular.	Complex setup with multiple simulation tools, limitation for adding intelligence to the models
MobIoTSim	Mobile IoT device simulation	Emulates devices, generates real-time data. Connects to cloud gateways.	May not fully replicate real device behavior.
RelIoT	Reliability in IoT networks	Integrates modules for power, performance, temperature. Estimates device reliability.	Needs support for more complex reliability models.
MoSIoT	IoT healthcare monitoring	MDE for scenario creation. Supports commercial IoT hubs.	Focus on healthcare, may not cover other IoT areas.
Hybrid Simulation-Based in [144]	Large-scale IoT applications	Combines simulation and real-life testing. Utilizes PADS methodology.	Focused on system level, may not address detailed IoT protocols.
SimulateIoT	IoT system design and simulation	DSL for scalable IoT systems. Model-Driven Development.	Limited node mobility and hardware simulation.
SimulateIoT-FIWARE	IoT simulation on FIWARE	Extends SimulateIoT for FIWARE. Generates code for specific FIWARE technology.	Tailored to FIWARE, limited other platform applicability.

eling across physical, networking, and application layers, incorporating cloud, fog, and edge computing models to simulate data flow and processing. IoTNetSim is equipped to model various protocols, including cellular, WiFi, LoRa, and ZigBee, enhancing studies on system performance. Its detailed simulation capabilities also hint at potential applications in creating virtual urban platforms, exploring urban IoT ecosystems. Although not explicitly mentioned, the platform's architecture suggests it could simulate SIoT-specific communication strategies. IoTNetSim emphasizes realistic simulations with features for simulating network and battery failures, highlighting its scalability for large-scale IoT in-

frastructures. Its modular design indicates compatibility with numerous IoT development tools and programming languages, making it a versatile tool for IoT system research and development. Real-world application examples, such as monitoring natural environments and responding to disasters, showcase IoTNetSim's utility in guiding users from conceptual design to detailed evaluation of complex IoT systems. While it demonstrates its prowess in facilitating the evaluation of large-scale systems, it currently has limitations in supporting certain sensor types due to the complexity of modelling their mobility.

The large-scale NB-IoT (Narrowband Internet of Things): described in [146] is a robust machine-to-machine traffic simulator tailored for studying IoT application performance in extensive environments like smart cities. Its unique aspect is integrating real geographical data from smart city open data projects, creating a virtual urban landscape where devices interact with the telecommunications infrastructure. This approach enhances the realism of IoT application simulations and aids in understanding their performance and potential issues. It adeptly simulates a wide array of IoT components, including NB-IoT and LTE devices, capturing their interactions, energy consumption, and mobility within urban ecosystems. This simulator enhances realistic modeling of IoT applications, from utility monitoring to urban mobility, by integrating real-world geographical data, allowing for precise device positioning within the simulated city. Its multi-layered architecture focuses on connection procedures and packet transmissions, efficiently simulating extensive networks without sacrificing speed or accuracy. Tailored for NB-IoT and LTE technologies, it aligns with 3GPP specifications, offering insights into network efficiency and scalability. The simulator's systematic approach in creating a virtual urban platform starts from real geographical data collection, facilitating the simulation of device interactions across the urban environment. While focusing on NB-IoT and LTE, its architecture suggests flexibility in supporting various IoT protocols and standards, showcasing exceptional scalability for city-wide simulations with numerous eNodeBs and UEs through a discrete-event simulation approach. Programmed in Python, it's compatible with other IoT development tools, encouraging integration with data analysis algorithms. The simulator provides a robust framework for examining IoT application performance in urban environments, enabling scenario creation that reflects the complex network behavior at the city scale. However, since the simulator's focus on NB-IoT and LTE (Long Term

Evolution technology) devices and its physical layer abstraction limit, its applicability to other IoT device types and communication protocols, potentially providing an incomplete perspective on IoT application behaviour in real-world scenarios.

ASSIST (Agent Simulator for Social Smart Things): outlined in [147], creates a unique simulation tool for the Internet of Things (IoT), aiming to establish a Social Network-inspired environment for IoT entities. It empowers users to define smart Internet of Everything (IoE) entities, characterize their attributes, and establish social connections among them. ASSIST's foundation is a cognitive middleware containing a Social Network of Agents, including SIoT Agents and a Broker Agent, representing the instantiated IoE entities. The SIoT Agents are autonomous, intelligent entities that employ ontologies for knowledge representation and employ a Publish/Subscribe pattern for service exposure and resource consumption. The Broker Agent oversees communication between SIoT Agents and manages their status. The simulator employs a flooding mechanism for SIoT Agents to find required services and a deterministic approach to forming social IoT clusters, distinguishing it from probabilistic methods. It integrates diverse IoT components like sensors and actuators as agents, allowing for rich interactions and social connections based on shared goals or ownership. The simulator excels in modeling IoT applications across scenarios, from environmental monitoring to energy management, using ontologies for knowledge representation and supporting dynamic social connections through a Publish/Subscribe pattern. ASSIST accommodates common IoT communication protocols such as CoAP, MQTT, and HTTP, suggesting its utility in virtual urban platform development and smart city solutions. It adheres to IoT standards and employs semantic web technologies for interoperability, testing IoT environments under various conditions to assess network resilience. Demonstrating scalability, ASSIST can manage extensive networks, indicating compatibility with semantic web technologies and various IoT development tools. A use case involving cultural heritage protection through WSNs, UAVs, and anomaly detection apps showcases its capability to simulate complex social interactions among IoT devices for effective service discovery and collaboration. However, it primarily supports Social Internet of Things (SIoT) environment simulation and SIoT Agent behaviour.

GVSoC: [148] is an event-driven simulator tailored for RISC-V-based IoT processors, offering a balance between accuracy and speed. It combines efficient C++ models with flexible Python configuration scripts, making it a valuable tool for researchers focusing on highly parallel and heterogeneous RISC-V-based IoT processors. Unlike the slow but accurate cycle-by-cycle simulators or the fast but less informative simulators, GVSoC delivers reasonably fast simulation with accuracy, enabling rapid exploration of diverse configurations, crucial for Design Space Exploration (DSE). It's highly configurable and flexible, with open-source availability for the research community. It supports a comprehensive range of IoT components, including processors and peripherals, enabling the simulation of complex systems with multicore and multi-memory levels. GVSoC excels in IoT application modeling, particularly for evaluating performance and energy efficiency in applications using deep neural networks and near-sensor data analytics. Its multi-layered architecture facilitates high-fidelity modeling of system components, including detailed simulations of the PULP platform. While focusing on structural and functional capabilities, GVSoC implies support for various communication protocols and standards, essential for architectural exploration. Its simulation features are particularly suited for Design Space Exploration, offering rapid configuration exploration with a significant speed-up over cycle-accurate simulations, and maintaining errors typically below 10 percent. GVSoC's scalable architecture supports extensive IoT systems simulation, adaptable for urban infrastructure modeling. Utilizing C++ and Python, it provides a flexible and open-source development environment conducive to community research and IoT processor simulation. Documentation showcases real-life application examples, underscoring GVSoC's role in accurate performance estimation and design exploration for low-power IoT system design. However, it currently focuses on RISC-V processors, lacking support for other architectures like ARM or MIPS, and it's designed for near-sensor data analytics applications, including Deep Neural Networks (DNNs).

LoRa-MAB: [149] is a versatile Python-based simulator tailored for decentralized learning resource allocation within IoT networks, specifically focusing on LoRaWAN applications. It employs an event-driven simulation framework using the Simply library to replicate LoRa link behaviour across various network scenarios, including the capture effect and inter-SF collision. After a simulation, the simulator provides valuable insights

into the network's packet delivery ratio and overall energy consumption. LoRaWAN, a low-power, wide-area network technology commonly employed in IoT applications, constitutes simulation's primary area. This technology encompasses end devices, gateways, and network servers, forming a star-of-stars network topology. Gateways connect to the network server through IP-based backhaul networks, and LoRa modulation enables energy-efficient, long-range communication. Additionally, the LoRaWAN protocol offers security features such as encryption and authentication to safeguard data integrity and privacy.

MyiFogSim: [150] is a virtual machine (VM) migration simulator in fog computing that extends the iFogSim simulator. The simulator is designed to support VM migration policies for mobile users, which involves migrating VMs to cloudlets based on user position. This approach can result in lower latencies and better quality of experience (QoE) for users. MyiFogSim incorporates a range of new classes, notably the Coordinate class that depicts map coordinates on a Cartesian plane, and the ApDevice class, an extension of the FogDevice class from iFogSim, equipped with the functionalities and duties of a wireless network access point. Additionally, the simulator introduces a VM migration strategy for mobile users within a fog computing environment, contrasting it with a scenario lacking VM migration. The outcomes of the simulations indicate that implementing the migration policy can lead to reduced latency compared to a setup that does not employ the migration policy.

The co-simulator: discussed in [151], is a versatile platform developed for assessing the impact of emerging technologies on smart grids. It integrates two established simulators, Gridlab-D for power systems and CORE for communication networks. To tackle synchronization and interaction challenges between these components, the co-simulator employs a Graphic User Interface (GUI) for efficiency, software emulation for fidelity, and an Ethernet-tunnel-based distributed module for scalability. Co-simulator serves as a robust instrument for utility companies and policymakers to implement new IoT devices or strategies in upcoming smart grid frameworks. This platform offers flexibility and scalability, enabling experiments across various scenarios. It can run Linux applications on virtual nodes, using lightweight virtualization, and supports real-time and non-real-time

modes, making it a valuable tool for studying the effects of smart grid technologies, particularly when developing a new simulator from scratch is complex and time-consuming.

The large scale simulation: platform presented in [152] offers an efficient Java-based solution for simulating large-scale IoT systems in urban settings. Its strength lies in simulating thousands of geographically distributed devices, a crucial aspect of extensive IoT deployments that require rigorous testing. Unlike traditional IoT simulators focused on low-level networking, this platform provides a high level of generality, modelling devices with multiple network interfaces and various mobility, network, and energy consumption patterns. This encourages code reuse and efficient development. However, it primarily concentrates on the application-layer perspective of IoT systems, which may limit its suitability for testing low-level networking aspects, as it abstracts device interactions more than specific data transmission and routing details.

The IoT software infrastructure: proposed in [143], facilitates energy management and simulation in a city district by enabling the integration of diverse data sources and IoT devices. It incorporates real-time building energy profiles, environmental sensor data, and building/grid models, allowing for comprehensive energy consumption monitoring and management. The platform employs REST-based request/response and MQTT-based publish/subscribe communication paradigms, leveraging the LinkSmart OpenSource Middleware. It excels in modelling real-world IoT devices and ICT systems, integrating heterogeneous data sources, and simulating energy policies for district-level energy optimization while following the microservices paradigm. Nevertheless, further expansion is required to support additional IoT devices and data sources, like smart meters, weather sensors, and traffic sensors.

The dynamic co-simulation of IoT components using a multi-agent system: [137] is an innovative approach for simulating complex IoT systems in a modular and adaptable manner. This approach enables the separate simulation of IoT components in different simulation tools, with agents representing each component, allowing them to join a running co-simulation dynamically. The connection between agents and simulation tools is facilitated through an interface concept. While promising, this approach involves us-

ing multiple simulation tools and the development of agents for each IoT component, which can be complex and time-consuming. Additionally, it lacks a complete solution for adding intelligence to the models, as it mainly optimizes the simulation process by reducing message traffic rather than enhancing agent intelligence.

RelIoT: [153] is a framework for end-to-end reliability simulation in IoT networks, focusing on energy efficiency and reliability optimization. It integrates power, performance, temperature, and reliability modules into the widely used ns-3 network simulator. This framework estimates device temperature and reliability, a unique feature compared to other network simulators. It aims to balance energy consumption and Quality-of-Service (QoS) constraints, considering reliability as a design parameter that can be optimized. It offers runtime adjustments for the trade-off between performance and reliability and has minimal performance overhead for scalability. While validated with real-world experiments, RelIoT could benefit from support for more complex reliability models and diverse device types and applications, particularly those in challenging environments.

MoSIoT: [154] is an innovative approach designed for modelling and simulating Internet of Things (IoT) healthcare monitoring systems, particularly for people with disabilities. It utilizes the principles of model-driven engineering (MDE) to facilitate the creation of customizable and efficient healthcare monitoring scenarios. This framework allows healthcare professionals to easily simulate complete IoT healthcare scenarios, tailored for various disabilities and diseases, and subsequently generate final IoT systems. Key features of MoSIoT include a set of models for scenario simulation, integration with enterprise cloud architecture for data simulation, and support for commercial IoT hubs like Azure IoT Central. The framework's emphasis on customization, ease of use for non-technical users, and its potential for improving remote healthcare monitoring make it a significant contribution to the field of IoT and healthcare technology.

The hybrid simulation-based testing approach: in [144] combines simulation and real-life testing to evaluate large-scale IoT applications, aiming to effectively simulate the interactions between local entities (LEs). It utilizes the IEEE-standardized Parallel and Distributed Simulation (PADS) methodology, particularly implemented through the

Gaia/Artis IoT simulator, which can run thousands of virtual LEs in parallel. The main challenge addressed is the scalability required to manage real-time interactions among numerous LEs, achieved partly through cloud-based infrastructure and efficient synchronization mechanisms between real-life and simulation environments. However, the approach is primarily focused on testing large-scale Internet of Things (IoT) applications at the system level, and its main goal is to effectively facilitate interactions between local entities (LEs) in large IoT environments. So, it does not directly impact the use of renewable energy sources.

SimulateIoT: [155] is a Domain Specific Language (DSL) that streamlines the design, code generation, and execution of IoT simulation environments. It offers time and cost savings for developers by enabling the creation of scalable IoT systems without substantial hardware and software investments. Developers define models with numerous Node elements, generating code that adheres to recognized software architecture patterns like publish-subscribe and Docker containers. SimulateIoT's flexibility allows for easy adaptation to different target technologies. Key features include Model-Driven Development for handling IoT system complexity, the ability to define various nodes and policies with granularity values, and support for both CloudNode and FogNode. However, the node mobility in this simulator has been partially developed, and the hardware simulation is only managed by the size attribute at ProcessNode, which implies several constraints to avoid creating specific software elements. Second, the current version of the simulator IoT environment only allows defining connected nodes by TCP/IP, and it assumes that connectivity is guaranteed.

SimulateIoT-FIWARE: [156] is a domain-specific language (DSL) that enables the design, code generation, and execution of IoT simulation environments on the FIWARE platform. The language is based on the SimulateIoT metamodel [155], which defines the concepts and relationships required to model IoT systems. SimulateIoT-FIWARE extends the SimulateIoT to include FIWARE-specific concepts and relationships, such as context brokers, IoT agents, and IoT devices. The language also includes a set of M2T transformations that generate code for specific FIWARE technology, such as Orion Context Broker and IoT Agents. SimulateIoT-FIWARE is flexible and scalable and based

on open-source technology, allowing developers to choose the necessary components and integrate them with other technologies as required. However, SimulateIoT-FIWARE is tailored to the FIWARE platform, which may limit its applicability to other IoT platforms.

EdgeCloudSim: [157] is a simulation framework tailored for evaluating IoT services over Edge and Cloud systems. It excels in its fine-grained analysis, covering service time, resource utilization, energy consumption, and service reliability. Notably, it accommodates mobile devices with diverse hardware characteristics, expanding its applicability. The framework stands out for its dual consideration of computational and network aspects, encompassing WLAN and WAN communication models, device mobility, load generation, and virtual machine usage models. EdgeCloudSim's capacity to detail IoT service provisioning and the Edge/Cloud trade-off empowers researchers and practitioners. Furthermore, it introduces novel functionalities, like dynamic device hardware configuration, improved output, and expanded service simulation. However, its limitation lies in potentially missing nuances of diverse hardware features, potentially leading to inaccuracies in simulation results.

IoTSim-Edge: [158] is a simulation framework addressing IoT and edge computing challenges. It enables data-driven decision-making in smart environments like homes and transport. The framework models various aspects, including device diversity, communication protocols, mobility, and battery features. It supports mobile IoT devices and handles handoffs between edges for consistent communication. IoTSim-Edge is built on existing simulators, capturing the complete behaviour of IoT and edge computing systems. It allows for modelling various IoT protocols and their energy consumption profiles and provides a new abstraction for IoT application graph modelling. However, it doesn't consider the energy consumption of the entire IoT and edge computing infrastructure.

SimulateIoT-Mobile: detailed in [159], is a model-driven development tool to simplify the simulation of complex Internet of Things (IoT) environments with mobile nodes. Developers can employ it to model, validate, generate, and simulate IoT systems with mobility characteristics. The tool's metamodel, built on the Eclipse modelling Frame-

work (EMF), defines IoT system structure, while the Graphical Concrete Syntax provides a visual representation for enhanced understanding. This tool is an extension of SimulateIoT [155] and effectively addresses the absence of mobile node modelling capabilities. It utilizes the MQTT protocol for publish/subscribe communication and integrates an MQTT mobility management model to handle mobility within IoT systems. However, the tool has limitations, such as the assumption of guaranteed connectivity, which may not reflect real-world conditions where connectivity can be intermittent or disrupted.

PIoT: [160] is a large-scale simulator designed to assess the network performance of IoT devices in a city-wide context. It includes a front-end for simulation configuration, a back-end for modelling millions of IoT devices using cellular infrastructure, and geographical and application databases. The architecture employs a grid structure based on a realistic city model and supports various elements like NB-IoT, network slicing, MEC, and beamforming. PIoT enables the testing of applications and KPI data extraction for AI and ML algorithm development. However, its primary focus is on network performance evaluation, with less emphasis on IoT device energy sources.

ABS-SmartComAgri: [9] is a novel simulator for smart communication protocols in the context of precision agriculture. This is an open-source, agent-based tool designed to efficiently manage pesticide usage in agriculture by implementing smart communication protocols. It allows the simulation of various strategies, including broadcast, neighbour, and low-cost neighbour protocols, to optimize electric power, crop health, pesticide consumption, and overall network performance. The simulator is unique in its application to precision agriculture, explicitly targeting the reduction of pesticide usage and energy consumption while maintaining crop health. It is a valuable tool for testing and developing communication protocols before deploying them in real-world wireless sensor networks, contributing significantly to advancing smart agricultural practices.

FS-IIoTSim: [161] is an innovative network simulation framework specifically designed for the performance evaluation of Industrial Internet of Things (IIoT) systems. Characterized by its flexibility and scalability, the tool adeptly supports both existing and modified communication protocols, addressing the complexities inherent in IIoT environ-

ments. FS-IIoTSim is structured into three integral components: scenario modelling, performance evaluation, and a user interface. Scenario modelling facilitates the creation of detailed industrial sensor network models and the generation of trace files containing device operation logs. The performance evaluation component analyzes these trace files, focusing on key metrics such as throughput, latency, and energy consumption.

IoTSim-Osmosis: as detailed in [141], is an advanced simulation framework designed for deploying Internet of Things (IoT) applications within integrated edge-cloud environments. Rooted in osmotic computing principles, it enables dynamic workload transfer between cloud data centers and edge devices, driven by performance and security triggers. This framework uniquely caters to the complexities of IoT applications and the heterogeneity present in integrated edge-cloud environments. Notably, it is the first framework to offer unified modelling and simulation for intricate IoT applications in such diverse settings. While existing frameworks support cloud-edge integration, none can directly accommodate osmotic computing due to IoT application intricacies and environmental diversity. IoTSim-Osmosis empowers researchers to evaluate end-to-end IoT application performance using osmotic computing concepts comprehensively. A case study on electricity management and billing demonstrates its potential. However, the framework has limitations, including expanding the wireless communication layer and incorporating support for security and privacy simulation. Assuming fixed IoT device locations impacts accuracy, necessitating further research to address varying signal factors and mobility. Additionally, research efforts are required to develop models and algorithms for security and privacy simulation.

IoTSim-Osmosis-RES: [25], as an extension of IoTSim-Osmosis[141], encompasses a range of features pertinent to sustainable and autonomic IoT ecosystems, offering an assessment of various factors such as solar radiation levels, the utilization of renewable energy sources, the adoption of low-emission sources, and the battery capacity of IoT devices. It aims to simulate osmotic computations within the context of renewable energy sources and autonomic agents, thereby facilitating the analysis of distributed management algorithms. IoTSim-Osmosis-RES distinguishes itself as a unique IoT simulator capable of encompassing energy management, diverse power sources, and network infras-

Table 3.3: Classification of IoT Simulators Based on their Operational Domains

Simulator	Edge Modelling	Fog Modelling	Cloud Modelling
iFogSim	✗	✓	✓
MyiFogSim	✗	✓	✓
SimIoT	✓	✗	✓
IoTSim-Edge	✓	✗	✓
IoTSim-Osmosis	✓	✗	✓
IoTSim-OsmosisRES	✓	✗	✓
IOTSim	✓	✗	✓
CupCarbon	✓	✗	✗
MobIoTSim	✗	✓	✗
SimulateIoT	✓	✓	✓
Contiki-Cooja	✓	✗	✗

tructure. In contrast to other simulators, it can simulate fluctuating weather conditions and the utilization of renewable energy sources, while also affording an easily extensible system that empowers researchers to specify their own virtual machine (VM) and power management policies. This framework is further characterized by its support for Software-Defined Networking (SDN), IoT devices, IoT device batteries, and renewable energy sources.

3.4.2 Classifications of IoT Simulators

IoT simulators are categorized into distinct types based on their operational level within the overarching IoT architecture and the specific location where both data processing and simulation tasks are executed, into cloud, fog, and IoT device simulators [162]. Table 3.3 shows these three categries and some example simulators for each one.

Cloud Simulators: Cloud IoT simulators primarily emulate cloud-based IoT platforms and services. Their design is rooted in understanding the intricate interactions between IoT devices and cloud services. They help evaluate the performance and efficiency of cloud resources, scheduling algorithms, and application deployments. This includes the mechanisms of data storage and processing in the cloud, as well as the intricacies of cloud-based analytics. Typically focuses on large-scale simulations involving multiple data centers and thousands to millions of VMs. They are used to evaluate cloud service models (IaaS, PaaS, SaaS), resource allocation strategies, and energy efficiency. Simula-

tors of cloud computing emphasize the complexity of data centers, virtualization, and the management of large-scale resources. Examples of such simulators include the CloudSim Simulator. The foundation of these simulators lies in the services provided by cloud platforms, encompassing aspects like device management, data storage, analytics, and machine learning. The following are some examples of cloud simulators:

- **IOTSim**[158] is a versatile simulator implemented on top of CloudSim, primarily focused on modelling and simulating multiple IoT applications in shared cloud data centers. It covers various modules, including IoT application modelling, MapReduce data processing, cloud data center management, and network simulation. It offers a layered architecture that facilitates the simulation of various IoT components, including VMs on data center nodes with diverse hardware configurations, enabling detailed modeling of IoT applications from smart cities to healthcare systems with big data technologies like MapReduce. IOTSim excels in processing large volumes of data using parallel processing technologies, simulating network and storage delays critical for IoT applications, thereby providing a realistic model for the execution of parallel and distributed applications. Its protocols are tailored for efficient big data processing in cloud computing environments, supporting both batch and stream models, and demonstrating scalability and adaptability for complex IoT environments. The architecture integrates CloudSim's core engine with layers for big data and user code, enhancing the simulation of network and storage delays. IOTSim's design is suited for a range of IoT applications, from building automation to wearable tech, highlighting its ability to simulate batch-oriented applications with high accuracy and offering a cost-effective solution for IoT solution development in cloud computing environments. IOTSim excels in accurately simulating IoT application performance and resource scalability in cloud environments but lacks support for stream-oriented IoT applications.

- **SimIoT** [163] is a simulation toolkit designed to analyze cloud computing systems in the context of IoT. It extends SimIC by including user submissions from IoT devices like sensors and smartphones. It models various entities and interactions, including users, data centers, and virtual machines, with a focus on real-time constraints. The toolkit offers message-exchanging optimization and supports heterogeneity, making

it versatile for various cloud configurations. Although valuable for healthcare and information processing scenarios, SimIoT lacks a real-world IoT implementation, physical device simulation, mobility support, and explicit energy efficiency measures.

Edge Simulators: Edge simulators specifically focus on the computing resources that are located at the edge of the network, closer to the source of data generation which are IoT devices. Unlike cloud or fog computing, edge computing aims to process data at or near the source, significantly reducing latency and bandwidth use. These simulators are crucial for evaluating the performance, reliability, and operational efficiency of edge computing environments. They model the dynamics of data processing, storage, and application execution at the network's edge, enabling researchers and developers to explore various edge deployment scenarios, resource management strategies, and application behaviors in real-time conditions. Edge simulators are designed to address challenges such as limited computing resources, network connectivity variations, and the seamless integration of edge computing with cloud and fog layers. This allows for the optimization of IoT applications and services that require immediate data processing and decision-making capabilities. Examples of edge computing simulators include EdgeCloudSim and IoTSim-Edge, which offer detailed modeling tools for edge computing scenarios, from single-edge nodes to complex, multi-tier architectures involving edge and cloud collaborations. The following are some examples of edge simulators:

- **CupCarbon**, introduced in [164], is a cutting-edge platform for designing and simulating IoT Wireless Sensor Networks (SCI-WSN) in Smart Cities. It addresses the increasing prevalence of radio communication systems and technological advancements enabling IoT. CupCarbon stands out from traditional simulators by offering realistic modelling of radio channels and interferences, accounting for deployment environments, supporting mobile nodes, and enabling behavioural analysis in practical scenarios. It's a versatile tool with a multi-agent environment for mobility scenarios and event generation. It offers realistic modeling of radio channels and interferences, supports dynamic mobile nodes, and includes a comprehensive range of IoT components such as microcontrollers, sensors, batteries, and radio modules optimized for energy efficiency. The platform supports ZigBee 802.15.4 for WSN

applications, and long-distance communication technologies like LoRa and SigFox, facilitating direct communication to base stations and reducing the need for extensive sensor node networks. CupCarbon's novel architecture integrates environmental factors, mobility, and accurate radio propagation models for urban settings, enhancing network deployment studies and interference detection. It supports physical layer communication standards (ZigBee, WiFi, LoRa) for evaluating link quality and transmission conditions. The simulator's 2D/3D virtual urban platform aids in planning smart city projects by simulating sensor network deployments, node mobility, and radio propagation. With its discrete event simulation kernel, CupCarbon scales to simulate dense sensor networks, crucial for future smart cities. Its modular, Java-based design ensures compatibility with various IoT development tools, facilitating customization and integration. CupCarbon's utility in simulating smart city environments, showcasing sensor deployment, mobility modeling, and communication link visualization, underscores its comprehensive tool-set for designing, simulating, and visualizing IoT network dynamics in urban settings. Although it represents a significant advancement in wireless sensor network simulation for IoT and Smart City applications, it has limitations, including integrating only two radio propagation models and potential challenges with complex scenarios.

- **Contiki-Cooja** [165] is a network simulation tool derived from the Contiki operating system. Developed in Java, it enables users to specify both large and small Contiki motes (sensor network nodes) for deployment across the network. After running a simulation, users can access crucial network data, including mote outputs and timelines. It's worth noting that motes can be custom-defined using template options. These simulations primarily emphasize hardware and network challenges rather than IoT scenarios or communication models like publish-subscribe.

- **EdgeMiningSim** in [166], offers a simulation-driven approach for IoT data mining within edge computing. This methodology addresses IoT applications' unique challenges in data mining, aiming to empower domain experts with actionable insights for decision-making in dynamic IoT scenarios. The simulator is adaptable to various IoT applications, encompassing algorithmic, infrastructural, and contextual aspects often studied in isolation. It employs an interactive and iterative ap-

proach, balancing technical objectives and business interests. It simulates a diverse IoT ecosystem, including sensors and edge servers, integrating algorithmic, infrastructural, and contextual aspects for a comprehensive architectural design, network modeling, and topology definition. Employing a multi-layered architecture, EdgeMiningSim offers realistic modeling of device and application characteristics, such as mobility and energy consumption. While not extensively detailed, the simulator accommodates various IoT communication protocols and strategies, including task offloading and edge server management. It highlights the role of simulation in the early stages of IoT deployment, supporting the planning and evaluation of virtual urban platforms. The simulator includes mechanisms like flooding for efficient information dissemination among nodes and supports lightweight, decentralized data processing techniques. Demonstrating high scalability, EdgeMiningSim is suitable for large-scale IoT deployments, with a modular architecture indicating compatibility with numerous development tools. Through a smart environmental monitoring case study, EdgeMiningSim exemplifies its ability to simulate real-life IoT data mining scenarios, bridging the gap between theory and practice, and underscoring its value for researchers and practitioners. While promising, EdgeMiningSim does require substantial computational resources, which may pose challenges for resource-constrained IoT devices with limited processing power and memory.

IoT Simulators: IoT simulators simulate the behaviour of IoT devices, networks, and services. They are used to evaluate the performance, scalability, and reliability of IoT systems. Their main features are device modelling, network communication, data generation and processing, etc. They are used to test IoT protocols, device-to-device communication, and scalability of IoT networks. Simulators of IoT computing focus on the heterogeneity of devices, network protocols, and scalability challenges. They are designed to handle many IoT devices, potentially in the order of millions, although not all IoT simulators simulate the cloud. Thus, comprehensive simulation environments must be developed that represent the entire journey of data from its origins in IoT devices to its eventual processing and storage in a data center in the cloud. The following are some examples of IoT simulators:

- In [167], the authors propose a **multi-level IoT simulator** designed for smart terri-

tories, emphasizing scalability and advanced modelling techniques. The approach aims to promote the development of sustainable services in non-metropolitan areas. The simulator offers a two-level simulation approach, encompassing IoT systems' physical (sensors and actuators) and logical (software components) aspects. While the methodology shows promise in enhancing services for decentralized regions, it faces challenges in handling large-scale IoT environments. Its distinguishing feature is the comprehensive modeling of both physical and logical scenarios, essential for the development of sustainable services in non-metropolitan areas. Physical scenarios encompass the tangible components of IoT systems, such as sensors and actuators, which interact directly with the environment. An illustrative example is the implementation of a smart market, where various producers utilize these devices to advertise product availability, enabling customers to engage interactively with the marketplace in real-time. This scenario demands precise simulation of physical interactions and device deployments, ensuring the efficient operation of sensors and actuators within a dynamic market environment. Conversely, logical scenarios address the software-driven aspects of IoT systems, including data processing algorithms, communication protocols, and decision-making mechanisms. These scenarios are pivotal for simulating the complex, intangible interactions that govern the behavior of IoT systems. In the context of the smart market, logical simulation involves modeling the publish/subscribe mechanisms that facilitate information exchange between producers and consumers. It also encompasses the detailed simulation of wireless communication protocols, crucial for providing customers with timely information on product availability and guiding them through the market.

The multi-level simulation approach, integrating coarse-grained and fine-grained simulations, is instrumental in addressing the intricacies of such scenarios. At the coarse level (Level 0), the simulation provides a broad overview of the smart territory, encompassing general interactions and behaviors. The fine-grained level (Level 1) delves into detailed simulations of specific areas, such as the smart market, focusing on intricate wireless communications and proximity-based interactions. This dual-level methodology ensures scalability and maintains high levels of detail where necessary, thereby facilitating the intricate simulation of IoT environments for the advancement of smart services in decentralized regions. This simulator de-

signed to support the development and optimization of smart services in cities and decentralized areas, incorporates a comprehensive range of IoT components such as sensors, RFID devices, and mobile terminals. It employs a two-level simulation approach: a coarse simulation for general behaviors and interactions, and a more detailed simulation for specific aspects like wireless communications, utilizing tools like OMNeT++. This simulator supports various protocols crucial for realistic IoT scenarios, such as epidemic dissemination protocols, and integrates tools like MASON and SUMO for modeling urban systems and intelligent traffic control. While not explicitly mentioning a flooding mechanism for Service IoT (SIoT) Agents, its flexible architecture suggests compatibility with various communication mechanisms. It addresses scalability challenges through adaptable programming frameworks, ensuring compatibility with a broad range of development tools. It is capable of modeling complex scenarios, like a smart market, illustrating its potential to significantly contribute to smart city services and strategies.

- Ref. [168] introduces a **3D virtual environment simulator** for IoT-based smart house systems, enabling controlled testing and evaluation. The simulator features virtual sensors replicating real-world conditions and an autonomous agent generator to simulate human-like behaviour in the smart house. It is a cost-effective tool for testing and evaluating IoT-based smart house systems, enhancing design and issue identification. It features a wide array of virtual sensors and an autonomous agent generator to accurately replicate real-world conditions and human-like behavior in smart houses. This simulator supports a broad range of IoT components, such as motion detection and temperature sensors, providing a comprehensive tool for architects and engineers to configure smart house layouts intuitively via a GUI-based interface. Its innovative motivation-based behavior planning method for generating autonomous agents ensures realistic interactions within smart environments, suggesting its potential use in larger urban planning projects. Although it does not specify communication protocols, its design implies compatibility with common IoT technologies. The GUI facilitates easy design and testing of complex systems, indicating scalability and offering insights into optimal sensor placement and smart house dynamics. The development environment is flexible, hinting at compatibility with various IoT tools and mainstream programming environments,

enabling thorough testing and refinement of smart house systems before real-world implementation. However, further improvement in virtual sensor accuracy through machine learning is suggested.

Fog Simulators: Fog computing is an extension of cloud computing that brings computation closer to the data source which are IoT devices. Fog simulators help in modelling and evaluating the performance of fog nodes, their interaction with cloud and edge devices, and the overall system efficiency. The main features here are hybrid modelling (combining features of cloud and IoT), latency evaluation, local data processing, etc. They are designed to handle a mix of large-scale cloud resources and numerous edge devices. They are used to evaluate data offloading strategies, fog node placement, and hybrid cloud-fog architectures. Simulators of fog computing deal with the complexity of integrating both cloud and edge components, ensuring seamless data flow and processing. Some of the important and widely used fog simulators are: FogTorch [169], EmuFog [170], EdgeCloudSim [157], FogNetSim++ [171], iFogSim [15], and YAFS [172]. Listed below are a few examples of fog simulators:

- **MobIoTSim** [173] is a mobile IoT device simulator designed to facilitate the learning, testing, and development of IoT applications more efficiently. It can emulate IoT devices, generate real-time sensor data, and respond to messages using popular IoT protocols and data formats. Users can create IoT environment simulations with custom settings and connect the simulator to cloud gateways, like IBM Bluemix, for device management and notifications. However, MobIoTSim may not fully replicate real IoT device behaviour and lacks support for simulating network errors, recording and replaying cases, and connecting real devices. It also doesn't explicitly simulate device mobility. These additional features could enhance its realism and utility for IoT testing.

