# Towards Scalable Architectures in oneM2M-based Interoperability deployments in Smart Cities

Thesis submitted in partial fulfillment
of the requirements for the degree of

*Master of Science*
*in*
*Computer Science and Engineering*
by Research

by

VJS Pranavasri
2020121001
vjs.pranavasri@research.iiit.ac.in

Advisor: Dr. Deepak Gangadharan
Co-Advisor: Dr. Karthik Vaidhyanathan

International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500 032, INDIA

June 2024

Copyright © VJS Pranavasri, 2024
All Rights Reserved

International Institute of Information Technology
Hyderabad, India

# CERTIFICATE

It is certified that the work contained in this thesis, titled ***Towards Scalable Architectures in oneM2M-based Interoperability deployments in Smart Cities*** by *VJS Pranavasri* has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Advisor: Dr. Deepak Gangadharan

To

My Family and Friends

# Acknowledgments

Completing this thesis has been a profoundly challenging and enriching experience, and there are many who have guided and supported me through this journey. I would like to express my deepest gratitude to each of them.

While the constraints of this paper limit me from mentioning everyone individually, it does not diminish the appreciation I hold for their contributions. Their support, whether through insightful discussions, constructive criticism, or simply being a source of encouragement, has been invaluable to me.

First and foremost, I extend my sincere thanks to my advisor, Dr. Deepak Gangadharan, for his belief in me and his invaluable assistance throughout my research. His guidance was crucial in shaping a solid foundation for my work and ensuring my timely graduation. Dr. Gangadharan provided me with numerous opportunities, offered constant support, and delivered critical reviews that significantly contributed to my research's success.

I am immensely grateful to Dr. Karthik Vaidhyanathan for his exceptional support and mentorship. Not only for serving as my co-advisor but also for his inspirational influence on both my academic and personal life. Dr. Vaidhyanathan's guidance not only directed my research path but also empowered me to surmount significant obstacles. He instilled in me a profound sense of confidence and dedication to my interests and aspirations. Throughout my research, Dr. Vaidhyanathan provided invaluable guidance, offering crucial insights and nurturing a spirit of intellectual curiosity. Words fall short in expressing my appreciation for his mentorship and presence throughout this journey.

I also wish to thank Mrs. Anuradha Vattem and all my lab mates at the Smart City Living Lab of IIIT-H for enabling my work. Their support was pivotal in allowing me to research and validate my work through practical, real-life use cases. My heartfelt appreciation goes out to my dedicated co-authors – Francis Leo, Mogadali Ushasri, Pal Gaurav, and Vaddhiparthy Suhas – whose assistance in setup, testing, and extensive result analysis was invaluable to my research.

On a personal note, special thanks are due to Prince Varshney, Aditya Hari, and Snehal Ranjan for their unconditional friendship and emotional support during some of the most challenging times of my research journey. Their encouragement and understanding have been indispensable, and I am truly grateful for their presence in my life.

Lastly, I owe a debt of gratitude to my parents. Their endless love, belief, and support have been my constant source of strength and courage through all the ups and downs of this journey. Their faith in me has been unshakeable, and for that, I am eternally grateful.

To all of you who have been part of my journey, directly or indirectly, thank you. This accomplishment is not only mine but also yours.

# Abstract

Smart Cities represent the nexus of innovation in urban planning, leveraging the Internet of Things (IoT) to foster enhanced urban living. However, the diversity of IoT devices and protocols often creates interoperability barriers. oneM2M, a global standard, addresses this challenge by providing a common service layer that enables seamless communication and data exchange between heterogeneous IoT devices along with features for device management, security, and semantic interoperability. Its resource-based model and support for various communication protocols make it a key enabler for interoperable smart city solutions. This enables a wide range of intelligent smart city applications, including coordinated traffic management, intelligent energy grids, and responsive public services. This thesis zeroes in on the pivotal aspect of interoperability within Smart Cities, with a particular focus on oneM2M-based interoperability systems. Through a detailed exploration of oneM2M standards and their application in the Smart City context, this research underscores the importance of a unified, standards-based approach to IoT integration, highlighting the role of interoperability in unlocking the full potential of Smart City initiatives.

In addressing the critical challenge of enhancing latency and performance in oneM2M-based systems within Smart City frameworks, this thesis recognizes the limitations of existing architectures, which often impede scalability and efficiency. A novel distributed, multi-layered data platform architecture for oneM2M based interoperability platforms is proposed, aiming to demonstrate significant improvements in latency and overall system performance while ensuring scalability across diverse and large-scale Smart City deployments. The research encompasses a comprehensive performance analysis of this architecture versus a centralized alternative within a real-world Smart City Living Lab deployment. Additionally, an exploratory analysis of open-source oneM2M-based interoperability systems (Mobius, OM2M, ACME) using this large-scale system offers insights into architectural choices and their suitability for smart city implementations. Another challenge that becomes fairly evident is the complexity that arises in the usage of the said interoperability solutions based on oneM2M. To increase the ease of use and of the oneM2M based solutions within the IoT ecosystems, this thesis presents the City IoT Operating Platform (ctOP). This platform, meticulously engineered as a lightweight oneM2M wrapper, is designed to enhance interoperability, streamline IoT device integration, and provide robust functionality such as user management and security within real-world smart city deployments.

This contribution is pivotal for the IoT field, challenging the status quo of existing oneM2M system design paradigms and proposing a more adaptable, scalable, and user-friendly alternative. Through rigorous analysis, real-world deployment scenarios, and the development of the ctOP platform, this

research contributes to both the academic discourse on Smart City development and provides actionable insights for practitioners and policymakers seeking to implement or refine IoT interoperability solutions in urban settings. The ultimate goal is to pave the way for more interconnected, sustainable, and resilient Smart Cities, where technology serves as a backbone for addressing complex urban challenges.

# Contents


Chapter Page

1 Introduction . . . 1
1.1 Motivation . . . 1
1.2 Contributions . . . 3
1.2.1 Scalable Architecture for oneM2M-Based Smart City Deployments . . . 3
1.2.2 Architecture and Performance Analysis of Open-source oneM2M Platforms . . . 3
1.2.3 ctOP: A Real-World Deployment of a oneM2M-Compliant Smart City System 4
1.3 Organisation of Thesis . . . 4

2 IoT in Smart Cities . . . 6
2.1 Introduction to IoT . . . 6
2.2 Architecture of IoT . . . 7
2.3 Smart Cities and IoT . . . 8
2.4 Importance of Interoperability in IoT . . . 10
2.5 Introduction to oneM2M . . . 10
2.6 Smart City at IIITH . . . 11

3 Interoperability in Smart Cities: A literature survey . . . 14
3.1 Introduction to Interoperability Frameworks and Standards in Smart Cities . . . 14
3.2 Scalable and Distributed Architectures for Interoperability . . . 15
3.3 Case Studies and Real-World Deployments . . . 15
3.4 Gaps in Current Research . . . 15

4 Designing a Scalable Architecture for oneM2M . . . 18
4.1 Introduction . . . 18
4.2 Proposed Architecture . . . 19
4.2.1 Data Monitoring Layer (DML) . . . 19
4.2.2 Data Enhancement Layer (DEnL) . . . 20
4.2.3 Data Storage Layer (DSL) . . . 20
4.2.4 Data Exchange Layer (DEL) . . . 20
4.3 Implementation . . . 21
4.3.1 Data Monitoring Layer . . . 21
4.3.2 Data enhancement Layer . . . 21
4.3.3 Data Storage Layer . . . 22
4.3.4 Data Exchange Layer . . . 22
4.4 Results . . . 22

4.4.1 Improvements in the Data Monitoring Layer . . . 22
4.4.2 Improvements in Data Enhancement and Control Layer . . . 23
4.4.3 Improvements in Data Exchange Layer . . . 24
4.5 Discussion . . . 24
4.5.1 Lessons Learned . . . 24
4.5.2 Threats to Validity . . . 25
4.6 Summary . . . 25

5 Performance Analysis of Multiple oneM2M Systems . . . 28
5.1 Introduction . . . 28
5.2 Motivation . . . 29
5.3 Study Design and Execution . . . 30
5.3.1 Goals . . . 30
5.3.2 Research Questions . . . 30
5.3.3 Experiment Design . . . 30
5.3.4 Experiment Candidates . . . 32
5.3.4.1 OM2M Architecture . . . 32
5.3.4.2 ACME Architecture . . . 33
5.3.4.3 Mobius Architecture . . . 35
5.4 Results . . . 36
5.4.1 Synthetic Workload . . . 36
5.4.1.1 Response Time and Latency . . . 36
5.4.1.2 CPU Stats . . . 37
5.4.2 Real Workload . . . 37
5.4.2.1 Response Time and Latency . . . 37
5.4.2.2 Emulation of Real System . . . 37
5.4.2.3 Stress Test . . . 38
5.5 Discussion . . . 39
5.5.1 Threats to Validity . . . 41
5.6 Summary . . . 41

6 Real World Deployment . . . 42
6.1 Introduction . . . 42
6.2 Motivation and Design Philosophy . . . 42
6.3 ctOP Architecture . . . 43
6.4 Features and Functionalities . . . 44
6.5 Implementation and Deployment . . . 45
6.5.1 Technical Overview . . . 46
6.5.2 Implementation Strategy and Deployment Process . . . 46
6.5.3 Naming Convention for Nodes . . . 46
6.5.4 Real-World Applications . . . 47
6.6 Workflow . . . 48
6.7 Future Directions . . . 48
6.7.1 Transition to Mobius for Performance Enhancement . . . 48
6.7.2 Clustered/Distributed Deployment Approach . . . 49
6.7.3 Implementation of City-Level OM2M Instances . . . 49

7 Conclusion . . . . . 50
7.1 Conclusion . . . . . 50
7.2 Future Work . . . . . 51

*Appendix A*: Low Level Workflow Diagrams for ctOP . . . . . 52

*Appendix B*: ctOP Pages . . . . . 57

Bibliography . . . . . 59

# List of Figures


Figure Page

2.1 IoT Layer Architecture . . . 7
2.2 oneM2M Layered Architecture with CSEs [11] . . . 9
2.3 Smart City Living Lab Data Architecture [1] . . . 12

3.1 Smart City Architecture at IIIT Hyderabad [8] . . . 16
3.2 Implementation Technologies at Smart City, IIIT Hyderabad [8] . . . 17

4.1 Proposed architecture of the Distributed OM2M . . . 26
4.2 Implementation of the Distributed OM2M . . . 27
4.3 Thingsboard Rule Chain . . . 27
4.4 OM2M retrieval performance characteristic . . . 27

5.1 OneM2M Architecture . . . 29
5.2 OM2M Architecture . . . 33
5.3 ACME Architecture . . . 34
5.4 Mobius Architecture . . . 35
5.5 Different request pattern comparison . . . 36
5.6 Emulation of the real workload . . . 37
5.7 CPU usage statistics with respect to number of concurrent users . . . 39

6.1 The ctOP architecture . . . 43
6.2 The ctOP data model . . . 44
6.3 High-Level Overview of ctOP Workflow . . . 47

A.1 Data Posting . . . 52
A.2 User Auth . . . 53
A.3 Domain Management . . . 54
A.4 Nodes Management . . . 55
A.5 Sensor Type Management . . . 56

B.1 ctOP Deployment Screenshots . . . 58

# List of Tables


Table Page

4.1 Comparision of performance across different load balancers for retrieval . . . . . 23
4.2 Comparision of performance across different load balancers for insertions . . . . . 23
4.3 Comparison of insertion performance through RESTful API vs our DEnL consisting of Data Pre-Processing and Publish Subscribe . . . . . 24
4.4 Comparison of retrieval performance from om2m to the implemented middleware for the latest data . . . . . 24

5.1 Stress Testing POST Requests on Real Workload . . . . . 38
5.2 Stress Testing GET Requests on Real Workload . . . . . 38