- **iFogSim** [15] is an academic toolkit focused on simulating resource management policies in IoT, Edge, and Fog computing environments. It evaluates policy impact on latency, energy use, and network congestion metrics. This toolkit can model complex fog environments, aiding the assessment of real-time IoT applications. iFogSim comprises four key components: application, network, resource

models, and a simulation engine. This resource is valuable for academic research into IoT, Edge, and Fog computing, particularly regarding resource management policy assessment. However, iFogSim does not support mobility.

- **Mercury** [174] is an open-source framework tailored for simulating real-time fog computing scenarios, emphasizing low latency, high data throughput, and 5G capabilities. It provides a detailed structural and behavioural model, offering insights into edge infrastructure and quality of service (QoS) optimization. Mercury is particularly suitable for data stream analytics applications and federated computation offloading. However, it is important to note that in the first approach of the scenario under study, cloud computing is not included in Mercury's fog model. The framework is primarily focused on investigating how location awareness and mobility impact the Quality of Service (QoS) and operational costs. This analysis aids in optimizing both the sizing and functioning of edge infrastructure necessary for supporting services that rely heavily on computation offloading.

Therefore, while there are overlaps, each type of simulator is tailored to address the unique challenges and characteristics of its respective domain. Some of the differences between these simulators are related to the communication layers or assumptions concerning IoT/Edge/Cloud/Fog. For example, fog simulators primarily focus on layers 4 to 7 of the OSI model Figure 3.1, as these are most relevant to the data processing, application protocols, and communication management functions of fog computing. On the other hand, edge simulators are only simulating layers 2 to 5 and layer 7 as Software Defined Networks Figure 3.1.

3.5 Applications of IoT Simulators

IoT applications cover a wide range of domains, each addressing specific challenges and requirements. In this section, we will discuss some of the current applications of the Internet of Things based on IoT simulators, including smart homes, healthcare, smart agriculture, transportation [175], smart cities, and smart industries. Smart services provided by the Internet of Things greatly benefit human life. Most activities can be performed anywhere, anytime and provide instant decision-making for efficient management. As

stated in [130], the most researched application is healthcare, followed by transportation and the environment. The other areas include utilities [176], military [177], safety [175], education [178], and financial [179].

- **Smart Cities:** IoT simulators are essential for urban planners and developers in crafting interconnected, intelligent urban environments. These simulators are instrumental in a range of applications, from traffic management to environmental monitoring, by leveraging data from diverse IoT devices such as vehicles, bicycles [14], and satellites [13], [11], [12]. In traffic management, IoT simulators use data from vehicles equipped with sensors to model and improve urban traffic flows, reducing congestion and enhancing road safety. Bicycle-sharing systems in cities benefit from IoT simulators that analyze data from GPS-equipped bicycles [14]. These simulators help in understanding cycling patterns, identifying high-demand areas, and planning the expansion of cycling infrastructure. Satellite imagery and data play a crucial role in large-scale environmental monitoring, offering a macro perspective on urban development, green spaces, and pollution levels. IoT simulators that incorporate satellite data [10] can predict weather patterns, monitor air quality, and assess the impact of urban planning decisions on the environment.

  Hence, all the IoT simulators tools in smart city applications excel in modelling traffic management systems to alleviate congestion and enhance public transportation efficiency, while also optimizing waste collection by predicting bin fill levels, reducing operational costs. They enable adaptive street lighting, adjusting brightness based on pedestrian and vehicular movement for energy savings. Additionally, they are crucial in environmental monitoring, tracking pollution levels and providing real-time weather updates for disaster preparedness. In the broader scope of smart city development, these simulators are adept at environment modelling, creating accurate digital representations of urban spaces, offering data analytics for informed decision-making, and facilitating emergency response planning [25, 180].

- **Smart Homes:** In the domain of residential technology, IoT simulators play a pivotal role in envisioning the future of smart homes. They are instrumental in modelling energy consumption, enabling homeowners to comprehend and minimize their energy usage, thereby reducing their environmental impact and utility costs. A sig-

nificant aspect of their application lies in testing home automation features, such as adaptive lighting and appliance control, for both user-friendliness and operational efficiency. This ensures seamless integration of these technologies into daily life. In the realm of smart home development, the focus areas include device integration, ensuring seamless connectivity and interoperability among various smart devices; user interaction, emphasizing intuitive interfaces and control mechanisms for an enhanced user experience; and robust security measures, which are essential for protecting sensitive data and maintaining resident privacy. These elements collectively contribute to the advancement of smart home technologies, optimizing both functionality and user engagement [141, 168, 181].

Additionally, these simulators are valuable in assessing home security systems and identifying potential vulnerabilities to safeguard against intrusions. For example, consider a smart building equipped with IoT sensors for monitoring temperature, occupancy, and security. These sensors are typically battery-powered to allow placement flexibility and reduce installation costs associated with wiring. An adversary targets these IoT devices with a battery drain attack, intending to disrupt the building's monitoring capabilities.

- **Healthcare:** The integration of IoT in healthcare represents a significant transformation, particularly evident in the use of simulators for various applications. These simulators are crucial in modelling remote patient monitoring systems, which are essential for accurately tracking vital signs and health metrics, with built-in mechanisms to trigger alerts in case of anomalies. They also play a vital role in optimizing elderly care systems, monitoring movement, activities, and health metrics to facilitate timely interventions. Additionally, medication adherence systems are tested for their efficacy in providing timely reminders and ensuring the consistency and accuracy of medication routines.

  In the broader context of healthcare simulation, key features include user interaction, which emphasizes intuitive interfaces for both healthcare professionals and patients to ensure efficient workflow and patient engagement. Security measures are of utmost importance to safeguard sensitive patient data and adhere to healthcare regulations. Data analytics is leveraged to extract meaningful insights from

extensive medical data, enhancing patient care and informing research. Emergency response simulations are integral for training and preparing healthcare personnel to handle critical situations effectively. Furthermore, the simulation of medical procedures and Electronic Health Records (EHR) is vital for the education and training of medical professionals and for the testing and refinement of healthcare systems, ensuring their efficacy and reliability [154, 182–184]. Consider a healthcare system where IoT devices are integral to monitoring medical equipment health, tracking patient care metrics, and regulating energy use. These devices constantly feed data into a central system, facilitating predictive maintenance of medical machinery and guaranteeing the highest standards of patient care. It is possible for attackers to result in rapid battery depletion or to falsify the sensor's battery status. Consequently, medical equipment could run without proper oversight, potentially leading to costly malfunctions or, more critically, compromising the safety of patients.

- **Agriculture:** Modern agriculture heavily relies on technology, and IoT simulators play a crucial role here. Precision farming applications can be modeled to ensure that variables like soil moisture and crop health are accurately monitored, leading to optimized irrigation and fertilization schedules. When stimulated, livestock monitoring can help design systems that track the health, location, and movement of animals, ensuring their well-being and reducing losses.

  Agriculture simulators prioritize device integration, environment modelling, weather simulation, data analytics, precision farming, pest and disease simulation, and resource management. Device integration allows farmers to monitor and control various agricultural equipment and sensors. Environment modelling involves creating digital representations of farmland to analyze and optimize conditions. Weather simulation aids in assessing the impact of weather patterns on crop growth. Data analytics helps in making data-driven decisions for crop management and yield optimization. Precision farming involves the precise application of resources like water, fertilizers, and pesticides, increasing efficiency. Pest and disease simulation assist in identifying and mitigating threats to crops. Resource management ensures efficiently utilising land, water, and other resources [9, 130, 185–187].

- **Industrial IoT (IIoT):** The industrial sector's efficiency and productivity gains with

IoT are significant. Predictive maintenance applications, when simulated, can help industries predict machinery wear and tear, reducing downtime and maintenance costs. Supply chain monitoring can be optimized to track goods and materials in real time, ensuring timely deliveries and reducing losses. Energy consumption optimization, when modelled, can lead to significant cost savings and reduced carbon footprints for industries. In the Industrial IoT domain, features centre around device integration, sensor and network simulation, data analytics, predictive maintenance, security, and scalability. Device integration facilitates the connection and management of industrial devices and equipment. Sensor and network simulation allow the testing and optimization of IoT sensor networks. Data analytics leverages industrial data for predictive maintenance, optimizing machinery performance and minimizing downtime. Security measures are critical in safeguarding industrial processes and sensitive data.

Consider a scenario in a smart industrial complex where IoT devices monitor machines, track inventory, and even manage access controls. These devices, often battery-powered to ensure uninterrupted service, are the lifeblood of operations, ensuring efficiency and safety. Within this complex, battery-operated IoT sensors are strategically placed on cargo containers to monitor temperature, ensuring that sensitive goods are stored optimally. These sensors relay real-time data to a central system, which makes necessary adjustments. Recognizing the critical nature of these sensors, cyber attackers target the battery management systems of these IoT devices. By exploiting vulnerabilities, they can either cause premature battery drainage or manipulate the sensors to report inaccurate battery levels. This could lead to goods being stored at incorrect temperatures due to perceived sensor outages or malfunctions, potentially resulting in significant financial losses [188]. In addition, scalability is crucial to ensure that IoT solutions can grow to accommodate expanding industrial operations [161, 189].

Discussion: When we compare IoT simulators from various domains Table 3.4, we observe that they emphasize various features differently. In smart home simulators for example, IoTSim-Osmosis, the primary focus lies on user interaction, security, and energy efficiency. Smart city simulators such as IoTSim-OsmosisRES are designed to excel

Table 3.4: Comparison of the Features of IoT Simulators Across IoT Applications

	FS-IIoTSim	IoTSim-Osmosis	IoTSim-OsmosisRES	ABS-SmartComAgri	MoSIoT
Smart City:					
Traffic Simulation			✓		
Energy Management		✓	✓		
Smart Home:					
User Interaction		✓			
Security Simulation	✓	✓			✓
Health Care:					
Patient Monitoring					✓
Electronic Health Records					✓
IIoT:					
Fault Detection	✓				
Remote Monitoring and Control	✓				
Agriculture:					
Pest and Disease Sim				✓	
Weather Simulation			✓	✓	

in environment modelling and traffic and weather simulation. On the other hand, patient monitoring, medical procedures, and EHR simulation are vital features of IoT simulators in the healthcare domain. However, simulators in smart homes and smart cities do not support these features. Smart home and smart city simulations may have similarities focusing on energy efficiency, but they can't be used in healthcare, since they don't support patient monitoring or medical procedures. In contrast, however, agriculture simulators prioritize weather simulation, precision farming, and pest and disease simulation, which is not able to support medical procedures feature in a healthcare domain. Lastly, the central features in the industrial IoT domain include predictive maintenance, fault detection, and remote monitoring and control. This means that IoT simulators don't support other features from other domains like pest and disease simulation.

3.6 Assessment Metrics for IoT Simulator Performance and Comparative Analysis

3.6.1 Evaluation Criteria for IoT Simulator Performance

Evaluating the performance of an IoT simulator involves assessing its ability to model, replicate, and predict the behaviour of real-world IoT devices and networks accurately and efficiently. The most critical evaluation criteria applied to measure the performance of an IoT simulator: scalability, realism, latency measurement, network topology flexibility, device behaviour modelling, protocol support, resource consumption, interactivity, extensibility, integration with other tools, performance metrics reporting, ease of use,

reliability and stability, mobility modelling, and environment modelling. In the next paragraph, we will discuss why these metrics are critical to evaluating IoT simulators.

Scalability is the most important criterion. As IoT networks can consist of many devices, simulators must handle large-scale simulations without performance degradation. So, they need to be able to simulate thousands to millions of devices, maintain performance under increased load, and provide consistent results in large-scale scenarios.

In **Realism** metric, a simulator primarily aims to mimic real-world scenarios. So, the closer the simulation is to reality, the more valuable its predictions. Hence, they need to accurately replicate device behaviours, real-world communication patterns, and potential device or network failures.

Moreover,**Latency** measurement is a critical aspect because, in IoT, timely data delivery can be crucial. So, simulators must measure and report latencies accurately. Therefore, IoT simulators must measure time delays in data transmission, account for network congestion, and provide insights into potential bottlenecks.

Also, **Network Topology Flexibility** is an important factor since real-world IoT deployments have diverse network topologies. Simulators should support and easily switch between different configurations. Hence, supporting various network shapes (star, mesh, tree, etc.), easing of reconfiguring topologies, and modelling of multi-tiered architectures are necessary features in IoT simulators.

Device behaviour modelling is another criteria, IoT devices can have varied behaviours based on their roles, energy sources, and capabilities. So, simulators should be able to simulate different states (active, sleep, error), model energy consumption patterns, and replicate device-specific functions.

Protocol support is another essential metric in evaluating IoT simulators. Different IoT solutions employ different communication protocols. Thus, a good simulator should be able to support a variety of IoT protocols. The inclusion of popular IoT protocols like MQTT, CoAP, and HTTP, the extensibility to add new protocols and the accurate modelling of protocol behaviours are important capabilities for IoT simulators.

In **The resource consumption** parameter, a simulator's efficiency can be gauged by the

computational and memory resources it consumes. So, low CPU and RAM usage, optimization for different hardware configurations, and minimization of unnecessary overheads are important features in IoT simulators.

For **The interactivity** metric, the real-time interactions enable users to modify scenarios on-the-fly and observe outcomes. So, IoT simulators should be able to have live modification capabilities, real-time visualization of network status, and instant feedback mechanisms.

In addition, **The extensibility** aspect is important. Simulators can be future-proofed by adding new features without overhauling their core. As a result, IoT simulators should have a modular architecture, be able to support plugins and extensions and be easy to integrate new features into. So, integration with other tools is a very important criteria, which is enhancing a simulator's capabilities by interfacing with external software tools. Hence, IoT simulators should support integration with data analysis tools and visualization platforms and be compatible with standard data formats.

Performance metrics is reporting another essential aspect to evaluate the IoT simulator. Users should get detailed insights into various performance aspects of the simulated network. Hence, simulators in IoT environments should support comprehensive reports on metrics like throughput, packet loss, energy consumption, metrics visualisation, and exportable data formats.

Moreover, **Ease of use** is an important factor since a user-friendly simulator ensures that users can focus on simulation tasks instead of attempting to master the software. So, it should have an intuitive graphical user interface (GUI), well-documented user manuals, and availability of tutorials or guides.

Reliability and stability are also important criteria of the simulator, as the simulator's consistent performance ensures that the users can trust the results. As a result, simulator tools must have minimal software crashes, robust error-handling mechanisms, and stable performance under various scenarios.

Furthermore, **Mobility modelling** is an essential factor in IoT simulations. Simulating movement and its impact on connectivity is vital for moving IoT devices. Simulators should support accurate modelling of device trajectories, the impact of mobility on connection stability, and the simulation of varied speeds and patterns.

Lastly, **Environment modelling** is another important criterion. Environmental factors can influence IoT device performance, so modelling physical obstacles, interference sources, and varying signal strengths based on environmental conditions are important features in IoT simulators. In conclusion, each criterion's depth varies based on the simulator's purpose, targeted users, and the specific domain of IoT it addresses.

In the previous part, we answered one of our research questions, which is the evaluation criteria applied to measure the performance of the IoT simulators. We identified a broad range of evaluation criteria. However, for our comparison analysis part, we have chosen to focus on some of these criteria that we believe are highly relevant to the current and emerging trends in IoT simulation. The next section compares the most recent IoT simulators based on some of the evaluation criteria. We focus on the criteria that are challenged in this domain.

3.6.2 Comparison of IoT Simulators

The most recent and commonly used IoT simulators and information are included in table 3.5. The table summarises the studied IoT simulators, focusing on their programming languages, availability, scalability, and support for SDN, IoT device mobility, energy models, and renewable energy sources.

Table 3.5: Overview Comparison of IoT Simulators

Simulation Platform	Language	OpenSource	Mobility	SDN	EnergyModel	Scalability	RES⋆
iFogSim	Java	✓	✗	✗	✓	✗	✗
MyiFogSim	Java	✓	✓	✗	✗	✓	✗
SimIoT	Java	✓	✗	✗	✗	✗	✗
IOTSim	Java	✓	Limited	✗	✓	✓	✗
CupCarbon	Java	✓	Limited	✗	✓	✓	✗
IoTNetSim	Java	✓	Limited	✗	✓	✓	✗
SimulateIoT	Java	✓	Limited	✗	✓	✗	✗
EdgeCloudSim	Java	✓	✓	✗	Limited	✗	✗
IoTSim-Edge	Java	✓	✗	✗	✗	✓	✗
IoTSim-Osmosis	Java	✓	✗	✓	✓	✓	✗
NB-IoT Simulator	Python	✓	✗	✗	✗	✓	✗
IoTSim-OsmosisRES	Java	✓	✗	✓	✓	✓	✓
SimulateIoT-Mobile	Java	✓	✓	✗	✗	✓	✗

⋆RES: Renewable Energy Sources

The column Language means the programming language used to code the experiments. The availability metric categorized simulators as an available (open-source) or a commercial. The mobility indicates whether the simulator has support for moving IoT devices. The SDN column (SDN) indicates simulator support for Software-Defined Networking. The energy Model feature identifies if there is already a model (for cost and/or energy)

implemented or if the user can add it via built-in features. The scalability metric measures the simulator's ability to scale and simulate a wide range of IoT devices and network sizes. The RES aspect focuses on the ability of the simulator to support renewable energy sources. A ✓indicates that the simulator reports that metric, while ✗indicates that the simulator does not.

From the table, we can see that all the studied IoT simulators are designed using Java, except for the NB-IoT simulator, which is designed using Python. And all our studied IoT simulators are open-source. Mobility support is a critical requirement for IoT applications and services, considering they are usually attached to users or devices that are moving between different access points at the edge of the communication infrastructure. Out of the simulators analyzed, only MyiFogSim, EdgeCloudSim, and SimulateIoT-Mobile support mobility. On the other hand, IOTSim, SimulateIoT, CupCarbon, and IoTNetSim can partially model an IoT environment where nodes are moving throughout the system. For example, IoTNetSim has limitations in supporting certain sensor types due to the complexity of modelling their mobility.

Software-Defined Networking (SDN) presents an effective approach for managing dynamic network settings, particularly in scenarios involving numerous connected devices and varied applications. SDN employs software-based controllers or application programming interfaces (APIs) to interact with the underlying hardware infrastructure, guiding the flow of network traffic [141]. Among the simulators that were analyzed, IoTSim-Osmosis and IoTSim-OsmosisRES are the only ones that support SDN. As part of these simulators support for SDN, it provides mechanisms for SD-WAN (Software-Defined Wide Area Networking) networking. In these simulators, IoT, edge, and cloud ecosystems can be integrated, and the edge can be virtualized as well as incorporating SDN-aware infrastructure. Similarly, a cloud data center can include hosts that are virtualized as well as networks that are aware of SDN. These three simulators holistically integrate all these components in order to provide researchers with the necessary support to evaluate the performance of an end-to-end IoT application using the osmotic computing concept. Using SDN, network switches can be centrally managed with fine-grained traffic management capabilities. Moreover, network elements can be dynamically programmed and controlled through a central controller [25].

For energy modelling support, iFogSim, IOTSim, CupCarbon, IoTNetSim, SimulateIoT,

IoTSim-Osmosis, and IoTSim-OsmosisRES are supporting energy models. While these simulators offer some form of energy modelling, their depth and focus vary. CupCarbon, focusing on wireless sensor networks in smart cities, offers detailed energy modelling for sensor nodes. iFogSim and IOTSim, on the other hand, provide energy models more suited for fog and cloud computing environments, respectively. EdgeCloudSim authors' mentioned the support of energy consumption models for mobile and edge devices, as well as cloud datacenters, as a feature that needs to be developed. In contrast, the IoTSim-Osmosis framework evaluates the energy consumption of IoT applications holistically, taking into account the energy usage of different components in the edge-cloud continuum. It is important to note that in one of their scenarios, they neglected the battery consumption of the IoT devices due to static data generation. In contrast, IoTSim-OsmosisRES support only battery managment for IoT devices without including the battery support of edge devices. Despite this, IoTSim-OsmosisRES supports energy modelling for RES (renewable energy sources), such as photovoltaic panels, and enable the evaluation of AC (Autonomic Computing) algorithms for osmotic computing. AC can help manage the available renewable energy sources efficiently. They are able to assess various parameters, such as the level of solar radiation, usage of the RES, usage of low-emission sources, and the IoT device battery capacity. The framework also enables the evaluation of the impact of different adaptation algorithms on the performance of the IoT system in terms of utilising renewable energy sources. As a result, IoTSim-OsmosisRES is the only IoT simulator that supports RES.

There are limitations in the ability to simulate large-scale environments with the IoT simulators under study, including iFogSim, SimIoT, SimulateIoT, and EdgeCloudSim. The other IoT simulators studied support scalability at a variety of levels. SimulateIoT-Mobile is designed to model and simulate complex IoT environments, but no information or testing exists. However, IoTSim-Osmosis is designed to handle the heterogeneity of integrated edge-cloud environments and the complexity of IoT applications. The scalability of IoTSim-Osmosis is demonstrated in terms of time and memory consumption, but no large-scale scenario has been tested using this simulator. For IoTSim-OsmosisRES, using historical data to realize the simulation environment can help improve the framework's scalability by reducing the simulation's computational overhead. Also, using decentralized management can help improve the framework's scalability by distributing the manage-

ment of the IoT system across multiple nodes. On the other hand, NB-IoT Simulator provides the clearest statement regarding scalability among the studied IoT simulators. It is specifically designed to handle large-scale networks spanning over an entire city. Also, it can model hundreds of base stations and thousands of devices and is based on realistic databases obtained from smart city open data projects.

These existing IoT simulators present a variety of methodologies for evaluating the performance of IoT-based applications in various environments, including edge computing and fog computing. Various metrics and parameters are used in these simulators to measure aspects such as energy consumption, scalability, and overall system performance. IOTSim and IoTSim-Osmosis focus on metrics like VM computing cost, network cost, battery consumption, and the dynamics of IoT devices within osmotic and conventional computing environments. SimulateIoT-Mobile emphasizes the impact of jitter on Quality of Service (QoS), indicating the importance of network quality in IoT applications. IoT-NetSim and others like NB-IoT Simulator provide insights into the simulation of network behaviors, battery usage, and the efficiency of data transmission among IoT devices.

Scalability testing across these simulators varies widely, from manipulating datacentre configurations and VM numbers in IOTSim to evaluating the performance of IoT applications based on the number of IoT devices and their operational dynamics in IoTSim-Osmosis. Energy consumption parameters also differ, with simulators like iFogSim comparing deployment on fog devices versus cloud data centers, while RES focuses on the utilization of renewable energy sources and energy-efficient data processing algorithms. IoTSim-Edge, SimIoT, and EdgeCloudSim each bring unique perspectives on the simulation of edge computing environments, highlighting the challenges of managing limited resources such as computational power, battery life, and network bandwidth. CupCarbon, on the other hand, delves into the simulation of network behaviors and energy consumption, emphasizing the effects of sensor connectivity on battery life.

Given the heterogeneity in the datasets used by these simulators and the diversity of metrics measured, it becomes evident that making a quantitative comparison between them is not straightforward. Simulators have been designed with a specific focus on IoT applications across a variety of computing paradigms. As a result of the differences between the simulated environments, as well as the varying approaches taken to measure performance and efficiency, it is difficult to establish a uniform basis for comparison. Since

each simulator provides valuable insights into certain aspects of IoT system performance, a direct quantitative comparison between them is not possible due to the differences in their datasets and metrics. It is clear from this diversity that there is a wide variety of considerations in IoT research and development, which highlights the need for generic simulation tools Section 3.7.2.1 to provide more versatile and integrated simulation solutions in the future.

while several IoT simulators are available, to the best of our knowledge, we have not found an IoT simulator that deals with security issues around the Internet of Things, specifically batter-draining attacks. This limitation in existing simulation capabilities strongly motivates the need for the development of a secure simulation framework Section 3.7.2.2, which would focus on the security aspects of IoT, especially concerning energy-related attacks. By incorporating the simulation of battery-draining attacks and other security threats, researchers and developers can gain valuable insights into the resilience of IoT systems against malicious actions, enabling the design of more robust and secure IoT solutions.

3.7 Challenges and Future Trends

3.7.1 Current Challenges

According to our analysis review, most IoT simulators are designed to support a particular scenario with their abstractions. Hence, no generic IoT simulator can simulate any scenario from any domain in the IoT world. Moreover, many challenges still need to be addressed in the field of Internet of Things simulation.

Mobility, an essential feature of IoT, is not comprehensively represented in many IoT simulators. Most tools offer limited mobility modelling capabilities, underscoring the urgent need for advanced simulators. These advanced simulators should be capable of accurately depicting the dynamic nature of IoT topologies, especially for complex simulation environments such as smart vehicles, wearables, and individuals on the move. Given the rising prevalence of mobile IoT devices, future simulators are anticipated to incorporate sophisticated mobility models. These models might leverage real-time data and predictive analytics to emulate intricate mobility patterns and scenarios. The development of advanced simulators that can more accurately and comprehensively represent

IoT deployments' dynamic and mobile nature is urgently required, perhaps integrating real-world mobility data or leveraging machine learning to predict mobility patterns.

Additionally, the recent integration of **SDN** into existing simulators highlights a gap in current research. As SDN and Network Function Virtualization (NFV) become integral to IoT networks, simulators must natively support these technologies, offering a holistic view of network dynamics and management. The absence of supporting these technologies in IoT simulators may prevent valuable insights into IoT paradigms centred on SDN. Given the increasing importance of SDN in modern network environments, researchers should prioritize its comprehensive representation in simulators, possibly by developing modules or plugins that can be integrated into existing tools.

Energy modelling is crucial to global sustainability efforts, yet many IoT simulators currently lack comprehensive energy models. This absence is particularly concerning given the pivotal role of energy consumption and management in real-world IoT deployments. Without these models, simulators may produce results that fail to accurately represent the energy dynamics of actual IoT devices and networks. Consequently, there's a critical need for future simulators to not only focus on energy consumption but also on generation, storage, and optimization, with a particular emphasis on renewable energy sources. To the best of our knowledge, IoTSim-Osmosis-RES is the only IoT simulator that can simulate floating weather conditions, renewable energy sources and provide easily extendable system that enables researchers to specify their own virtual machines and power management policies.

The **scalability** issue remains unresolved for a significant subset of simulators. In the IoT world, environments are growing fast, and if the support for large-scale environments in IoT simulators is limited, it could negatively impact fundamental research efforts, including those targeting large-scale, urban, or even transcontinental IoT frameworks. To address scalability concerns, researchers should focus on enhancing the scalability of simulators, possibly by applying distributed computing techniques or cloud-based platforms, allowing for the simulation of vast IoT networks spanning cities or even continents. Moreover, integrating real-world data streams, especially those emanating from smart city infrastructures, into simulations presents a complex set of challenges. Smart city infrastructures continuously generate vast amounts of data, which not only vary in volume and velocity but also in format, given the diverse sources like sensors, cameras, and traffic

systems. A significant challenge is ensuring that simulators can seamlessly process, interpret, and handle this heterogeneous data. Moreover, the real-world data can often be incomplete, thus requiring robust preprocessing and validation mechanisms within the simulators. There are also temporal and spatial intricacies to consider, as data often comes time-stamped and geo-tagged, requiring simulators to manage both dimensions adeptly. Challenges also arise in real-time data integration, efficient storage and retrieval mechanisms, contextual interpretation of data, and managing feedback loops where simulations might influence and alter real-world operations. Addressing these complex challenges is essential for enhancing the realism and applicability of future IoT simulations.

With the development of **Vehicular Cloud Computing** (VCC), vehicles interact, collaborate, and can leverage their collective computational power more effectively. By transforming vehicles into nodes of a dynamic cloud, VCC harnesses their computational, storage, and sensing capabilities to offer real-time, context-aware services. This significant change, where vehicles consume, produce, and maintain content, challenges the traditional cloud infrastructure's role. Instead, it consists of a self-organized, adaptive cloud formation that evolves based on the proximity and availability of neighbouring vehicles.

Given this, an important future trend is beginning to emerge for IoT simulators: the need to simulate vehicular environments accurately. Using IoT simulators along with traffic simulators to simulate Vehicular Ad-hoc Networks (VANET) environments allows smooth integration of mobility, IoT devices, heterogeneity, and battery management in highly heterogeneous and dynamic environments. It is to our knowledge that no research has been conducted on this topic.

The limited exploration of this intersection between IoT and VANET simulation indicates an urgent need for more in-depth and diverse investigations. Given the significant potential of VANETs and their increasing relevance in modern urban settings, further research is essential. An expanded research scope would both validate the existing study and offer a broader perspective, capturing details potentially not captured by one study. Given the limited research integrating IoT simulators with traffic simulation for VANET environments, there's a clear necessity for more extensive investigations in this domain.

The integration of real-world data into IoT simulators raises important issues related to

data security and privacy. *To the best of our knowledge, no simulator supports simulating attacks such as battery drain, as a simulator has to support battery and communication models. There is a lot of research on detecting attacks on log files that are not simulated in real time with an actual attacker using a real communication infrastructure. However, as the Internet of Things environment expands, several security threats are introduced, requiring simulators to simulate these vulnerabilities accurately. The very nature of data from smart city infrastructures and other IoT deployments is often sensitive. This data can encompass personal information about citizens, traffic patterns, energy consumption metrics, and more. Unauthorized access or breach of such data can have significant consequences regarding individual privacy and broader infrastructural operations.*

To address this challenge, researchers will need to invest more time and effort in developing IoT simulators that are focused on these issues. Simulators should be designed with built-in encryption modules to ensure data security during transit and while in storage. Additionally, as simulators integrate real-world data, data anonymization and sanitization processes should be implemented to prevent the loss of personally identifiable information. Beyond just data protection, simulators should also be equipped to emulate various threat scenarios. Researchers can learn how real-world IoT deployments might respond to security threats by simulating potential attack vectors, vulnerabilities, and corresponding mitigation strategies. This proactive approach, where potential breaches are identified and addressed in a simulated environment, can pave the way for more resilient IoT systems in the real world.

Moreover, while the different applications have distinct feature priorities, *it is important to acknowledge that no single simulator identified in the literature comprehensively addresses all these features across all domains. The unique requirements of each domain present a challenge for simulator developers, making it essential for researchers to carefully select or develop simulators tailored to specific use cases within these diverse application areas. Addressing this limitation could pave the way for more versatile and integrated simulation solutions in the future. This limitation in existing simulation capabilities strongly motivates the need for the development of a generic simulation framework Section* 3.7.2.1.

3.7.2 Future works: Conceptualizing a new simulator

3.7.2.1 Generic Simulator by Supporting OSI layers:

The IoT simulators may address aspects related to different OSI layers Figure 3.1, but each simulator does not cover all layers across all application domains comprehensively. This limitation is due to the vast diversity of requirements, protocols, and technologies across different IoT applications, as shown in Figure 3.1, which makes it challenging to develop a one-size-fits-all simulator. For instance, smart city IoT simulators like IoTSim-OsmosisRES, in order to accurately simulate traffic and energy management, cover a broad range of the OSI model. While the main focus might be on higher layers: Application, Presentation, Session, and Network due to their direct relevance to application-specific simulations, but the physical layer is not simulated in these simulators. On the other hand, simulators focusing on agriculture may excel in simulating the Application layer's pest and disease simulation and weather simulation but lack the depth to simulate lower layers like the Data Link or Network layers, which are crucial for detailed network communication simulations required in smart city applications.

However, despite the varying focus on these features within each domain and supporting different OSI layers, it's important to note that no simulator identified in the literature can comprehensively address all these feature requirements across all domains. In other words, no existing IoT simulator can simulate any scenario from any domain in the IoT world. It is possible to move from one domain to another, as in IoTSim-OsmosisRES [25] that moved from the smart home domain to the smart cities domain. But still, it is not able to simulate a scenario in the healthcare domain, which requires adding a new agent to it. This is because there is no existing IoT simulator that is able to simulate all the OSI layers. This gap highlights a significant challenge in the field of simulation, as there is currently no single platform able to simulate any scenario across these diverse IoT applications.

Moreover, the specialized focus of each simulator on certain OSI layers determined by its target application domain reveals a fundamental gap in the field of IoT simulation: the lack of a versatile, multi-layer, cross-domain simulator. Hence, this highlights a significant research opportunity in the field of simulation technology: developing advanced simulators capable of dynamically adapting to simulate any given scenario in any IoT application

by supporting and simulating all OSI layers. Researchers and practitioners should be aware of this limitation when selecting simulators for specific use cases and consider the development of more versatile simulators as a valuable area for future work in the field of simulation research.

This can be achieved by considering a simulator that supports reliable IPv6 simulation as [43]. An OMNeT++ simulator is constructed from modules that communicate using message passing. An hierarchical system structure can be created by nesting modules, which makes it possible to construct complex systems from simpler components in a clear and manageable manner. So, the framework is designed to be easily extended with new protocols or devices. The users are able to create their own modules or modify existing ones to meet their specific requirements. Hence, since every block in OMNET++ is simulated, it is possible to combine different modules in order to have a new simulator that is able to simulate different scenarios in IoT applications. There are existing works that combined modules with OMNET++ such as [190, 191]. As a result, Omnet++ acts as a platform that schedules events based on specific distributions, and its modular nature makes it possible to advocate creating a simulator that simulates a wide range of Internet of Things scenarios by combining different modules. Achieving this would represent a significant step forward in simulation technology, allowing researchers, developers, and practitioners across the IoT ecosystem greater flexibility and utility.

3.7.2.2 Secure Simulator by Cybersecurity Enhancements:

Based on the current version of IoTSim-OsmosisRES [25], it is notable that the simulation capabilities are limited, particularly in terms of simulating battery-draining attacks and extending the simulator with new agents, such as attackers and cyber-security diagnostic tools. This simulator cannot support direct communications between IoT devices, as it only facilitates interactions between IoT devices and Cloud nodes via Edge nodes. This architectural limitation is significant given that many current battery-draining attacks involve direct communication between IoT devices in IoT environments. By engaging in overly extensive, unnecessary communication requests or other energy-consuming activities, attackers target another IoT device to drain the battery. To address this, it is necessary to set up a whole IPv6 infrastructure in the simulator, which is currently not available. This gap in functionality means that IoTSim-OsmosisRES cannot effectively

model complex attack scenarios or provide a thorough understanding of security attacks, especially those involving direct communication between IoT devices, which are increasingly common in modern IoT networks. A fundamental understanding of the security of IoT networks can be obtained only by significantly revising such a simulator: due to the simulator's limited interaction capabilities and the missing IPv6 infrastructure, its applicability is currently constrained. So, the new secure simulatoris needed that can address this limited interaction capabilities, therefore, support cyber-security detection algorithms.

Due to the importance of simulating cyber threats in different IoT scenarios Section 3.5, such a simulator should also be able to mimic non-mobility-based IoT nodes. This should be trivial, as this boils down to assuming that such devices will never change their position. In summary, the use case involves deploying edge gateways to detect and mitigate IoT device battery-draining attacks. This is motivated by the need to protect the longevity and functionality of these devices, which are increasingly integral to the operation of modern smart environments.

3.8 Conclusions

It can be concluded that most existing IoT simulators are designed specifically to support specific scenarios with their unique abstractions, indicating a notable gap in having a generic IoT simulator capable of simulating any scenario in different IoT applications. Furthermore, through our systematic analysis, we identify a critical gap in the current IoT simulation frameworks: that is, no simulator supports simulating attacks especially battery depletion attacks, which are growing in importance in IoT systems. Furthermore, there is no study integrating IoT simulators with traffic simulation for simulating VANET environments, indicating that a thorough investigation in this area is necessary. In addition, considering that only one IoT simulator support the integration of renewable energy sources, as we emphasize the importance of environmental sustainability, this finding is particularly noteworthy. Sustainable practices have become increasingly important to the global community, and integrating renewable energy sources in IoT simulations has become a critical need that has remained largely unmet.

As a result of this study, it becomes clear that advanced, versatile platforms for IoT

simulation are urgently required. Besides bridging the current gap by supporting simulations across all OSI layers, such platforms should also provide the capability to simulate a wide range of cyber threats, including battery depletion attacks. Given these challenges, there's an urgent demand for tools that can adapt to the continuously evolving requirements of the Internet of Things. Addressing these issues will enable more comprehensive and accurate simulations and open a variety of opportunities for researchers to make significant contributions to the field. The future development of IoT simulators is likely to be dynamic and multifaceted, reflecting the complexities and opportunities of the rapidly advancing IoT landscape. The existence of this gap highlights the urgent need for future developments in IoT simulators to incorporate renewable energy models, aligning them more closely with environmental sustainability objectives.

4

Platform Toward Integrating IoT Simulations with Traffic Simulations: SUMOtoOsmosis

Contents

4.1 introduction

Because real-world performance testing is very expensive and not comprehensive, today's valuations are based on analysis and simulation through traffic simulators. However, the simulation relies on a mobility model that represents the movement pattern of mobile users, including their location, velocity, and acceleration over time. A realistic mobility model should consider the characteristics of a real-world scenario either by taking a real-world map or by considering Satellite images of Google Earth to simulate a realistic network [192]. In VANET network vehicles are equipped with a radio interface and communicate directly with each other and the road traffic infrastructure. Though, the realistic simulation for VANETs is a challenging task because that network has high mobility of vehicles, for this issue VANETs research have the requirement of a group of simulators to analyze the traffic and networks. The purpose of using these simulators is to achieve the simulation results more precisely and accurate as a real-time system [193]. Research is being carried out in the field of VANET such as analyzing data dissemination in VANETs, identifying and studying routing protocols in VANET in terms of highest delivery ratio and lowest end-to-end delay, etc. The issues of Security and Privacy also demand great attention [194]. The study of Mobility Models and their realistic vehicular model deployment is a challenging task. To investigate the impact of traffic information systems, two options are available. First, traffic simulators can be extended with application code and a simplified model for wireless communication. Second, existing network simulators can be coupled with existing traffic simulators. The coupling of existing and well-known simulators is favored as it is believed that the wireless communication characteristics significantly influence data transfer, and an oversimplified transmission model may lead to flawed results [195].

4.2 Motivation and Limitations

The significant limitation of traffic simulators is the complexity of their calibration and the commercial license that needs to be acquired, which could be significantly onerous in some cases. For these reasons, a large community started to work on the development of free vehicular mobility models tailored to network research and that provide interactions

with network simulators. In contrast to research on simulators and their coupling, only few research results exist that verify positive effects of VANET communication.

All existing VANET simulators combine a traffic simulator and a network simulator; however, according to the literature review made in 2, no work has been done on coupling traffic simulation with IoT simulation. Using IoT simulators in conjunction with traffic simulation allows more accurate evaluation of the energy efficiency of vehicles. An examination of different battery consumption patterns may assist in determining the most effective strategy for minimizing the amount of energy consumed by network communications.

As a result, the hypothesis is that while coupling traffic simulators and network simulators has been effective for simulating VANET environments, integrating traffic simulators with IoT simulators could enhance the simulation's realism and utility. There are tools that currently couple traffic simulators with network simulators for VANET simulations. The question of whether the IoT simulator, *IoTSim-Osmosis* [196], can be coupled with the road traffic simulator *SUMO* [197] for simulating VANET scenarios is posed.

Hence, this chapter aims to fill the gap in the literature by investigating the possibility of bridging a non-mobility IoT simulator with a traffic simulator. Additionally, how this could be accomplished and what is required to achieve it will be examined. The proposed framework is designed to meet these two goals: coupling IoTSim-Osmosis simulator with traffic simulator (SUMO), and enhancing IoTSim-Osmosis capabilities to be able to simulate moving IoT devices. IoTSim-Osmosis, which was at the time of working on this, the most recent IoT simulator available as well as the first simulation framework that enables unified modelling and simulation of complex IoT applications over heterogeneous edge-cloud environments, provides all the necessary functionality to perform this coupling simulation more accurately.

The remainder of this chapter is structured as follows. In alignment with the objective to integrate an IoT simulator with a traffic simulator, Section §4.3 offers an overview of the most common traffic simulators, while Section §4.4 provides a thorough analysis of IoT simulators. After that, Section (§4.4.1) outlines the advantages that are obtain from the proposed coupling idea, and Section (§4.5) discusses the challenges that must be addressed in order to achieve this research's objective. Next, Section (§4.6) describes

Table 4.1: Comparison of Traffic Simulators.

	License	Vehicle Model	Space Domain	HeterogeneousTraffic Behavior
VISSIM	Commercial	**Micro**	Continuous	✗
AIMSUN	Commercial	**Micro**, Macro, Meso	Discrete, Continuous	✗
FreeSim	**Open Source**	**Micro**, Macro	Discrete, Continuous	✗
MATSIM	**Open Source**	**Micro**	Continuous	✗
HeteroSim	**Open Source**	**Micro**	Discrete	✓
SUMO	**Open Source**	**Micro**	Discrete, Continuous	✓

the methodology for developing the proposed simulator *SUMOtoOsmosis*. Section (§4.7) describes the experimental setup and results for the proposed simulator. Finally, the conclusion and future directions are presented in Section (§4.8).

4.3 Traffic Simulators

Traffic simulators, which are the depiction of real-world scenarios, are called *mobility generators* and contain the simulated infrastructure and event information, such as vehicle speed, type, origin, destination, arrival time, arrival rate, maximum density, number of lanes, speed limits, capacity, intersection type, queuing, service distribution, service rate, traffic signs, and location. The ability of traffic simulators to model complex traffic ranging from a single intersection to specific regions at a city-wide scale will provide valuable insights into traffic analysis and modeling [198]. Commercial and open-source traffic simulators can offer differing levels of granularity. These traffic simulators can, therefore, be categorized into three categories, namely macroscopic, microscopic [199, 200], and mesoscopic, depending on their level of representation [198]. The first vehicle mobility model is *Macro(scopic)*. This mobility pattern is more concerned with the motion constraints like roads, crossroads, traffic lights, traffic density, traffic flows, streets, and vehicle distribution. The second one is *Micro(scopic)*, which is based on the constraints such as neighbouring vehicles, pedestrians, types of cars, and vehicle behaviour. The third vehicle mobility model is *Meso(scopic)*, it fills the gap between microscopic and macroscopic models. The mesoscopic models simulate aggregate vehicle flow based on individual vehicle behaviour rules. The following paragraphs will discuss the most commonly used traffic simulators listed in table 4.1.

VISSIM is a commercial tool for simulating microscopic traffic [201]. It can model integrated roadway networks and modes, such as buses, light rail, heavy rail, trucks, pedestrians, and bicycles. VISSIM implements the component object model (COM) interface, which allows users to develop and deploy custom tools in VISSIM using C++, Visual Basic, or Python [202]. Internally, the model includes two elements that interact via an interface: a *microscopic* traffic flow model simulating car following as well as lane changing, and a signal state generator updating the signal status for the next simulation step. The simulator can total delay, stopped-time delay, stops, queue lengths, fuel emissions, and fuel consumption.

AIMSUN is a commercial tool [203]. AIMSUN simulating real-world traffic scenarios in an urban network for the purpose of building and validating traffic structures, public transportation networks, and new transportation infrastructure. This can simulate cars following others, lane changing, and gap acceptance models. These behaviours are continuously modelled throughout the simulation. It is capable of identifying many different types of vehicles, while traffic demand can be modelled using traffic flows and turning ratios, as well as origin-destination matrices and route selection models. AIMSUN copes with many roadway types on arterials and freeways networks; it can also model the impact of incidents and Variable Message Signs (VMS). AIMSUN is combined with a simulation environment called GETRAM that contains four elements: a traffic network editor (TEDI), a network database, a simulation module, and an application programming interface [204].

FreeSim [205] is a *open-source* simulator including both *macroscopic* and *microscopic* features. FreeSim acts as an intelligent transportation system collecting data from individual vehicles travelling along a freeway. Each vehicle runs as a separate thread and communicates with the simulator in a manner of their choosing. During freeway segments, vehicles can transmit their current speed and location to a central server. According to the current speeds, they can also request the shortest or fastest path to their destination. As soon as the vehicles receive an updated path, they can decide whether or not to modify their current route. The graph data structure in FreeSim can express different freeway systems. The distances between freeway segments (edges) and the nodes (vertices) are needed to represent a freeway graph. The freeway system will be displayed in a GUI using the nodes and vertices based on the distance and speed of each segment. The extensibility

provided by FreeSim makes it possible for programmers to create customary algorithms for running on the simulator's data. The edges in the graph can be updated during a simulation using either synthetic or real data, such as data collected by a transportation organization. The system may then monitor the progress of individual vehicles as well as the overall state of the freeway system to see how travel times and congestion vary with the updated data. The progress of individual vehicles or the status of the freeway system can be tracked to see how the travel times and congestion change with the updated data.

MATSim [206] was built to simulate large-scale agent-based transportation systems. Several modules, be they ready-made or customised, make up the framework, which can be used separately or together. It can simulate 24 hours traffic flow data in a few minutes thus enabling the simulation on real-world scenarios.

HeteroSim is a microscopic traffic simulator [207] for heterogeneous traffic simulations handling mixed traffic flows such as cars, motorcycles, bicycles, buses, and trucks. It can satisfactorily replicate *heterogeneous traffic behaviour* when vehicles move without lane discipline on roads. This simulator can be used to evaluate such traffic management measures as segregating vehicles on major urban roads, providing buses and bicycles with exclusive lanes, and banning specific categories of vehicles from certain stretches of road. It can only simulate specific road sections while achieving real-time accuracy.

SUMO is a free and open-source simulation tool [197]. In 2001, the Institute for Transportation Systems developed it at the German Aerospace Centre. Besides being highly portable, this simulator also simulates vehicular traffic on a *microscopic* level. With its lane-changing capabilities and large road network support, it can simulate both time-*discrete* and space-*continuous* vehicle movements on multi-lane streets. In addition, accident scenarios can be simulated. SUMO can set any traffic scenario as it is an intermodal traffic simulator considering public transportation, traffic road networks, and pedestrians. This comes with multi-modal traffic and automated driving. Furthermore, the traffic management feature allows the modelling of vehicle detection loops and video detectors to manage and control traffic through traffic lights, monitoring vehicles' behaviours and adjusting traffic parameters such as vehicles' speed limits. This extends the set of emulable *heterogeneous traffic behaviour* from HeteroSim. SUMO can import road networks from other traffic simulators such as VISUM, Vissim, and MATsim.

Table 4.2: Summary Comparison of IoT Simulators

Simulator Name	SDNSupport	IoTDevices	Mobility	EnergyManagement
iFogSim	✗	✓	✗	✗
MyiFogSim	✗	✓	✗	✗
SimIoT	✗	✓	✗	✗
IoTSim-Edge	✗	✓	✓	✗
IoTSim-Osmosis	✓	✓	✗	✓

Table 4.1 provides a comparison between the most common used traffic simulators based on four features: license type, vehicle model, space domain, and heterogeneous traffic behavior. For license type, VISSIM and AIMSUN are commercial software. FreeSim, MATSIM, HeteroSim, and SUMO are open sources. For vehicle model feature, all the compared simulators are micro simulators except AIMSUN and FreeSim. FreeSim includes both macroscopic and microscopic features and AIMSUN includes all of the three: macroscopic, microscopic, and mesoscopic. For space domain feature, VISSIM, MATSIM, and SUMO are continues models. However, HeteroSim is a discrete system.

4.4 IoT Simulators

Based on the scope and the degree of architectural layer coverage, IoT simulators can fit into three categories: *i)* full stack simulators, providing end-to-end support for all the constituents of the IoT framework, *ii)* big data processors in the context of IoT data, and *iii)* network simulators [208].

Table 4.2 presents a summary of the IoT simulators that were studied in the literature review chapter 2, focusing on their main characteristics. The following features have been taken into account in this comparison: (a) SDN support, (b) IoT devices support, (c) mobility, and (d) IoT device battery support. As illustrated in the table, all simulators support IoT devices. It can be seen that only one IoT simulators support IoT device mobility, which is IoTSim-Edge. On the other hand, IoTSim-Osmosis supports SDN, IoT devices, and IoT device battery.

4.4.1 Advantages from Integrating Traffic Simulators with IoT Infrastructure

Modelling real use case scenarios where each IoT device is embedded in a VANET vehicle allows a complete simulation of edge computing in an efficient simulation environment while providing a single data storage and analysis point where the network edge are closer to the IoT devices. Syncretism between IoT and traffic simulators enables the orchestration of cloud, edge and/or SDN-based networks and the abstraction of complex IoT applications generated by vehicular traffic. As a result, each component of the IoT applications is covered holistically. Through the use of osmotic computing, researchers can evaluate the performance of these applications from end-to-end.

As VANET environments contain a large number of connected vehicles and they are in highly heterogeneous form, Software-Defined Networking (SDN) is a suitable solution for dealing with dynamic network environments, especially those with a large number of connected devices and heterogeneous applications. SDN is a networking method that uses software-based controllers or application programming interfaces (APIs) that can communicate with underlying hardware infrastructure and direct network traffic [196]. Using SDN, network switches can be centrally managed with fine-grained traffic management capabilities. With SDN, network elements can be dynamically programmed and controlled through a central controller [209].

4.5 Challenges in Linking Traffic Simulation with IoT Simulation

The literature review shows no simulator bridging IoT and traffic information, and the previous section proved the usefulness of doing so. In order to overcome this limitation, it is necessary to address the challenges listed in the following paragraphs.

Different Infrastructures Compared with other types of networks, current VANET simulators do not provide networking infrastructure supporting mobile nodes that are free to move in any direction according to the prescriptions of a traffic simulator. Also, traffic simulation is based on the speed and movement of the nodes: vehicles are moving in

disparate directions and looking for routes in the city's layout [210]. So, the very high mobility of nodes, limited energy reserves, and lack of central management are challenges in traffic simulation. On the other hand, IoT simulation is based on a graph of multi-layered architecture (comprising IoT, edge, cloud and SD-WAN), focusing on the management between the layers [196]. Network simulation considers power in IoT devices rather than their movements. So, there is a challenge in linking these two different infrastructures in one framework.

Complexity of traffic environments Due to the heavy flow of vehicles and the high heterogeneity of static and dynamic characteristics, traffic networks have very complex topologies. Vehicle speeds and locations vary widely in heterogeneous traffic conditions. Moreover, due to barriers on nearby highways, vehicles might be potentially disconnected. Hence, the difficulty of traffic simulation is the variation in geographic temporal aspects of resource function calculations over time. Additionally, vehicles in heterogeneous traffic do not follow lane discipline; therefore, they interact with vehicles at the front and sides. As a result, traffic analysis becomes very complex [210].

Simulation Accuracy Recent years have seen a rapid increase in new technologies emerging in VANETs. These new technologies, from self-driving vehicles to 5G, enhance the bandwidth, latency, and reliability of VANET applications, which enables their adoption in the real world [210]. For example, 5G networks are more susceptible to weather conditions than their predecessors. In this sense, extending IoT simulators to encompass these new conditions could be challenging. As a whole, the quality of traffic simulation strongly depends on the accuracy of underlying models. Noticeably, the degree of realism has increased over the last years, and some traffic simulators now include modules that incorporate signal attenuation, different antenna patterns, and environmental diffraction. So, by linking IoT simulators with traffic simulators, IoT simulators must include modules that support all these new technologies.

4.6 Methodology

The purpose of this section is to describe the approach for developing VANET simulation framework that merges the use of traffic simulators with IoT simulators. The proposed

framework, *SUMOtoOsmosis*, leverages state-of-the-art simulators from both domains to provide a comprehensive and accurate evaluation of vehicular energy efficiency and communication patterns. Thus, it incorporates well-known models for road traffic microsimulation with a comprehensive selection of models of network protocols. For traffic simulation systems, there are various software have been developed. But for heterogeneous traffic behaviors, as in [198], authors stated that among the 29 traffic simulators analyzed, only two could simulate heterogeneous traffic behaviour: HeteroSim, the discrete simulation system, and SUMO, the continuous one. Results in [211] showed that continuous models are more accurate than discrete models, as the former are guaranteed to find optimal solutions. So, with SUMO, simulation results are generally considered to be more accurate. Specifically, given sufficient time points, continuous-time models are guaranteed to find the optimal solution. Furthermore, since HeteroSim was developed as an academic initiative, it can only simulate a part of a road with close to realistic results. As a result, HeteroSim is not scalable. In light of SUMO's features and capabilities, the platform has been selected as a suitable platform for achieving the objectives of this research.

On the IoT simulators side, various frameworks have been introduced for modelling and simulating IoT environments and evaluating resource management. From the studied IoT simulators in table 4.2, IoTSim-Osmosis is the only IoT simulator considering energy management, while others can neither simulate energy sources, nor provide an easily extendable system to allow researchers to specify their own VM and power management policies. The *IoTSim-Osmosis* is considered the most comprehensive IoT simulator since it possesses the most features and capabilities among other simulators.

Accordingly, *SUMOtoOsmosis* framework is composed of the traffic simulator SUMO [197] and the IoT simulator IoTSim-Osmosis. In the IoTSim-Osmosis simulator, there is no direct communication between IoT devices; they need to interact with the edge datacentre which is in the cloud, which are VM1, VM2, and VM3 in 4.1. To simulate that, the edge data center acts as a traffic light, whereas IoT devices act as vehicles. The traffic light (the edge) can have a number of connected IoT devices (vehicles), generating data over a particular time interval, Fig. 4.1 illustrate this emulation step. Each vehicle obtains a battery with an integrated consumption policy. The data from each vehicle is forwarded to the traffic lights nearby. The distance at which the traffic light will

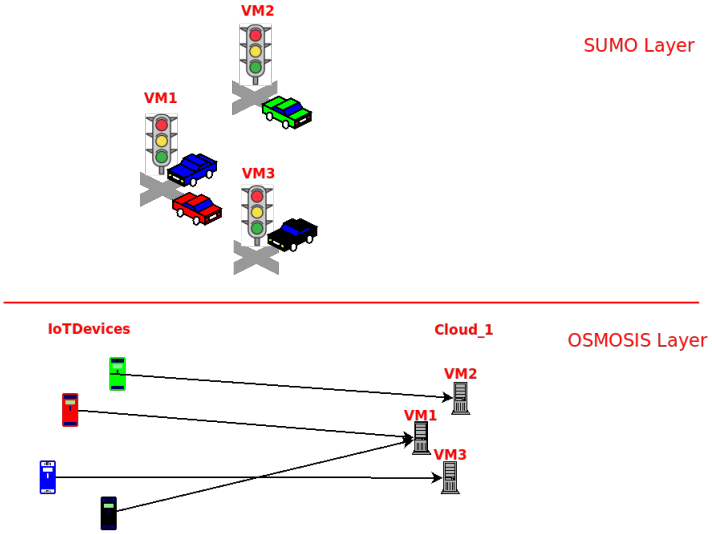


Figure 4.1: Emulating Osmosis layer to SUMO layer

detect vehicles is set. Every vehicle consistently senses its surrounding environment over a given time interval, sending its data to the near traffic light. The data format includes a vehicle's identical type, location, speed, slope, and other such information.

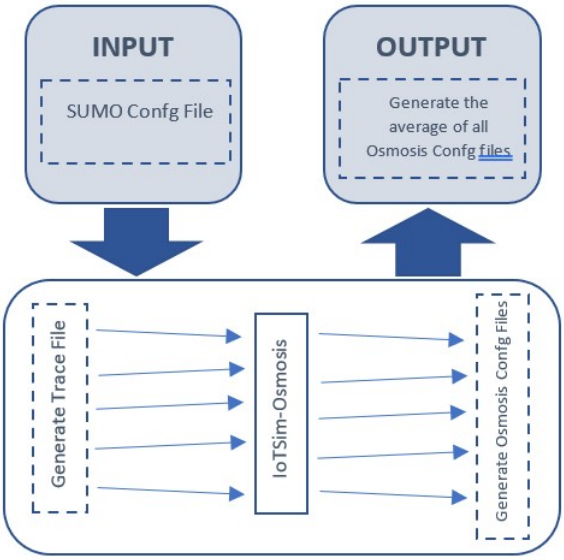


Figure 4.2: SUMOtoOsmosis Architecture

As a first step, the proposed simulator, *SUMOtoOsmosis*, takes a configuration file from SUMO as an input. And generates the trace file that has information for the vehicles appearing in the simulation at a given position in a given simulation time.

In the second step, the proposed system invokes the IoTSim-Osmosis simulator at each time frame on the trace file and generates the configuration file. On this step, traffic lights detect vehicles within a selected distance and process their data. This distance is

considered to be a major parameter for traffic light battery consumption. The output for this step will be a summary of all the transactions between vehicles and traffic lights. Figure 4.2 shows the architecture of the SUMOtoOsmosis system.

4.7 Experiment

In this section, the experimental setup and results of the proposed *SUMOtoOsmosis* framework is presented. Implementation details, data handling, and results of testing the system on the Hamburg dataset are described to demonstrate the effectiveness of the proposed approach in simulating vehicular communication and load balancing in VANET environments. *SUMOtoOsmosis*'s code bridges both SUMO and the Osmosis pipeline and, given as an input the scripts for SUMO, it returns a table similar to Osmosis. The Osmosis codebase is extended to generate an actual CSV file. The simulation times from Osmosis are also aligned with the ones from SUMO. Implementation-wise, KD-Trees are used to store the positions of the vehicles, and traffic lights are used as query points for finding the nearest vehicles. In the CSV, IOTDeviceName stands for the name of the vehicle, while AppID, AppName, and DestinationVmName refer to the specific traffic light within the SUMO simulation. The program's behaviour can be changed via the YAML configuration file, thus including the parameters for both the SUMO and the IoTSim-Osmosis simulations.

The proposed system *SUMOtoOsmosis* has been tested with Hamburg dataset[1]. Fig. 4.3 shows the number of vehicles communicating with a traffic light per simulation time from Hamburg experiment.

In Fig. 4.4, shows the distribution of the vehicles between traffic lights after balancing the load between them. To demonstrate the variation in the number of vehicles communicating with each traffic light, the load was balanced. Different numbers of vehicles were generated for each traffic light by a script, and then the trace file was produced. The Hamburg scenario was simulated using the proposed *SUMOtoOsmosis* platform, and the results were plotted to show how the number of vehicles rises and falls over time for each traffic light.

[1] μ`https://github.com/DLR-TS/sumo-scenarios/tree/main/TAVF-Hamburg`

Figure 4.3: Number of vehicles communicating with a traffic light per simulation time

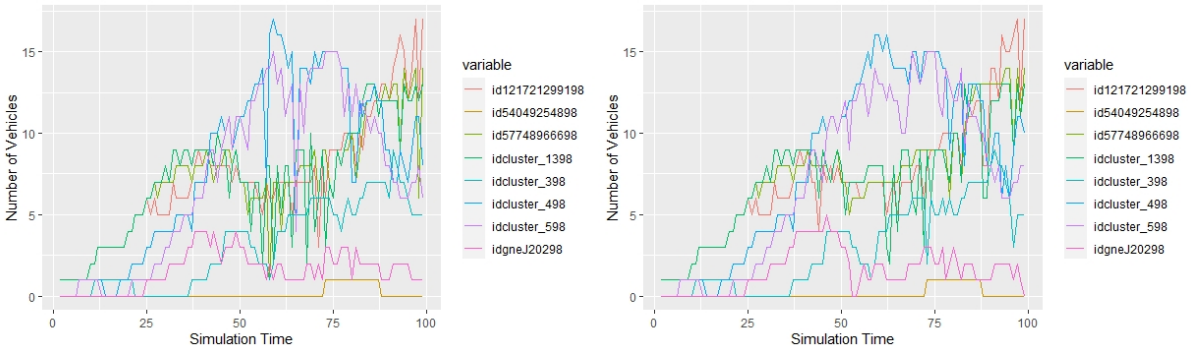


Figure 4.4: Different distributions of the vehicles between traffic lights with load balancing

4.8 Conclusion and Future Works

In conclusion, as this is a preliminary experiment for simulating traffic environments based on the IoT infrastructure, the simple extension for a mobility scenario could be provided. In this chapter, the architecture for the proposed system have been provided. From the architecture 4.2, it appears that the proposed platform is not able to continuously simulating the scenario and needs to invoke the IoTSim-Osmosis simulator at each time frame, which is not realistically. Also, the proposed system have been tested using Humburg dataset, and provided a result only for the communication time. However, the system doesn't provide the buttery information. So, to take the battery information into account, it is necessary to build a new simulator. In conclusion, to the best of the knowledge, this is the first study and research to bridge traffic simulators with IoT simulators. However, the major limitation for this work is that it is not continuously simulating the scenario which is not realistically. Also, it doesn't take into account the battery information. As

a result, the future work for this project will involve building a new simulator that will support battery consumption and act more realistically than bridging two simulators.

5

Platform for Energy Efficiency Monitoring Electrical Vehicle in Real World Traffic Simulation: SimulatorBridger

Contents

5.1 Introduction

The advancement of technology leads us towards a new area of communication connecting two or more vehicles to exchange data within an IoT ecosystem. This drives us through a new VANET (Vehicular Adhoc Network) framework making the conventional transport system safer, full of infotainment, convenient, and smart. In VANET research, researchers aim to develop a simulator platform that provides *quick* and *cost-efficient* transmission of data for passenger safety and comfort. As described earlier, VANET simulators consist of two components: Traffic Simulators (software environments that generate vehicle movements in trace files) and Network Simulators (used to test the performance of networking protocols). The network simulators are used to build communication topologies, evaluate network protocols, and exchange routing information between the nodes after importing the traces of mobility models generated by traffic simulators. These mobility models, which are the depiction of real-world scenarios, are called *traces* and contain the simulated infrastructure and event information such as vehicle speed, type, origin, destination, arrival time, arrival rate, maximum density, number of lanes, speed limits, capacity, intersection type, queuing, service distribution, service rate, traffic signs, location, etc. Green and orange paths in Figure 5.1a provide a minimalistic view of these traces where only the vehicle's geographical position is considered. The traces generated with the mobility generator, microscopic or macroscopic [199, 200], are imported into the network simulator and generate a vehicular program: boxes surrounding paths in Figure 5.1b provide a depiction of the vehicular programs associated with simulated IoT agents.

SimulatorBridger extends this traditional coupling of traffic and network simulators by integrating SUMO and IoTSim-OsmosisRES to evaluate key performance indicators (KPIs) such as energy consumption and communication latency. Unlike existing platforms, SimulatorBridger provides mobility support for IoT devices, enabling a more realistic evaluation of urban vehicular environments. SimulatorBridger improves upon traditional frameworks by providing a seamless integration of mobility and IoT devices in highly heterogeneous and dynamic environments. The architecture supports plug-and-play functionality, allowing the inclusion of other traffic simulators if required. Preliminary results show that the vehicular battery consumption distribution closely aligns with packet distribu-

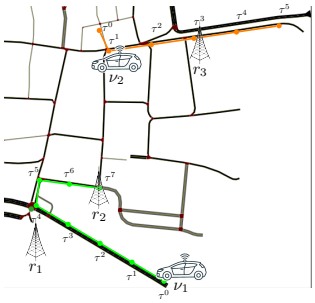
tion trends, demonstrating the fidelity of the simulator. Battery consumption is a critical metric in evaluating the energy efficiency of IoT-enhanced vehicular systems. Addressing this issue early ensures a comprehensive understanding of energy demands in vehicular IoT networks, contributing to the sustainability and efficiency of urban environments. Furthermore, this framework highlights the potential for testing diverse communication policies aimed at reducing packet processing overhead and network congestion. These results underscore SimulatorBridger's role as a potential digital twin for realistic urban mobility scenarios, empowering policymakers to test various network configurations and optimize traffic distributions.

5.1.1 Objective and motivation

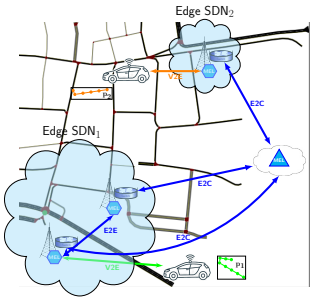
The growing complexity of urban vehicular environments has motivated the development of simulation platforms that can effectively integrate mobility and IoT technologies. The previous chapter (§4) introduced a bridging platform that highlights the need for seamless coupling between traffic and IoT simulators. This work builds upon that foundation by proposing a new simulation framework, SimulatorBridger, which combines the state-of-the-art IoT simulator, IoTSim-OsmosisRES [25], with the traffic simulator SUMO [197]. This coupling allows for an accurate representation of vehicular IoT systems, addressing limitations in existing simulation tools.

IoTSim-OsmosisRES, as discussed in Chapter 2, is a unique IoT simulation platform that incorporates energy management, diverse power sources, and network infrastructure. However, its current implementation assumes IoT devices are stationary, which limits its applicability to dynamic vehicular environments. Motivated by these limitations, this research aims to extend the capabilities of IoTSim-OsmosisRES by enabling support for IoT device mobility, coupled with realistic battery consumption modeling. Battery consumption modeling in this context is crucial for understanding how communication and mobility patterns affect energy usage in IoT-enabled vehicles. By including detailed battery consumption metrics, the simulator provides insights into optimizing energy efficiency while maintaining system performance. The traffic simulator SUMO provides the mobility traces necessary for modeling vehicular movements, forming the basis for interactions between IoT devices and network infrastructure.

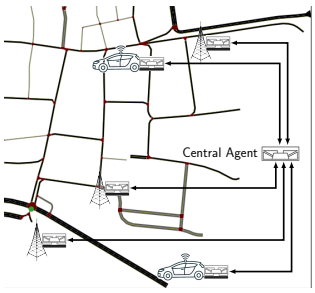
SimulatorBridger is designed to achieve the following objectives: (1) integrate IoTSim-



(a) Output of the SUMO simulator, associating each vehicle ν to a trace (paths in green or orange) determining their location $\vec{\omega}_\nu^i$ at each simulation frame τ^i. RSU r are associated with fixed geographical coordinates $\vec{\omega}_r$.



(b) Osmotic Applications (MEL representation) remarking the Vehicle2Edge (V2E), Edge2Edge (E2E), and the Edge2Cloud (E2C) communication channels. Boxes denoted as p_1 and p_2 associated to each vehicle indicate the "simulation programs" retaining the mobility information.



(c) Supporting the Osmotic Agent configuration from IoT-Osmosis-RES with centralized training and decentralized execution for dynamically updating the communication strategies of the IoT devices. This enables us to support dynamic load balancing in the future.

Figure 5.1: Osmotic Computing in car traffic scenario in Newcastle Upon Tyne.

OsmosisRES with SUMO to simulate dynamic vehicular environments, (2) extend IoTSim-OsmosisRES to support mobility for IoT devices, and (3) enable the simulation of energy consumption in heterogeneous IoT-enabled vehicles. By achieving these goals, the proposed framework offers a more realistic representation of the interactions between mobility, communication, and energy consumption in urban IoT ecosystems.

Furthermore, SimulatorBridger's modular design allows for flexibility in its implementation. The framework supports a plug-and-play approach, enabling the integration of alternative traffic simulators while retaining its core functionality. This flexibility ensures that SimulatorBridger can adapt to the evolving demands of vehicular IoT research.

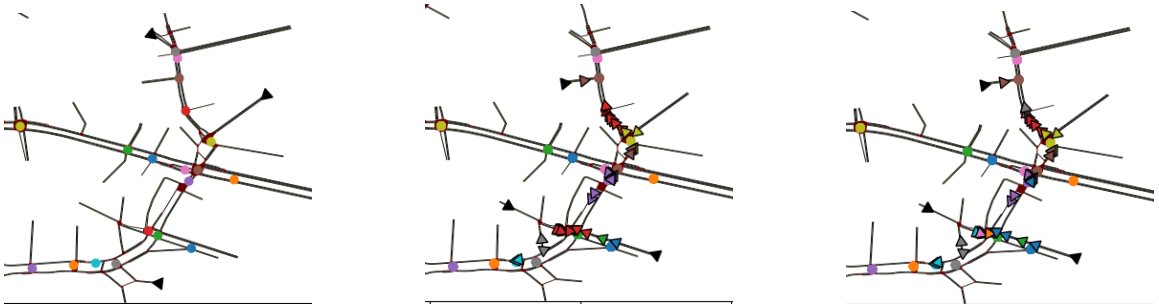
Preliminary results (§5.5) demonstrate the correctness of the simulator, showing that vehicular battery consumption trends closely align with packet distribution patterns. Simulations also reveal that increasing the number of communication packets results in network congestion, highlighting bottlenecks in processing. These findings validate the fidelity of SimulatorBridger as a digital twin for realistic urban mobility scenarios. By providing a platform to test different network configurations and traffic distributions, this framework empowers policymakers and researchers to explore strategies for minimizing network communication overhead and improving system efficiency.