# Abbreviations

| | |
|---|---|
| ACP | Access Control Policy |
| ACS | Autonomic Computing Service |
| AE | Application Entitiy |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| AQ | Air Quality |
| AWS | Amazon Web Services |
| CBOR | Concise Binary Object Representation |
| CM | Crowd Monitoring |
| CoAP | Constrained Application Protocol |
| CORS | Cross-origin resource sharing |
| CPU | Central Processing Unit |
| CRUD | Create, Read, Update, Delete |
| CSF | Common Service Function |
| ctOP | City IoT Operating Platform |
| DEL | Data Exchange Layer |
| DEnL | Data Enhancement Layer |
| DML | Data Monitoring Layer |
| DSL | Data Storage Layer |
| EM | Energy Monitoring |
| ETL | Extract, Transform, Load |
| GQMA | Goal Question Metric Approach |
| HTTP | Hyper Text Transfer Protocol |
| IIITH | IIIT Hyderabad |
| IN-CSE | Infrastructure Node Common Service Entity |
| IoT | Internet of Things |
| IUDX | Indian Urban Data Exchange |
| JSON | JavaScript Object Notation |
| KPI | Key Performance Indicator |

| | |
|---|---|
| LC | Least Connections |
| LoRaWAN | Long Range Wide Area Network |
| LPWAN | Low Power Wide Area Network |
| M2M | Machine-to-Machine |
| ML | Machine Learning |
| MoHUA | Ministry of Housing and Urban Affairs |
| MQTT | Message Queuing Telemetry Transport |
| OM2M | Open Machine-to-Machine |
| OSGi | Open Service Gateway Initiative |
| RAM | Random Access Memory |
| REST | Representational State Transfer |
| RR | Round Robin |
| SCL | Service Capability Layer |
| SCM | Smart City Mission |
| SCRC | Smart City Research Center |
| SL | Solar Monitoring |
| SR | Smart Rooms Monitoring |
| TLS | Transport Layer Security |
| WE | Weather Monitoring |
| Wi-Fi | Wireless Fidelity |
| Wi-SUN | Wireless Smart Utility Network |
| WM | Water Monitoring |
| WN | Wi-SUN Smart Street Lamps |
| XML | Extensible Markup Language |

# List of Related Publications

[P1] **Pranavasri VJS**, Francis Leo, Mogadali Ushasri, Pal Gaurav, Vaddhiparthy Suhas, Vattem Anuradha, Vaidhyanathan Karthik and Gangadharan Deepak. **"Scalable and Interoperable Distributed Architecture for IoT in Smart Cities"**, in proceedings of *IEEE 9th World Forum on Internet of Things (WF-IoT)*, 2023.

[P2] **Pranavasri VJS**, Francis Leo, Pal Gaurav, Mogadali Ushasri, Vattem Anuradha, Vaidhyanathan Karthik and Gangadharan Deepak. **"Exploratory Study of oneM2M-based Interoperability Architectures for IoT: A Smart City Perspective"**, in proceedings of *21st IEEE International Conference on Software Architecture (ICSA)*, Software Architecture in Practice (SAIP) Track, 2024.

# Related Software Artifacts:

[P3] **ctOP (City IOT Operating Platfotm),** a city level scalable interoperability platform following oneM2M standards. *GitHub Repository*. Available online: `https://github.com/ctOP-IIITH/`.

*Chapter 1*

# Introduction

This chapter explores the transformative impact of the Internet of Things (IoT) and Smart City initiatives on the future of urbanization. It examines the challenges that arise as cities expand, and how IoT innovations offer solutions for optimizing resource management, improving the quality of life for residents, and reducing environmental impact. The chapter then delves into the critical importance of data-driven decision-making in Smart Cities and introduces the Smart City Living Lab at IIITH [1] as a pioneering initiative to address these challenges. The need for robust data management and interoperability within IoT ecosystems is emphasized, setting the stage for the thesis's focus on architectural enhancements for oneM2M-based smart city solutions.

## 1.1 Motivation

Smart Cities are becoming increasingly common as urbanization and technological advancements continue to reshape our world. The surge in global urbanization presents multifaceted challenges, including heightened demand for resources, increased environmental pollution, and exacerbated strain on infrastructure and public services. As urban areas continue to expand, traditional city planning and resource management approaches are proving inadequate to sustainably meet the needs of growing populations. This is where the concept of Smart Cities, underpinned by the Internet of Things (IoT), plays a transformative role. By integrating digital technology into urban management, Smart Cities aim to enhance the efficiency of city operations, improve the quality of life for residents, and minimize environmental impact. IoT, with its network of interconnected devices and sensors, enables real-time data collection, analysis, sensing, actuation, and seamless communication between diverse nodes, facilitating informed decision-making and proactive management of urban environments. Smart Cities are becoming increasingly common as urbanization and technological advancements continue to reshape our world.

The role of Smart Cities and IoT extends beyond mere technological innovation; it represents a comprehensive approach to addressing the sustainability and livability challenges of urbanization. For instance, IoT-enabled smart grid technologies can optimize energy consumption, reducing the overall carbon footprint of cities. Similarly, smart water management systems can monitor and control water

usage in real time, ensuring efficient use of this precious resource. In transportation, IoT applications can streamline traffic flow, enhance public transportation systems, and reduce congestion, significantly lowering urban areas' environmental impact and improving residents' quality of life. Furthermore, smart city initiatives contribute to economic development by attracting businesses that value sustainability and innovation, creating a virtuous cycle of growth and improvement. Thus, Smart Cities and IoT are not just technological trends but essential components of a sustainable urban future, offering solutions to some of the most pressing challenges of our times.

One critical challenge that emerges post the conceptualization of a smart city is the utilization of data to engineer solutions tailored to urban challenges. In the context of India's diverse landscape, this task assumes an added layer of complexity. The diverse needs of the Indian populace demand customized solutions, a realization that served as a catalyst for establishing the Smart City Living Lab at IIIT Hyderabad (IIITH). This facility, a pioneering Smart City testbed, is designed to implement, test, and refine solutions in a real-world environment, thereby enabling the development of technologies that are specifically catered to the Indian demographic. The lab's objective is to harness data collected from IoT devices across the smart city ecosystem to innovate and address various urban issues effectively.

However, a fundamental challenge across smart cities and IoT ecosystems is ensuring interoperability between the diverse range of devices and systems that utilize different protocols and standards. This encompasses not only the seamless transfer of data between devices (nodes) but also the ability of these devices to understand and interact with each other effectively. The sheer volume and variety of data, coupled with the heterogeneity of devices and their protocols, demand a robust architecture to ensure interoperability and seamless interaction. It is this foundational infrastructure that the Smart City Living Lab exemplifies by providing a platform to experiment with these crucial elements. To fully realize the potential of this solution, scaling its interoperability and data streaming architecture becomes paramount.

This thesis delves into the intricacies of this architecture, examining its current state, analyzing its effectiveness, and proposing architectural enhancements following the oneM2M standard. oneM2M is a global initiative that provides a standardized framework for IoT interoperability, ensuring compatibility and communication between diverse devices and systems. By adhering to this standard, the proposed architectural improvements aim to advance IoT interoperability within the smart cities framework, ensuring that solutions are not only innovative but also scalable and adaptable to the evolving needs of urban environments. By focusing on these architectural improvements, the thesis aims to contribute to the broader discourse on smart city development, particularly in the context of India's unique urban challenges. Through a detailed analysis and proposed enhancements, this research endeavors to pave the way for more resilient, efficient, and robust smart cities.

## 1.2 Contributions

This thesis addresses the challenge of achieving seamless interoperability within smart city ecosystems, where diverse IoT devices and systems need to communicate and collaborate effectively. It makes three key contributions towards this goal:

### 1.2.1 Scalable Architecture for oneM2M-Based Smart City Deployments

The increasing prevalence of IoT devices and the emergence of smart cities have placed significant demands on existing IoT frameworks. This research analyzes the current architecture of oneM2M, a prominent IoT interoperability standard, within the context of a real-world smart city deployment at IIIT Hyderabad. By identifying potential bottlenecks and analyzing their causes, this thesis proposes a redesigned scalable and distributed architecture. This novel architecture, comprising Data Monitoring, Storage, Enhancement, and Exchange Layers, facilitates efficient data handling, high throughput, and cross-platform compatibility. Performance evaluations demonstrate significant improvements compared to the existing centralized architecture, with up to 41.23% increase in throughput and 29.19% reduction in latency for data insertion. These findings pave the way for more robust and scalable oneM2M deployments in smart cities. The following are major proposals presented in this work:

- Analyzing the existing architecture of oneM2M in our smart city deployment
- Finding potential bottleneck causes and analyzing them
- Proposing a scalable and distributed architecture for oneM2M-based interoperability platforms
- Comparing the performance of the new architecture with an existing one

### 1.2.2 Architecture and Performance Analysis of Open-source oneM2M Platforms

While oneM2M provides a standardized framework for interoperability, various open-source platforms implement it with different architectural approaches. This thesis conducts a detailed analysis of three prominent platforms – Mobius, OM2M, and ACME – within the context of our smart city deployment. The analysis focuses on their architectural characteristics, architectural differences, support for oneM2M standards. Additionally, the research compares their performance under different scenarios, utilizing relevant metrics to assess their effectiveness in real-world deployments. This comparative analysis provides valuable insights for stakeholders seeking to implement oneM2M-based solutions, offering guidance on selecting the most suitable platform based on specific quality constraints and project requirements. We summarize the work as follows:

- Architecture analysis and comparison of three different open-source oneM2M platforms: OM2M, Mobius and ACME.
- Performance analysis of the three platforms under multiple load conditions

- Co-relating performance to architectural differences.

### 1.2.3 ctOP: A Real-World Deployment of a oneM2M-Compliant Smart City System

This thesis goes beyond Proof of Concept analysis by delving into the practicalities of deploying a oneM2M-based platform in a large-scale smart city environment. It discusses the complexities involved specifically the technical hurdles. Furthermore, the work outlines the process of establishing a data model and interoperability standard based on the oneM2M framework and the specific needs followed at our smart city deployment. This majorly comprises developing a user-friendly, scalable system designed for extensive deployments, with features that facilitate device management, user management and access control policy, and subscription to data, amongst others. Our contributions in real-world deployment underscore the practical significance of interoperable IoT solutions, offering practical insights and a roadmap for achieving interoperability in real-world scenarios. The process of development/deployment of ctOP is as follows:

- Analyzing the complexity of setting up a oneM2M-based platform
- Establishing a data model and interoperability standard based on our oneM2M-based smart city deployment
- Developing an easy-to-use oneM2M compliant system for large-scale city and state-wide deployments of smart cities

These three contributions, while distinct in their focus, are interconnected by the overarching goal of enabling seamless interoperability within smart cities. The proposed scalable architecture lays the foundation for efficient data handling and communication, while the comparative analysis of oneM2M platforms provides guidance for choosing the most suitable solution. Finally, the real-world deployment and interoperability standard development demonstrate the practical application of these concepts and offer valuable insights for future smart city initiatives.

## 1.3 Organisation of Thesis

The remaining thesis is organised as follows:

- Chapter 2 introduces the concept of the Internet of Things (IoT) and discusses its importance in modern technology. The chapter then explores how IoT technologies are leveraged in smart cities to enhance urban living, providing a bridge to the following discussions on interoperability and standards. It further underscores the critical importance of interoperability within the IoT ecosystem and introduces the oneM2M standard as a solution for achieving this interoperability, outlining its role in facilitating communication among diverse IoT devices and platforms.

- Chapter 3 delves into the existing body of research and developments in the field of IoT interoperability, specifically within the realm of smart cities. It surveys various approaches comparing, their methodologies, applications, and efficacy in enhancing interoperable communication. This chapter highlights pivotal studies and projects that have shaped the landscape of IoT interoperability, setting the stage for understanding the significance of the proposed architecture and analyses in subsequent chapters.

- Chapter 4 proposes a new scalable and distributed architecture specifically designed for oneM2M-based interoperability solutions. This chapter outlines the design considerations and benefits of this architecture, emphasizing its potential to support the growing demands of smart city infrastructures.

- Chapter 5 offers a comparative analysis of three open-source oneM2M implementations: OM2M, Mobius, and ACME. It evaluates their architectures, performance, and suitability for smart city applications, providing insights into their strengths and weaknesses.