5.1.2 Use Case Scenario

The recent Car-as-a-Service paradigm [212] has remarked the possibility of using the cars' IoT sensors not only for the personal purposes of travellers but, with the driver's agreement, for the benefit of the entire community. As most modern cars are equipped with navigation systems and video cameras, cars can collect videos and images of their surroundings and geo-tag such information. This might be extremely useful for any town hall highly committed to road surface remaking for car safety so that they can spot tarmac conditions before they degenerate into potholes [26]. To do so, the data needs to be collected from cars and then streamed towards the main data centre, Figure 5.1, where an AI model is going to detect the seriousness of the situation [27]. Road-Side Units (RSU) might first collect the data from the cars through 5G antennae supporting massive low-latency communication and stream it towards the primary data centre thanks to the core networks. Communication between 5G antennae is supported by optical fibres realising low-latency communications. As a result of the data collection process, a massive amount of data will stream every second from the cars towards the data centres, which will severely increase during rush hours when road congestions might happen with higher probability [28]. Figure 5.2 represents a portion of the simulated traffic from Hamburg. [1] Due to the massive volume of data generated by the high number of vehicles during rush hours, any road congestion will constitute a communication bottleneck degrading the overall communication performance [213]. To achieve success in these situations, the traffic load needs to be balanced [214] to reduce communication delays. Given that the traffic flow might be redirected at run-time, this motivates the representation of each RSU as a specific Edge [209]. The exploitation of Osmotic Computation [215] eases communication flow management, thus dynamically establishing new streams. This also motivates the definition of a simulator bridging traffic simulation to network communication systems.

The remainder of this chapter is structured as follows. Section §5.2 details the design of our proposed simulator framework. Next, in section (§4.4.1), we discuss how we implement the simulator and provide an overview of its key components. In the following Section (§5.5), the results of the experiment are discussed and analyzed. Conclusion and future works are discussed in Section (§5.6).

[1] μ`https://github.com/DLR-TS/sumo-scenarios/tree/main/TAVF-Hamburg`



(a) Some road lanes (black) and some RSU (colored circles). Black triangles are vehicles not associated with an RSU.

(b) Associating each semaphore with the nearest cars within the radius of θ : cars and RSU with the same colour communicate.

(c) After exploiting a load balancing algorithm, the cars are distributed to other semaphores in the neighbourhood.

Figure 5.2: A subset of the Sumo TAV Hamburg Dataset for mobility

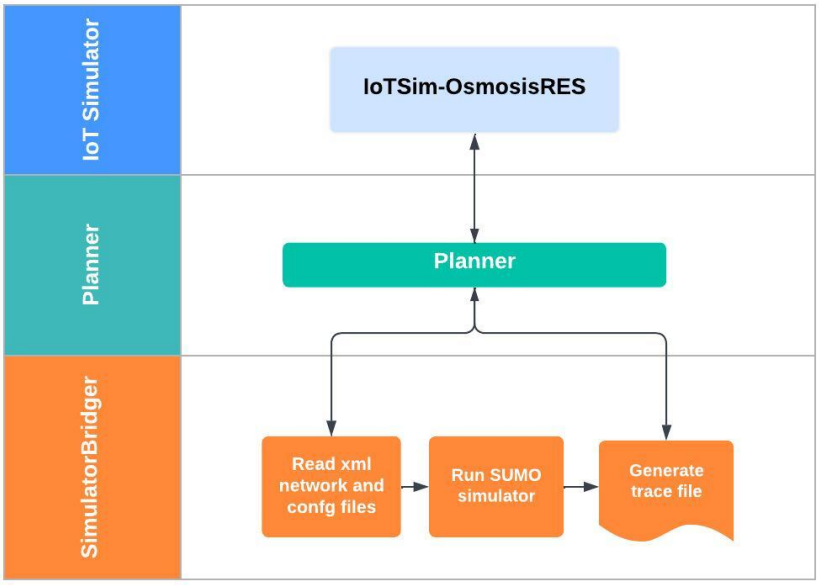


Figure 5.3: Architecture of SimulatorBridger

5.2 Design of the Simulator

In this section, we detail the design of our proposed simulator framework that integrates traffic simulation and IoT simulation. Our framework twins traffic simulation and IoT simulation. Thus, it incorporates well-known models for road traffic micro-simulation with a comprehensive selection of models of network protocols. By running the vehicular generator, we collect all the vehicular geographical information and the position of the Edge node (§5.2.1). This is required to set up the network topologies associated with each Edge node and determine the vehicular program to be injected into the IoT devices in

the IoT simulator (§5.2.2). At each simulation time, each IoT device will query a centre to determine whether the IoT device shall to establish communication with the cloud through an edge device or not (§5.2.3). Upon confirmation, the IoT device directly contacts the MEL router associated with the edge node of choice, establishing communication between the IoT device and the cloud through one of its hosts (§5.2.4). The simulator leverages Software-Defined Networking (SDN) and Software-Defined Wide Area Networking (SD-WAN) protocols to facilitate traffic management between IoT, edge, and cloud environments. It abstracts physical-layer protocols like WiFi or LTE to focus on high-level metrics such as energy efficiency and communication latency. While this design ensures computational efficiency, future extensions could incorporate protocol-specific details for more granular analysis.

5.2.1 Vehicular Data Collection

First, the simulator collects the mobility information generated by the traffic generator. This simulator might be configured by setting the begin τ_b and the end τ_e simulation time, as well as a temporal granularity δ defining the sampling rate of the vehicular information. This allows do determine the time at the i-th simulation step for each vehicle ν as the following relation:

$$\tau_\nu^i = \begin{cases} \tau_\nu^{i-1} + \delta & i > 0 \\ \tau_b & \text{oth.} \end{cases}$$

At the end of the traffic generation, the VANET simulator returns a list of pairs associating each simulation time τ_ν^i to a geographical location $\vec{\varpi}_\nu^i$ for each vehicle ν. These vehicular mobility traces serve as critical inputs to the simulator, forming the foundational data for subsequent network topology configurations and IoT simulation. Figure 5.1a provides a graphical depiction of such traces, highlighted in either green or orange. From the network topology, we might also infer the geographical position of each RSU of interest r alongside its position $\vec{\varpi}_r$ and its communication radius ρ_r. This operation is performed by a simulator wrapper, which provides a standard interface for generating the data of interest in a uniform representation independently from the specific traffic simulator of choice. These mobility traces serve as the core input for setting up the network topologies, enabling the next stages of IoT simulation. Although the simulator is

currently confined to urban vehicular mobility patterns, its modular framework allows for integration with ad-hoc or non-urban mobility patterns. For instance, it can be adapted to simulate pedestrian movement, drone navigation, or off-road vehicle mobility by incorporating custom mobility datasets. This flexibility broadens the simulator's applicability beyond urban settings.

5.2.2 Software Defined Network Configuration

The next step assumes that each vehicle ν is associated with just one single IoT device potentially communicating with the cloud infrastructure. We collect the sequence of pairs $[(\tau_\nu^0, \vec{\varpi}_\nu^0), \ldots, (\tau_\nu^n, \vec{\varpi}_\nu^n)]$, where $\tau^n = \tau_e$, as a vehicular program ν that will be injected within the IoTOsmosis-RES simulator.

This will allow the IoT device to update its position at each instant of the SUMO simulation time in the network simulator. Fig. 5.1b represents such vehicular programs as minimizations of the traces represented in Fig. 5.1a.

Contextually, we define Software-Defined Wide Area Networks for the RSU nodes. To do so, we first define an undirected graph $G = (V, E)$, where the vertices V list the RSU nodes r obtained from the VANET simulator, and the edges establish a communication channel between two distinct RSU r and r' if and only if they can both communicate between each other:

$$\forall r, r' \in V. \quad (r, r') \in E \leftrightarrow r \neq r' \wedge \|\vec{\varpi}_r - \vec{\varpi}_{r'}\| \leq \min(\rho_r, \rho_{r'})$$

For each strongly connected component of such graph, we establish a distinct Edge SDN with an associated Datacenter and Software-Defined Network Controller, where each RSU is described as an Edge device associated with at least one host. These Edge SDNs enable a robust network configuration, ensuring efficient communication between vehicles, RSUs, and cloud nodes. Each light blue cloud in Fig. 5.1b represents a distinct Edge SDN, for which only the Edge nodes are remarked. We also set up a Cloud network towards which each IoT device will communicate to send the sensed data, as well as an SDWAN network bridging each Edge SDN to a Cloud SDN. Both these structures are engulfed in the white cloud in Fig. 5.1b. Last, we completely set up the edge connectivity between these networks by setting up Edge2Edge and Edge2Cloud communication links. VehicleToEdge links will be established at simulation time.

5.2.3 Osmotic agents module

The osmotic agents' module is exploited for simulating network discovery procedures and determining each IoT device which should be the best edge node for establishing the communication. Each IoT device queries a central agent to decide its communication pathway dynamically. This cannot be necessarily determined at the local level, as each IoT device has only the information of which edges it can communicate with, but it doesn't know which is going to be the best neighbouring agent towards which establish the first-mile communication towards for reaching the cloud. An IoT device acts as an reinforcement agent, which sends the information about the observed environment to the central agent. In particular, each IoT device sends its geographical position, while the geographical position of each edge device is known a priori. This dynamic querying process allows for efficient routing and communication setup, directly impacting the KPIs of energy efficiency and communication latency. We implemented two specific types of central agents:

- NearestCentralAgent: for each querying IoT device, the central agent will always respond with the nearest Edge device to the IoT that is within mutual signal coverage. This strategy does not require any alteration to the SDN Routing Policy of each Edge Network, which might still exploit the SHORTESTPATHMAXIMUMBANDWIDTH discussed in [216].

- FlowDataCenterAgent: after associating each IoT device to a possible edge device, the agent runs a multi-source and multi-target minimum-cost flow problem for establishing the best strategy to minimize the channel communication. This algorithm will then return for each IoT device the best edge device. This also requires updating the SDN Routing Policy MaximumFlowRoutingPolicy associated with each network via its SDN Controller, thus returning the paths calculated by the Central Agent.

Upon reception of a non-empty edge device, the IoT will generate a novel Osmotic Application and start communication towards the cloud through the selected edge node. Figure 5.1c illustrates the bidirectional communication flow between IoT or SDN routers

and the Central Agent. The simulator assumes proximity-based connectivity, where successful communication is established if the IoT device is within range of an RSU. However, detailed electromagnetic propagation models, such as free-space path loss or fading effects, are not included. This abstraction simplifies the simulation while focusing on high-level metrics. For scenarios requiring detailed propagation modeling, external tools or extensions could be incorporated into the framework.

5.2.4 Dynamic Adaptive Routing

In osmotic computing, each IoT device directly interacts with MicroELements (MEL) software components that might be instantiated anywhere in a given network associated to Edges [25]. To faithfully represent real communication scenarios, an IoT device must directly establish communication with a precise edge node [217]: the Adaptive Routing in [25] was further extended to directly select the MEL associated with a given edge node instead of picking one in a round-robin fashion. As IoT-OsmosisRES further associates a MEL to one or more hosts, the routing algorithm picks one of the hosts running the MEL as its first-mile communication point. Contextually, the host establishes the communication channel with the cloud network indicated by the IoT device, thus starting the communication. The routing process adapts dynamically based on the IoT device's position and network conditions, ensuring energy-efficient and low-latency communication pathways. The routing mechanism uses SDN and SD-WAN technologies to optimize communication pathways. While lower-layer protocol implementations like WiFi or LTE are abstracted, this approach enables efficient evaluation of system-wide metrics, such as routing efficiency and energy consumption. Future improvements could incorporate protocol-specific details to study their impacts on network performance.

5.2.5 Inputs, Outputs, and Key Performance Indicators (KPIs)

The simulator relies on vehicular mobility traces, IoT device configurations, and network topology data as inputs. It outputs energy consumption metrics, communication latency, and network efficiency indicators. Key performance indicators (KPIs) such as energy efficiency, latency, scalability, and network reliability are used to evaluate the simulator's performance under diverse scenarios. The KPIs are assessed under idealized network

conditions, focusing on the influence of mobility and energy consumption on system performance. Although scalability is not explicitly analyzed in this chapter, the simulator's modular design ensures its capacity to handle larger networks through SDN and SD-WAN integration.

5.3 Implementation in Java

In this section, we discuss how we implemented our simulator, SimulatorBridger, which is developed using the Java programming language. We provide an overview of the key components and modules, highlighting their roles in creating a comprehensive simulation environment that integrates traffic and IoT simulations. The proposed simulator, SimulatorBridger, is available in Github [2]. The NetworkTopology.java class in the SimulatorBridger-IOTSimOsmosisRES module is crucial for setting up the network layer in the simulation. It generates a topological network, which is used to simulate latency in network traffic. The IoTDevice.java class represents IoT devices within the IoTSimOsmosisRES framework. The CloudletScheduler class is responsible for scheduling tasks in the cloud. And EdgeDeviceManager class is managing edge devices.

The implementation of the SimulatorBridger platform contains several modules and a large number of Java source files. Here's an overview of its structure:

SimulatorBridger-core component contains shared dependencies and core functionalities used across the project. It serves as the foundational codebase that other modules in the project rely on.

SimulatorBridger-traffic-information-collector involves running the traffic simulator and collecting data from it. It processes the output of the simulation, identifying IoT nodes and Edge nodes. The Edge nodes are used for interactions by the IoT nodes.

SimulatorBridger-central-agent-planner provides a theoretical omniscient algorithm capable of scheduling time as required. It also generate potential network connectivity information based on the IoT and Edge information provided by the simulation.

SumoOsmosisBridger is an example that bridges all the simulations together with Dynamic IoTSimOsmosisRES simulator. It illustrates how the components can be integrated in a seamless manner.

[2] μ`https://github.com/jackbergus/SimulatorBridger/releases/tag/v0.1`

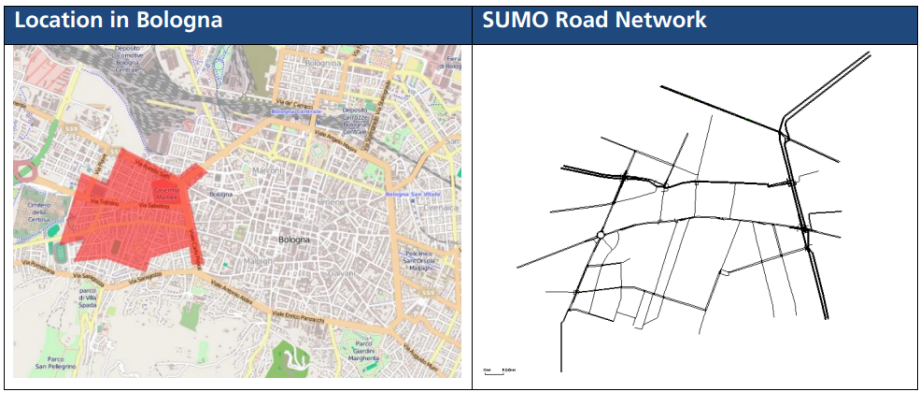


Figure 5.4: Bologna Dataset [16]

Each of these components plays a specific role in integrating and processing traffic data within the IoTSimOsmosisRES environment. The process starts with data collection and processing, followed by planning and scheduling using the central agent planner, and finally, the integration of all these elements through the SumoOsmosisBridger to work with Dynamic IoTSimOsmosisRES. This integration allows for a comprehensive simulation environment that can handle complex IoT and traffic data scenarios.

5.4 Dataset and Experiment Setup

This section covers running the Bologna dataset [3], through our proposed simulator, where vehicles have an embedded IoT device and RSU are Edge nodes associated to MELs. Fig. 5.4 shows the SUMO network of this dataset. The dataset has 16 RSUs and 239 vehicles. The Bologna scenario includes the area around the football stadium and was set up to simulate the mobility of big events such as football matches or concerts.

Our experiments orchestrate a traffic simulator (SUMO) with a network simulator (IoTOsmosis-RES) with each simulation second in the former corresponding to a second in the latter. We arbitrarily set the IoT battery consumption rate to 1.02% when a vehicle communicates with a MEL. We assumed embedded IoT devices cannot be re-charged to better analyse the simulator's correctness by correlating the number of packets sent by an IoT device with its battery consumption. When a vehicle was near an RSU in the traffic simulator, we scheduled a new communication between the IoT device embedded in the vehicle and the Cloud via the Edge node representing the RSU in the network simulator.

[3] μhttps://github.com/DLR-TS/sumo-scenarios/tree/main/bologna/acosta

This schedule is selected according to the granularity δ of the traffic simulator: in our scenario, this is set to start every 1 second. In this time frame, the IoT device in the vehicle sends one single packet. The simulation time for these plots begins at the start of the traffic simulator and ends when the last MEL has successfully sent all the packets to the cloud.

Our simulator assumes each RSU is an Edge device containing multiple MELs. As per the previous discussion, the network simulator was extended to establish direct communication between IoT and Edge devices by resolving an available MEL associated with the Edge device. A round-robin policy selects an available MEL in the Edge node to ease each MEL's workload. For this experiment, the maximum simulation time of IoTOsmosis-RES was 3898.4s, corresponding to 100% of the simulation, starting at the same time, the maximum time for the SUMO simulator was 186.7s (τ_e), with vehicles no longer able to enter the urban environment after around 55% of the overall traffic simulator time, or 100s. From this, as the number of vehicles in the simulation decreases, the number of starting communications will also decrease, resulting in a normal distribution for both the number of communicating vehicles and the number of packets sent. The traffic simulation takes less than 5% of the total simulation time. This is due to packet network delays which delayed communication patterns even though the IoT devices sent no new packets. We consider all communications between an IoT vehicle and a MEL.

The simulation assumes that each vehicle sends a packet to the RSU every second while within communication range. This periodic transmission replicates real-world scenarios where vehicles share sensor readings, status updates, or environmental data with roadside infrastructure. Metrics such as delay, jitter, and throughput are calculated by aggregating data from all packets transmitted, ensuring a realistic assessment of network performance under varying traffic conditions.

5.5 Analysis and Results

In this section, we analyze and discuss the key findings from the Bologna dataset experiment conducted on the SimulatorBridger platform. The results of our simulation are meticulously examined, focusing on critical performance metrics and the effectiveness of various strategies implemented throughout the process. This analysis not only high-

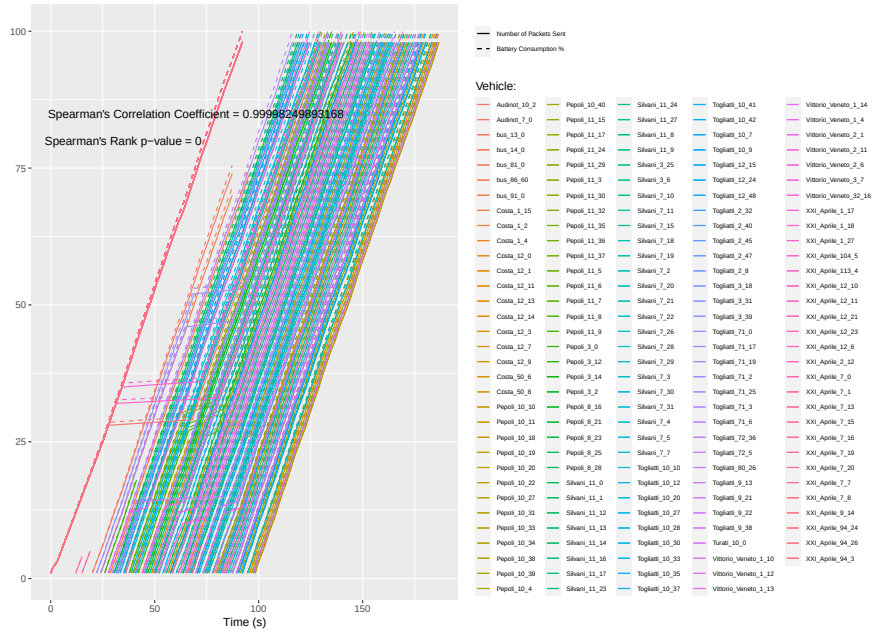


Figure 5.5: Correlation between battery consumption and number of packets being sent per vehicle within the simulation.

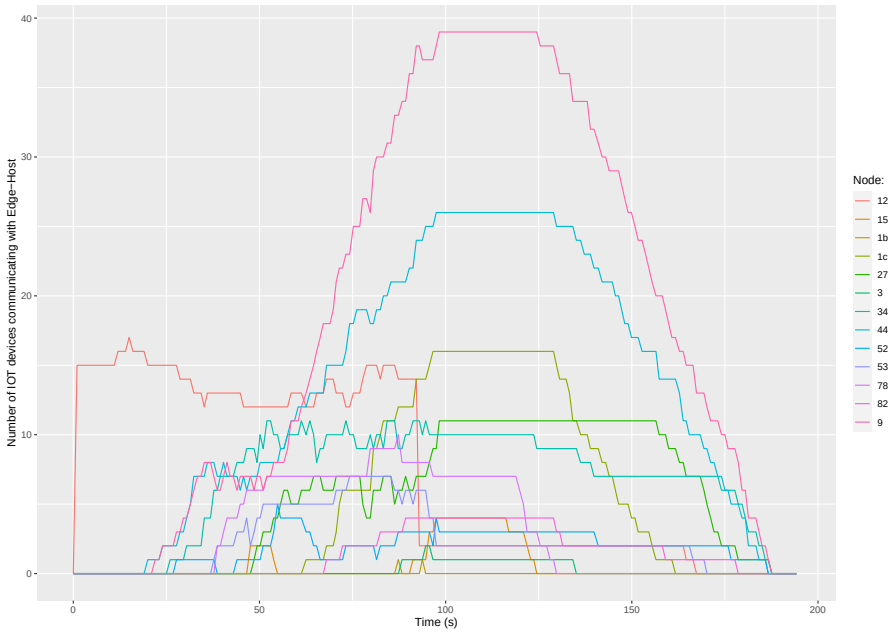


Figure 5.6: Number of IoT devices communicating with an Edge per simulation time.

lights the successes and challenges encountered but also provides valuable insights into the practical implications and potential improvements for future iterations.

Fig 5.5 shows that each of the 163 embedded IoT devices (in distinct colours) shows an almost perfect correlation between the number of packets sent (solid line) and its battery consumption (dashed). The Spearman correlation between those for each vehicle is almost 1 , with a p-Value of 0 , thus indicating a very strong correlation between

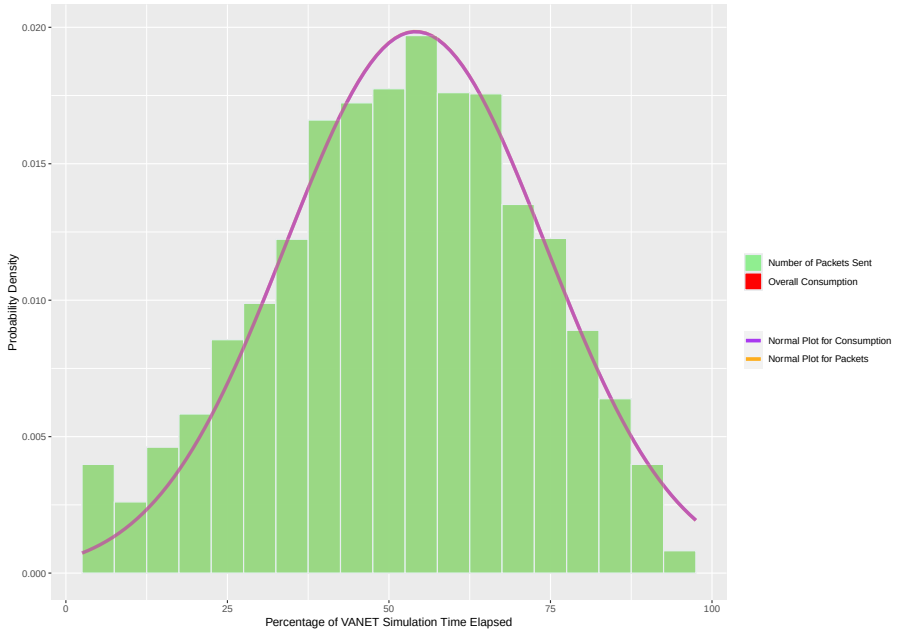


Figure 5.7: The distribution of packets sent in the network follows the same trend as the overall IoT battery consumption.

these two. From Fig 5.6, for most of the RSUs the number of connected to each has an approximate normal-like distribution, with most plateauing after the 100s mark, which is at about the 55% mark in VANET simulation time mentioned earlier. A maximum number of 39 IoT devices were connected to a single MEL and its Edge node at any time. This figure also explicitly shows vehicles starting at an RSU, as Edge#12 has 15 connected vehicles from the very beginning of the simulation. This figure also shows the simulation time used for these plots, the simulation time starts at time 0 seconds and then stops around 190 seconds once all the RSUs have no more connected vehicles. Next, we tested if there was a strong correlation between the number of vehicles in the simulation within a given time interval, with the overall battery consumed within that same time interval, as common-sense suggests that an increase of the number of the vehicles should match an increase of communications between a IoT and Edge devices, thus reflecting in an increase of the overall battery consumption. We found no strong correlation between these two aspects as, even though more vehicles lead to an increase of communications, this does not necessarily entail that either more communications should take place (e.g., vehicles might not be in a region covered by RSUs) or even that all vehicles have the same number of communications within the same time interval.

Following this, any potential correlation between packets sent from vehicles and the over-

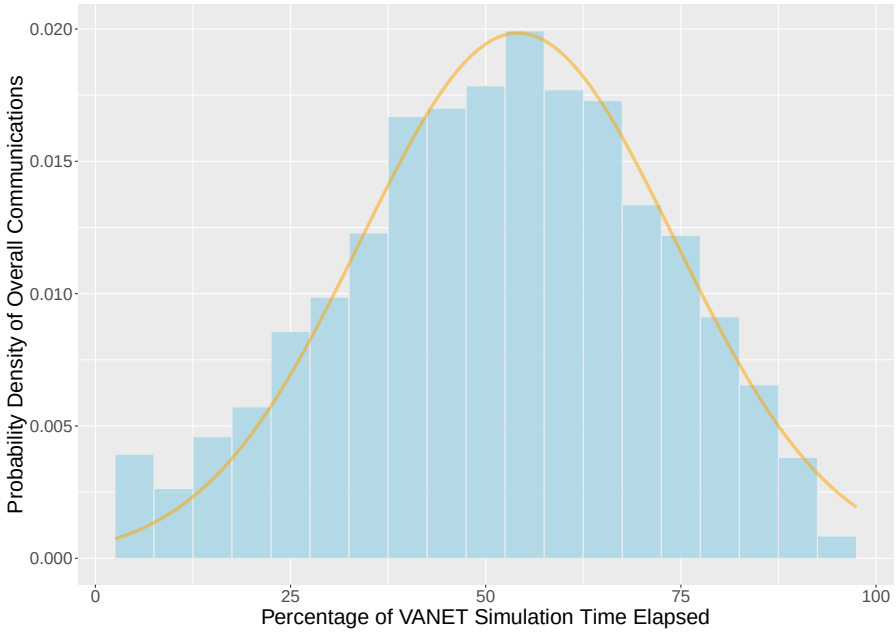


Figure 5.8: The distribution of the number of communications starting within each time interval.

all battery consumption was investigated. The inclusion of battery consumption as a key performance metric demonstrates the simulator's capability to assess energy efficiency in dynamic vehicular scenarios. This analysis underscores the relationship between communication loads and energy demands, providing actionable insights for designing energy-efficient IoT networks. Fig 5.7 shows the probability density of the overall battery consumed by all vehicles in the simulation (within each 5% time interval of simulation time), along with probability density of the number of packets sent by all vehicles (always within each 5% time interval of simulation time). From Fig 5.7 both sets of data not only closely follow a normal distribution, but also are closely correlated with each other, as both distributions and both normal plots completely overlap one another. These two variables correlating with each other was expected, as in the current simulator setup, the packets being sent are responsible for the battery consumption, and the fact that we found these two variables do in fact correlate indicates that the simulation is behaving as expected. Fig 5.7 shows a spike occurring at the $35 - 45\%$ for both the overall consumption and the number of packets sent. This spike can be explained in terms of Fig 5.8 showing the distribution of the communications starting within each traffic and VANET simulation time interval: this spike is also present for the number of starting communications within this same time interval. Given the current simulator settings of a vehicle sending

a single packet as a result of communication with an RSU, this like-for-like behaviour of communications and packets sent is not only expected but demonstrates the simulator is functioning correctly. The reason more packets were sent and therefore why more battery was consumed in this time interval is due to more communications occurring within this time interval. Fig 5.8 also validates our preliminary hypothesis observing that the number of starting communications should drop after a 55% elapse of the Sumo simulation time.

While the overall distribution of communication start times follows a normal pattern across the entire simulation (Figure 5.7), localized variations occur due to differences in RSU placement, vehicle density, and traffic flow patterns (Figure 5.8). These variations reflect the heterogeneous nature of urban traffic environments, where RSU utilization intensity is influenced by proximity to high-traffic areas and vehicle trajectories.

We might also observe another spike at the beginning of the simulation where the influx of traffic starts entering the simulation while approaching RSUs, thus justifying the increase of the overall number of communications. The explanation for the initial drop-off between the first two intervals, $0 - 5\%$ and 510%, can be explained with the first interval being artificially high as a node is placed very close to where vehicles initially join when the simulation starts. This behaviour can also be seen on Fig 5.6 as Edge#12 having 15 connected vehicles just 1 second into the simulation.

From Fig 5.9 vehicles at the start of the simulation enter from the most southwestern entrance, which, when factoring in the 100 m radius of communication of the nodes, explains how vehicles are able to connect to Edge#12 (i.e. the light blue Node 12 in the bottom left of the plot). After 190 seconds in the simulation, the Sumo simulation stops, and therefore we observed no vehicles connected to any of the RSUs. In fact, the IoTOsmosis-RES still has to wait for all the Edge nodes to send their packets to the cloud while receiving an acknowledgement for this. The discrepancy between the end of the traffic simulator and the Network simulator also matching the end of the VANET simulator has been cut off for the sake of legibility. This processing of all the packets by the network infrastructure therefore takes over 3700 seconds: we refer to this as *shutdown time*. Fig 5.10 shows how this shutdown time is affected by the number of total communications in the simulator: the more time steps in the simulation, or the longer vehicles are allowed to navigate in the traffic simulator, the more the communications between vehicles and RSUs take place. This matches the intuition that more time for

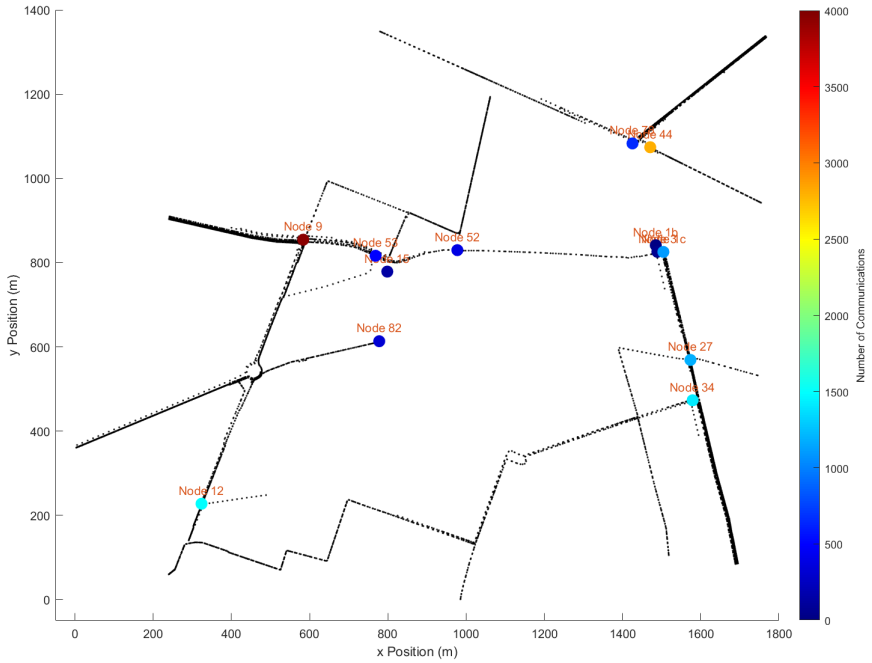


Figure 5.9: Showing Edge displacements as RSU in the Bologna Dataset: their colour represents the intensity of the undergoing communication. Black lines show the trajectory of the vehicles carrying embedded IoT devices.

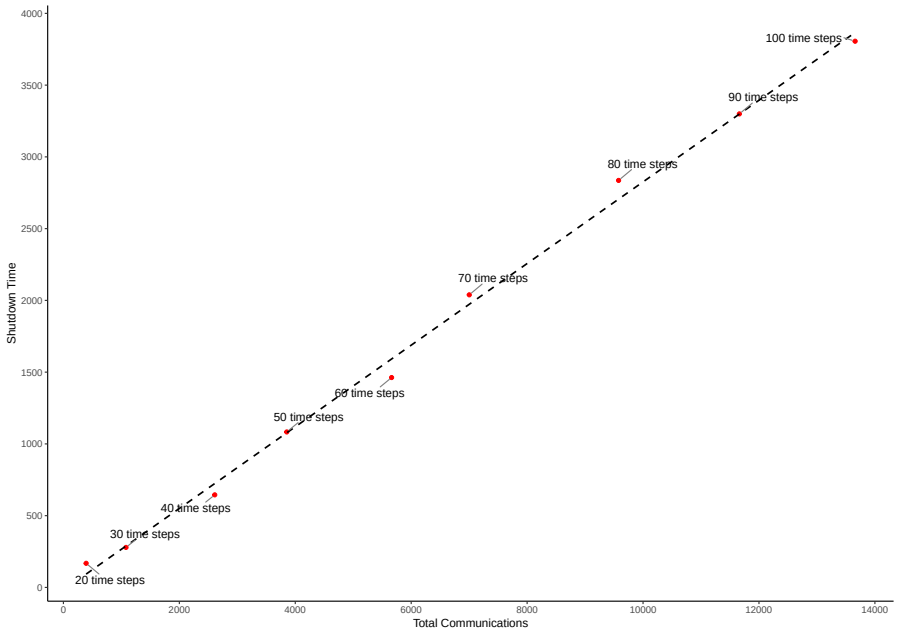


Figure 5.10: Correlation between simulation time, number of undergoing communications, and their effect to the shutdown time.

potential communications entails an increase in the number of communications. The more relevant result from this figure with respect to the shutdown, however, shows that more communications lead to a longer and longer shutdown time.

5.5.1 Scalability Analysis

The scalability of the SimulatorBridger framework is a critical factor in determining its capability to simulate increasing vehicular densities and communication loads efficiently. By analyzing the existing experimental results, we assess how the simulator performs as the number of vehicles (IoT devices) and their interactions grow over time.

Load Distribution Across RSUs: As shown in Figure 5.6, the number of vehicles communicating with each edge node (RSU) follows a normal-like distribution, with most RSUs stabilizing their connections after approximately 100 seconds of simulation time. The maximum number of vehicles connected to a single RSU is 39, highlighting the simulator's ability to manage a significant number of simultaneous connections without overloading any single edge node. This even distribution of vehicle connections across RSUs reflects the simulator's effective dynamic load-balancing mechanisms, which prevent resource bottlenecks.

Impact of Communication Density on Shutdown Time: Figure 5.10 illustrates the relationship between the total number of communications and the shutdown time, which corresponds to the time taken for all communication processes to terminate after the vehicular simulation ends. The increase in shutdown time is primarily due to packet processing delays within the network infrastructure. During the vehicular simulation, vehicles generate communications that must be processed by the RSUs and the cloud. Once the traffic simulation (SUMO) ends, the IoT simulation (IoTSim-OsmosisRES) continues to process any remaining packets in the communication queues. This sequential packet processing between IoT devices, RSUs, and the cloud extends the shutdown time as communication density increases. This behavior aligns with expected scalability challenges in large-scale IoT simulations, where higher communication loads lead to longer processing times.

Scalability Insights from Existing Experiments: The analysis of Fig. 5.9 and Fig. 5.10 underscores the scalability of the simulator in handling urban vehicular scenarios. The modular architecture of SimulatorBridger, which decouples mobility modeling from IoT interactions, ensures that communication demands and network performance are efficiently managed. The results indicate that even as the system load increases, the simulator processes all interactions and communications predictably, providing a robust platform

for testing vehicular IoT systems at scale.

Observations on Upper Bound Performance: The experiments demonstrate strong scalability under the tested conditions, with a maximum of 163 vehicles. However, the increase in shutdown time as communication density grows suggests that scalability is bounded by the computational resources available for processing queued communications at the cloud level. This observation is consistent with the expected behavior of large-scale IoT simulation frameworks.

5.6 Conclusion and future works

Due to the high mobility of vehicles in VANETs, realistic simulation is a challenging task. This paper proposes a novel simulator, SimulatorBridger, by bridging the IoT network simulation with the traffic simulation. The efficacy of SimulatorBridger is validated using a case study for urban roads in Bologna city. Results show the various capabilities of SimulatorBridger in terms of vehicular network lifetime, vehicle battery, and energy consumption. As demonstrated in Fig.5.6 and Fig.5.10, SimulatorBridger effectively manages load distribution across RSUs and scales predictably with increased communication density. According to our results, SimulatorBridger is also scalable in terms of vehicle count and simulation time. Furthermore, our framework can be easily extended to support a load balancing scheme between traffic lights, thus minimising load imbalances in the communication network while improving energy management and implementation time. A cooperative approach for load balancing among the network might be used when the traffic light receives more requests than the maximum number, resulting in high traffic or load. As highlighted in Fig. 5.10, the correlation between communication density and shutdown time suggests the importance of implementing advanced load balancing strategies to mitigate these challenges. Also, vehicular network lifetime is increased by reducing energy consumption which is necessary to balance energy in traffic lights. In addition, the simulator we propose combines IoTSim-OsmosisRES with any potential traffic simulator. This modularity enables the integration of alternative simulators to explore diverse traffic conditions and mobility patterns, broadening the framework's applicability. Then for future work, a different traffic simulator can be used in conjunction with IoTSim-OsmosisRES. This modularity allows for exploring different mobility patterns and

expanding the simulator's capabilities. Scalability in size remains a key area for further exploration. While the current simulation focuses on urban environments with a limited number of vehicles, future studies will aim to evaluate the simulator's performance under larger and more complex scenarios, such as multi-city networks with thousands of vehicles. Testing scalability with expanded datasets will validate the framework's capacity to support increasingly demanding vehicular IoT ecosystems. Additionally, the modular design of SimulatorBridger ensures that it can accommodate new traffic simulators and mobility datasets, paving the way for broader applications. As highlighted in Fig. 5.10, the network infrastructure faces significant delays due to the way packets are processed and transmitted between IoT devices and the cloud. Our future work will identify the specific causes of the long shutdown times and alleviate any problems in the infrastructure to reduce the shutdown time of the simulation. Specifically, we will explore whether optimizing packet routing policies, implementing advanced network configurations, or adopting dynamic load-balancing strategies can address these challenges. Lastly, future work will prioritize scaling SimulatorBridger to handle larger datasets and more complex vehicular environments, further demonstrating its robustness in supporting extensive simulations.

6

SimulatorBridgerDfT: A Real-Data Simulator for IoT-Osmotic Interactions

Contents

6.1 Introduction

Advanced simulation platforms are essential in urban transportation research, providing critical tools for evaluating and optimizing traffic management strategies. These platforms are pivotal in enhancing traffic flow, safety, and overall urban mobility. Several simulators have been developed to model these networks, each with varying capabilities and limitations. VANET simulators such as Veins[22], VENTOS [48], and TraNS [56] are among the most commonly used tools for simulating vehicular networks. These simulators typically integrate traffic simulation engines like SUMO [197] to generate vehicular mobility patterns and subsequently simulate network communications. Veins, for instance, is a well-established framework that combines SUMO and OMNeT++ [43] to enable detailed vehicular network simulations. VENTOS, for example, focuses on scalable simulation environments, offering capabilities for high-fidelity modeling of large-scale vehicular networks by utilizing efficient algorithms and distributed simulation techniques. On the other hand, TraNS bridges the gap between traffic and network simulators by allowing dynamic interaction between them, which is crucial for accurate evaluation of VANET protocols under realistic traffic conditions. While most platforms rely on traffic simulators, none of these platforms supports the direct ingestion of real-world traffic data being collected through sensors in smart cities[1]. Unlike the former, the latter does not retain unique vehicular information due to privacy constraints and merely record the number of vehicles occurring at a specific location per unit of time.

Integrating traffic data from real-world environments can also be challenging and requires significant preprocessing, resulting in limited accessibility and usability within mainstream simulation applications. Although NS-3 and OMNeT++ can be used to import external mobility data, this process requires extensive data formatting. It cannot directly import large-scale datasets such as those provided by the Department for Transport (DfT) in the United Kingdom.

This gap underscores the need for a simulation platform that can seamlessly integrate diverse real-world traffic datasets, offering a more accurate reflection of urban traffic patterns and their impact on vehicular communication networks.

This chapter seeks to extend the capabilities of SimulatorBridger [1] by incorporating

[1] μ`https://newcastle.urbanobservatory.ac.uk/`

various real-world traffic datasets, such as those provided by the United Kingdom's Department for Transport (DfT) [2], thereby moving beyond exclusive reliance on the SUMO traffic simulator.

The core motivation of this chapter is to prove the flexibility and usability of the *SimulatorBridger* platform by enabling it to process different traffic data formats. By integrating real-world traffic data from the DfT into *SimulatorBridger*, our platform will improve its ability to simulate a wide range of urban traffic scenarios more accurately and fill the gap of existing VANET simulators. Additionally, there is a need to investigate the impact of using different types of traffic data on the performance of simulations. Specifically, we are interested in understanding how the detailed car traces produced by SUMO compare to the static data from DfT in terms of their effect on communication delays within the simulation. This understanding is crucial for optimizing the design and application of simulation tools in urban traffic research.

This chapter introduces an enhanced simulation platform, *SimulatorBridgerDfT*, designed to simulate real-world traffic scenarios by integrating the dynamic IoT simulator *IoTSimOsmosisRES* [25] with real traffic data from the DfT in CSV format. This improved version builds upon the original *SimulatorBridger* [1], which connected the dynamic *IoTSimOsmosisRES* with *SUMO* [197], but now leverages various data formats.

As a result of this enhancement, *SimulatorBridger* will be capable of processing multiple data formats, such as CSV data from the Department for Transport's website, instead of relying solely on *SUMO* for traffic data generation. This capability increases the simulator's versatility and applicability across a range of urban traffic scenarios. Therefore, this study examines whether *SimulatorBridger* can effectively process different real traffic data formats, including DfT data, without depending on *SUMO*. Furthermore, the study aims to determine whether we can rightfully estimate real-time communication times by not necessarily relying on the precise location of each IoT device at each instant of time as in our previous work [1], and whether just knowing the number of devices communicating per time unit would suffice. As a result, valuable insight into the design and application of urban traffic simulation tools can be gained.

To evaluate these enhancements, we address the following research questions:

[2] μ`https://roadtraffic.dft.gov.uk/downloads`

- **RQ1 (Section: 6.4.1): Does counting IoT device connections suffice to estimate real-time communication times rather than tracking specific locations?**
- **RQ2 (Section: 6.4.2): Can *SimulatorBridger* process different formats of real traffic data without relying on SUMO for data generation?**

Contributions

This chapter makes several key contributions to the field of urban traffic simulation:

Extension of SimulatorBridger: The SimulatorBridger platform has been enhanced to support a variety of real traffic data formats. This extension enables the simulator to utilize real-world traffic data in CSV format from sources such as the Department for Transport (DfT), significantly improving the platform's functionality and usability, which is unprecedented in current VANET simulator literature.

Development of SimulatorBridgerDfT: The design and development of SimulatorBridgerDfT represents a major advancement, integrating the dynamic IoTSim-OsmosisRES with DfT traffic data. This new version increases the application's flexibility, making it applicable to a broader range of urban traffic scenarios, thereby extending its utility in diverse research and planning contexts.

Efficient Real-Time Communication Estimation via IoT Connection Counts: This research establishes that counting IoT device connections can serve as an accurate method for estimating real-time communication times in vehicular networks. The ability to rely on connection counts rather than precise location tracking simplifies the simulation process, reducing computational complexity and enhancing the practicality of real-time communication analysis. This finding broadens the potential use cases for traffic simulations, particularly in scenarios where data granularity is limited, making it a scalable solution for large-scale traffic management and IoT applications.

Evaluation of Data Format Handling: This chapter addresses the research question of whether SimulatorBridger can process different formats of real traffic data without

relying on SUMO for data generation. The evaluation demonstrates that the platform can effectively handle DfT traffic data, showcasing its robustness and adaptability in managing various data formats.

By addressing these contributions, this chapter advances the field of VANET simulation, offering a more functional and flexible tool for researchers and practitioners in traffic management and urban planning.

The remainder of this chapter is structured as follows. Following a discussion of the related works, Section §6.2 details the real traffic data collection and analysis processes. Next, Section §6.3 provides a comprehensive overview of the design and architecture of *SimulatorBridgerDfT*. The experimental setup, execution, and results are discussed and analyzed in Section §6.4. In Section §6.5, the research questions are addressed. The conclusion and future work is provided in Section §6.6.

6.2 Data Collection

This section outlines the processes of collecting and analyzing real traffic data from the UK Department for Transport (DfT). There are two types of data: the automatic traffic counters (ATC) and the average annual daily flow (AADF). The ATC provides average speed and traffic flow rates for four types of vehicles every 15 minutes, where vehicle types are classified by length. The AADF provides average daily flow by manual counting, and the data contains the actual types of vehicles [218]. Due to the correctness guarantees ensured by the manual curation of the latter dataset, we used AADF's estimated annual average daily flows (AADFs) for major and minor roads for the period of 2000-2022. It shows the average over a full year of vehicles passing a sensor point in the road network daily. The dataset contains 4815505 rows, each row represents a vehicle passing in front of a Road Side Unit (RSU), e.g. a traffic light. To answer our research question, we consider each RSU as a edge node receiving communications starting from IoT devices located within vehicles. By contextualizing this scenario within a Smart-City environment, we wonder whether an average Cloud infrastructure could bear the recorded amount of daily communications.

By comparing the real traffic data (DfT) with those generated by the traffic simulator SUMO, the trace file in SUMO divides the vehicles in time-steps of a second. While the

DfT dataset does not have a time-based division, each row represents a RSU, its location, the number of vehicles that communicated with the RSU, the date, the year, and the time. Moreover, the time of each communication in the DfT dataset is expressed in hours instead of seconds, as in the SUMO traffic data output.

6.3 SimulatorBridgerDfT's Architecture

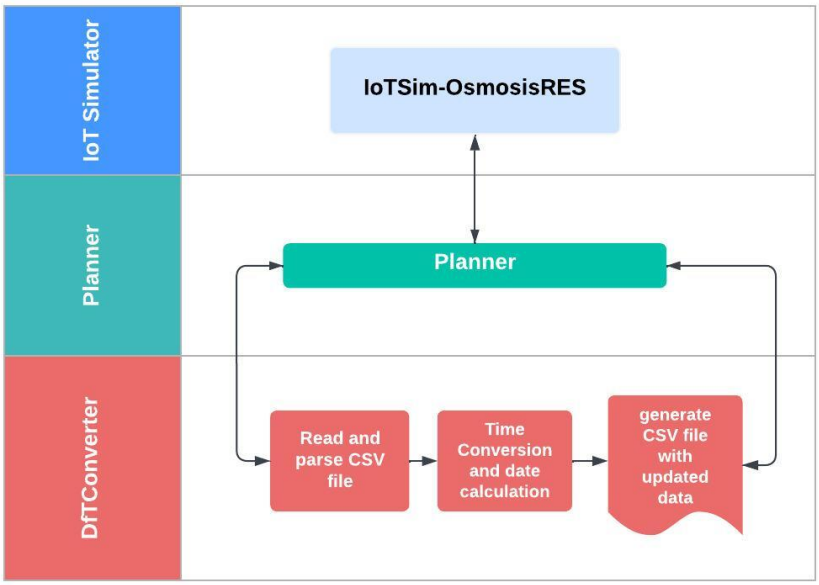


Figure 6.1: Architecture of SimulatorBridgerDfT

The original version of SimulatorBridger [1] bridges vehicular generator SUMO, providing the location in time for both RSU and vehicles, an extension of IoTSim-OsmosisRES [25], where all mobile IoT devices establish communication to the cloud depending on the proximity to a RSU-Edge node. The computation is carried out over an osmotic architecture, and vehicles transmit with a given transmission rate. SimulatorBridgerDfT extends the former by mainly implementing a new Java class, `DfTConverter`, which serves as a Bridge pattern [219] to convert the DfT representation towards the internal SimulatorBridger, thus acting as a vehicular generator estimating the number of vehicles present from the AADF data. This allows SimulatorBridger to simulate network traffic conditions from real-world Smart-City data. As SimulatorBridger was conceived modularly (Figure 6.1), where each component can be easily reconfigurable through configuration files that might load different classes at run time, this allows us to easily extend the former to support

this novel scenario.

6.3.1 Architectural Modifications in SimulatorBridgerDfT

Introducing real-world traffic data from the UK Department for Transport (DfT) into SimulatorBridgerDfT required minor modifications to the existing architecture. However, these changes did not necessitate a complete overhaul of the system. The reasons for this are rooted in the original design principles, which emphasise the modularity and adaptability of the phases discussed below.

Edge/RSU Locations SimulatorBridger initially supported ingesting RSU nodes acting as the Edges of an Osomotic architecture by reading the urban layout information encoded in the SUMO datasets. By extracting the position of each RSU, we can also determine the relative position of each IoT device and the former, thus determining whether any communication should be established. Furthermore, we ensure that each Edge/RSU node stays active for the entire simulation. This allowed the existing RSU framework to be reused without modification, thereby preserving the overall architectural structure. When integrating DfT data, the DfTConverter module now processes the traffic information recorded at each RSU, and records the geographical position of the former to determine the position of a novel Edge node. We then pertain the same correctness guarantees as in the former simulation without requiring any extension of the former RSU/Edge representation.

Dynamic Vehicle Generation A key feature of the original SimulatorBridger architecture is its ability to simulate dynamic IoT vehicles that can move across various locations within the city at different times. This capability is rooted in the system's flexible design, which allows for continuously updating vehicle positions as they roam through the simulation environment. Because the architecture is already equipped to handle the movement of individual vehicles in a time-dependent manner, it naturally extends to scenarios where multiple vehicles are generated at specific locations and times based on real-world data.

When integrating DfT datasets, which provide detailed records of vehicle counts at particular locations and times, the existing system can generate the corresponding number

of vehicles at those points without requiring any major structural changes. Since SimulatorBridger is already designed to dynamically update the position of a single IoT vehicle, it can similarly manage the simultaneous generation of multiple vehicles at any given location and time. This is achieved by adapting the input data to match the real-world conditions provided by the DfT datasets, rather than altering the underlying architecture.

Given this tool's flexibility, the present chapter aims to test its scalability while considering the simulation of vehicles coming from realistic traffic patterns, all while maintaining the original framework of SimulatorBridger.

6.3.2 `DfTConverter` ***Bridger***

As shown in Figure 6.1, the DfTConverter module plays a crucial role in the SimulatorBridgerDfT architecture by parsing and processing DfT traffic data. The module begins by loading CSV files from the DfT, which contain detailed information on vehicle counts, geographical locations, timestamps, and RSU interactions. The DfTConverter utilizes the OpenCSV library to extract relevant columns from the dataset, such as vehicle counts and timestamps, and stores this data in a structured format that the simulator can process. This data is then used to align the traffic patterns within the simulation, ensuring that the real-world data accurately informs the behavior of virtual vehicles and RSUs within the simulated environment. As shown in Figure 6.1, the processing flow highlights the conversion and integration steps, ensuring that the data is consistent with the simulation's requirements.

Time Conversion and Calculation To align the DfT dataset with the simulator's time units, it's crucial to convert timestamps from hours to seconds. The DfT dataset timestamps are initially in hours, indicating the hour of the day when an event occurred. The conversion process begins by extracting the date and time from the DfT dataset to ensure consistency with the simulation, which uses seconds. The conversion is performed using the equation:

$$\mathsf{conv}_D(d) = \mathit{timedate}(d) - \min_{d' \in D} \mathit{timedate}(d')$$

where $\mathsf{conv}_D(d)$ represents the converted time for the event d in seconds, $\mathit{timedate}(d)$ is

the timestamp of the specific event d as a datetime value considering both day hour and date information, and $\min_{d' \in D} timedate(d')$ denotes the minimum timestamp in the entire DfT dataset D. This minimum timestamp serves as the reference point for the conversion, so to ensure that the simulation starts at time zero. The DfTConverter module ensures that SimulatorBridgerDfT operates with high temporal accuracy by converting hours to seconds and synchronizing them with the Unix epoch. This function then yields the new time interval for the simulation, which is comprised between the first timestamp, being mapped to zero, and the last simulation time.

Vehicle Generation Creating a realistic simulation environment requires the simulator to generate virtual vehicles given the rows in the DfT dataset. The simulator creates virtual vehicles corresponding to each communication event recorded in the dataset, considering each vehicle's geographic location and direction information. Given that all vehicles, when detected, are associated with a specific RSU, we can freely assume that all vehicles within the simulation will communicate. Given the specificity of our Osmotic computation, we can freely assume that each vehicle will communicate with one designated Cloud virtual machine [25, 220].

RSU Integration and Edge network Roadside Units (RSUs), which are traffic lights, serve as vital communication points within the simulator. Through the DfTConverter class, unique RSUs are identified, generated, and seamlessly integrated into the simulator's environment. RSU integration is demonstrated in the following algorithm: identifying unique RSUs within the dataset, creating RSU objects with location and communication attributes, and integrating RSUs into the simulation environment as communication points. As, in this scenario, each Edge network will contain one single RSU component, we can freely assume that its architecture comprises one switch node and a Gateway node for connecting to the backbone network.

Osmotic Architecture Generation A network of connections is constructed between vehicles and RSUs to facilitate effective communication. The entire communication infrastructure is based on multiple Edge networks, one per RSU from the DfT dataset, a Cloud network that processes all the communication requests from the IoT clients, and a

backbone network that connects the two exploiting a *bus topology*. In our previous implementation [1], the overall number of processing units within the cloud was proportional to the number of expected devices to ensure better scalability. As this might lead to an excessive amount and number of resources that are scarcely handled by our simulator, we extended the former also to support a fixed amount of cloud nodes. Furthermore, the cloud architecture is a simple generalization of the common *tree topology*, where the root is the main gateway (or data centre) and the core leaf nodes correspond to the nodes running virtual machines processing the information from the IoT nodes; *links aggregation* policies are enacted by adding intermediate buffer switches between the gateway and the core leaf nodes [221], to increase the overall network throughput and ensure network redundancy [222].

6.4 Experimental Analysis

In this section, we present the experimental setup and execution of the extended proposed version, *SimulatorBridgerDfT*, which is available in Github [3].

6.4.1 Evaluating Real-Time Communication Estimations Using SUMO-Generated Mobility Traces and DfT Dataset

Given both that mobile and static IoT devices will contribute to internet traffic, we wonder whether we can rightfully estimate real-time communication times by not necessarily relying on the precise location of each IoT device at each instant of tine as in mobility traces (MTs) generated by SUMO [1], and whether just knowing the number of devices communicating per time unit such as in DfT dataset would suffice.

We investigate this by first generating realistic traffic patterns expressed as MTs through SUMO. We choose to consider those occurring in Bologna[4], from which we derived a dataset solely comprising the number of communications per time through edge modes – guaranteeing a comparable number of communication. The MT generated from SUMO contains further vehicular information, such as vehicle velocity and rotations, lane occupation, and urban mobility patterns, which might be ignored as they should not affect

[3] μhttps://github.com/jackbergus/SimulatorBridger/releases/tag/v0.5
[4] μhttps://github.com/DLR-TS/sumo-scenarios/tree/main/bologna/acosta

the communication behaviour. As of now, there is no traffic dataset available in both DfT format and MT format. And due to the fact that DFT datasets are in CSV format, we need to convert Bologna datasets from MT format to CSV format.

MCD Methodology:

We used Minimum Common Denominator (MCD) methodology which is available in GitHub[5]. This comprises both the conversion from the MT to CSV format where we lose IoT trace information, for then reconstructing such information from the information available from the CSV aggregated data. The MCD approach is designed to integrate and compare traffic data from different sources, such as MT and DfT datasets, ensuring consistency in the number of connections per simulated time. The method involves converting MT data to CSV format using SimulatorBridger, which captures the location of edge nodes and the number of communications per time tick. This conversion allows the simulation to operate under the same assumptions as the MT scenario, enabling direct comparisons between datasets.

In the simulation, CSV data is used to model the total number of connections at each network edge, without tracking individual vehicle locations. The MCD approach then generates vehicles for each connection at each time step, simulating communication behaviors similar to the original MT data. The method also includes parsing edge node information and connection counts, creating and mapping IoT devices to their respective edges based on the number of connections. This process allows the simulation to replicate real-world communication scenarios with a focus on the consistent and scalable modeling of IoT-enabled networks.

A comparison is made between Bologna datasets analyzed and compared in MT and CSV/MCD scenarios, with a particular focus on communication behavior, running time, and processing time:

1. **IoT Device Density Distribution:** We examine IoT density distribution over the simulation time for both MCD and MT scenarios. From Figure 6.2 and 6.3, we notice variations within the number of IoT and battery consumption trends on MCD and MT. The figure illustrates that while the MCD configuration exhibits

[5] μ`https://github.com/jackbergus/SimulatorBridger/releases/tag/v0.4`

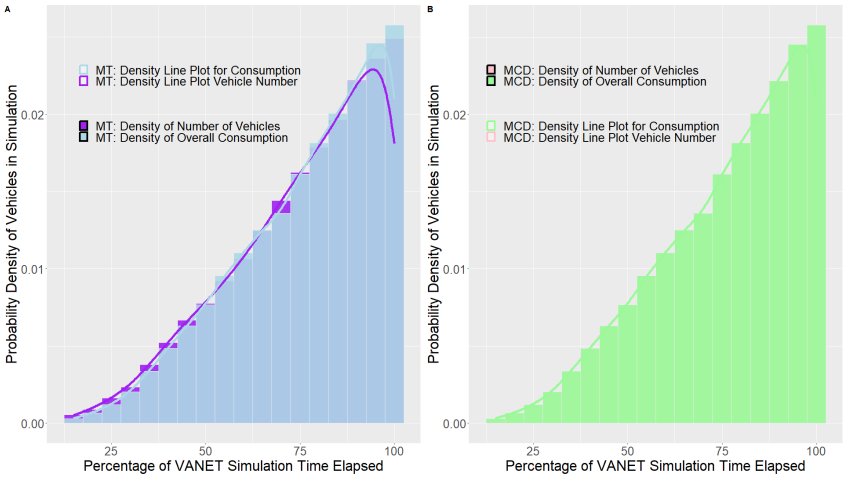


Figure 6.2: Probability Density of Vehicles in Simulation

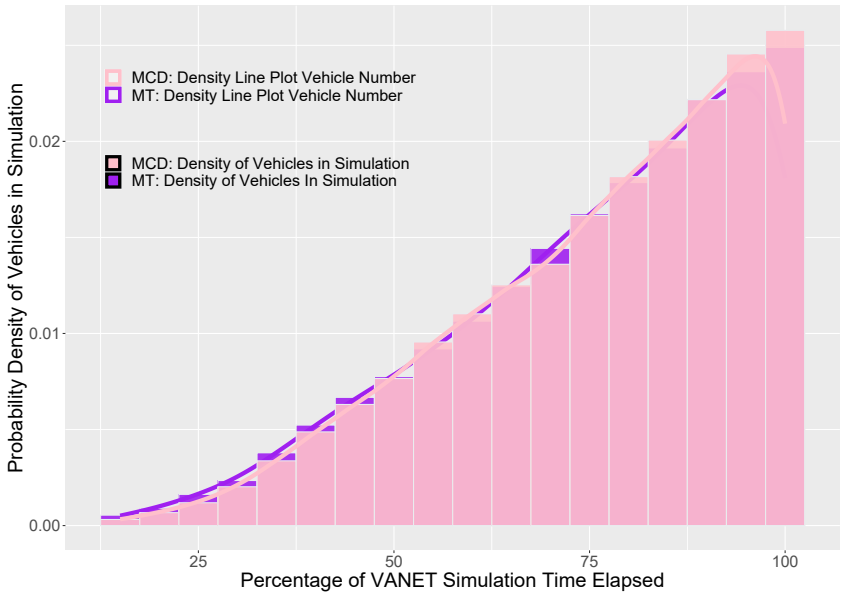


Figure 6.3: Probability Density of Vehicles in Simulation

a straightforward linear relationship, the MT one exhibits more varied behaviour. The peaks and troughs in density plots indicate subtle differences in how each format handles vehicle data. The stability in the MCD data can be explained by the static nature of the MCD method, where vehicles are represented only once at discrete time step rather than as continuous traces. Consequently, the data does not capture the detailed movement and interactions of vehicles over time, leading to a more uniform and predictable density pattern. Hence, the detailed movement data provided by the trace files in MT, contrasts with the more static data provided by the MCD method. This dynamic representation in MT allows for more precise modeling of traffic flow and vehicle interactions, potentially resulting

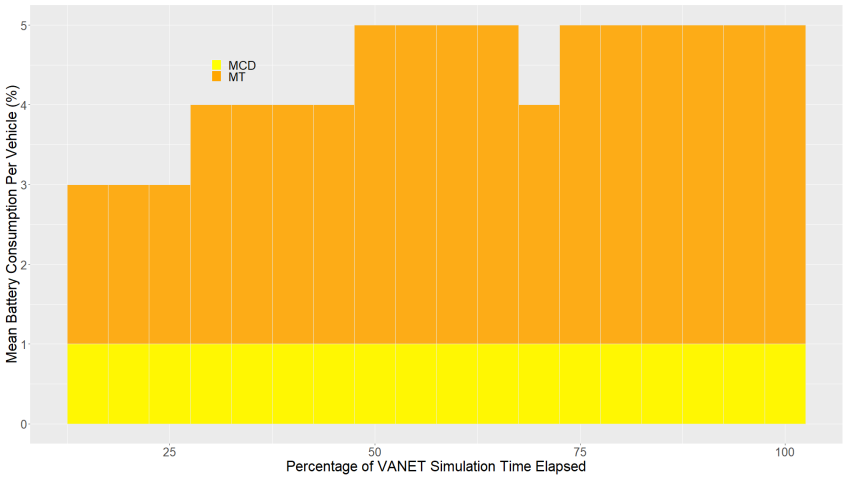


Figure 6.4: Average Buttery Consumption Per Vehicle

in more accurate visualizations of variations in vehicle density.

2. **IoT Battery Information:** We also consider results for IoT battery consumption. To facilitate the analysis, the IoT battery consumption rate has been set arbitrarily at 1.02% per vehicle communication. Based on the assumption that embedded IoT devices cannot be recharged, we correlated the number of packets sent by an IoT device with its battery consumption in order to better analyze the simulator's correctness. However, different battery parameters could lead to varying results in the simulation. The battery parameters of this experiment, such as maximum battery capacity, battery sensing rate, and battery sending rate, have been set to specific values in order to make the current generic dataset consistent with our previous experiment discussed in [1].

   Results from Figure 6.4 show a stable trend over time in both scenarios. Based on the MCD scenario, the power consumption is predictable and displays a consistent energy usage pattern. This consistency in the MCD scenario is due to the simulator's method of creating new unique vehicle at each simulation time step. In the MCD configuration, vehicles are present only once per simulation time step, which ensures that vehicles are present only once during the simulation. On the other hand, the MT scenario shows a higher degree of variability in power consumption. This variability is a direct consequence of the detailed and dynamic interactions between vehicles represented in MT. MT provides comprehensive movement data for each vehicle, allowing a single vehicle to exist at multiple points throughout the simulation. This detailed movement data captures the complex interactions and

variations in vehicle behaviour over time, leading to variations in power consumption. Accordingly, while MCD provides a stable power consumption pattern due to its simplified vehicle generation process, MT offers a more detailed and variable pattern that reflects the intricate interactions between vehicles over time. This variability, highlights the impact of vehicle movements on battery life and underscores the need for precise IoT data to accurately capture and analyze detailed aspects of IoT devices, such as battery consumption. Aggregated data alone cannot provide the necessary granularity to infer detailed insights, emphasizing the importance of detailed simulation data for comprehensive analysis.

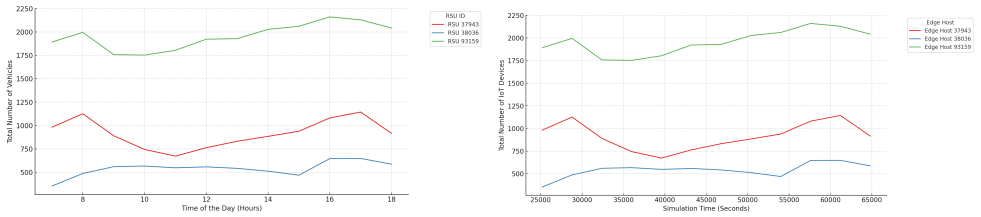
Comparing the communication patterns between the two scenarios: MT and CSV/DfT, our results illustrate the simulator's ability to accurately replicate communication patterns using both detailed mobility traces (MT) and CSV/DfT dataset generated by MCD approach, which considers only the number of communications per time unit.

6.4.2 Correlation Analysis Between Simulated and Real-World Traffic Data Using SimulatorBridgerDfT

To answer the second research question, we analyze the ability of SimulatorBridger to process real traffic data in CSV format by examining the communication patterns of vehicles in the Manchester dataset. The analysis is based on the comparison between the real-world DfT dataset and the simulated data generated by the extended simulator *SimulatorBridgerDfT*. It is important to determine whether the simulated data correlates with the actual traffic counts recorded by the RSUs in real time.

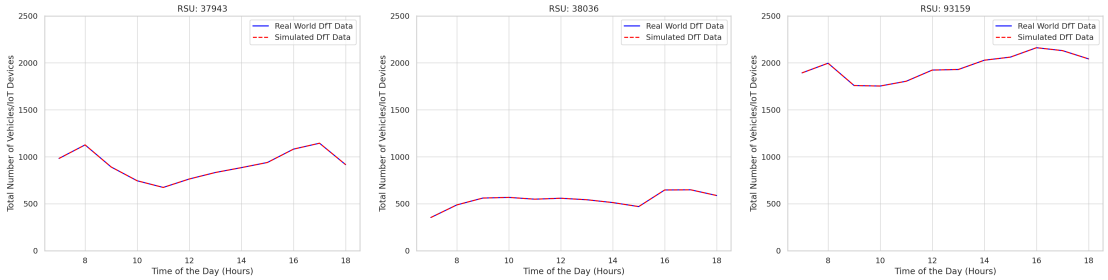
The simulator starts after the transformation above via the `DfTConverter`, which considers vehicles occurring within a single day. We consider interactions between 5G RSUs Edges and the Cloud in an Osmotic architecture, where IoT devices initiate communications [1].

The experiments, as shown in Figure 6.6, compare the communication patterns of three different RSUs between 7 AM and 6 PM using real-world DfT data (straight line) and simulated data generated by the proposed platform *SimulatorBridgerDfT* (dashed line). Regarding the traffic patterns from Figure 6.6a, RSU #37943 shows vehicle counts peaking around 8 AM and rising again around 4 PM. RSU #93159 Figure 6.6c experiences



(a) Real World DfT Dataset (b) Simulated DfT Dataset

Figure 6.5: Communication Pattern for Real World DfT Dataset and Simulated DfT Dataset



(a) RSU #37943 (b) RSU #38036 (c) RSU #93159

Figure 6.6: Comparison of IoT Device Counts and Vehicle Counts for Different RSUs

the highest traffic volumes among the three RSUs of interest: it also demonstrates a strong peak in vehicle counts around 8 AM, followed by stabilization throughout the day. We also consider RSU #38036 Figure 6.6b from a less trafficked area, while exhibiting different rush hour times. Determining whether the simulated data correlates with the actual traffic counts recorded by the RSUs in real time is essential. We investigate the impact of the number of IoT devices on the average communication time to determine the correlation between the two.

1. **Validating Simulated Traffic with Real-World Traffic Data:**

   The two Figures 6.5a, 6.5b provided above display the communication patterns for three different RSUs between 7 AM and 6 PM. The first plot 6.5b represents the simulated data using SimulatorBridgerDfT, while the second plot 6.5a represents the real-world data from the DfT dataset. In these plots, the number of vehicles/IoT devices interacting with each RSU over time is tracked for both the real-world data and the simulated data . Figure 6.6 illustrates the communication patterns for three distinct RSUs. We validate a preliminary simulator correctness by checking whether the amount of vehicles described in the dataset per unit of time matches the number of actual vehicles running within the simulation. Such plots trivially show that the

software re-engineering of the tool and the data parsing and processing that came with it correctly retain the original traffic volume per RSU, as they exhibit the exact same number of vehicles.

For each RSU, the simulated data closely mirrors the real-world vehicle counts, demonstrating the simulator's capability to accurately replicate real-world traffic patterns. This accuracy is further examined by looking at each RSU individually in Figure 6.6. For example, RSU 37943 6.6a shows that both simulated and real-world data exhibit similar trends, with vehicle counts peaking around 8 AM and rising again around 4 PM. The almost identical patterns observed in the first and second plots confirm that SimulatorBridgerDfT can accurately simulate real-world traffic fluctuations during these critical periods.

The relatively stable trend observed throughout the day underscores the simulator's ability to maintain accuracy even in scenarios with less dynamic traffic flows (RSU #38036) 6.6b , ensuring that the simulated data remains representative of real-world conditions.