- Chapter 6 describes the practical contribution of this thesis through the development of an accessible oneM2M-compliant platform. This platform is designed for straightforward deployment in large-scale scenarios, from city-wide to state-wide projects, illustrating the feasibility and scalability of the proposed solution.

- Chapter 7 concludes the thesis by summarizing the key findings and contributions. It reflects on the impact of the work within the broader IoT and smart city domains and suggests directions for future research.

*Chapter 2*

# IoT in Smart Cities

This chapter embarks on a journey through the foundational concepts of the IoT, a transformative force reshaping our world by integrating digital intelligence into everyday objects. We begin with global definitions, navigating through the multi-layered architecture of IoT, and exploring its applications across sectors. We then direct our focus towards IoT in smart cities, where it serves as the backbone for innovative urban solutions enhancing sustainability, efficiency, and quality of life. Central to this narrative is the importance of interoperability among IoT devices and systems, a challenge addressed by the oneM2M standard, which aims to ensure seamless communication and integration. Readers seeking to delve deeper into the technical, architectural, and application-specific aspects of IoT can go through the following references: [2–4]. Additionally, the oneM2M initiative's official documentation [5] offers detailed information on standards for IoT interoperability, crucial for advancing IoT solutions in smart cities and beyond.

## 2.1 Introduction to IoT

The Internet of Things IoT stands as a beacon of the fourth industrial revolution, heralding an era where digital and physical realms converge to spawn interconnected environments brimming with intelligence. At its core, IoT embodies the concept of imbuing everyday objects with connectivity and computational capabilities, enabling them to collect, transmit, and process data autonomously. This paradigm shift not only redefines human interaction with technology but also promises to streamline operations across diverse sectors, including healthcare, agriculture, urban development, and manufacturing. By facilitating direct integration of the physical world into computer-based systems, IoT paves the way for improved efficiency, economic benefits, and reduced human exertion.

IoT is built upon a set of fundamental concepts that emphasize seamless connectivity, intelligent automation, and the extensive use of data analytics to derive meaningful insights from vast networks of interconnected devices. These principles enable not just the remote monitoring and management of devices but also the creation of systems that can learn, adapt, and potentially act autonomously. The evolution of IoT can be traced back to the advent of RFID technology [6] and the pioneering work on connected appliances in the late 20th century. Since then, IoT has expanded exponentially, fueled

by advances in sensor technology, miniaturization, wireless communication, and cloud computing. This progression has transformed IoT from a niche concept into a global phenomenon that influences virtually every sector of the economy.

## 2.2 Architecture of IoT

The architecture of the IoT is a foundational framework designed to support the vast and varied applications of IoT devices and systems. A few of these can be seen in the existing literature [6–8]. It encapsulates the journey of data from its initial collection by sensors to its ultimate utilization in applications, governed by a series of interconnected layers that manage communication, processing, and storage. This structure, while flexible to accommodate different use cases, generally comprises several key layers. Notably, despite the wide range of applications, a striking conceptual similarity exists across different IoT architectures. Drawing upon the work of A. Taivalsaari [9], we present an overview of an end-to-end architecture for IoT solutions in Figure 2.1.

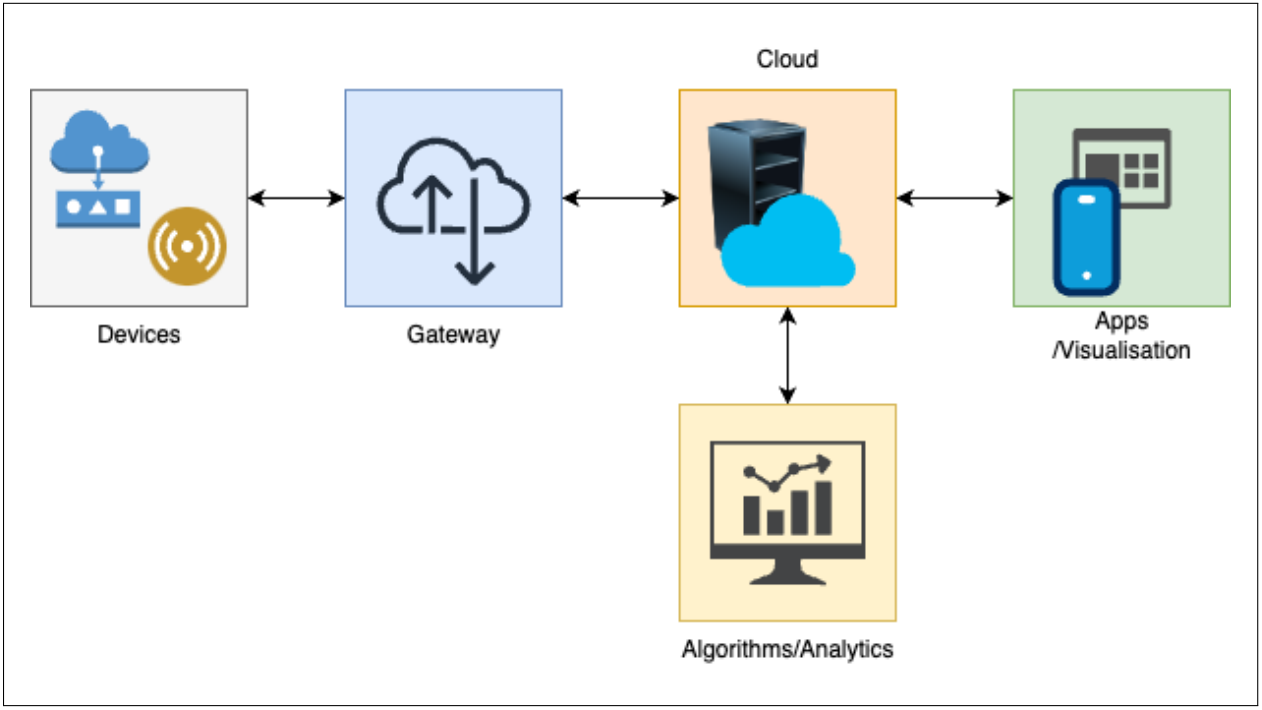



**Figure 2.1** IoT Layer Architecture

- **IoT Devices/Nodes:** At the base of the IoT architecture are the nodes, sensors, or IoT devices that act as the primary data sources. These elements are equipped with the necessary sensors and actuators to collect environmental data or perform specific actions based on the instructions received. They are the frontline players in the IoT ecosystem, capturing real-time data from the physical world, which can range from temperature readings to complex multimedia inputs.

- **Gateway:** Once data is collected, it needs to be transmitted to other parts of the IoT system for processing and analysis. The gateway facilitates this by providing the protocols and technologies necessary for data transmission. This layer encompasses a wide range of technologies, including traditional Wireless Fidelity (Wi-Fi), Bluetooth, Zigbee, and newer protocols like Low Power Wide Area Network (LPWAN) technologies that are specifically designed for IoT applications requiring low power consumption and long-range communication.

  **Cloud:** Serving as the central hub of the IoT architecture, the cloud layer (also referred to as middleware layer) manages the reception of data from various devices and orchestrates its flow within the system. It ensures data is appropriately routed, processed, and made available to the upper layers. This layer often includes data processing capabilities, enabling the preliminary analysis, filtering, or aggregation of data before it reaches the application layer. It also manages device registration, authentication, and security, ensuring that communications are secure and that devices are properly authenticated.

- **Apps/Visualisation:** This, referred to also as the application layer, is where data becomes actionable insight. It consists of software applications that utilize the processed data to perform specific functions or services for the user. This could range from simple notifications about sensor readings to complex analytics and decision-making tools that leverage AI and machine learning to offer predictive insights. The application layer is the most visible part of the IoT ecosystem to end-users, delivering the tangible benefits of IoT technology through user interfaces and interactions.

- **Algorithms and analytics:** Integral to the IoT architecture is the analytics layer, which also acts as a pre-processing unit to provide data to the visualisation layer, which could be used for features such as notifications mentioned above.

Each layer of the IoT architecture plays a crucial role in ensuring the seamless operation of IoT systems, from data collection to actionable insights. While this architecture provides a general framework, specific implementations can vary widely depending on the particular requirements of each IoT application, from smart homes to industrial IoT systems. Understanding this layered architecture is essential for developing, deploying, and managing effective and efficient IoT solutions.

## 2.3 Smart Cities and IoT

The integration of IoT within smart cities epitomizes the transformative potential of this technology to redefine urban living [10]. By embedding IoT devices across various segments of urban infrastructure, from traffic management systems to waste disposal, energy distribution, and public safety networks, cities become more than just conglomerates of structures; they evolve into interconnected, intelligent ecosystems. These IoT devices collect, analyze, and act upon data in real-time, enabling city administrators to optimize resources, improve services, and enhance the quality of life for residents.

For instance, IoT-enabled traffic lights can adjust signal timings based on real-time traffic conditions, reducing congestion and improving road safety. Similarly, smart energy meters can help in managing energy consumption more efficiently, contributing to the sustainability goals of the urban area.

The seamless operation of these diverse IoT applications within the smart city framework underscores the critical importance of interoperability. Without the ability to communicate and exchange data across different systems and platforms, the potential of IoT in smart cities cannot be fully realized. Interoperability ensures that devices from various manufacturers and systems of different standards can work together, enabling a cohesive and integrated urban IoT ecosystem. As we delve into the next section, "The Importance of Interoperability in IoT," it becomes clear that developing and adhering to common standards and protocols is not just beneficial but essential for unlocking the full potential of smart cities. This foundation enables not only the efficient and effective implementation of IoT solutions but also paves the way for future innovations and scalability in the urban landscape.

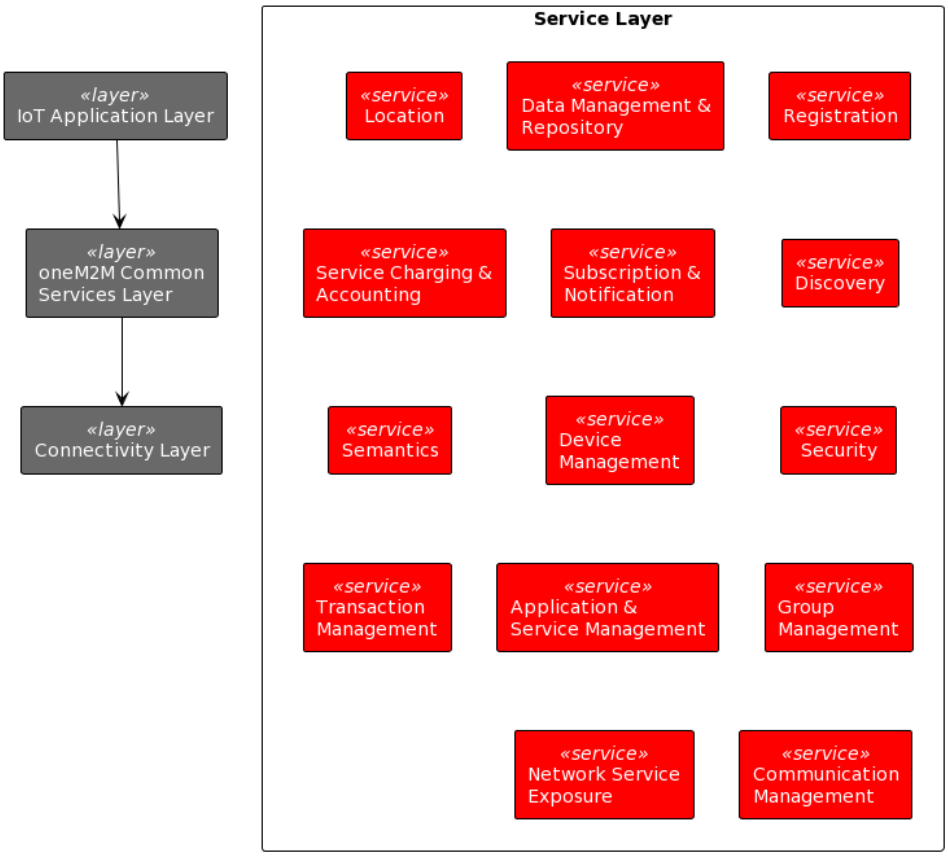



**Figure 2.2** oneM2M Layered Architecture with CSEs [11]

## 2.4 Importance of Interoperability in IoT

Interoperability within the Internet of Things (IoT) is foundational to unlocking the full potential of smart technologies, especially in the context of smart cities. The challenge of device heterogeneity, with myriad IoT devices from different manufacturers operating on various platforms and standards, necessitates a unified approach to ensure seamless communication. Interoperability is critical for integrating these diverse systems, allowing them to communicate effectively and share data without compatibility issues. This seamless interaction is vital for creating efficient, scalable, and flexible IoT ecosystems, where data from one device can be used to inform actions in another, regardless of their proprietary protocols or platforms. Without interoperability, the fragmentation of IoT systems leads to silos of inaccessible information, severely limiting the ability to leverage data for comprehensive insights and actions.