2. **Impact of Cloud Traffic on User Experience**

We now analyse how the number of vehicles initiating communication with the cloud via RSU affects the average communication time for the vehicles. Given that the network provides a limited amount of resources, we might expect that an increase in vehicular traffic at a specific time of the day will sensibly affect the communication patterns. We can say that the Osmotic Architecture is really robust if such correlation is kept linear and if there are no sudden communication lags that severely affect the communication times after the decrease in the number of vehicles communicating with the RSU. An analysis of IoT device loads across various RSUs provides valuable insights into how network performance varies throughout the day as a function of device load. Mapping the spikes in simulation time data to specific hours of the day shows a clear relationship between the observed changes in communication times and the number of connected IoT devices.

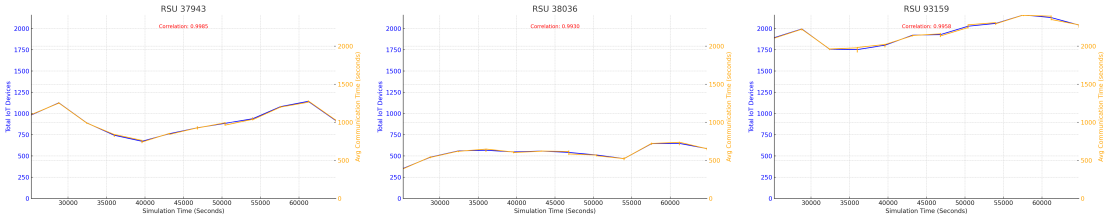
Figures 6.7a, 6.7b, 6.7c show the relationship between the total number of IoT devices communicating with the Edge/RSU and the average communication times. Based on the plots, there is a very strong positive correlation (close to 1) between the

total number of IoT devices and the average communication time across the RSUs. The Pearson correlation coefficients between the total number of IoT devices and the average communication time for each RSU are: RSU 37943: 0.9985, RSU 38036: 0.9930, RSU 93159: 0.9958. This suggests that as the number of communications increases, the average communication time also tends to increase.

RSU #93159 6.7c, in particular, exhibited the highest number of connected IoT devices during the simulation, with spikes occurring between the 28,000 to 30,000-second mark (approximately 7:45 AM to 8:20 AM) and again between 50,000 and 60,000 seconds (corresponding to 1:50 PM to 4:00 PM). These spikes align closely with typical rush hours, which coincide with increased vehicular traffic and a corresponding rise in IoT device connections to the RSUs. Further analysis of real time hourly data supports these findings. The highest IoT device counts for RSU #93159 (Figure 6.6c) occur between 7:00 AM and 8:00 AM, with another spike between 3:00 PM and 5:00 PM. These peak activity periods directly correlate with the spikes observed in the simulation data, suggesting that the increased number of IoT devices during these hours significantly elevates network load, leading to congestion and longer processing times.

However, even with the generation of a substantial amounts of data requiring considerable resources for processing and transmission, the resulting Osmotic architecture proved to be relatively robust to an increase in vehicles. As we do not remark peeks in communication delays when the traffic volume decreases, this remarks that a Smart City infrastructure should be able to tolerate several communicating vehicles communicating with the cloud comparable to realistic traffic volumes.

Thus, rush hour peeks might not seriously impact user experience. This simulator proves crucial in simulating applications that transmit real-time data in smart cities, such as autonomous vehicle systems, smart city infrastructure, and emergency response services, as those applications are particularly vulnerable to significant delays. For instance, if an autonomous vehicle needs to receive a traffic signal update and there is a delay, it could hinder its ability to respond promptly, potentially compromising safety or efficiency.



(a) RSU 37943 (b) RSU 38036 (c) RSU 93159

Figure 6.7: Total Communications and Average Communication Time for Different RSUs

6.5 Discussion

This section aims to explore and answer the research questions posed in this study, based on the analysis and experiment results. Each subsection will present the findings and analyses related to each research question.

RQ1: Would it be sufficient to estimate real-time communication times by counting IoT device connections rather than tracking precise locations?

The results indicate that the MCD approach can indeed provide a reliable estimation of real-time communication times. This is evident from the consistent and predictable patterns observed in IoT device density and battery consumption across simulations using the MCD method. The MCD-generated data, though simplified, maintains consistency in the number of communications per time unit, which is critical for modeling and estimating communication behavior in IoT networks.

However, the analysis also highlights the limitations of the MCD approach. The MT data, with its detailed vehicle movement and interaction information, offers a more nuanced and variable representation of communication patterns. The variability observed in MT scenarios, particularly in IoT device density and battery consumption, underscores the importance of capturing the dynamic nature of vehicle movements and interactions to more accurately reflect real-world conditions. This detailed information, which is lost when converting to a simpler CSV format, can be crucial for applications where precise modeling of traffic flow and vehicle interactions significantly impacts communication performance.

Thus, while the MCD approach offers a scalable and computationally efficient method for estimating communication times, it may not fully capture the complexities of real-time

communication in dynamic IoT environments. For applications where detailed interaction data is less critical, the MCD method provides a sufficient approximation. However, for scenarios requiring a higher degree of accuracy and granularity, reliance on detailed MT data may still be necessary. This suggests that the choice of methodology should be aligned with the specific requirements of the application, balancing the need for computational efficiency with the level of detail required for accurate communication modeling.

RQ2: Is SimulatorBridger capable of processing different formats of real traffic data without relying on SUMO for such data generation?

The detailed analysis of the Manchester dataset provides clear evidence that SimulatorBridgerDfT successfully processes real traffic data in CSV format, without requiring SUMO for data generation. Through the examination of RSU communication patterns, it is evident that SimulatorBridgerDfT accurately processes and simulates real traffic data from the DfT dataset, replicating real-world traffic flows and RSU communication behaviors with high fidelity.

The plots offer compelling evidence that the DfTConverter module within SimulatorBridgerDfT effectively integrates real-world DfT data into the simulation environment. The simulator's ability to closely match the real-world traffic patterns across all three RSUs underscores its versatility and robustness in handling different traffic data formats. This close alignment between real-world and simulated data provides a definitive answer to the second research question. It confirms that SimulatorBridgerDfT is not only capable of processing various traffic data formats but also of delivering valuable insights into real-world traffic and communication dynamics. The visual consistency between the simulated outputs and the actual traffic data validates the effectiveness of the DfTConverter module in achieving realistic simulations.

As a result of this comprehensive evaluation affirms that SimulatorBridgerDfT is highly flexible and effective in processing diverse traffic data sources, successfully addressing the second research question.

6.6 Conclusion and future works

This research successfully extends the SimulatorBridger platform by integrating real-world traffic data from the UK's Department for Transport, resulting in the development of *SimulatorBridgerDfT*. This enhanced capability not only improves the accuracy and realism of traffic simulations but also fills a critical gap in the existing VANET simulator literature, where real-world data integration has been largely absent. The development of the SimulatorBridgerDfT platform marks a crucial step forward, allowing the use of CSV-format data to simulate comprehensive traffic behaviors with greater accuracy and realism. A key component of this platform is the DfTConverter module, which processes raw DfT datasets efficiently to create a realistic and dynamic simulation environment that closely simulates actual urban traffic conditions.

The analysis conducted in this chapter confirms that *SimulatorBridgerDfT* accurately simulate real-world traffic patterns; this simulator proves also the effectiveness of Osmotic architectures for dealing with considerable network traffic initiated by a realistic number of devices. The results highlight the significant impact of IoT device load on network performance, especially during peak traffic periods, without severely affecting communication times in non-rush hour fragments.

In addition, this study examined whether the detailed movement patterns generated by SUMO provide a significant advantage over static data provided by the Department for Transport in terms of communication delays over the static data. Based on our findings, precise node locations and movements may not significantly impact communication patterns from the simulator's perspective, as the interaction with stationary Edge devices remains consistent regardless of vehicle movement. Based on this insight, the DfT datasets have the potential to be used for effective and scalable communication analysis without requiring granular movement information.

With future works, *SimulatorBridgerDfT* can be enhanced by optimizing its scalability and performance, which are critical for handling larger datasets and complex traffic scenarios. An expanded simulation environment will be created by incorporating data sources such as weather and accident reports, as well as social media. The use of advanced network management techniques is essential to reducing communication delays caused by IoT device loads, particularly during peak traffic periods. Furthermore, smart

cities can enhance service quality by studying network performance's effect on user experience. With real-time simulation and feedback capabilities, the platform will manage traffic dynamically, further increasing its utility for Smart City planning.

7

Towards an IoT Simulator Supporting Cyber-Threat Detection Algorithms: IoTSimSecure

Contents

7.1 Introduction

The increasing use of IoT devices in various sectors, from smart homes to industrial environments (§7.2.3), has led to an increased focus on their security and functionality. A critical aspect of IoT device functionality is battery life, which can be severely impacted by malicious activities such as battery-draining attacks. These attacks, often in the form of flooding the device with unnecessary tasks or requests, can lead to rapid battery depletion, making the device useless and potentially disrupting the service it provides.

Consider a smart city infrastructure where IoT devices control traffic lights, monitor environmental conditions, and even manage public transportation schedules, thus entailing the presence of mobility IoT devices [223]. These devices communicate in real-time with central servers, ensuring smooth city operations. Cyber adversaries, aware of the computational limits of these city-wide IoT devices, launch a coordinated computational escalation attack. They flood devices with complex, resource-intensive tasks, aiming to overload their processing capabilities. A traffic light, for instance, overwhelmed by such an attack, could fail to function during peak traffic, leading to traffic jams and potential accidents. Furthermore, an adversary might also target these IoT devices with a battery drain attack [224], intending to disrupt the building's monitoring capabilities. The attack involves sending a flood of wireless communication requests to the sensors, causing them to activate and transmit data more frequently than necessary, leading to rapid battery depletion.

To mitigate this, a system is required that can detect unusual patterns of activity that may indicate a battery-draining attack. Security policies can be used to determine the rules and behaviours indicative of such an attack. These policies are then translated into a detection model that monitors IoT device activity for signs of battery-draining attacks.

The detection model can be continuously updated as new types of attacks are identified, and these updates can be propagated to all MELs entities residing at all edges of the network [141] to maintain a robust defence against battery attacks. This approach ensures that the IoT devices can operate securely and maintain their intended battery life, which is essential for the reliability of the services they provide.

7.1.1 Motivation

To the best of our knowledge, there is currently no IoT simulator that addresses security concerns, especially those related to battery-draining attacks, as IoT device battery support is scarcely supported by current IoT and Osmotic simulators. However, it is very essential that researchers develop an IoT simulation tool that support security and protect the operational integrity and longevity of IoT devices within the smart world, not only smart office building. Every smart scenario around us have battery-powered devices that need to be protected from battery drain attacks.

7.1.2 Contribution

It should be noted from the outset that this simulator was not implemented, and the contributions detailed below are limited to conceptual design and potential applications. The work presented in this chapter makes several significant contributions to the field of IoT security, particularly in the area of battery-draining attacks:

Comprehensive Analysis of Battery-Draining Attacks: The chapter provides an in-depth exploration of the various types of battery-draining attacks, detailing their mechanisms and the ways in which they compromise the functionality and longevity of IoT devices. This analysis highlights the operational and financial impacts of such attacks, underscoring the urgent need for effective mitigation strategies.

Introduction of IoTSimSecure: The chapter introduces IoTSimSecure, a novel simulation framework specifically designed to detect the security challenges posed by battery-draining attacks. IoTSimSecure stands out by offering advanced features for detecting and preventing such attacks, thus providing a robust simulator platform for cybersecurity measures.

Support for Diverse Detection Algorithms: IoTSimSecure supports a range of detection algorithms, including threshold-based detection and Exponential Weighted Moving Average (EWMA) techniques. This flexibility allows for comprehensive analysis and testing of various security strategies, thus enhancing the simulator's ability to develop effective countermeasures against battery-draining attacks.

Realistic Use Case Scenarios: The chapter presents detailed use case scenarios in smart buildings, industrial complexes, and healthcare systems, illustrating the potential consequences of battery-draining attacks. These scenarios demonstrate the critical need for a dedicated IoT security simulator and provide practical contexts for applying and testing IoTSimSecure.

As a result, this chapter not only identifies and addresses a critical vulnerability in IoT systems but also contributes a novel and adaptable simulation tool that enhances the resilience and security of these systems. The insights and tools provided by this research are essential for ensuring the sustained operation and protection of IoT devices in our increasingly connected world. By addressing a significant gap in current IoT simulation tools and proposing a comprehensive solution, this research lays the groundwork for future advancements in IoT security. The development of IoTSimSecure as a modular, extensible simulator opens up greater opportunities for protecting IoT ecosystems from cyber threats.

After providing a literature review of current IoT simulators (§7.2.2), we observe that the major gap across these is that no simulator concerns with cyber-security simulations, especially battery depletion attacks (§7.2.1), which are of growing importance in IoT systems. For this, we propose a novel IoTSimSecure simulator framework (§7.3), supporting both IoT battery, attackers as agents (§7.3.2), while providing a testbed for existing cyber-security AI algorithms acting as further distinct agents within the simulation running over MicroELements of our osmotic architecture (§7.3.1). Our work can serve as a foundation for the development of more advanced security features for IoT simulators as well as strategies for mitigating attacks by researchers.

7.2 Related Works

7.2.1 Battery-Draining Attacks

Battery drain attacks not only disrupt the immediate functionality of the devices but also have long-term financial and operational impacts due to the immediate need for battery replacement and potential security breaches during downtime. These types of attacks are specifically designed to deplete the energy reserves of IoT devices, compromising their

functionality and longevity. They range from jamming communications to creating unnecessary network traffic and manipulating data routing to increase power consumption.

Denial of Sleep Smart devices can be forced to expend additional energy through various attacks. Authors in [225] investigate three specific denial of sleep attacks in their work. The first, termed 'Ding Dong Ditching,' is a composite attack that integrates four different denial of sleep strategies, increasing the possibility of at least one attack draining the device's battery. The initial attack is a jamming strategy that disrupts network functionality for users by emitting an RF signal, known as a constant jammer. This approach continuously transmits random data bits, disregarding MAC layer protocols by not waiting for an idle channel. The subsequent phase involves broadcasting, where an adversary dispatches a ping to each host, resulting in an ICMP response. This data can then be exploited for more sophisticated attacks, potentially controlling other network nodes. The final component is the droplet attack, which involves transmitting the start of an 802.15.4 frame and then stopping, resulting in the receiver remaining ready. Employing these attacks in isolation or combination can effectively breach defences and deplete the energy reserves of peripheral devices [226].

Flooding Attack The predominant form of flooding attack is known as 'Hello' Flooding. In this scenario, a malicious RPL (IPv6 Routing Protocol for Low-Power) node generates large volumes of traffic by dispatching DIS (DODAG Information Solicitation) messages to other nodes, prompting them to reply with DIO (DODAG Information Object) messages, which contain information about the rank of the broadcasting node, the Objective Function (OF), and the Destination Oriented Directed Acyclic Graph ID. This creates network congestion and drains the nodes' energy. Furthermore, if the attacker's broadcast range is sufficiently large, they may deceive nodes into believing they are neighbours, resulting in misdirected packet forwarding and further network congestion [227].

Vampire Attack Vampire attacks impose a continuous load on the limited battery resources of a device, leading to its failure. There are two main types of attacks: the stretch attack and the carousel attack. The stretch attack uses packet headers to route data along an unnecessarily long path within a network, resulting in an increase in energy

consumption even without other attacks. Carousel attacks, on the other hand, loop packets within the network before reaching their destination. It may be difficult to detect and counter these attacks when they are used alone, as they are often unnoticed and challenging to detect [228]. Trying to detect these attacks often results in defence mechanisms consuming more power than the attacks they're trying to stop. One proposed solution is implementing a zero-power notification system, which would raise an alert upon detecting an attack without significantly increasing power usage [229].

7.2.2 IoT Simulators

As previously outlined in IoT simulators literature review chapter (3), a comprehensive analysis of current IoT simulators has been conducted, evaluating them across several critical dimensions such as implementation language, availability, scalability, and specialized support for features pertinent to IoT device mobility, Software-Defined Networking (SDN), energy models, and the integration of renewable energy sources, as shown in table 3.5. This analysis highlighted the varied capabilities of simulators like MyiFogSim, EdgeCloudSim, and SimulateIoT-Mobile in terms of mobility support, as well as the unique positioning of IoTSim-Osmosis and its extension, IoTSim-OsmosisRES, in supporting SDN and energy-efficient IoT deployments. The findings from this review are instrumental in understanding the landscape of IoT simulation tools, serving as a foundation for our subsequent discussions on the need for enhanced security features within these simulators. The detailed examination of these simulators, including their limitations and strengths in simulating complex, scalable IoT environments, is vital for identifying gaps in current research and development efforts, particularly in the context of security within IoT ecosystems. For a more in-depth exploration of these simulators and their comparative analysis, readers are referred back to the Literature Review chapter (3).

Limitation of SimulatorBridger As SimulatorBridger [1] is the extension of IoTSim-OsmosisRES for mobility scenarios with support for IoT device batteries in the context of Osmotic computing, we then pick this one as a potential candidate for an extension for supporting Cyber-Threat detection algorithms. Before further extending SimulatorBridger, we need to assess which realism requirements are posed by the simulation and detection of the attacks and whether the best-fit simulator can be extended to take up

the challenge. A negative response to this will then motivate the definition of a new simulator.

Based on the current version of SimulatorBridger, it is notable that the simulation capabilities are limited, particularly in terms of simulating battery-draining attacks and extending the simulator with new agents, such as attackers and cyber-security diagnostic tools. This simulator cannot support direct communications between IoT devices, as it only facilitates interactions between IoT devices and Cloud nodes via Edge nodes. This architectural limitation is significant given that many current battery-draining attacks involve direct communication between IoT devices in IoT environments. By engaging in overly extensive, unnecessary communication requests or other energy-consuming activities, attackers target another IoT device to drain the battery. To address this, it is necessary to set up a whole IPv6 infrastructure in the simulator, which is currently not available. This gap in functionality means that SimulatorBridger cannot effectively model complex attack scenarios or provide a thorough understanding of security attacks, especially those involving direct communication between IoT devices, which are increasingly common in modern IoT networks. A fundamental understanding of the security of IoT networks can be obtained only by significantly revising such a simulator: due to the simulator's limited interaction capabilities and the missing IPv6 infrastructure, its applicability is currently constrained. So, we need to propose a new simulator that can address this limited interaction capabilities, therefore, support cyber-security detection algorithms.

7.2.3 Further Use Case Scenarios

Due to the importance of simulating cyber threats in IoT scenarios, such a simulator should also be able to mimic non-mobility-based IoT nodes. This should be trivial, as this boils down to assuming that such devices will never change their position.

Smart Building Consider a smart building equipped with IoT sensors for monitoring temperature, occupancy, and security. These sensors are typically battery-powered to allow placement flexibility and reduce installation costs associated with wiring. An adversary targets these IoT devices with a battery drain attack, intending to disrupt the building's monitoring capabilities [141, 181, 230].

Industry 4.0 Consider a scenario in a smart industrial complex where IoT devices monitor machines, track inventory, and even manage access controls. These devices, often battery-powered to ensure uninterrupted service, are the lifeblood of operations, ensuring efficiency and safety. Within this complex, battery-operated IoT sensors are strategically placed on cargo containers to monitor temperature, ensuring that sensitive goods are stored optimally. These sensors relay real-time data to a central system, which makes necessary adjustments. Recognizing the critical nature of these sensors, cyber attackers target the battery management systems of these IoT devices. By exploiting vulnerabilities, they can either cause premature battery drainage or manipulate the sensors to report inaccurate battery levels. This could lead to goods being stored at incorrect temperatures due to perceived sensor outages or malfunctions, potentially resulting in significant financial losses [188, 189].

e-Health Consider a healthcare system where IoT devices are integral to monitoring medical equipment health, tracking patient care metrics, and regulating energy use. These devices constantly feed data into a central system, facilitating predictive maintenance of medical machinery and guaranteeing the highest standards of patient care. It is possible for attackers to result in rapid battery depletion or to falsify the sensor's battery status. Consequently, medical equipment could run without proper oversight, potentially leading to costly malfunctions or, more critically, compromising the safety of patients [154, 182–184].

In summary, the use case involves deploying edge gateways to detect and mitigate IoT device battery-draining attacks. This is motivated by the need to protect the longevity and functionality of these devices, which are increasingly integral to the operation of modern smart environments.

7.3 Envisioning the IoTSimSecure Simulator

The proposed IoTSimSecure simulator should support various detection strategies, including threshold-based detection. IoTSimSecure would support the monitoring and analysing network traffic patterns in real-time, allowing EWMA [231] and threshold-based techniques to detect anomalies indicative of potential flooding attacks. The latter

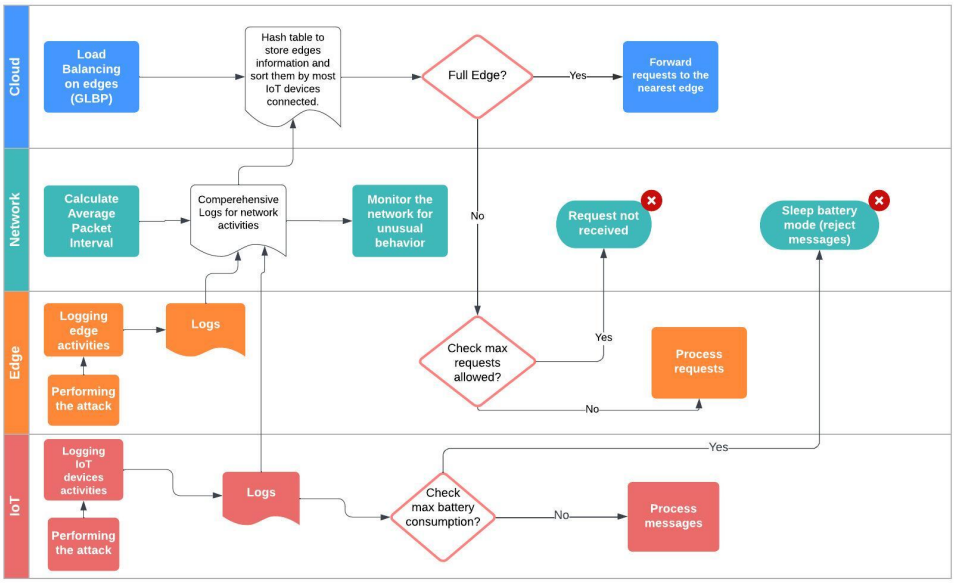


Figure 7.1: The Proposed IoTSimSecure Architecture

aims to identify unusual spikes in network interactions that could signify an attempt to drain the battery of IoT devices through excessive communication requests. Furthermore, this simulator requires the explicit presence of a perpetrator orchestrating the attack, as the cyber-security algorithm should also be tested for differentiating between malicious attempts to attack nodes in the network from any unintentional device malfunctioning affecting the nodes, from the absence of both behaviours. This is a required feature, as detecting the presence of the attack is just the first step before applying counteracting measures for neutralising the attacker. As this simulator should schedule activities for each agent and given that SimulatorBridger, the fittest simulator, does not simulate communications as pervasively as required to mimic packet transmission correctly, we cannot extend this simulator despite its support for IoT battery consumption. We then envision a new simulator IoTSimSecure supporting different malicious attacks and attack detection algorithms, so to better outline, Figure 7.1 describes our proposed IoTSimSecure.

7.3.1 MEL for Cyber-Threat recognition

In osmotic computing, each IoT device directly interacts with MicroELements (MEL) software components that might be instantiated anywhere in a given network associated with Edges [141], providing a dynamic and efficient interface for real-time data processing and decision-making at the network's edge. In the context of Osmotic computing, this helps distribute the computation across the network while minimising the responsiveness

of IoT apps without requiring to send data to the cloud for pre-filtering steps. In the Cybersecurity realm, this might be further extended to better auditing the network in sensible spots to be protected. This computation capability is then crucial for the timely detection and mitigation of battery attacks and positions MEL as an ideal platform for hosting cyber-police agents. Using MEL's computational resources and proximity to data sources, these agents can monitor IoT device traffic effectively, ensuring security measures are enforced promptly and efficiently without imposing additional latency or battery drain by directly auditing the environment where the attack is occurring. As a result, implementing the cyber-police agent within MEL aligns perfectly with its role in managing and securing IoT environments, as this will increase the chance of detecting the attacks as the controlling algorithm will be pervasive throughout the network. This will then allow future works for also testing distributed versions of cyber-attack prevention algorithms.

For instance, a MEL could rapidly process incoming data from IoT sensors and apply the detection model to identify potential battery-draining activities. If an attack is detected, the MEL can take pre-defined actions, such as alerting administrators or temporarily isolating the affected device to prevent further battery depletion.

The simulation can support responsive and adaptive security measures by injecting an external AI security model into a robust IoT simulator. This integration will reduce the risk of battery drain attacks and ensure the reliability and accessibility of energy sources such as batteries. As a result, IoT devices are protected from drain battery attacks, ensuring continuous and uninterrupted operation.

Although flooding attacks and selective forwarding attacks have different objectives and effects on the network, both involve sending many messages to the target node. As a result, to detect these attacks, we need to define a security policy that specifies acceptable usage patterns and energy consumption rates for IoT devices. The policy includes thresholds for the number of requests a device should handle within a specific interval of time and the expected battery life under normal operating conditions. So, we need a detection model based on the security policy, which includes patterns and signatures of known battery drain attacks. The model might exploit external machine learning algorithms trained on typical and attack scenarios to differentiate between regular traffic and potential battery drain activities. Also, by setting a threshold for the number of

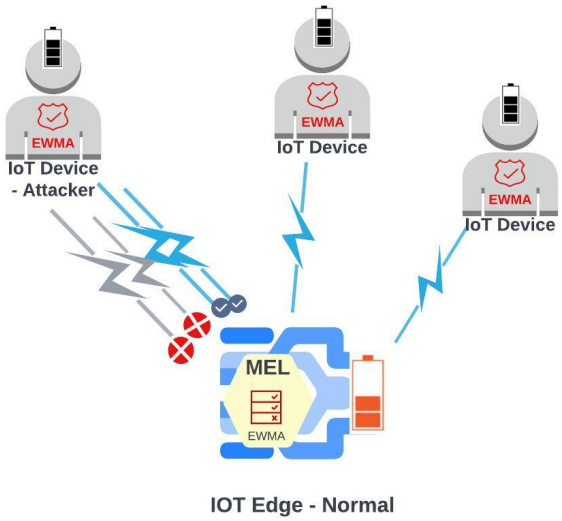
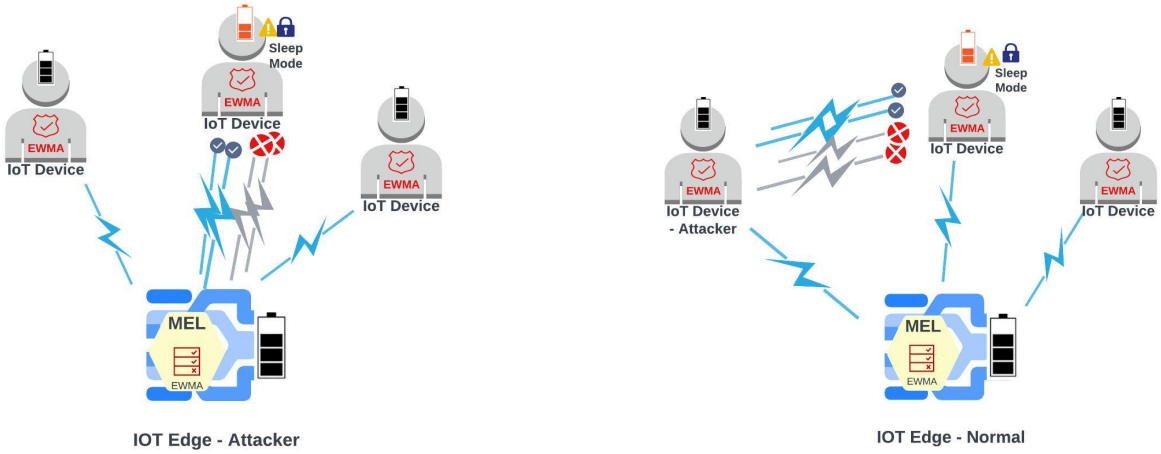


Figure 7.2: Scenario 1: Attacking the battery of Edge device: implementing EWMA for number of requests received

requests an edge can accept, the network can be prevented from becoming overwhelmed. In addition, the maximum battery consumption that each IoT device can reach should be set at a certain threshold. Edge devices with computational capabilities, such as GPU acceleration, are deployed within the proximity of clusters of IoT devices. They are responsible for real-time monitoring of IoT traffic, applying the detection model to identify anomalies indicative of a battery drain attack. The edge devices continuously analyze the incoming and outgoing traffic of the IoT devices.

7.3.2 Cyber-Threat simulation

After discussing the potential cyber-threats to be simulated in our simulator in real-time, we outline how these should be detected and run within our simulator IoTSimSecure. To increase the amount of realism, each decision by the prosecuting agent should also be simulated in real-time, as the time taken to come up with a decision should also correspond to the progress of the attack itself. This is also not considered by the current simulator, as this would require the agent to sense the environment as it appears after the decision process. This will then require yielding capabilities, allowing the agent to temporarily halt the internal computation and then resume it before sensing the updated environment. These capabilities are currently not supported by SimulatorBridger, as the CPU time will only regard the packets transmission time but not other internal processing.



(a) Edge device is attacking

(b) IoT device is attacking

Figure 7.3: Scenario 2: Attacking the battery of IoT device: implementing EWMA for battery consumption of the IoT device

a) Load Balancing Algorithm In our network countermeasures, we have designed a flexible load-balancing algorithm for the MELs in edge devices within SimulatorBridger. This algorithm is structured to support the Software-Defined Networking (SDN) efficiently. This aspect will also enable us to inject either the Gateway Load Balancing Protocol (GLBP) - a Cisco protocol - or a simpler mechanism based on maximum support thresholds of received requests. This design ensures that the implementation can seamlessly switch between these two methods (GLBP or threshold-based load balancing) in a plug-and-play manner. The load balancing scheme between edges will be achieved as follows: When the edge receives more requests than the maximum number at that time, resulting in high traffic or load, it then shares its work by using its nearest neighbours. We then envision extending our current load-balancing algorithm as follows within each edge node: a hash table is used to store edge information and sort them by most IoT devices connected. The maximum number of IoT devices connected to an edge is set so that if the edge has this number of IoT devices connected, it will be considered a full edge. Then, the full edge will stop processing IoT devices' requests and forward them to the nearest edge. Then, the status of the nearest edge will be rechecked to decide if it can process the request or re-allocate it again to the other nearest edge. As a result, the load balancing method helps minimize load imbalances on the edge and improves energy management, preventing an attacker from either depleting the battery or incrementing the Edge energy consumption if directly connected to the grid. When using GLBP, the

algorithm enables rerouting tasks from a device under attack to a backup device to ensure that operations remain uninterrupted, thus always requiring a back-up agent for re-setting dynamically the DNS routing algorithm. GLBP facilitates the automatic selection and simultaneous use of multiple available gateways, along with the distribution of traffic among them. It allows for even load sharing across all routers in the GLBP group. At this stage, SimulatorBridger structurally supports one gateway per network, thus requiring a further extension of the simulator architecture. Alternatively, when employing the threshold-based method, the algorithm should be able to manage the load by monitoring the number of active connections. Once an edge device reaches its maximum capacity of IoT device connections, it will stop processing new requests and forward them to the nearest edge. This process involves checking the status of the nearest edge to determine if it can handle the additional load or if the request should be reallocated to another edge. Both methods aim to minimize load imbalances on the edge and enhance energy management, thereby preventing battery depletion or grid energy consumption due to excessive load or potential attacks.

b) Average packet interval Each device is configured to communicate with each other at specific intervals which, in SimulatorBridger, are pre-determined by the IoT nodes while configuring a cloud App. In a realistic set-up, malicious devices could instead send more requests in a shorter period. To detect this behaviour, we consider the time intervals of all nodes in the network and calculate an average time interval that is considered normal for the network. The node that exceeds the standard packet interval threshold will therefore be considered as malicious. To do this, the detection algorithm running over a MEL should establish a baseline by analyzing the usual communication patterns of IoT devices in historical data from real-world networks to understand typical traffic patterns, packet intervals, and behaviour under various conditions. Collecting data on number of requests per second for a period that is long enough to capture the variations in everyday activity, including peak and off-peak hours, weekdays, and weekends. This includes how often devices broadcast messages like speed, location updates (beacon messages), and other safety messages under normal traffic conditions. The baseline must consider the context, such as density, urban vs. rural environments, and time of day. Then, log the number of requests per second. and set logging intervals by determining how often to log this data,

for example, log the number of requests every minute. Analyzing it to understand the patterns of normal network activity by calculating the average number of requests per second for different times of the day and days of the week. Following this, it is necessary to identify patterns by observing those that tend to repeat, such as a high level of traffic during business hours or a low level of traffic at night. We should now establish a baseline that represents normal network behaviour, including the normal range of activity for each metric. For example, during weekdays from 9 AM to 5 PM, the normal range of requests per second might be 1000-1500. IoT networks are highly dynamic, with frequent topology changes due to device mobility. Therefore, the average interval calculation might need to be adaptive and calculated over different time windows or scenarios to represent normal behaviour. Hence, the next step is setting thresholds based on the baseline for various metrics, allowing for some flexibility to avoid false positives. So, it is vital to set upper limits and lower limits. For example, if the average number of requests per second during peak time is 1500, set an alert threshold at 2000 requests per second for the upper limits, and if the traffic never falls below 500 requests per second during peak time as the lower limits, a sudden drop below this threshold will also cause an alert. In the event that this threshold is exceeded, it may indicate a potential intrusion or attack.

c) Attacking the edge device In the context of IoT network security, edge nodes may be exposed to attacks aimed at causing network disruptions and increasing energy consumption. To enhance the simulation capabilities of SimulatorBridger in this regard, several key modifications are proposed: performing the Exponential Weighted Moving Average (EWMA) algorithm in MELs, located at the edges, and setting the threshold above which traffic is considered anomalous or potentially malicious. EWMA is a widely used method for detecting flooding attacks, it can be used to monitor network traffic and identify unusual spikes or patterns indicative of an attack by calculating moving averages. Our proposed IoTSimSecure will set a threshold for the number of requests a MEL in the edge can accept so the edge can be prevented from becoming overwhelmed by a draining battery attack. The Microelements (MELs) at the edge can detect spikes in traffic by monitoring the rate of incoming requests/packets. This can be achieved by maintaining a count of the packets received over a fixed period, which is then used to calculate the EWMA. The EWMA serves as a smoothed representation of traffic flow,

allowing the system to detect anomalies against the usual pattern. A dynamic value for the threshold in the EWMA algorithm will be considered: continuously calculating the EWMA for a particular metric is monitoring. The metric that is considered is the number of IoT devices connected to the edge. So, the threshold value will be dynamic based on the number of connected devices. The MELs will keep an internal log of traffic, which could involve recording information on packets or requests over time. This log would use a sliding window approach, where only the most recent data within a specific time frame is kept, to manage memory usage efficiently and focus on current traffic patterns. The scenario in Figure 7.2 illustrates how an IoT device could attack an edge device's battery. An IoT device sends many requests to the edge device, so the limit of the number of requests that the edge can accept is reached, thus exceeding the threshold in MEL. This means there is an increase in the number of requests the edge receives compared to the number of devices connected. Hence, an attack may occur and try to deplete the edge battery. Thus, the MEL in the edge will accept only requests below the limit allowed, which is the threshold, marked with a grey tick (✓) in the figure, and reject all other requests, marked with a red cross (✗) in the figure. This requirement also postulates the need for accessing and updating memory stored in secondary memory, which is not currently in place within the current simulator.

d) Logging procedures To detect battery-draining attacks effectively, this logging system needs to be expanded to capture a comprehensive record of all network activities. This extended logging should provide an overview of network interactions by collecting information on communication patterns, frequency of requests, and any anomalies in data transmission. Obtaining access to these extensive logs is necessary for the IoT attack prevention algorithm, a critical component of the network's security framework. By accessing this log and using its data, the algorithm could identify changes from established network behaviour patterns, suggesting the occurrence of potential security attacks. Moreover, the algorithm analyses metrics such as unusual spikes in data requests or irregular communication patterns to effectively identify the difference between normal network operations and potential attack scenarios. This approach not only enhances the detection of complex attacks but also plays a part in developing proactive defence mechanisms, ensuring a robust security framework for IoT networks. A comprehensive logging system

and advanced analytical capabilities are essential for protecting IoT devices against an increasingly complex environment of security attacks. The algorithm would utilize this data to analyze network behaviour, identifying variations from established patterns that may indicate potential attacks. The algorithm's ability to distinguish between normal operations and possible attacks is key to this process, based on metrics such as unusual spikes in data requests or irregular communication patterns. Upon detecting a potential attack, the algorithm must decide whether to continue monitoring the messages or launch a response protocol, which could include temporarily stopping message monitoring for further analysis or modifying network parameters. The flexibility to re-schedule or adjust the monitoring process is crucial, allowing the system to respond to varying attack levels and maintain network integrity dynamically. This advanced approach to logging and analysis is essential in enhancing the security of IoT devices against complex attacks, ensuring a robust and responsive defence mechanism within IoT networks.

e) Support for other algorithms in addition to EWMA The IoTSimSecure framework should be flexible, allowing different algorithms to be plugged in and used for traffic analysis. To detect attacks, this tool could provide an interface for algorithms to access traffic logs and perform the necessary computations. The plugged-in algorithm would analyze the data within the window to calculate the EWMA/thresholds and other statistical measures. If the traffic within the current window varies significantly from the calculated average or expected pattern, it could be flagged as anomalous. The thresholds in the proposed IoTSimSecure system are two: one is in the MELs in the edge devices to calculate the number of requests allowed to be received by the edge, and the other threshold calculation is in IoT devices that identify the maximum power consumption it is capable of consuming. The threshold value in IoT devices will be fixed and set at the configuration step before the simulation starts. The scenario for attacking the battery of an IoT device is illustrated in Figure 7.3. If an attacker, which can be an edge device or an IoT device, tries to deplete an IoT device battery, the battery consumption will reach the maximum limit/threshold. Hence, this device is set to sleep battery mode to protect the battery from being depleted so it will not receive the rest of the packets from the attacker. This then postulates the need for the simulator to realistically simulate the possibility of IoT nodes transmitting packets and receiving them from the network.

This is not encompassed by the current SimulatorBridger simulator, as it does not fully implement the IPv6 communication stack.

7.4 Conclusion

In conclusion, our exploration of the current IoT simulation ecosystems has not only identified a critical gap in security capabilities but has also led to the establishment of a novel IoTSimSecure framework designed to mitigate battery depletion attacks on IoT devices. The proposed IoTSimSecure simulator, which will effectively support signature-based and anomaly-based detection methods, represents a significant advancement in the field of IoT simulations. As we look to the future, our efforts will be channelled towards the actual implementation of a new simulator, *IoTSimSecure*, which, to support all of the previous, will require a massive refactoring of the current state-of-the-art simulator, SimulatorBridger. This therefore makes us lean towards the creation of a less monolithic and more modular architecture, which could allow further possible extensions in the future, thus requiring IoTSimSecure to be a completely new simulator. This goes hand in hand with choosing data lakes as the preferred storage for logging data, thus allowing us to provide better analytics including time series monitoring [232]. We are planning on conducting extensive validation experiments to improve the detection algorithms, assuring that they are both accurate in identifying real attacks and robust against false positives. Further, we aim to expand the IoTSimSecure's capabilities to address a broader range of IoT-specific attacks, thereby providing a comprehensive security solution. The development of an open-source library of attack signatures and an adaptable anomaly detection engine will also be essential in fostering a collaborative environment for ongoing research. Ultimately, the realization of this IoTSimSecure will not only improve IoT simulations against emerging security attacks but will also serve as a benchmark for the development of real-world IoT security simulators, protecting the future of our increasingly connected world.

8

Conclusion and future works

Contents

8.1 Thesis Summary

With the development of VANETs and the integration of the IoT, transportation systems could be revolutionized by enabling the control and communication of networked vehicles in real time. As a result of such a smart transition, traffic management and road safety will be enhanced, and advanced services such as autonomous driving and intelligent transportation systems will be facilitated. While VANETs and IoT present a number of opportunities, they are also accompanied by several challenges, particularly with regard to mobility, scalability, and battery consumption. As of now, there is no tool that provides comprehensive and continuous simulations that include both traffic and IoT components. In addition, important factors such as battery consumption are often overlooked.

This thesis addresses these critical challenges and proposes novel solutions in the field of IoT and VANET simulations, with a focus on energy efficiency, security, and integration of emerging technologies. Following is a summary of the contributions of this thesis:

Chapter 2 first provides an overview of VANETs and their architecture. It then demonstrates various existing VANET simulators and their applications. It next explains the different ways of testing and evaluating VANET designs, architectures, and optimization solutions. Based on the findings of our literature review, it is found that the existing environment of VANET simulators lacks comprehensive capabilities in several key aspects. Firstly, none of the currently available VANET simulators have demonstrated the ability to model and simulate the integration of renewable energy sources within the vehicular network, specifically in terms of recharging the batteries of vehicles while they are in motion within an urban environment. Furthermore, our review highlights a notable gap in VANET simulation domain concerning the incorporation of security effects within the network. This identified gap in the current state of VANET simulation tools emphasizes the need for further research and development efforts to address these limitations and enhance the accuracy and reliability of VANET simulations in relation to renewable energy integration, mobile battery recharging, and security attacks.

Chapter 3 discusses advances and challenges in IoT simulators, which are critical to modeling, analyzing, and optimizing IoT systems. In this chapter, key contributions and comparative analyses of various open-source simulators are highlighted, with a focus

on their capabilities, limitations, and applications in smart cities, healthcare, agriculture, and industrial IoT. Evaluation criteria include scalability, realism, latency, network topology flexibility, device behavior modelling, protocol support, resource consumption, interactivity, extensibility, integration with other tools, performance reporting, ease of use, reliability, mobility modelling, and environment modelling. In spite of their importance, current simulators are challenged in the area of mobility modelling, SDN, comprehensive energy modelling, scalability, and cyber security threat simulations. As a result of the chapter's emphasis on the need for versatile simulators that support all OSI layers and enhanced cybersecurity measures, it underscores the importance of advanced platforms for meeting the evolving demands of the IoT landscape and facilitating comprehensive, accurate simulations.

Chapter 4 introduces a holistic modeling and integration approach for bridging IoT simulation with traffic simulation. As a preliminary experiment, it addresses the question: Can we bridge IoT simulation and traffic simulation? The proposed system architecture demonstrated initial steps in integrating IoTSim-Osmosis with the traffic simulator SUMO. As a result of this chapter, the groundwork has been laid for simulating VANET environments based on IoT infrastructure by identifying key requirements and proposing initial solutions.

Chapter 5 proposes SimulatorBridger, a novel platform designed to integrate traffic and IoT simulations for energy efficiency monitoring in electric vehicles within real-world traffic scenarios. The chapter highlights the roles of traffic simulators and network simulators in generating vehicle movement traces and evaluating network protocols. The primary motivation for developing SimulatorBridger is addressed, noting its capabilities to simulate battery consumption, and moving IoT devices which is limitation in the current IoT simulator: IoTSim-OsmosisRES. The chapter also provides an overview of the design and implementation of the SimulatorBridger framework, detailing its key components, vehicular data collection, software-defined network configuration, osmotic agents module, and dynamic adaptive routing. An experimental setup using the Bologna dataset is discussed, demonstrating the platform's ability to handle complex IoT and traffic data scenarios. Preliminary results are analyzed, showing the correlation between battery consumption and packet transmission, validating the simulator's functionality.

Chapter 6 presents the enhancement of the SimulatorBridger platform through the inte-

gration of real-world traffic data from the Department for Transport (DfT), leading to the development of SimulatorBridgerDfT. By incorporating DfT traffic data in CSV format, the new version of the simulator enables more accurate and flexible simulations of urban traffic scenarios. The chapter details the design and implementation of the DfTConverter module, which processes the raw DfT data into a format suitable for simulation. During this chapter, we examine the impact of using real-world traffic data on the performance of simulations, particularly in terms of communication delays within vehicular networks. Through a series of experiments, we demonstrate that *SimulatorBridgerDfT* effectively replicates real-world traffic patterns, showing a strong correlation between simulated and actual data across different Roadside Units (RSUs). The study also highlights the significant impact of increased IoT device loads on network performance, particularly during peak traffic periods, emphasizing the need for efficient network management in smart cities. This work underscores the simulator's versatility and its potential for advancing traffic management research and applications. Additionally, the chapter explores whether estimating real-time communication times through IoT device connection counts, rather than tracking precise locations, is sufficient. The findings suggest that this simplified approach can effectively estimate communication behaviors, making it a practical solution for large-scale simulations. Future enhancements will include more diverse datasets and advanced analytics to further refine the simulation's fidelity and utility.

Chapter 7 introduces IoTSimSecure, a novel simulation framework designed with the goal of addressing the security concerns caused particularly by battery-draining attacks on IoT devices. It outlines the increasing threat posed by such attacks in various sectors, including smart cities, industrial environments, and healthcare systems. The chapter discusses the limitations of current IoT simulators, particularly in simulating security scenarios and highlights the need for a dedicated tool to enhance the robustness of IoT systems. IoTSimSecure aims to fill this gap by supporting diverse detection algorithms, real-time traffic analysis, and realistic use case scenarios. The proposed simulator leverages MicroELements (MEL) for efficient data processing and decision-making at the network's edge, providing a robust platform for developing and testing cyber-threat detection strategies. IoTSimSecure supports a range of detection algorithms, including threshold-based detection and Exponential Weighted Moving Average (EWMA) techniques. This flexibility allows for comprehensive analysis and testing of various security strategies, enhancing

the simulator's ability to develop effective countermeasures against security attacks, particularly battery-draining attacks.

Overall, this thesis provides significant contributions to the fields of IoT and VANET simulations by identifying critical gaps, proposing novel frameworks, and enhancing existing simulation tools to address the current challenges in energy efficiency, security, and integration of advanced technologies.

8.2 Future Research Directions

Even though this thesis has made significant strides in advancing the fields of IoT and VANET simulations, future research could provide even greater enhancements to the capabilities and applications of the frameworks. For future exploration, the following directions are suggested:

Enhanced Mobility Modeling In the future, more sophisticated mobility models should be developed that can accurately simulate real-world traffic scenarios, including pedestrian movement, public transportation interaction, and multi-modal transportation. The integration of these models into existing simulators will allow us to gain a better understanding of traffic dynamics and their impact on IoT and VANET systems.

Integration of Renewable Energy Sources As sustainable energy solutions become increasingly important, future work should explore the integration of renewable energy sources, such as solar and wind power, into VANET and IoT simulations. The objective of this project is to model the dynamic processes involved in harvesting, storing, and consuming energy in vehicular networks. To develop energy-efficient transportation systems, it is important to investigate how renewable energy affects network performance and vehicle battery life.

Advanced Cybersecurity Measures While IoTSimSecure has been introduced in its architectural design, its full development and execution are yet to be realized. The future work for IoTSimSecure involves implementing the proposed simulator framework, focusing on conducting validation experiments to improve detection algorithms, ensuring accuracy

in identifying real attacks and robustness against false positives. In addition, developing an open-source library of attack signatures and an adaptable anomaly detection engine to foster a collaborative environment for ongoing research. The next phase of research should include not only the completion of IoTSimSecure, but also the expansion of its capabilities to address a broader range of IoT-specific attacks, providing a comprehensive security solution. To enhance the robustness of IoT and VANET systems against these evolving attacks, advanced attack detection algorithms, machine learning techniques, and blockchain-based security solutions should be integrated into the simulation frameworks. A wide range of cyber threats should be recognized in real-time by advanced attack detection algorithms. The methods used in this regard include anomaly detection, pattern recognition, and behavioral analysis in order to detect malicious activity. Machine learning models can be used to predict and identify security breaches by analyzing large datasets and learning from past attack patterns. In addition, these models can enhance the adaptive capabilities of security systems, enabling them to respond effectively to new and unknown threats. It is also possible to create secure, decentralized systems that are resistant to tampering and fraud by exploring the use of blockchain technology. IoT and VANET communications can be enhanced with the use of blockchains that can provide secure data storage, transparent transaction records, and enhanced trust. A focus on these advanced cybersecurity measures in future research will assist in developing more robust and secure IoT and VANET systems. By implementing IoTSimSecure along with these enhancements, modern transportation networks will be more secure against a wide range of complex cyber attacks.

Scalability and Performance Optimization With the continuing growth of IoT and VANET deployments, scalability remains a significant issue. Simulation tools must be optimized to handle large-scale networks with millions of devices and vehicles. Addressing these challenges requires improving computational efficiency, reducing latency, and enhancing real-time processing capabilities. Exploring cloud-based and distributed computing approaches can also mitigate scalability issues. Additionally, future work for the SimulatorBridger simulator should identify specific causes of long shutdown times in network infrastructure and explore different packet routing policies or network configurations to resolve these problems. Implementing load balancing schemes between traffic lights to

minimize load imbalances and improve energy management is also recommended.

Comprehensive Energy Consumption Modeling Future research should develop comprehensive models that account for all aspects of energy consumption in IoT and VANET environments in order to achieve more accurate simulations. As well as battery consumption, energy costs associated with data transmission, processing, and storage must also be considered. Researchers can identify strategies to minimize energy consumption and extend the operational life of IoT devices and vehicular networks by incorporating detailed energy profiles of various devices and network components.

Interoperability and Standardization It is recommended that future work aim to increase the interoperability of simulation tools through the development of standardized interfaces and protocols that facilitate the seamless integration of different simulators and data sources. As a result, researchers will be able to conduct more comprehensive and comparative studies across a variety of platforms. Furthermore, establishing common standards will promote collaboration and data sharing within the research community, contributing to further advancements in the field.

Real-World Validation and Field Trials Towards bridging the gap between simulation and real-world applications, future research should focus on validating simulation results using real-world testing and deployments. Implementing pilot projects in collaboration with industry partners and municipalities will provide valuable insight into the practical challenges and benefits of the proposed solutions. Validating simulation models in the real world will also contribute to the enhancement and improvement of their accuracy and reliability.

Ethical and Social Implications As the use of IoT and VANET technologies expands, it is essential that the ethical and social implications of these technologies be considered. There is a need for future research to examine the potential impact of these technologies on human rights, privacy, and security. To ensure public trust and acceptance, it is essential to develop frameworks for ethical decision-making and ensure that the benefits of smart transportation systems are distributed equally.

Expanding SimulatorBridgerDfT Capabilities *SimulatorBridgerDfT* can be investigated along two interrelated dimensions, both aimed at enhancing the platform's realism and efficiency. *First*, the simulator will be expanded to include a broader range of datasets and more advanced analytical tools. By incorporating additional data sources, such as weather conditions, social events, and environmental factors, the simulation can achieve higher fidelity, offering more nuanced insights into how various factors influence traffic patterns and communication behaviors in urban environments. Additionally, integrating real-time data processing capabilities will allow the simulation of dynamic traffic scenarios, which are essential for effective traffic management and emergency response planning.

Second, it will be important to explore which parameters and configurations are most suitable for realistic simulations of Department for Transport (DfT) traffic data. This line of inquiry should include an analysis of the memory requirements for simulating cloud infrastructure, particularly in relation to the number of IoT devices involved. Understanding the limits of different cloud configurations and determining a manageable cloud configuration size will be crucial for optimizing the balance between simulation realism and computational efficiency. Together, these two approaches will significantly advance the capabilities of *SimulatorBridgerDfT*, making it a more versatile and powerful tool for urban traffic simulation.

Exploring 6G Integration The development of **6G** technology presents significant opportunities to enhance the capabilities of the *SimulatorBridger* platform, particularly in simulating complex IoT and vehicular network scenarios. The integration of Integrated Sensing and Communication (ISAC) within 6G networks can be utilized to enhance the real-time data acquisition capabilities of SimulatorBridger, allowing for more accurate modeling of dynamic vehicular environments and traffic conditions [233]. Additionally, the adoption of the terahertz (THz) spectrum for high-speed data transmission can be incorporated into the simulator to model ultra-low latency communication scenarios, essential for applications like autonomous vehicles and emergency response [234]. Furthermore, incorporating AI-driven green communication strategies will enable the simulation of advanced resource management techniques, optimizing energy consumption and integrating renewable energy sources, which align with the sustainable focus of SimulatorBridger [235]. Lastly, the enhanced network architecture of 6G, with its emphasis on edge

computing and seamless connectivity, can be simulated to evaluate its impact on reducing communication bottlenecks and improving load balancing in smart city applications [236]. This future work aims to extend the capabilities of SimulatorBridger, making it a comprehensive tool for evaluating next-generation vehicular networks in smart cities.

By addressing these future research directions, the field of IoT and VANET simulations can continue to evolve, driving innovation and enabling the development of smarter, more efficient, and secure transportation systems.

8.3 Conclusion

This thesis has made significant contributions to the field of VANET simulation by developing novel frameworks that integrates IoT simulators with traffic simulators. The proposed frameworks addresses several critical challenges and provides a more comprehensive tool for assessing vehicular energy efficiency and communication patterns. Through extensive literature review, modeling, simulation, and evaluation, the research has laid the foundation for future advances in the integration of VANETs and IoTs. It is hoped that the findings of this thesis will enhance smart city applications, improve traffic management, and ensure the sustainability of transportation systems in the future.

References

[1] R. Almutairi, G. Bergami, G. Morgan, and R. Gillgallon, "Platform for energy efficiency monitoring electrical vehicle in real world traffic simulation," in *2023 IEEE 25th Conference on Business Informatics (CBI)*. IEEE, 2023, pp. 1–8.

[2] R. Almutairi, G. Bergami, and G. Morgan, "Advancements and challenges in iot simulators: A comprehensive review," *Sensors*, vol. 24, no. 5, p. 1511, 2024.

[3] ——, "Poster: Iotsimsecure: Towards an iot simulator supporting cyber-threat detection algorithms (forthcoming, accepted)," in *8th IEEE International Conference on Fog and Edge Computing, ICFEC 2024, Philadelphia, PA, ISA, May 6-9, 2024*. IEEE, 2024.

[4] ——, "Systematic literature review of vanet simulators: Comparative analysis, technological advancements, and research challenges," in *1st International Symposium on Parallel Computing and Distributed Systems (PCDS2024)*. IEEE, September 2024.

[5] ——, "Special issue on multidisciplinary sciences and advanced technology simulatorbridger: System for monitoring energy efficiency of electric vehicles in real-world traffic simulations," *Journal of Engineering Research and Sciences*, vol. 3, pp. 33–40, 2024.

[6] ——, "Approximating real-time iot interaction through connection counting: A qos perspective," in *Proceedings of the Twelfth International Workshop on e-Health Pervasive Wireless Applications and Services (e-HPWAS'24), in conjunction with the 20th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2024)*. IEEE, October 2024. [Online]. Available: μhttp://www.wimob.org/wimob2024/

[7] ——, "Simulatorbridgerdft: A real-data simulator for iot-osmotic interactions," in *Proceedings of the 12th International Conference on Information Technology: IoT and Smart City (ICIT 2024)*. ACM, December 2024. [Online]. Available: μhttps://icit.org/

[8] R. Gillgallon, R. Almutairi, G. Bergami, and G. Morgan, "Simulatororchestrator: A 6g-ready simulator for the cell-free/osmotic infrastructure," *Sensors*, vol. 25, no. 5, p. 1591, 2025.

[9] I. García-Magariño, R. Lacuesta, and J. Lloret, "Abs-smartcomagri: An agent-based simulator of smart communication protocols in wireless sensor networks for debugging in precision agriculture," *Sensors*, vol. 18, no. 4, p. 998, 2018.

[10] W. Jiang, Y. Zhan, X. Xiao, and G. Sha, "Network simulators for satellite-terrestrial integrated networks: A survey," *IEEE Access*, 2023.

[11] O. B. Yahia, Z. Garroussi, O. Bélanger, B. Sansò, J.-F. Frigon, S. Martel, A. Lesage-Landry, and G. K. Kurt, "Evolution of high throughput satellite systems: Vision, requirements, and key technologies," *arXiv preprint arXiv:2310.04389*, 2023.

[12] Y. Zhu, T. Delamotte, and A. Knopp, "Geographical noma-beamforming in multi-beam satellite-based internet of things," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[13] J. Chu, X. Chen, C. Zhong, and Z. Zhang, "Robust design for noma-based multi-beam leo satellite internet of things," *IEEE Internet of Things journal*, vol. 8, no. 3, pp. 1959–1970, 2020.

[14] J. L. M. Puyol and V. M. Baeza, "Bicycle sharing system using an iot network," in *2021 Global Congress on Electrical Engineering (GC-ElecEng)*. IEEE, 2021, pp. 131–135.

[15] H. Gupta, A. V. 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," *Softw. Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, 2017.

[16] L. Bieker, D. Krajzewicz, A. Morra, C. Michelacci, and F. Cartolano, "Traffic simulation for all: a real world traffic scenario from the city of bologna," in *Modeling Mobility with Open Data: 2nd SUMO Conference 2014 Berlin, Germany, May 15-16, 2014*. Springer, 2015, pp. 47–60.

[17] K. Ashton *et al.*, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[18] S. Bosmans, S. Mercelis, J. Denil, and P. Hellinckx, "Challenges of modeling and simulating internet of things systems," in *Advances on P2P, Parallel, Grid, Cloud and Internet Computing: Proceedings of the 13th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2018)*. Springer, 2019, pp. 457–466.

[19] F. Qu, Z. Wu, F.-Y. Wang, and W. Cho, "A security and privacy review of vanets," *IEEE Trans Intell Transp Syst*, vol. 16, no. 6, p. 2985–2996.

[20] E. C. Eze, S. Zhang, and E. Liu, "Vehicular ad hoc networks (vanets): Current state, challenges, potentials and way forward," in *2014 20th international conference on automation and computing*. IEEE, 2014, pp. 176–181.

[21] O. Carsten and A. H. Jamson, "Driving simulators as research tools in traffic psychology," in *Handbook of traffic psychology*. Elsevier, 2011, pp. 87–96.

[22] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved ivc analysis," *IEEE Transactions on mobile computing*, vol. 10, no. 1, pp. 3–15, 2010.

[23] K. Lee, U. Lee, and M. Gerla, "Survey of routing protocols in vehicular ad hoc networks," in *Advances in Vehicular Ad-hoc Networks: Developments and Challenges*. IGI Global, 2010, pp. 149–170.

[24] J. Jakubiak and Y. Koucheryavy, "State of the art and research challenges for vanets," in *2008 5th IEEE Consumer Communications and Networking Conference. IEEE*, p. 912–916.

[25] T. Szydlo, A. Szabala, N. Kordiumov, K. Siuzdak, L. Wolski, K. Alwasel, F. Habeeb, and R. Ranjan, "Iotsim-osmosis-res: Towards autonomic renewable energy-aware osmotic computing," *Software: Practice and Experience*, vol. 52, no. 7, pp. 1698–1716, 2022.

[26] R. A. Eaton, R. H. Joubert, and E. A. Wright, "Pothole primer – a public administrator's guide to understanding and managing the pothole problem," US Army Corps of Engineers – Cold Regions Research & Engineering Laboratory, Special Report Vol. 81–21, December 1989.

[27] M. H. Asad, S. Khaliq, M. H. Yousaf, M. O. Ullah, and A. Ahmad, "Pothole detection using deep learning: A real-time and ai-on-the-edge perspective," *Advances in Civil Engineering*, vol. 2022, p. 9221211, Apr 2022.

[28] M. Saleem, S. Abbas, T. M. Ghazal, M. Adnan Khan, N. Sahawneh, and M. Ahmad, "Smart cities: Fusion-based intelligent traffic congestion control system for vehicular networks using machine learning techniques," *Egyptian Informatics Journal*, 2022.

[29] S. Al-Sultan, M. Al-Doori, A. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network," *J Netw Comput Appl*, vol. 37, p. 380–392.

[30] S. Mussa, M. Manaf, K. Ghafoor, and Z. Doukha, "Simulation tools for vehicular ad hoc networks: A comparison study and future perspectives," in *2015 International Conference on Wireless Networks and Mobile Communications (WINCOM). IEEE*, p. 1–8.

[31] P. K. Shrivastava and L. Vishwamitra, "Comparative analysis of proactive and reactive routing protocols in vanet environment," *Measurement: Sensors*, vol. 16, p. 100051, 2021.

[32] R. K. Jurgen, *V2V/V2I communications for improved road safety and efficiency*. SAE International, 2012, vol. 154.

[33] F. Martinez, C. Toh, J.-C. Cano, C. Calafate, and P. Manzoni, "A survey and comparative study of simulators for vehicular ad hoc networks (vanets," *Wirel Commun Mob Comput*, vol. 11, no. 7, p. 813–828.

[34] E. Spaho, L. Barolli, G. Mino, F. Xhafa, and V. Kolici, "Vanet simulators: a survey on mobility and routing protocols," in *2011 International Conference on Broadband and Wireless Computing, Communication and Applications. IEEE*, p. 1–10.

[35] J. Yang, J. Wang, and B. Liu, "An intersection collision warning system using wi-fi smartphones in vanet," in *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*. IEEE, 2011, pp. 1–5.

[36] G. B. Araujo, M. M. Queiroz, F. de LP Duarte-Figueiredo, A. I. Tostes, and A. A. Loureiro, "Cartim: A proposal toward identification and minimization of vehicular traffic congestion for vanet," in *2014 IEEE symposium on computers and communications (ISCC)*. IEEE, 2014, pp. 1–6.

[37] L. Zhou, Y. Zhang, K. Song, W. Jing, and A. V. Vasilakos, "Distributed media services in p2p-based vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 2, pp. 692–703, 2010.

[38] R. Doolan and G.-M. Muntean, "Ecotrec—a novel vanet-based approach to reducing vehicle emissions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 608–620, 2016.

[39] S. Xiao, X. Ge, Q.-L. Han, and Y. Zhang, "Resource-efficient platooning control of connected automated vehicles over vanets," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 579–589, 2022.

[40] H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti, "Vanet security challenges and solutions: A survey," *Vehicular Communications*, vol. 7, pp. 7–20, 2017.

[41] M. S. Sheikh, J. Liang, and W. Wang, "A survey of security services, attacks, and applications for vehicular ad hoc networks (vanets)," *Sensors*, vol. 19, no. 16, p. 3589, 2019.

[42] G. Carneiro, "Ns-3: Network simulator 3," in *UTM Lab Meeting April*, vol. 20, p. 4–5.

[43] A. Varga and O. Ltd., *OMNeT++ Discrete Event Simulator*, 2019, version 5.5.1. [Online]. Available: μhttps://omnetpp.org/

[44] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "Traci: an interface for coupling road traffic and network simulators," in *Proceedings of the 11th Communications and Networking Simulation Symposium*, p. 155–163.

[45] D. Eckhoff, C. Sommer, and F. Dressler, "On the necessity of accurate ieee 802.11 p models for ivc protocol simulation," in *2012 IEEE 75th Vehicular Technology Conference (VTC Spring). IEEE*, p. 1–5.

[46] R. Riebl, H.-J. Günther, C. Facchi, and L. Wolf, "Artery: Extending veins for vanet applications," in *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS). IEEE*, p. 450–456.

[47] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, and R. Cigno, "Plexe: A platooning extension for veins," in *2014 IEEE Vehicular Networking Conference (VNC). IEE*, p. 53–60.

[48] M. Amoozadeh, B. Ching, C.-N. Chuah, D. Ghosal, and H. Zhang, "Ventos: Vehicular network open simulator with hardware-in-the-loop support," *Procedia Comput Sci*, vol. 151, p. 61–68.

[49] TETCOS, *Cellular Network User Manual*, 2019, μhttps://www.tetcos.com/downloads/v12.2/NetSim_User_Manual.pdf (Accessed: 2024-08-30).

[50] K. Roscher, S. Bittl, A. Gonzalez, M. Myrtus, and J. Jiru, "ezcar2x: rapid-prototyping of communication technologies and cooperative its applications on real targets and inside simulation environments," in *11th Conference Wireless Communication and Information*, p. 51–62.

[51] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille, and D. Scheuermann, "Car2x communication: securing the last meter-a cost-effective approach for ensuring trust in car2x applications using in-vehicle symmetric cryptography," in *2011 IEEE Vehicular Technology Conference (VTC Fall). IEEE*, p. 1–5.

[52] A. Festag and S. Hess, "Etsi technical committee its: news from european standardization for intelligent transport systems (its)-[global communications newsletter," *IEEE Commun Mag*, vol. 47, no. 6, p. 1–4.

[53] A. Tomandl, D. Herrmann, K.-P. Fuchs, H. Federrath, and F. Scheuer, "Vanetsim: an open source simulator for security and privacy concepts in vanets," in *2014 International Conference on High Performance Computing & Simulation (HPCS). IEEE*, p. 543–550.

[54] A. Sliman, K. Madi, A. Khadour, B. Maala, and A. Ahmad, "Fabrication attack effect on medical applications based on q4 vanets," *Int J Comput Sci Trends Technol (IJCST*, vol. 5, no. 2, p. 1–4.

[55] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Comput*, vol. 7, no. 4, p. 12–18.

[56] M. Piorkowski, M. Raya, A. L. Lugo, P. Papadimitratos, M. Grossglauser, and J.-P. Hubaux, "Trans: realistic joint traffic and network simulator for vanets," *ACM SIGMOBILE mobile computing and communications review*, vol. 12, no. 1, pp. 31–33, 2008.

[57] J. Härri, F. Filali, C. Bonnet, and M. Fiore, "Vanetmobisim: generating realistic mobility patterns for vanets," in *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks. ACM*, p. 96–97.

[58] L. Huang, K. Matsuura, H. Yamane, and K. Sezaki, "Enhancing wireless location privacy using silent period," *IEEE Wireless Communications and Networking Conference, 2005*, vol. 2, p. 1187–1192.

[59] A. Beresford and F. Stajano, "Mix zones: User privacy in location-aware services," in *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second. IEEE*, p. 127–131.

[60] A. Stentz, *Optimal and efficient path planning for partially known environments.* In: Intelligent Unmanned Ground Vehicles. Springer.

[61] A. Tomandl, D. Herrmann, K.-P. Fuchs, H. Federrath, and F. Scheuer, "Vanetsim: An open source simulator for security and privacy concepts in vanets," in *2014 International Conference on High Performance Computing & Simulation (HPCS).* IEEE, 2014, pp. 543–550.

[62] S.-Y. Wang, C.-L. Chou, and C.-M. Yang, "Estinet openflow network simulator and emulator," *IEEE Communications Magazine*, vol. 51, no. 9, pp. 110–117, 2013.

[63] S. Wang and C. Lin, "Nctuns 6.0: A simulator for advanced wireless vehicular network research," in *2010 IEEE 71st Vehicular Technology Conference*, Taipei, pp. 1–2,.

[64] B. Schünemann, "V2x simulation runtime infrastructure vsimrti: an assessment tool to design smart traffic management systems," *Comput Netw*, vol. 55, no. 14, p. 3189–3198.

[65] The Network Simulator-NS-2, "The Network Simulator-NS-2," μhttp://www.isi.edu/nsnam/ns, available online: μhttp://www.isi.edu/nsnam/ns (accessed on 28-August-2024).

[66] R. Mangharam, D. Weller, R. Rajkumar, P. Mudalige, and F. Bai, "Groovenet: A hybrid simulator for vehicle-to-vehicle networks," in *2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, 2006, pp. 1–8.

[67] S.-Y. Wang and C.-C. Lin, "Nctuns 6.0: a simulator for advanced wireless vehicular network research," in *2010 IEEE 71st Vehicular Technology Conference*. IEEE, 2010, pp. 1–2.

[68] G. De Marco, M. Tadauchi, and L. Barolli, "Cavenet: Description and analysis of a toolbox for vehicular networks simulation," in *2007 International Conference on Parallel and Distributed Systems*. IEEE, 2007, pp. 1–6.

[69] X. Ge, Z. Li, and S. Li, "5g software defined vehicular networks," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 87–93, 2017.

[70] A. Hussein, I. Elhajj, A. Chehab, and A. Kayssi, "Sdn vanets in 5g: An architecture for resilient security services," in *2017 Fourth International Conference on Software Defined Systems (SDS). IEEE*, p. 67–74.

[71] W. Wang, Y. Chen, Q. Zhang, and T. Jiang, "A software-defined wireless networking enabled spectrum management architecture," *IEEE Commun Mag*, vol. 54, no. 1, p. 33–39.

[72] H. Farhady, H. Lee, and A. Nakao, "Software-defined networking: a survey," *Comput Netw*, vol. 81, p. 79–95.

[73] W. B. Jaballah, M. Conti, and C. Lal, "A survey on software-defined vanets: benefits, challenges, and future directions," *arXiv preprint arXiv:1904.04577*, 2019. [Online]. Available: μhttps://arxiv.org/abs/1904.04577

[74] M. Ushakova, Y. Ushakov, I. Bolodurina, D. Parfenov, L. Legashev, and A. Shukhman, "Research of productivity of software configurable infrastructure in vanet networks on the basis of models of hybrid data transmission devices," in *2020 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC). IEEE*, p. 1–9.