The need for standards and protocols in achieving interoperability cannot be overstated. Standards provide a common language and set of rules for device interaction, ensuring that different systems can understand and exchange data with each other. Protocols define the specific methods and processes for data transmission and reception, enabling the reliable and secure exchange of information. Together, standards and protocols form the backbone of interoperable IoT systems, facilitating the integration of disparate devices and platforms into a cohesive network. This integration is particularly crucial in smart cities, where the coordination of various IoT applications — from traffic management and environmental monitoring to energy distribution and public safety — is essential for optimizing urban operations and enhancing citizen well-being.

## 2.5 Introduction to oneM2M

one Machine-to-Machine (M2M) [oneM2M] stands as a beacon in the quest for IoT interoperability, representing a global initiative designed to establish a standardized framework that spans across industries and applications. Launched by eight of the world's leading telecommunications standards development organizations, alongside contributions from numerous industry stakeholders, oneM2M aims to address the pressing need for a common IoT standard that ensures seamless communication and interoperability among the myriad of IoT devices and applications. The initiative's objectives are rooted in facilitating the efficient development and deployment of IoT systems, reducing market fragmentation, and accelerating the delivery of IoT services to users and industries worldwide.

The structure of oneM2M is built around a comprehensive set of specifications that cover various aspects of IoT systems, including application layer protocols, security, data management, and device management. These specifications are designed to be agnostic of the underlying hardware and transport layers, making them applicable across different network technologies and IoT platforms. This flexibility is instrumental in fostering the integration and deployment of smart city solutions, where a wide range of devices and applications must work in concert to achieve the desired outcomes. By providing a standardized approach to IoT interoperability, oneM2M plays a pivotal role in simplifying the development

process for vendors, enhancing the scalability of IoT solutions, and ensuring that smart city initiatives can leverage the full capabilities of interconnected devices to improve urban life.

The oneM2M framework adopts a three-tiered layered architecture consisting of the Application Layer, the Common Services Layer, and the Connectivity Layer. This structured approach is pivotal in facilitating the modular and scalable design of IoT systems, ensuring they are robust and can be integrated across various domains and platforms. A more detailed exploration of the oneM2M architecture, including its intrinsic architectural characteristics, is provided in Chapter 5. Figure 2.2 illustrates the Common Service Functions (CSFs) as defined by the oneM2M standards. These functions are crucial for achieving a seamless interaction across different IoT devices and platforms, underpinning the essence of interoperability within the IoT landscape. By defining a uniform set of service capabilities, oneM2M lays the groundwork for diverse IoT applications to communicate and collaborate.

Through its commitment to open standards and collaboration, oneM2M not only paves the way for the seamless interoperation of IoT devices but also lays the foundation for future innovation and growth in the IoT sector. As cities around the globe strive to become smarter and more responsive to the needs of their citizens, the principles and framework provided by oneM2M will be crucial in realizing the vision of truly interconnected, intelligent urban environments.

## 2.6 Smart City at IIITH

The Smart City initiative at IIITH exemplifies a pioneering approach to leveraging IoT technologies for urban management and development. Spread across a sprawling 66-acre campus, the IIITH Smart City Living Lab integrates a comprehensive network of IoT nodes distributed over eight verticals or (Application Entitiy (AE)s) to collect, process, and analyze data pertinent to various aspects of city life and operations. Verticals/AEs refer to distinct categories or domains within the smart city framework. Each focused on a specific set of objectives or services, such as air quality monitoring, energy management, or traffic flow.

Our lab focuses on several key verticals within the smart city domain, including Air Quality (AQ), Energy Monitoring (EM), Solar Monitoring (SL), Smart Rooms Monitoring (SR), Weather Monitoring (WE), Water Monitoring (WM), Wi-SUN Smart Street Lamps (WN) and Crowd Monitoring (CM). These verticals leverage specialized IoT sensors and devices to gather real-time data relevant to their respective areas. With over 370 deployed nodes, we collect valuable insights. For example, the Air Quality vertical utilizes sensors to measure pollutants and particulate matter, aiding in environmental health assessments. Similarly, the Water Monitoring verticals optimize the management of campus water resources, while Crowd Monitoring analyzes pedestrian flows to enhance safety and operational efficiency.

Underpinning the diverse functionalities of these verticals is a layered IoT architecture meticulously designed to support the seamless collection, transmission, and utilization of data, as can be seen in Fig 2.3:


**Figure 2.3** Smart City Living Lab Data Architecture [1]

- **Device Layer:** This foundational layer houses the physical IoT devices and sensors deployed across the smart campus, serving as the primary sources of data collection.
- **Communication Layer:** Supporting a variety of protocols such as Wi-Fi, 4G, 5G, Long Range Wide Area Network (LoRaWAN), and Wireless Smart Utility Network (Wi-SUN), this layer ensures robust and flexible connectivity options for the transmission of data from the devices to the processing hubs.
- **Data Lake:** At the core of the project's interoperability efforts, the Data Lake adopts the OM2M standard by Eclipse, adhering to oneM2M guidelines. This setup facilitates the aggregation, storage, and preliminary analysis of data from various sources, ensuring that it can be effectively shared and utilized across different verticals.
- **Data Exchange:** The Indian Urban Data Exchange (IUDX) serves as the platform for data exchange, enabling the secure and efficient sharing of information between the Smart City's systems and external stakeholders. This layer plays a crucial role in bridging the gap between data collection and application development.
- **Application Layer:** This topmost layer encompasses the various business applications and user-facing services that leverage the processed data to deliver actionable insights, enhance decision-making, and improve campus operations.

The Smart City Living Lab at IIITH not only demonstrates the profound impact of IoT technologies on urban management but also highlights the significance of interoperability and standardized frameworks such as oneM2M in realizing the full potential of smart city initiatives. By fostering collaboration between different verticals and ensuring seamless data integration, the project stands as a model for smart city development, paving the way for scalable, efficient, and intelligent urban ecosystems.

---

*Chapter 3*

# Interoperability in Smart Cities: A literature survey

In the evolving landscape of smart cities, interoperability emerges as a cornerstone, enabling diverse IoT systems and devices to seamlessly communicate and collaborate. This chapter does an in-depth exploration in the domain of interoperability within smart city environments. This survey aims to encapsulate the current state of research and practice in the field through a detailed review of existing frameworks, standards, and architectures. Additionally, by extending the discussion to include real-world deployments, this chapter not only highlights theoretical advancements but also sheds light on practical implementations and the tangible challenges they face. The forthcoming sections will methodically dissect these areas, presenting a coherent narrative that bridges the gap between conceptual models and their operational realities. This overview sets the stage for a nuanced understanding of interoperability's pivotal role in the smart city ecosystem, paving the way for future innovations and enhancements in this dynamic and critical field.

## 3.1 Introduction to Interoperability Frameworks and Standards in Smart Cities

Interoperability within smart cities is crucial for ensuring seamless communication and integration of diverse IoT systems and devices, underpinning efficient data exchange and integration. Sethi and Sarangi [12], along with Ibrahim et al. [13], have underscored the significance of developing interoperable frameworks that can address the myriad challenges posed by the IoT architecture, including security, interoperability, QoS, and scalability. Among multiple interoperability frameworks, such as AWS IoT Core [14] and Azure IoT [15], oneM2M[1] has emerged as a global standards initiative, offering a comprehensive IoT interoperability standard. The oneM2M standard, according to Alaya et al. [16], aims to improve M2M communication through the introduction of IoT-O, an ontology that enables semantic data interoperability, leading to better device interoperability and automated reasoning for autonomic behaviour. This integration addresses the constraints of current M2M standards. Within oneM2M standards, several interoperability architectures have evolved, presenting viable solutions to facilitate data

[1] https://onem2m.org/technical/published-specifications/release-2

exchange and integration in smart cities [8, 17]. The design and implementation of these interoperability frameworks are paramount in influencing the overall performance and service quality within smart city applications. The need for a robust interoperability layer becomes increasingly evident as the IoT device network within these ecosystems expands rapidly. Such a layer is indispensable, ensuring the smooth operation of smart city applications by enabling diverse IoT devices and systems to interact without barriers, thereby bolstering the scalability, security, and reliability of the smart city network.

## 3.2 Scalable and Distributed Architectures for Interoperability

The necessity for scalable and distributed architectures in smart cities cannot be overstated. Guth, Jasmin et al. [18] and Seungmyeong et al. [19] compare different IoT platform architectures, emphasizing the limitations of non-standardized ontology, centralized systems, and the traditional three-layer system. Our research advances the architecture proposed by Mante et al. [8], incorporating a distributed and scalable approach that significantly enhances performance for city-wide deployments.

## 3.3 Case Studies and Real-World Deployments

Real-world deployments offer invaluable insights into the practical implementation of IoT architectures in smart cities. Our architecture aligns with existing implementations [20] while incorporating these principles, supporting diverse industry scenarios and enforcing a distributed system. Srdjan et al. [21] presented an architectural reference model (ARM), inspiring our architecture's adoption of a common ontology and promoting semantic interoperability. Mante et al. [8] show our Living Lab architecture introducing a comprehensive structure for smart city applications that includes a Data Monitoring Layer (DML), Data Storage Layer (DSL), and Data Exchange Layer (DEL), demonstrating interoperable data management and sustainable smart city solutions. The implementation of this architecture at Smart City, IIIT Hyderabad, is depicted in Fig 3.1. Our existing implementation utilizes the said architecture with key technologies such as OM2M by Eclipse for DML, an Apache Server alongside Django for the DEL, and Postgres for the DSL, as illustrated in Fig 3.2.

## 3.4 Gaps in Current Research

While existing literature provides a solid foundation for understanding IoT architectures and their application in smart cities, gaps remain, particularly in achieving high scalability and interoperability while adhering to oneM2M standards across diverse IoT platforms. Although we have a robust architecture [8], the architecture's non-distributed nature limited its scalability, an issue our research addresses by proposing a distributed framework that offers enhanced performance and scalability. Our work [22] proposes a multi-layered distributed IoT data platform architecture that significantly improves performance and scalability, which will be further discussed in 4. However, there is scope for further

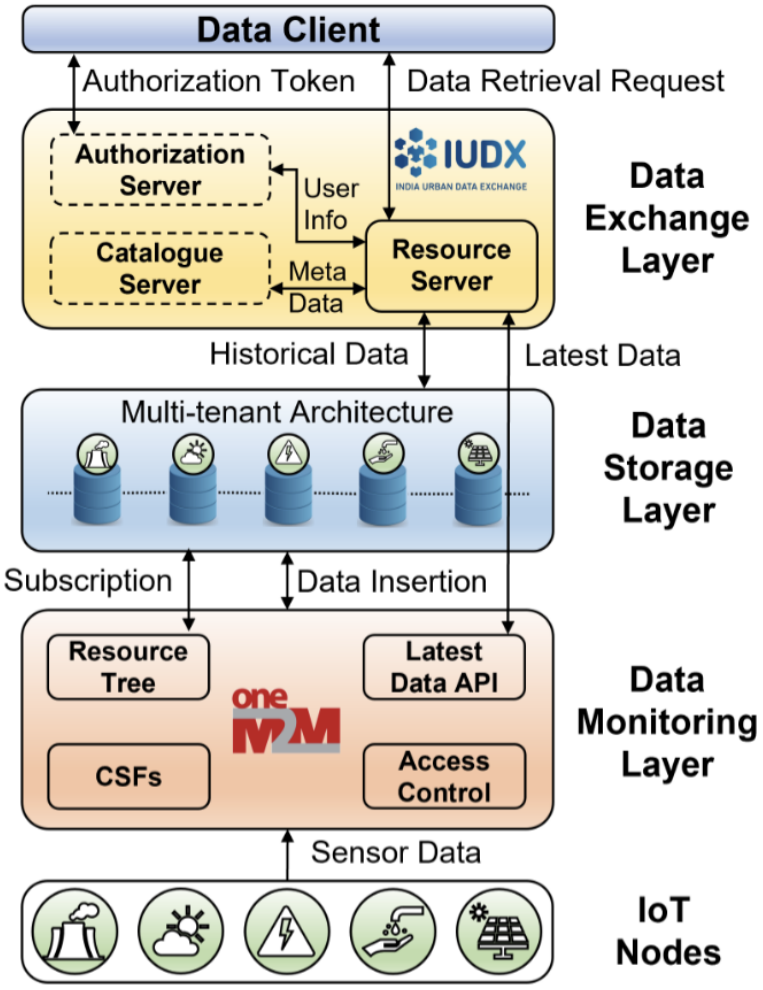



**Figure 3.1** Smart City Architecture at IIIT Hyderabad [8]

enhancements, especially in leveraging oneM2M standard systems for even greater performance gains. We then partake in this research further by doing a performance analysis to improvise the oneM2M performance, which can be seen in Chapter 5.