[75] M. Arif, G. Wang, V. Balas, O. Geman, A. Castiglione, and J. Chen, "Sdn based communications privacy-preserving architecture for vanets using fog computing," *Veh Commun*, vol. 26, no. 100265.

[76] W. Zhu, D. Gao, W. Zhao, H. Zhang, and H.-P. Chiang, "Sdn-enabled hybrid emergency message transmission architecture in internet-of-vehicles," *Enterp Inf Syst*, vol. 12, no. 4, p. 471–491.

[77] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Comput Commun Rev*, vol. 38, no. 2, p. 69–74.

[78] A. Kadhim and S. Seno, "Energy-efficient multicast routing protocol based on sdn and fog computing for vehicular networks," *Ad Hoc Netw*, vol. 84, p. 68–81.

[79] J. Weber, M. Neves, and T. Ferreto, "Vanet simulators: an updated review," *Journal of the Brazilian Computer Society*, vol. 27, no. 1, p. 8.

[80] P. Manzanares-Lopez, J. Malgosa-Sanahuja, and J. Muñoz-Gea, "A software-defined networking framework to provide dynamic qos management in ieee 802.11 networks," *Sensors*, vol. 18, no. 7.

[81] M. Amoozadeh, H. Deng, C.-N. Chuah, H. Zhang, and D. Ghosal, "Platoon management with cooperative adaptive cruise control enabled by vanet," *Veh Commun*, vol. 2, no. 2, p. 110–123.

[82] S. Garg, A. Singh, K. Kaur, G. Aujla, S. Batra, N. Kumar, and M. Obaidat, "Edge computing-based security framework for big data analytics in vanets," *IEEE Network*, vol. 33, no. 2, p. 72–81.

[83] A. Boukerche and V. Soto, "An efficient mobility-oriented retrieval protocol for computation offloading in vehicular edge multi-access network," *IEEE Trans Intell Transp Syst*, vol. 21, no. 6, p. 2675–2688.

[84] P. Zhou, T. Braud, A. Zavodovski, Z. Liu, X. Chen, P. Hui, and J. Kangasharju, "Edge-facilitated augmented vision in vehicle-to-everything networks," *IEEE Trans Veh Technol*, vol. 69, no. 10, p. 12187–12201.

[85] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans Veh Technol*, vol. 68, no. 4, p. 3061–3074.

[86] C. Sommer, S. Joerer, and F. Dressler, "On the applicability of two-ray path loss models for vehicular network simulation," in *2012 IEEE Vehicular Networking Conference (VNC). IEEE*, p. 64–69.

[87] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Ave: Autonomous vehicular edge computing framework with aco-based scheduling," *IEEE Trans Veh Technol*, vol. 66, no. 12, p. 10660–10675.

[88] A. Olmos, F. Vazquez-Gallego, R. Sedar, V. Samoladas, F. Mira, and J. Alonso-Zarate, "An automotive cooperative collision avoidance service based on mobile edge computing," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, p. 601–607.

[89] A. Roy and S. Madria, "Secured traffic monitoring in vanet," *arXiv preprint arXiv:1909.10057*, 2019. [Online]. Available: μhttps://arxiv.org/abs/1909.10057

[90] X. Kan, A. Ganlath, S. Ucar, K. Han, P. Tiwari, and K. Karydis, "Edge assisted misbehavior detection for platoons," in *2019 IEEE Vehicular Networking Conference (VNC). IEEE*, p. 1–4.

[91] K. Gilly, S. Alcaraz, N. Aknin, S. Filiposka, and A. Mishev, "Modelling edge computing in urban mobility simulation scenarios," in *2020 IFIP Networking Conference (Networking). IEEE*, p. 539–543.

[92] S. Shah, E. Ahmed, M. Imran, and S. Zeadally, "5g for vehicular communications," *IEEE Commun Mag*, vol. 56, no. 1, p. 111–117.

[93] R. Uzcátegui, A. Sucre, and G. Acosta-Marum, "Wave: A tutorial," *IEEE Commun Mag*, vol. 47, no. 5, p. 126–133.

[94] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-advanced for Mobile Broadband*. Academic Press, 2013.

[95] 3rd Generation Partnership Project (3GPP), "3gpp - 3rd generation partnership project," μhttps://www.3gpp.org/, 1998, accessed: September 6, 2024.

[96] A. Chehri, H. Chehri, N. Hakim, and R. Saadane, "Realistic 5.9 ghz dsrc vehicle-to-vehicle wireless communication protocols for cooperative collision warning in underground mining," *Smart Transportation Systems*, p. 133–141.

[97] A. Virdis, G. Nardini, and G. Stea, "5g new radio user plane simulation model for inet and omnet++," in *Proceedings of the 7th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2020)*. SciTePress, 2020.

[98] J. Zhang, H. Zhong, J. Cui, M. Tian, Y. Xu, and L. Liu, "Edge computing-based privacy-preserving authentication framework and protocol for 5g-enabled vehicular networks," *IEEE Trans Veh Technol*, vol. 69, no. 7, p. 7940–7954.

[99] J. Huang, Y. Qian, and R. Hu, "Secure and efficient privacy-preserving authentication scheme for 5g software defined vehicular networks," *IEEE Trans Veh Technol*, vol. 69, no. 8, p. 8542–8554.

[100] D. Chekired, M. Togou, L. Khoukhi, and A. Ksentini, "5g-slicing-enabled scalable sdn core network: Toward an ultra-low latency of autonomous driving service," *IEEE J Sel Areas Commun*, vol. 37, no. 8, p. 1769–1782.

[101] P. Dharanyadevi and K. Venkatalakshmi, "Proficient routing by adroit algorithm in 5g-cloud-vmesh network," *EURASIP J Wirel Commun Netw*, vol. 2016, no. 1, p. 1–11.

[102] H. Puttagunta and D. Agrawal, "Performance of 802.11 p in vanet at 5g frequencies for different channel models," in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). IEEE*, p. 0468–0472.

[103] A. Ibrahim, C. Math, D. Goswami, T. Basten, and H. Li, "Co-simulation framework for control, communication and traffic for vehicle platoons," in *2018 21st Euromicro Conference on Digital System Design (DSD). IEEE*, p. 352–356.

[104] S. Shadrin and A. Ivanova, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," *Avtomobil'. Doroga. Infrastruktura*, vol. 3, no. ue 21, p. 10.

[105] Q. Lu, T. Tettamanti, D. Hörcher, and I. Varga, "The impact of autonomous vehicles on urban traffic network capacity: an experimental analysis by microscopic traffic simulation," *Transp Lett*, vol. 12, no. 8, p. 540–549.

[106] J. Pereira and R. Rossetti, "An integrated architecture for autonomous vehicles simulation," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, p. 286–292.

[107] F. Alotibi and M. Abdelhakim, "Anomaly detection for cooperative adaptive cruise control in autonomous vehicles using statistical learning and kinematic model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3468–3478, 2020.

[108] Y. Li and Q. Liu, "Intersection management for autonomous vehicles with vehicle-to-infrastructure communication," *PLoS ONE*, vol. 15, no. 7.

[109] M. Teixeira, P. d'Orey, and Z. Kokkinogenis, "Simulating collective decision-making for autonomous vehicles coordination enabled by vehicular networks: A computational social choice perspective," *Simul Model Pract Theory*, vol. 98, no. 101983.

[110] D. Zehe, S. Nair, A. Knoll, and D. Eckhoff, "Towards citymos: a coupled city-scale mobility simulation framework," *5th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication*, vol. 2017, no. 03.

[111] K. Roscher and G. Maierbacher, "Reliable message forwarding in vanets for delay-sensitive applications," in *2016 International Symposium on Wireless Communication Systems (ISWCS). IEEE*, p. 199–203.

[112] S. Hadiwardoyo, J.-M. Dricot, C. Calafate, J.-C. Cano, E. Hernández-Orallo, and P. Manzoni, "Uav mobility model for dynamic uav-to-car communications in 3d environments," *Ad Hoc Netw*, vol. 107, no. 102193.

[113] A. Brummer, R. German, and A. Djanatliev, "On the necessity of three-dimensional considerations in vehicular network simulation," in *2018 14th Annual Conference on Wireless On-demand Network Systems and Services (WONS). IEEE*, p. 75–82.

[114] D. Montgomery and E. Foufoula-Georgiou, "Channel network source representation using digital elevation models," *Water Resour Res*, vol. 29, no. 12, p. 3925–3934.

[115] Z. Lu, G. Qu, and Z. Liu, "A survey on recent advances in vehicular network security, trust, and privacy," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 760–776, 2018.

[116] M. N. Mejri, J. Ben-Othman, and M. Hamdi, "Survey on vanet security challenges and possible cryptographic solutions," *Vehicular Communications*, vol. 1, no. 2, pp. 53–66, 2014.

[117] C. H. Quevedo, A. M. Quevedo, G. A. Campos, R. L. Gomes, J. Celestino, and A. Serhrouchni, "An intelligent mechanism for sybil attacks detection in vanets," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[118] H. Wang, J. Wu, W. Hu, and X. Wu, "Detecting and assessing anomalous evolutionary behaviors of nodes in evolving social networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 13, no. 1, pp. 1–24, 2019.

[119] L. Wu, J. Wang, K.-K. R. Choo, and D. He, "Secure key agreement and key protection for mobile device user authentication," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 319–330, 2018.

[120] F. Sailhan, T. Delot, A. Pathak, A. Puech, and M. Roy, "Fault injection and monitoring for dependability analysis of wireless sensor-actuators networks," in *Proceedings of the 4th Workshop Gestion des Données dans les Systèmes d'Information Pervasifs (GEDSIP) in conjunction with INFORSID*, 2010.

[121] M. Cinque, D. Cotroneo, C. Martino, S. Russo, and A. Testa, "Avr-inject: A tool for injecting faults in wireless sensor nodes," in *2009 IEEE International Symposium on Parallel & Distributed Processing. IEEE*, p. 1–8.

[122] D. Raposo, A. Rodrigues, J. Silva, and F. Boavida, "A taxonomy of faults for wireless sensor networks," *J Netw Syst Manag*, vol. 25, no. 3, p. 591–611.

[123] D. Buse and F. Dressler, "Towards real-time interactive v2x simulation," in *2019 IEEE Vehicular Networking Conference (VNC). IEEE*, p. 1–8.

[124] D. Buse, M. Schettler, N. Kothe, P. Reinold, C. Sommer, and F. Dressler, "Bridging worlds: Integrating hardware-in-the-loop testing with large-scale vanet simulation," in *2018 14th Annual Conference on Wireless On-demand Network Systems and Services (WONS). IEEE*, p. 33–36.

[125] R. Riebl, M. Monz, S. Varga, H. Janicke, L. Maglaras, A. Al-Bayatti, and C. Facchi, "Improved security performance for vanet simulations," in *4th IFAC Symposium on Telematics Applications*.

[126] M. A. R. Baee, L. Simpson, E. Foo, and J. Pieprzyk, "Broadcast authentication in latency-critical applications: On the efficiency of ieee 1609.2," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 11 577–11 587, 2019.

[127] European Telecommunications Standards Institute (ETSI), "Etsi website," μhttps://www.etsi.org/, accessed: 2024-08-30.

[128] M. Alaa, M. A. Abdala, and A. Al-Sherbaz, "Evaluation study of ieee 1609.4 performance for safety and non-safety messages dissemination," *International Journal of Enhanced Research in Science Technology & Engineering*, vol. 3, no. 11, pp. 29–36, 2014.

[129] U. Rajput, F. Abbas, H. Eun, and H. Oh, "A hybrid approach for efficient privacy-preserving authentication in vanet," *IEEE Access*, vol. 5, pp. 12 014–12 030, 2017.

[130] A. H. M. Aman, E. Yadegaridehkordi, Z. S. Attarbashi, R. Hassan, and Y.-J. Park, "A survey on trend and classification of internet of things reviews," *Ieee Access*, vol. 8, pp. 111 763–111 782, 2020.

[131] M. A. Jabraeil Jamali, B. Bahrami, A. Heidari, P. Allahverdizadeh, F. Norouzi, M. A. Jabraeil Jamali, B. Bahrami, A. Heidari, P. Allahverdizadeh, and F. Norouzi, "Iot architecture," *Towards the Internet of Things: Architectures, Security, and Applications*, pp. 9–31, 2020.

[132] V. Cristea, C. Dobre, and F. Pop, "Context-aware environments for the internet of things," *Internet of Things and inter-cooperative computational technologies for collective intelligence*, pp. 25–49, 2013.

[133] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 414–454, 2013.

[134] D. Kyriazis and T. Varvarigou, "Smart, autonomous and reliable internet of things," *Procedia Computer Science*, vol. 21, pp. 442–448, 2013.

[135] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, "Smart objects as building blocks for the internet of things," *IEEE internet computing*, vol. 14, no. 1, pp. 44–51, 2009.

[136] 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.

[137] T. Jung, P. Shah, and M. Weyrich, "Dynamic co-simulation of internet-of-things-components using a multi-agent-system," *Procedia CIRP*, vol. 72, pp. 874–879, 2018.

[138] A. Taufique, M. Jaber, A. Imran, Z. Dawy, and E. Yacoub, "Planning wireless cellular networks of future: Outlook, challenges and opportunities," *IEEE Access*, vol. 5, pp. 4821–4845, 2017.

[139] F. K. Shaikh, S. Zeadally, and E. Exposito, "Enabling technologies for green internet of things," *IEEE Systems Journal*, vol. 11, no. 2, pp. 983–994, 2015.

[140] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE communications surveys & tutorials*, vol. 20, no. 1, pp. 416–464, 2017.

[141] K. Alwasel, D. N. Jha, F. Habeeb, U. Demirbaga, O. Rana, T. Baker, S. Dustdar, M. Villari, P. James, E. Solaiman *et al.*, "Iotsim-osmosis: A framework for modeling and simulating iot applications over an edge-cloud continuum," *Journal of Systems Architecture*, vol. 116, p. 101956, 2021.

[142] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan *et al.*, "The prisma 2020 statement: an updated guideline for reporting systematic reviews," *International journal of surgery*, vol. 88, p. 105906, 2021.

[143] F. G. Brundu, E. Patti, A. Osello, M. Del Giudice, N. Rapetti, A. Krylovskiy, M. Jahn, V. Verda, E. Guelpa, L. Rietto *et al.*, "Iot software infrastructure for energy management and simulation in smart cities," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 832–840, 2016.

[144] S. Bosmans, S. Mercelis, J. Denil, and P. Hellinckx, "Testing iot systems using a hybrid simulation based testing approach," *Computing*, vol. 101, pp. 857–872, 2019.

[145] M. Salama, Y. Elkhatib, and G. Blair, "Iotnetsim: A modelling and simulation platform for end-to-end iot services and networking," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, 2019, pp. 251–261.

[146] V. Therrien, H. Mellah, V. Boutin, and B. Sansò, "A large-scale simulator for nb-iot," *IEEE Access*, vol. 10, pp. 68 231–68 239, 2022.

[147] P. Kasnesis, L. Toumanidis, D. Kogias, C. Z. Patrikakis, and I. S. Venieris, "Assist: An agent-based siot simulator," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 353–358.

[148] N. Bruschi, G. Haugou, G. Tagliavini, F. Conti, L. Benini, and D. Rossi, "Gvsoc: a highly configurable, fast and accurate full-platform simulator for risc-v based iot processors," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*. IEEE, 2021, pp. 409–416.

[149] D.-T. Ta, K. Khawam, S. Lahoud, C. Adjih, and S. Martin, "Lora-mab: A flexible simulator for decentralized learning resource allocation in iot networks," in *2019 12th IFIP wireless and mobile networking conference (WMNC)*. IEEE, 2019, pp. 55–62.

[150] M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "Myifogsim: A simulator for virtual machine migration in fog computing," in *Companion proceedings of the10th international conference on utility and cloud computing*, 2017, pp. 47–52.

[151] X. Li, Q. Huang, and D. Wu, "Distributed large-scale co-simulation for iot-aided smart grid control," *IEEE Access*, vol. 5, pp. 19 951–19 960, 2017.

[152] G. Brambilla, M. Picone, S. Cirani, M. Amoretti, and F. Zanichelli, "A simulation platform for large-scale internet of things scenarios in urban environments," in *Proceedings of the First International Conference on IoT in Urban Space*, 2014, pp. 50–55.

[153] K. Ergun, X. Yu, N. Nagesh, L. Cherkasova, P. Mercati, R. Ayoub, and T. Rosing, "Reliot: Reliability simulator for iot networks," in *Internet of Things-ICIOT 2020: 5th International Conference, Held as Part of the Services Conference Federation, SCF 2020, Honolulu, HI, USA, September 18–20, 2020, Proceedings 5*. Springer, 2020, pp. 63–81.

[154] S. Melia, S. Nasabeh, S. Lujan-Mora, and C. Cachero, "Mosiot: modeling and simulating iot healthcare-monitoring systems for people with disabilities," *International Journal of Environmental Research and Public Health*, vol. 18, no. 12, p. 6357, 2021.

[155] J. A. Barriga, P. J. Clemente, E. Sosa-Sánchez, and Á. E. Prieto, "Simulateiot: Domain specific language to design, code generation and execute iot simulation environments," *IEEE Access*, vol. 9, pp. 92 531–92 552, 2021.

[156] J. A. Barriga, P. J. Clemente, J. Hernández, and M. A. Pérez-Toledano, "Simulateiot-fiware: domain specific language to design, code generation and execute iot simulation environments on fiware," *IEEE Access*, vol. 10, pp. 7800–7822, 2022.

[157] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, 2018.

[158] 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.

[159] J. A. Barriga, P. J. Clemente, M. A. Pérez-Toledano, E. Jurado-Málaga, and J. Hernández, "Design, code generation and simulation of iot environments with mobility devices by using model-driven development: Simulateiot-mobile," *Pervasive and Mobile Computing*, vol. 89, p. 101751, 2023.

[160] A. D. Firouzabadi, H. Mellah, O. Manzanilla, R. Khalvandi, V. Therrien, V. Boutin, and B. Sansò, "Piot: A performance iot simulation system for a large-scale city-wide assessment," *IEEE Access*, 2023.

[161] H.-H. Lee, J.-H. Kwon, and E.-J. Kim, "Fs-iiotsim: a flexible and scalable simulation framework for performance evaluation of industrial internet of things systems," *The Journal of Supercomputing*, vol. 74, pp. 4385–4402, 2018.

[162] K. Bajaj, B. Sharma, and R. Singh, "Comparative analysis of simulators for iot applications in fog/cloud computing," in *2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*. IEEE, 2022, pp. 983–988.

[163] S. Sotiriadis, N. Bessis, E. Asimakopoulou, and N. Mustafee, "Towards simulating the internet of things," in *AINA*. IEEE, 2014, pp. 444–448.

[164] A. Bounceur, L. Clavier, P. Combeau, O. Marc, R. Vauzelle, A. Masserann, J. Soler, R. Euler, T. Alwajeeh, V. Devendra *et al.*, "Cupcarbon: A new platform for the design, simulation and 2d/3d visualization of radio propagation and interferences in iot networks," in *2018 15th IEEE annual consumer communications & networking conference (CCNC)*. IEEE, 2018, pp. 1–4.

[165] F. Österlind, *A sensor network simulator for the Contiki OS*. Swedish Institute of Computer Science, 2006.

[166] C. Savaglio and G. Fortino, "A simulation-driven methodology for iot data mining based on edge computing," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 2, pp. 1–22, 2021.

[167] G. D'Angelo, S. Ferretti, and V. Ghini, "Multi-level simulation of internet of things on smart territories," *Simulation Modelling Practice and Theory*, vol. 73, pp. 3–21, 2017.

[168] W. Lee, S. Cho, P. Chu, H. Vu, S. Helal, W. Song, Y.-S. Jeong, and K. Cho, "Automatic agent generation for iot-based smart house simulator," *Neurocomputing*, vol. 209, pp. 14–24, 2016.

[169] A. Brogi and S. Forti, "Qos-aware deployment of iot applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, 2017.

[170] R. Mayer, L. Graser, H. Gupta, E. Saurez, and U. Ramachandran, "Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures," in *2017 IEEE Fog World Congress (FWC)*. IEEE, 2017, pp. 1–6.

[171] T. Qayyum, A. W. Malik, M. A. K. Khattak, O. Khalid, and S. U. Khan, "Fognetsim++: A toolkit for modeling and simulation of distributed fog environment," *IEEE Access*, vol. 6, pp. 63 570–63 583, 2018.

[172] I. Lera, C. Guerrero, and C. Juiz, "Yafs: A simulator for iot scenarios in fog computing," *IEEE Access*, vol. 7, pp. 91 745–91 758, 2019.

[173] T. Pflanzner, A. Kertész, B. Spinnewyn, and S. Latré, "Mobiotsim: Towards a mobile iot device simulator," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. IEEE, 2016, pp. 21–27.

[174] R. Cárdenas, P. Arroba, R. Blanco, P. Malagón, J. L. Risco-Martín, and J. M. Moya, "Mercury: A modeling, simulation, and optimization framework for data stream-oriented iot applications," *Simulation Modelling Practice and Theory*, vol. 101, p. 102037, 2020.

[175] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[176] S. Pattar, R. Buyya, K. R. Venugopal, S. Iyengar, and L. Patnaik, "Searching for the iot resources: Fundamentals, requirements, comprehensive review, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2101–2132, 2018.

[177] T. M. Fernández-Caramés and P. Fraga-Lamas, "A review on the use of blockchain for the internet of things," *Ieee Access*, vol. 6, pp. 32 979–33 001, 2018.

[178] C. Perera, C. H. Liu, S. Jayawardena, and M. Chen, "A survey on internet of things from industrial market perspective," *IEEE Access*, vol. 2, pp. 1660–1679, 2014.

[179] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, 2019.

[180] C. Zhu, V. C. Leung, L. Shu, and E. C.-H. Ngai, "Green internet of things for smart world," *IEEE access*, vol. 3, pp. 2151–2162, 2015.

[181] N. Alshammari, T. Alshammari, M. Sedky, J. Champion, and C. Bauer, "Openshs: Open smart home simulator," *Sensors*, vol. 17, no. 5, p. 1003, 2017.

[182] K. Pareek, P. K. Tiwari, and V. Bhatnagar, "Fog computing in healthcare: A review," in *IOP Conference Series: Materials Science and Engineering*, vol. 1099, no. 1. IOP Publishing, 2021, p. 012025.

[183] A. Kumar, R. Krishnamurthi, A. Nayyar, K. Sharma, V. Grover, and E. Hossain, "A novel smart healthcare design, simulation, and implementation using healthcare 4.0 processes," *IEEE access*, vol. 8, pp. 118 433–118 471, 2020.

[184] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The internet of things for health care: a comprehensive survey," *IEEE access*, vol. 3, pp. 678–708, 2015.

[185] J. M. Talavera, L. E. Tobón, J. A. Gómez, M. A. Culman, J. M. Aranda, D. T. Parra, L. A. Quiroz, A. Hoyos, and L. E. Garreta, "Review of iot applications in agro-industrial and environmental fields," *Computers and Electronics in Agriculture*, vol. 142, pp. 283–297, 2017.

[186] A. Vangala, A. K. Das, V. Chamola, V. Korotaev, and J. J. Rodrigues, "Security in iot-enabled smart agriculture: Architecture, security solutions and challenges," *Cluster Computing*, vol. 26, no. 2, pp. 879–902, 2023.

[187] F. De Rango, N. Palmieri, A. F. Santamaria, and G. Potrino, "A simulator for uavs management in agriculture domain," in *2017 international symposium on performance evaluation of computer and telecommunication systems (SPECTS)*. IEEE, 2017, pp. 1–8.

[188] C. Zhang and Y. Chen, "A review of research relevant to the emerging industry trends: Industry 4.0, iot, blockchain, and business analytics," *Journal of Industrial Integration and Management*, vol. 5, no. 01, pp. 165–180, 2020.

[189] F. Javed, M. K. Afzal, M. Sharif, and B.-S. Kim, "Internet of things (iot) operating systems support, networking technologies, applications, and challenges: A comparative review," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2062–2100, 2018.

[190] K. Gilly, C. Bernad, P. J. Roig, S. Alcaraz, and S. Filiposka, "End-to-end simulation environment for mobile edge computing," *Simulation Modelling Practice and Theory*, vol. 121, p. 102657, 2022. [Online]. Available: μhttps://www.sciencedirect.com/science/article/pii/S1569190X22001277

[191] P. C. Cañizares, A. Núñez, A. Bernal, M. E. Cambronero, and A. Barker, "Simcan2cloud: a discrete-event-based simulator for modelling and simulating cloud computing infrastructures," *Journal of Cloud Computing*, vol. 12, no. 1, p. 133, 2023.

[192] Nidhi and D. Lobiyal, "Performance evaluation of vanet using realistic vehicular mobility," in *Advances in Computer Science and Information Technology. Networks and Communications: Second International Conference, CCSIT 2012, Bangalore, India, January 2-4, 2012. Proceedings, Part I 2.* Springer, 2012, pp. 477–489.

[193] R. C. Poonia and D. Bhargava, "A review of coupling simulator for vehicular ad-hoc networks," *International Journal Information Technology and Computer Science*, vol. 5, pp. 44–51, 2016.

[194] D. Lobiyal *et al.*, "Performance evaluation of realistic vanet using traffic light scenario," *arXiv preprint arXiv:1203.2195*, 2012.

[195] A. Wegener, H. Hellbruck, C. Wewetzer, and A. Lubke, "Vanet simulation environment with feedback loop and its application to traffic light assistance," in *2008 IEEE Globecom Workshops*. IEEE, 2008, pp. 1–7.

[196] K. Alwasel, D. N. Jha, F. Habeeb, U. Demirbaga, O. F. Rana, T. Baker, S. Dustdar, M. Villari, P. James, E. Solaiman, and R. Ranjan, "Iotsim-osmosis: A framework for modeling and simulating iot applications over an edge-cloud continuum," *J. Syst. Archit.*, vol. 116, p. 101956, 2021.

[197] P. Á. López, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. WieBner, "Microscopic traffic simulation using SUMO," in *ITSC*. IEEE, 2018, pp. 2575–2582.

[198] R. Ullah, K. Khattak, Z. H. Khan, M. Ahmad Khan, N. Minallah, and A. Khan, "Vehicular traffic simulation software: A systematic comparative analysis," *Pakistan Journal of Engineering and Technology*, vol. 4, no. 1, pp. 66–7, 03 2021.

[199] B. Boukenadil, "Importance of realistic mobility models for vanet network simulation," *arXiv preprint arXiv:1410.2450*, 2014.

[200] J. Harri, F. Filali, and C. Bonnet, "Mobility models for vehicular ad hoc networks: a survey and taxonomy," *IEEE communications surveys & tutorials*, vol. 11, no. 4, pp. 19–41, 2009.

[201] M. Fellendorf, "Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority," in *64th Institute of Transportation Engineers Annual Meeting*, vol. 32. Springer, 1994, pp. 1–9.

[202] T. Alghamdi, S. Mostafi, G. Abdelkader, and K. Elgazzar, "A comparative study on traffic modeling techniques for predicting and simulating traffic behavior," *Future Internet*, vol. 14, no. 10, 2022.

[203] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday, *Traffic Simulation with Aimsun*. Springer, 2010, pp. 173–232.

[204] M. R. Ullah, K. S. Khattak, Z. H. Khan, M. A. Khan, N. Minallah, and A. N. Khan, "Vehicular traffic simulation software: A systematic comparative analysis," *Pakistan Journal of Engineering and Technology*, vol. 4, no. 1, pp. 66–78, 2021.

[205] J. Miller and E. Horowitz, "Freesim - a free real-time freeway traffic simulator," in *ITSC*. IEEE, 2007, pp. 18–23.

[206] K. W. Axhausen, A. Horni, and K. Nagel, *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press, 2022/11/22/ 2016.

[207] V. T. Arasan and R. Z. Koshy, "Methodology for modeling highly heterogeneous traffic flow," *Journal of Transportation Engineering*, vol. 131, no. 7, pp. 544–551, 2005.

[208] 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, 2017.

[209] T. Szydlo, A. Szabala, N. Kordiumov, K. Siuzdak, L. Wolski, K. Alwasel, F. Habeeb, and R. Ranjan, "Iotsim-osmosis-res: Towards autonomic renewable energy-aware osmotic computing," *Software: Practice and Experience*, vol. 52, no. 7, pp. 1698–1716, 2022.

[210] A. Pell, A. Meingast, and O. Schauer, "Trends in real-time traffic simulation," *Transportation research procedia*, vol. 25, pp. 1477–1484, 2017.

[211] T. Loossens, F. Tuerlinckx, and S. Verdonck, "A comparison of continuous and discrete time modeling of affective processes in terms of predictive accuracy," *Scientific reports*, vol. 11, no. 1, pp. 1–11, 2021.

[212] A. Dahlinger, B. Ryder, and F. Wortmann, "Car as a sensor. paying people for providing their car data," in *Proceedings of the 5th International Conference on Internet of Things, Seoul, South Korea*, 2015.

[213] N. Baranasuriya, V. Navda, V. N. Padmanabhan, and S. Gilbert, "QProbe: Locating the bottleneck in cellular communication," in *CoNEXT*, 2015.

[214] J. Kang, X. Chen, D. Wu, Y. T. Xu, X. Liu, G. Dudek, T. Lee, and I. Park, "Hierarchical policy learning for hybrid communication load balancing," in *ICC 2021 - IEEE Int. Conf. on Communications*, 2021, pp. 1–6.

[215] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, "Osmotic computing: A new paradigm for edge/cloud integration," *IEEE Cloud Computing*, vol. 3, no. 06, pp. 76–83, nov 2016.

[216] K. Alwasel, D. N. Jha, E. Hernandez, D. Puthal, M. Barika, B. Varghese, S. K. Garg, P. James, A. Zomaya, G. Morgan *et al.*, "Iotsim-sdwan: A simulation framework for interconnecting distributed datacenters over software-defined wide area network (sd-wan)," *Journal of Parallel and Distributed Computing*, vol. 143, pp. 17–35, 2020.

[217] S. Häger, S. Böcker, and C. Wietfeld, "3D self-motion tracking services: Coalescence of mmWave beam orientations and phase information," in *2022 IEEE FNWF*, oct 2022.

[218] V. Chang, Q. A. Xu, K. Hall, O. T. Oluwaseyi, and J. Luo, "Comprehensive analysis of uk aadf traffic dataset set within four geographical regions of england," *Expert Systems*, vol. 40, no. 10, p. e13415, 2023.

[219] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns: Elements of reusable object oriented software," 1995.

[220] S. Laso, J. Berrocal, P. Fernandez, J. M. García, J. Garcia-Alonso, J. M. Murillo, A. Ruiz-Cortés, and S. Dustdar, "Elastic data analytics for the cloud-to-things continuum," *IEEE Internet Computing*, vol. 26, no. 6, pp. 42–49, Nov 2022.

[221] F. Li, W. Liu, W. Gao, Y. Liu, and Y. Hu, "Design and reliability analysis of a novel redundancy topology architecture," *Sensors*, vol. 22, no. 7, 2022. [Online]. Available: μhttps://www.mdpi.com/1424-8220/22/7/2582

[222] "Ieee standard for local and metropolitan area networks – link aggregation," *IEEE Std 802.1AX-2014 (Revision of IEEE Std 802.1AX-2008)*, pp. 1–344, 2014.

[223] Z. Lv, B. Hu, and H. Lv, "Infrastructure monitoring and operation for smart cities based on iot system," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1957–1962, 2019.

[224] R. Smith, D. Palin, P. P. Ioulianou, V. G. Vassilakis, and S. F. Shahandashti, "Battery draining attacks against edge computing nodes in iot networks," *Cyber-Physical Systems*, vol. 6, no. 2, pp. 96–116, 2020.

[225] K.-F. Krentz, C. Meinel, and H. Graupner, "Countering three denial-of-sleep attacks on contikimac." in *EWSN*, vol. 17, 2017, pp. 108–119.

[226] D. R. Raymond, R. C. Marchany, M. I. Brownfield, and S. F. Midkiff, "Effects of denial-of-sleep attacks on wireless sensor network mac protocols," *IEEE transactions on vehicular technology*, vol. 58, no. 1, pp. 367–380, 2008.

[227] V. P. Singh, S. Jain, and J. Singhai, "Hello flood attack and its countermeasures in wireless sensor networks," *International Journal of Computer Science Issues (IJCSI)*, vol. 7, no. 3, p. 23, 2010.

[228] E. Y. Vasserman and N. Hopper, "Vampire attacks: Draining life from wireless ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 2, pp. 318–332, 2011.

[229] N. Rathi, M. Kakani, M. El-Sharkawy, and M. Rizkalla, "Wearable low power pre-fall detection system with iot and bluetooth capabilities," in *2017 IEEE National Aerospace and Electronics Conference (NAECON)*. IEEE, 2017, pp. 241–244.

[230] W. Lee, S. Cho, P. Chu, H. Vu, S. Helal, W. Song, Y.-S. Jeong, and K. Cho, "Automatic agent generation for iot-based smart house simulator," *Neurocomputing*, vol. 209, pp. 14–24, 2016.

[231] J. S. Hunter, "The exponentially weighted moving average," *Journal of Quality Technology*, vol. 18, no. 4, pp. 203–210, 1986.

[232] Y. Zhao, I. Megdiche, F. Ravat, and V.-n. Dang, "A zone-based data lake architecture for iot, small and big data," in *Proceedings of the 25th International Database Engineering & Applications Symposium*, ser. IDEAS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 94–102.

[233] A. Kaushik, R. Singh, S. Dayarathna, R. Senanayake, M. Di Renzo, M. Dajer, H. Ji, Y. Kim, V. Sciancalepore, A. Zappone *et al.*, "Toward integrated sensing and communications for 6g: Key enabling technologies, standardization, and challenges," *IEEE Communications Standards Magazine*, vol. 8, no. 2, pp. 52–59, 2024.

[234] W. Jiang, Q. Zhou, J. He, M. A. Habibi, S. Melnyk, M. El-Absi, B. Han, M. Di Renzo, H. D. Schotten, F.-L. Luo *et al.*, "Terahertz communications and sensing for 6g and beyond: A comprehensive review," *IEEE Communications Surveys & Tutorials*, 2024.

[235] N. Alhussien and T. A. Gulliver, "Toward ai-enabled green 6g networks: A resource management perspective," *IEEE Access*, 2024.

[236] S. Islam, Z. A. Atallah, A. K. Budati, M. K. Hasan, R. Kolandaisamy, and S. Nurhizam, "Mobile networks toward 5g/6g: Network architecture, opportunities and challenges in smart city," *IEEE Open Journal of the Communications Society*, 2024.