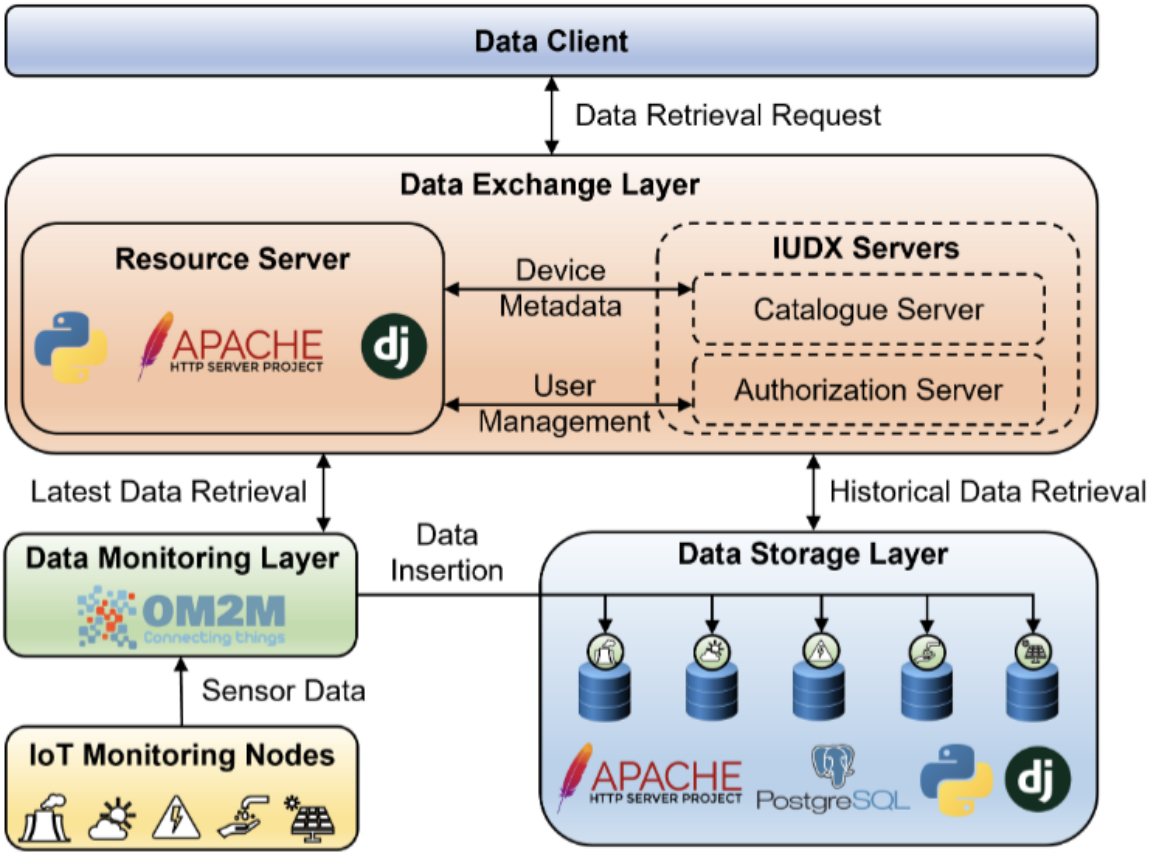



**Figure 3.2** Implementation Technologies at Smart City, IIIT Hyderabad [8]

*Chapter 4*

# Designing a Scalable Architecture for oneM2M

In this chapter, we explore a scalable and interoperable distributed architecture for IoT in Smart Cities, which introduces a multilayered data platform comprising Data Monitoring, Storage, Enhancement, and Exchange Layers. This architecture aims to address the challenges of increased sensor data and user demands by ensuring high throughput, scalability, and cross-platform compatibility. Implemented and evaluated using the Smart City Living Lab at IIIT Hyderabad, the architecture demonstrates significant improvements in throughput and latency, suggesting a promising approach for efficient and standardized IoT frameworks in urban development.

## 4.1 Introduction

The rise in urban populations and the increasing demand for urbanization in smart cities call for effective connectivity and automation solutions. As discussed earlier, interoperability plays a pivotal role in the implementation and interfacing of these solutions, which are key in tackling challenges like energy use, waste management, water quality, and pollution control [23] [24]. Smart cities strive to integrate various sectors with interconnected systems, enabling the real-time monitoring and management of essential services. This approach allows cities to make decisions based on data, improving efficiency, driving innovation, and including citizens in governance processes [25] [26].

Smart cities aim to reduce their carbon footprint [27] and make better use of resources through the use of IoT, data analytics, and connectivity. These technologies cover data collection and storage, control of IoT devices, and access control. However, adopting these technologies in smart cities brings challenges related to security and data management, including issues with collecting data efficiently, data silos that limit system integration, sensor errors, and the lack of standardized details for smooth data exchange among stakeholders [28].

To overcome these smart city challenges, we propose a distributed, multilayered data platform architecture designed for interoperable IoT systems. This architecture is made up of four layers: the Data Monitoring Layer (DML), Data Enhancement Layer (Data Enhancement Layer (DEnL)), Data Storage Layer (DSL), and Data Exchange Layer (DEL). Each layer contributes to effective data processing, storage, and sharing, enhancing the system's performance and ability to scale.

This work adds to the field of smart city interoperability by introducing a new architecture. This architecture consists of four layers: the Data Monitoring Layer (DML), Data Enhancement Layer (DEnL), Data Storage Layer (DSL), and Data Exchange Layer (DEL). Each layer plays a crucial role in enabling efficient data processing, storage, and exchange, improving overall performance and scalability. We implemented this using distributed instances of OM2M as the platform for interoperability and thoroughly examined performance metrics such as insertion, retrieval, and latency times. These evaluations offer insights into how well the architecture performs.

The following sections detail each layer in the proposed architecture (Section 4.2), describe the implementation process (Section 4.3), and discuss using distributed OM2M instances [29] as the interoperability platform. We assess the architecture's performance by looking at key indicators like data insertion and retrieval speeds and latency (Section 4.4). The thesis concludes with a discussion on the significance of our architecture and suggestions for future research (Section 4.6).

This research was conducted on the Smart City Living Lab[1]. By studying this system, we were able to gather valuable real-world data, which allowed us to evaluate the effectiveness of our proposed architecture. The experiments conducted within the Smart City Living Lab demonstrated efficient results and highlighted the potential of our architecture in addressing the complexities of smart city deployments.

## 4.2 Proposed Architecture

Our proposed architecture consists of four key layers: Data Monitoring Layer (DML), Data Enhancement Layer (DEnL), Data Storage Layer (DSL), and Data Exchange Layer (DEL) as illustrated in Fig.4.1. The DML receives data from IoT nodes through a load balancer and forwards it to the distributed oneM2M instances. This layer then forwards data to DEnL using the Common Service Function (CSF) of oneM2M. The DSL comprises a Historical Data Archive for older data and a temporary Data store for recent data, which are populated through the Extract, Transform, Load (Extract, Transform, Load (ETL)) process from the DEnL. The DEL includes an Authentication server, resource server, and Representational State Transfer (REST) Application Programming Interface (API)s for secure data retrieval.

### 4.2.1 Data Monitoring Layer (DML)

The DML acts as the entry point for data in the architecture. It receives data from a cluster of IoT nodes at predefined frequencies, with a load balancer efficiently distributing the incoming traffic among multiple oneM2M framework instances, making it a distributed system. The oneM2M interoperability layer collects and stores data using the built-in Common Service Functions (CSF). This information is stored as a `<contentInstance>` resource within the relevant `<container>` resource. These resources can be controlled collectively through `<group>` resources. Application Entities (`<AE>`)

[1] https://smartcitylivinglab.iiit.ac.in/

represent specific sensor network categories which are the parent of Container resource. Data forwarding to the DSL occurs via `<Subscription>` resources using a publish-subscribe mechanism. Each oneM2M instance internally maintains a database for metadata storage, resource management, and sensor data.

### 4.2.2 Data Enhancement Layer (DEnL)

The DEnL receives data from the DML and performs various data enrichment operations. It is equipped with functionalities to process and transform the incoming data. This layer may involve data cleansing, normalization, aggregation, or other operations to enhance the quality and value of the data. The DEnL layer shields the underlying components and technologies used by encapsulating the logic and operations required for data pre-processing and transformation. This layer eliminates the need for repetitive reformatting during data retrieval, which will be required if this layer doesn't exist. Ultimately, it optimizes the overall data flow and facilitates seamless integration with downstream systems. The DEnL enables the system to handle diverse data sources and adapt to evolving requirements without tightly coupling it to specific technologies or components.

### 4.2.3 Data Storage Layer (DSL)

The DSL is a critical component that is responsible for securely storing and managing data obtained from the DEnL. The DSL adopts a multi-tenant architecture with a focus on efficient and reliable storage. It consists of two main components: a Historical Data Archive and a temporary Data Store. The Historical Data Archive serves as a long-term storage solution for older data, ensuring data retention and compliance with regulatory requirements. On the other hand, the temporary Data Store holds recent data, enabling quick and efficient access for applications. The DEnL forwards data to both the Historical Data Archive and the temporary Data Store in parallel, ensuring redundancy and availability.

### 4.2.4 Data Exchange Layer (DEL)

The DEL acts as a secure interface for data clients to access the stored data within the architecture. It comprises an Authentication server, a resource server, and REST APIs.
The Authentication server handles user registration, Access Control Policy (ACP)s creation/deletion, and token generation for data retrieval, which ensures that only authorized clients can access the data by verifying their credentials. The Resource server hosts the stored data and provides the necessary functionalities for data retrieval. REST APIs are employed to facilitate communication and enable data clients to query the stored data efficiently.

## 4.3 Implementation

This section discusses the technology stack used to implement our proposed architecture. For our experiments, we simulated three verticals to emulate nodes sending requests to the data architecture. The architecture implementation can be seen in Figure 4.2. To handle the data from these nodes and to our data architecture, we deployed three similarly configured systems, each with three instances of Infrastructure Node Common Service Entity (IN-CSE)s. These instances received data from a load balancer, ensuring efficient distribution of data across the system. To enhance data quality, we integrated Thingsboard for data cleaning. For seamless data streaming, we employed the Kafka Engine, which facilitated the transfer of data to Elasticsearch [30] and Postgres. Elasticsearch served as the temporary data archive, while Postgres acted as the historical data archive.

### 4.3.1 Data Monitoring Layer

In the Data Monitoring Layer, we employed a load balancer as an entry point from the sensor nodes. It distributes incoming sensor data and delivers it to distributed OM2M instances, i.e. distributed IN-CSEs. The distributed IN-CSEs are physically apart from each other. In this scenario, we used Round Robin (RR) and the Least Connections (LC) approach separately to input data into the dispersed Open Machine-to-Machine (OM2M) IN-CSEs to evaluate its performance characteristics. As in the prior method, one MongoDB instance is utilized to accomplish the IN-CSE, resulting in data congestion; thus, In the current design, each IN-CSE has its own MongoDB instance as its Database. We use the oneM2M `<Subscription>` resource functionality to transfer data to the Data Enhancement Layer.

### 4.3.2 Data enhancement Layer

The Data Enhancement Layer seamlessly processes incoming data from various sources within the oneM2M framework using Thingsboard. It connects with the onem2m platform to receive real-time data in JavaScript Object Notation (JSON) format. The layer's main function is to extract and parse relevant telemetry information from the JSON payload, specifically focusing on the content information (`<con>`). Once extracted, the enhancement layer assigns appropriate parameters to the data to ensure compatibility with downstream layers and optimize data cleaning/pre-processing. The processed data is then seamlessly forwarded to the Data Storage Layer for transmission and storage. The design and implementation of the enhancement layer offer flexibility and extensibility, encapsulating device functionalities and rule chains to achieve a modular and scalable architecture. Fig 4.3 shows an example rule chain done in Thingsboard. The system's core functionalities include data ingestion, parsing, parameter assignment, and integration with the data storage layer, emphasizing the role of the enhancement layer.

### 4.3.3 Data Storage Layer

We continue the workflow from Thingsboard by sending formatted data to Kafka through a publish/subscribe mechanism. Kafka Connect API [31] is then leveraged to transmit this data to both Elasticsearch and PostgreSQL. Elasticsearch is our temporary data store, while PostgreSQL is our historical data archive. Elasticsearch is a state-of-the-art search store for faster data querying and searching and is one of the best options to use as a temporary database. It is one of the most popularly used data stores for efficient data ingestion and retrieval, as suggested by Vandikas [32] and Thacker [33] in their research.

Thingsboard in the DEnL formats data before storing it in Elasticsearch, eliminating the need for redundant reformatting when retrieving data. For each similar node, a topic will be created in Kafka, a respective table will be created in Postgres, and an index will be created in Elasticsearch. A Kafka connector would be created for each topic to ensure the data is being stored successfully in both data stores. This provides an efficient and reliable data storage layer for our implementation. The code for our implementation is available on GitHub [2]

### 4.3.4 Data Exchange Layer

In our most recent implementation, we included a resource server with a set of services. An authorization service that enables the user to request a token for data access. The data flow within the implementation can be seen in Fig.4.2. The APIs fetch data from the DML using the Resource service that decides which data store to perform the query based on the data requested from the data client. This enables the framework to maintain data freshness when querying live data and also supports archival retrievals. Further, we have leveraged a Python service based on Flask API (Gunicorn as the gateway) to implement the data exchange layer with Nginx as the open-source web server.

## 4.4 Results

We performed experiments to evaluate the effectiveness and efficiency of the proposed architecture. We employed three identical workstations as infrastructure nodes with common service entities (incse). These machines featured an x86_64 architecture, six cores, an Intel i5 Central Processing Unit (CPU), a minimum of 16GB Random Access Memory (RAM), and a 64-bit Ubuntu 18.04 operating system. Throughput and latency were measured to assess the architecture's performance under various workloads, including data insertion and retrieval.

### 4.4.1 Improvements in the Data Monitoring Layer

In this experiment, our primary objective was to replicate real-time data entry and retrieval events in the Interoperability Layer. We conducted performance tests for data insertion and retrieval to the DML

[2]https://github.com/vjspranav/KafkaConnect

over a duration of 3 hours for insertion and 1 hour for retrieval. Throughout the tests, we systematically increased the number of simultaneous requests/users while capturing various parameters [3].

For data retrieval, we tested varying parallel user counts in steps of 25 and requested data points in steps of 200 (ranging from 200 to 600). The results showed that our distributed architecture could effectively handle **95** concurrent users using Round Robin Load balancing and **116** concurrent users using the Least Connection Load balancing, with 0% downtime.

The architecture exhibited a remarkable **222.70%** increase in overall throughput and an average **69.08%** decrease in latency compared to the existing centralized architecture[4] as seen in Table 4.1. Fig 4.4 shows a detailed comparison of performance across different load balancers when compared with the centralized architecture for data retrieval.

In the data-insertion scenario, the proposed distributed architecture with LC load balancer effectively handled **51** concurrent devices, resulting in up to a **41.23%** improvement in throughput and an average **29.19%** decrease in latency compared to the centralized approach as shown in Table 4.2. This improvement was achieved by utilizing the LC load balancer, which dynamically allocates requests based on server load, optimizing resource utilization and employing a distributed system.

**Table 4.1** Comparision of performance across different load balancers for retrieval

| LoadBalancer | Throughput/s | Average Latency (ms) | Median Latency |
|---|---|---|---|
| None | 36.11 | 81329.23 | 8062.00 |
| RR | 95.98 | 30571.24 | 185.00 |
| LC | 116.53 | 25143.10 | 804.50 |

**Table 4.2** Comparision of performance across different load balancers for insertions

| LoadBalancer | Throughput/s | Average Latency (ms) | Median Latency |
|---|---|---|---|
| None | 36.45 | 2364.75 | 5185.50 |
| RR | 51.48 | 1674.28 | 47.00 |
| LC | 49.43 | 1742.75 | 924.00 |

### 4.4.2 Improvements in Data Enhancement and Control Layer

To provide a comparative perspective, we considered a direct alternative approach of utilizing a RESTful API to obtain the data and subsequently push it to the respective databases. In this context, we implemented a fastAPI-based middleware service to directly push the data into PostgreSQL and Elasticsearch. However, a performance analysis revealed that this direct alternative, while functional, exhibited slightly inferior performance when compared to our proposed DEnL.

Our DEnL, which involves data pre-processing and streaming through a publish-subscribe architecture consisting of ThingsBoard and Kafka, showcased notable performance improvements across

[3]Detailed results are available at: https://vjspranav.github.io/om2mreports

[4]This performance evaluation focuses on the om2m retrieval in a distributed setting

various aspects. The results depicted in Table 4.3 affirm the enhanced performance achieved by our implementation while retaining all the additional functionality provided by the rule chain of ThingsBoard. Thus, our chosen approach not only ensures data integrity but also delivers improved performance compared to a conventional Create, Read, Update, Delete (CRUD) method.

**Table 4.3** Comparison of insertion performance through RESTful API vs our DEnL consisting of Data Pre-Processing and Publish Subscribe

| Architecture | Throughput | Total transactions | Avg Latency |
|---|---|---|---|
| RESTful API | 153.55 | 730731 | 147.93 |
| Publish-Subscribe | 176.94 | 842024 | 70.39 |

### 4.4.3 Improvements in Data Exchange Layer

In the original implementation, the Data retrieval by the client happened directly through OM2M, which sends the latest data through REST API. Now that we propose using a temporary Data Store as our source for data, we compare how beneficial retrieval through this temporary Data Store would be when compared to retrieving directly through OM2M. We implement a simple service that will act as the middleware allowing us to query the latest data and do a fair data retrieval comparison.

We compare retrieval through our implementation of this latest data middleware directly with retrieval in OM2M, which allows us to showcase the real-life benefits of our new architecture. Retrieval through our new middleware gives a substantially high throughput. We see an increase of over 800% in both throughput and total number of requests. By utilizing ElasticSearch in our architecture, the query response time is significantly reduced, resulting in the observed substantial improvement. This comparison demonstrates the effectiveness of our architecture in overcoming performance limitations and achieving enhanced retrieval capabilities. The results can be seen in Table 4.4.

**Table 4.4** Comparison of retrieval performance from om2m to the implemented middleware for the latest data

| Service | Throughput/s | Total transactions |
|---|---|---|
| OM2M | 36.11 | 375556 |
| Proposed middleware | 332.36 | 3456209 |

## 4.5 Discussion

### 4.5.1 Lessons Learned

The results presented in this paper demonstrate the enhanced performance and scalability achieved by our distributed architecture, as shown in the performance comparison tables and graphs (Section 4.4). The specific technology stack utilized was chosen to ensure a fair comparison with an existing

non-distributed architecture that faced challenges when the number of users increased [8]. The aim was to address these issues and provide a more scalable solution. The architecture showcased in this paper exhibits notable improvements in handling increased user load, as evidenced by the results obtained in insertion and retrieval operations as shown in Section 4.4.1. The findings highlight the advantages of the proposed architecture in terms of improved performance and reduced load, even under conditions of increased user activity. We believe that this architecture can be improvised further by moving it into a micro-services-based architecture, ensuring we have very few dependencies across multiple services provided by the oneM2M standard.

### 4.5.2 Threats to Validity

Threats to *internal validity* concern the selection of workload parameters and measuring performance metrics. To this end, real workloads of the system have been used. Further, the same system configuration was used for the different experiments to prevent bias arising from any background processes. Threats to *external validity* concern the generalizability of the study across different IoT systems. Although our approach has been applied to a specific smart city IoT system, it uses techniques that can be generalized to more complex systems. Moreover, while this experiment provides valuable insights into the performance metrics of different architectures, the system configurations may impact them. To this end, we ensured the same base configuration was used across different experiments to ensure consistency.

## 4.6 Summary

This study describes an improved architecture for a distributed and scalable system based on the oneM2M smart city standards. We proposed a multilayered architecture consisting of a Data Monitoring Layer (DML) which is distributed and oneM2M compliant, a Data Storage Layer (DSL) consisting of temporary and historical data archives, and a Data Enhancement Layer (DEnL) which formats data before storage and the Data Exchange Layer (DEL ) which serves as a robust interface where the data clients can securely access the data. We have conducted the performance analysis and provided context on the increase in scalability and load-handling capability in the system.

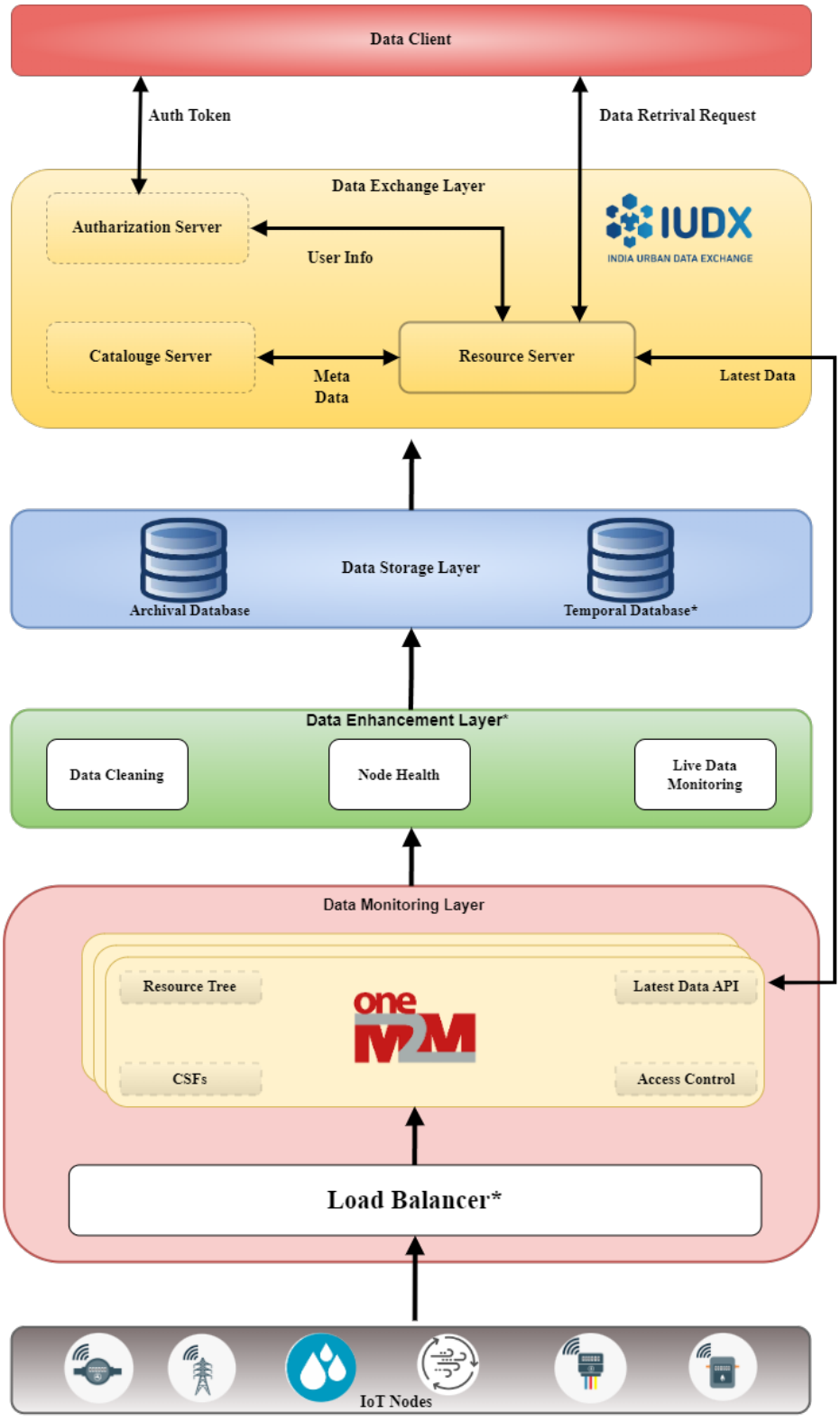



**Figure 4.1** Proposed architecture of the Distributed OM2M

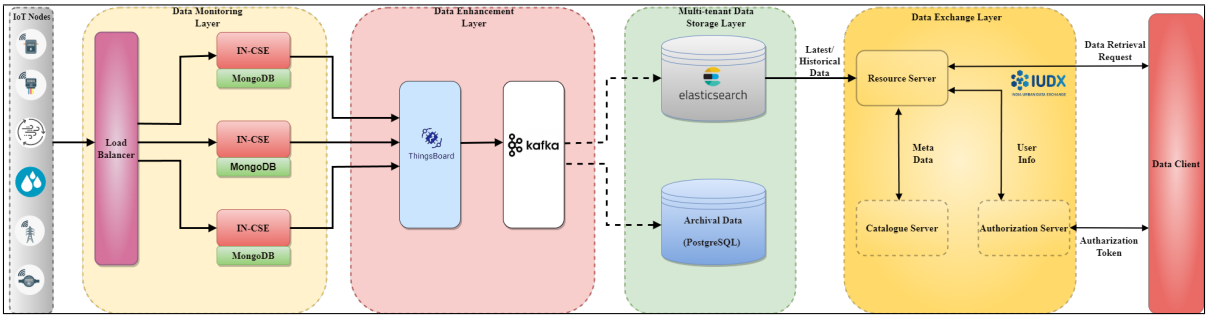



**Figure 4.2** Implementation of the Distributed OM2M

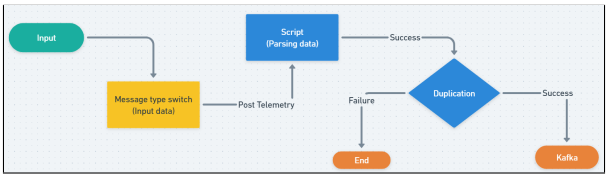



**Figure 4.3** Thingsboard Rule Chain

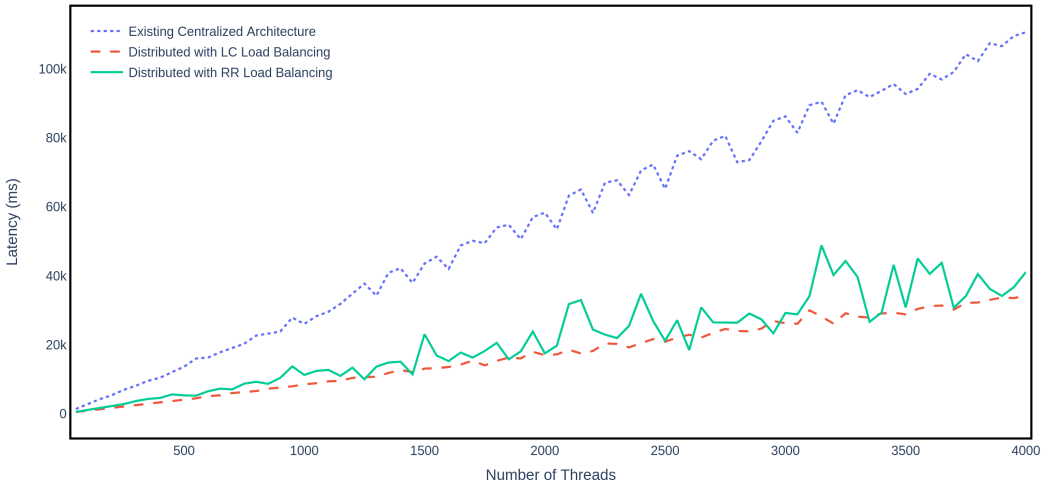



**Figure 4.4** OM2M retrieval performance characteristic

*Chapter 5*

# Performance Analysis of Multiple oneM2M Systems

In this chapter, we present an extensive evaluation of three prominent oneM2M based IoT interoperability platforms: OM2M, ACME, and Mobius, focusing on their performance in terms of latency, throughput, and resource utilization. Through a series of systematic experiments conducted within a Smart City Living Lab environment, we identify Mobius as the superior platform, consistently outperforming OM2M and ACME across all tested metrics. This comprehensive analysis not only underscores the importance of architectural design and concurrency management in IoT middleware but also offers valuable insights for developers and researchers aiming to optimize IoT systems for smart city applications.

## 5.1 Introduction

For the full potential of IoT to be realized, interoperability is crucial. It ensures seamless communication between diverse devices and systems, underpinning efficient data exchange and integration. Among multiple interoperability frameworks, such as Amazon Web Services (AWS) IoT Core [14] and Azure IoT [15], oneM2M[1] has emerged as a global standards initiative providing a comprehensive IoT interoperability standard. Within oneM2M standards, several interoperability architectures have emerged as potential solutions to facilitate data exchange and integration in smart cities [17]. The design and implementation of these interoperability frameworks play a crucial role in determining the overall performance and quality of services in smart city applications. The rapid growth of IoT devices within these smart urban ecosystems underscores the necessity for a robust interoperability layer.

Popular interoperability solutions for smart cities include Mobius [34], OM2M [29], and ACME [35]. Each of these solutions, notably open-source, differ in their technology stacks and approaches. We focus on open-source options for transparency and accessibility, enabling stakeholders to make informed decisions when selecting an interoperability solution for their smart city deployment. Real-world deployments are considered key for evaluation. We again rely on Smart City Living Lab at Smart City Research Center (SCRC), IIITH to test these solutions and explore their architectures thoroughly and gain practical insights.

[1] https://onem2m.org/technical/published-specifications/release-2

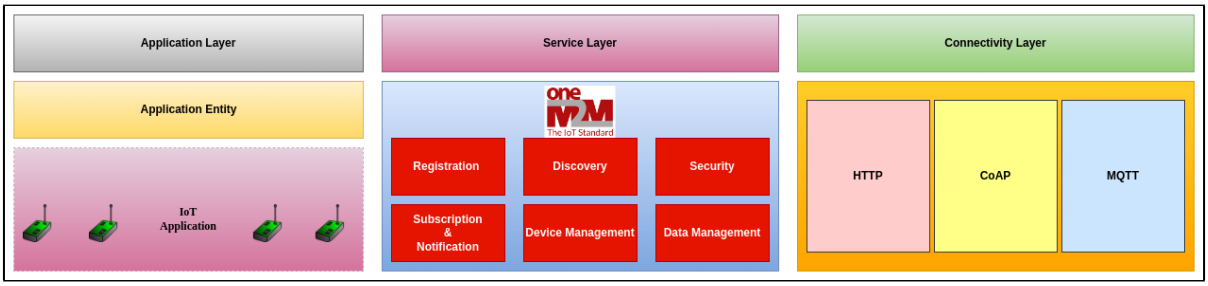



**Figure 5.1** OneM2M Architecture

This work is the first to explore state-of-the-art oneM2M-based interoperability solutions. Our evaluation shows Mobius consistently outperforming OM2M and ACME. In stress tests, Mobius scales exceptionally well and maintains low latency. This study aims to guide software architects to make informed choices for robust interoperability in smart cities.

## 5.2 Motivation

Our large-scale Living Lab provides a real-world testing ground for smart city solutions. As mentioned earlier, We rely on OM2M by Eclipse[2] as the oneM2M interoperability layer within our deployment. As our live deployment expanded, we observed the potential for a bottleneck developing due to the growing number of nodes incorporated into the system. This potential bottleneck motivates our research into the performance of different oneM2M-based interoperability architectures within large-scale IoT settings. Addressing this challenge became imperative, prompting us to explore potential solutions. We identified two primary avenues for improvement:

The first approach entails devising a distributed architecture that can efficiently handle the increasing number of nodes as covered in Chapter 4. The second avenue involves conducting a comprehensive comparison of existing interoperability solutions. We evaluate if specific architectural choices within interoperability frameworks significantly impact performance, allowing the selection of the best-suited solution for large-scale use. By scrutinizing prominent solutions, we aim to ascertain whether any of these solutions inherently provide a significant performance boost. Such solutions, when combined with a well-designed distributed architecture, have the potential to yield even more remarkable results.

This research goes beyond performance comparisons. We aim to understand if and how fundamental architectural differences of interoperability solutions influence their ability to handle vast IoT networks. Our insights will aid smart city stakeholders, in particular architects, in making informed choices for building scalable, efficient, and interoperable systems.

[2]https://www.eclipse.org

## 5.3 Study Design and Execution

In this section, we provide a comprehensive overview of the purpose and structure of this chapter, breaking down our research into three fundamental components: the experimental design, implementation, and results. To ensure a systematic and structured approach to our investigation, we adopt the Goal Question Metric Approach (GQMA), as proposed by Basili et al. in their seminal work [36]. This approach serves as the guiding framework for our research, allowing us to define the goals precisely, formulate research questions, and establish performance metrics.

### 5.3.1 Goals

The primary objective of this study is to assess and compare multiple interoperability architectures based on the oneM2M standards, shedding light on their key characteristics, performance metrics, scalability, and adaptability. These objectives can be defined as follows (following the GQMA):
***Analyze*** the architecture of interoperability systems for IoT
***For the purpose of*** finding out their impact on performance
***By performing*** multiple intensive experiments
***From viewpoint of*** software architects and researchers
***In context of*** Smart Cities

### 5.3.2 Research Questions

To achieve this goal, we formulate the following research questions:
**Research Question 1 (RQ1):** *What are the key characteristics and components of each interoperability architecture based on OneM2M standards?*
This question aims to provide an overview of the architectures under consideration, elucidating their fundamental features and functionalities.
**Research Question 2 (RQ2):** *How do different interoperability frameworks compare with respect to metrics such as latency, throughput and resource utilization?*
By investigating this question, we seek to evaluate the performance of these architectures across various scenarios, highlighting their strengths and limitations.
**Research Question 3 (RQ3):** *What are the scalability and adaptability aspects of each architecture to accommodate large-scale IoT deployments?*
This question delves into the scalability and adaptability of each architecture, addressing their ability to meet the demands of extensive IoT deployments within smart cities.

### 5.3.3 Experiment Design

We employed a systematic approach to conduct a thorough comparative analysis of the three interoperability systems (Mobius, OM2M, and ACME). This section outlines our methods for this analysis,

elucidating the data collection process, the evaluation criteria chosen, and the tools and frameworks utilized. To ensure a consistent testing environment and maintain uniformity across the systems, we set up all three systems using Docker. The host system featured an $x86\_64$ architecture with an Intel i5 12500 processor, 16 GB RAM, and Ubuntu 20.04. This hardware configuration choice provides a stable foundation for our comparative analysis. Additionally, the host system was connected via Ethernet to minimize network-related variations. For reference and transparency, we have made the Dockerized system specifications open source[3]. This step was crucial to precisely defining the system specifications for all three systems, ensuring that each operated within an equivalent technological context. Three important performance metrics—*latency, concurrent users, and requests per second*—were given priority in our research. IoT system efficiency and responsiveness can be evaluated using these measures. Real-time applications require low latency, and scalability is ensured by evaluating the system's ability to support numerous simultaneous users. Additionally, monitoring the number of incoming requests per second offers crucial information about how effectively the system processes data. We seek to ensure that IoT applications and systems satisfy the requirements of various use cases and user expectations by concentrating on these parameters, ultimately maximizing their performance and user experience.

For data collection, we needed to select a load testing tool that would allow us granular control and less overhead; we looked at the performance analysis performed by Pradeep et al. [37]. We leveraged Locust [38], a comprehensive testing framework renowned for its versatility and per-user control capabilities. This tool allowed us to configure various testing patterns to assess the performance of the three systems thoroughly. Our evaluation encompassed two distinct scenarios:

**1. Synthetic Workload:** In this scenario, we populated each of the OneM2M systems with three verticals (Application Entities - AEs) containing 150 nodes each, simulating a high-load testing environment. We conducted three types of tests—High Density Periodic, Sporadic, and a Poisson Distribution—for data retrieval (GET) and data population (POST). This combination resulted in 6 different variations of tests for each system, totalling 18 tests. Each test lasted 2 hours, and we collectively refer to these as "Pattern tests."

**2. Real Workload:** Here, we reconfigured the systems to mirror our production setup, comprising eight verticals and 370 nodes. Notably, real data from deployed nodes was used in these tests to replicate the conditions of a genuine production system. We conducted three types of tests in this scenario. First, the "Pattern" tests, as described earlier. Second, "Stress" tests where we incrementally increased the number of parallel users until either system crashes or performance degradation occurs, with one test each for GET and POST operations, resulting in 6 additional tests. Finally, "Emulation" tests, where we precisely simulated the data frequency generated by actual deployed nodes in each vertical over a two-hour duration. This test provided insights into how each system would perform in a production environment.

**Pattern Tests** encompasses three distinct test scenarios designed to evaluate the system's performance under varying conditions. These scenarios include:

[3]https://github.com/vjspranav/om2m-comparison

1. High-Density Periodic: In this scenario, a periodic surge of users simultaneously accesses the system. This test helps assess the system's response to sudden spikes in user activity.

2. Step Function: The "Step Function" scenario involves a gradual increase in the number of users, resembling a step-like pattern. This test enables us to gauge the system's sustained performance and stability as it accommodates incremental user loads.

3. Poisson: In this scenario, users post actions at random intervals ranging from 1 to 15 seconds. This test provides insights into the system's performance under unpredictable and sporadic conditions.

These pattern tests offer a comprehensive assessment of the system's responsiveness and reliability across various usage patterns and help understand its behaviour in real-world scenarios. In total, we conducted 48 comprehensive tests, all executed over Ethernet connections. In subsequent sections, we will present and analyze the results of these tests in conjunction with our examination of the architectural differences among the systems.

### 5.3.4 Experiment Candidates

The oneM2M architecture partitions IoT functionalities into three primary domains: the application layer, which emphasizes device-to-application connectivity; the services layer, a horizontal framework serving diverse industry applications with common IoT functions; and the connectivity layer, which is responsible for IoT device and endpoint communication. Fig 5.1 shows the overarching view of the layers. These layers collaborate to offer a standardized interface for application management and interaction, fostering collaborative intelligence in distributed IoT systems. All three systems we are about to discuss adhere to an architecture that aligns with the standards set forth by oneM2M. Our examination explores potential architectural distinctions beyond the variance in the employed technology stack as we strive to address the essence of **RQ1**.

#### 5.3.4.1 OM2M Architecture

The OM2M project proposes a modular architecture running on top of an Open Service Gateway Initiative (OSGi) layer [39], making it highly extensible via plugins. Fig 5.2 shows an overview of the OM2M architecture. OM2M is a project written in Java and owned by the Eclipse Foundation. It is built as an eclipse project using maven [40] and tycho [41]. It offers a flexible Service Capability Layer (SCL) that can be deployed in an M2M network, a gateway, or a device. The SCL comprises small, tightly coupled plugins, each offering specific functionalities. Plugins can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot. They can also detect the addition or removal of services via the service registry and adapt accordingly, facilitating the extension of the SCL. It enables the binding of multiple communication protocols, allowing interoperability with different devices and

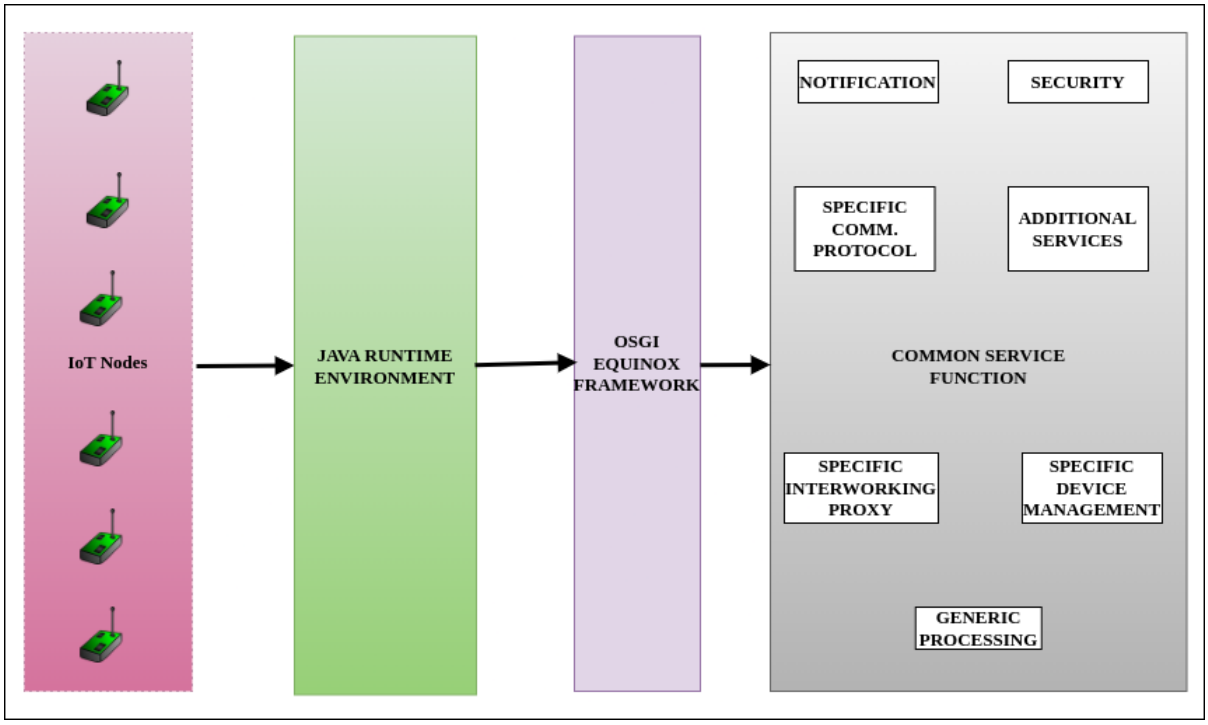



**Figure 5.2** OM2M Architecture

systems. The currently specified protocols are Hyper Text Transfer Protocol (HTTP), Constrained Application Protocol (CoAP), Message Queuing Telemetry Transport (MQTT), and WebSocket, and the specified serialization formats are Extensible Markup Language (XML), JSON, and Concise Binary Object Representation (CBOR) (Concise Binary Object Representation). OM2M embraces a RESTful methodology featuring open interfaces that facilitate the development of services and applications, irrespective of the underlying network. The architecture also includes an autonomic computing service called Autonomic Computing Service (ACS) for self-configuration of M2M communications. ACS operates as an expert system, emulating human decision-making abilities to solve complex problems by reasoning about knowledge. It enables dynamic discovery and self-configuration of M2M applications based on device profiles and application roles. We use version V1.4.1, which is the latest version at the time of writing the paper.

#### 5.3.4.2 ACME Architecture

ACME is a lightweight implementation primarily for educational and experimental purposes. It leverages Flask, a micro web framework in Python, and TinyDB, a NoSQL database, to create a modular architecture that offers flexibility and extensibility, as shown in fig 5.3. ACME covers a wide range of oneM2M service features, including AE registration, blocking and non-blocking requests, notifications, resource addressing, semantic queries, and time synchronization. This comprehensive feature

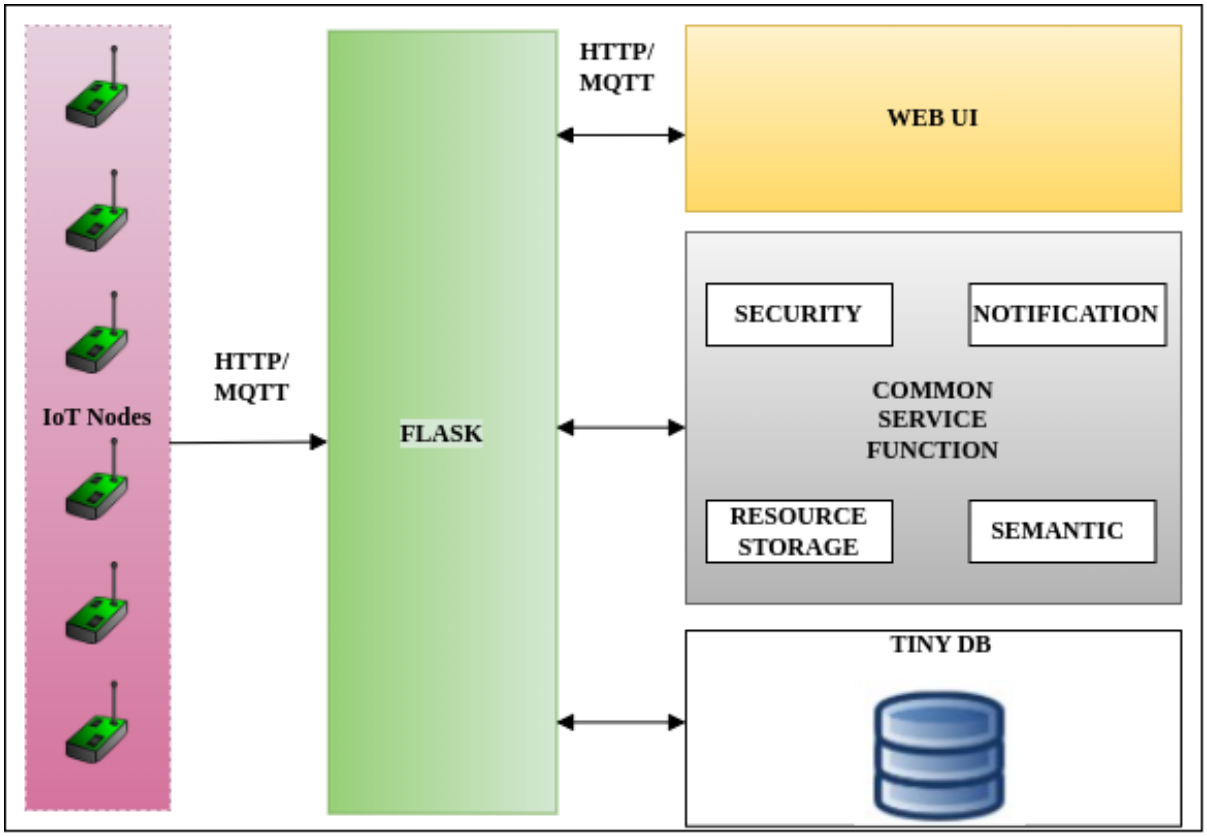



**Figure 5.3** ACME Architecture

set ensures efficient communication and resource management. It supports different protocol bindings, including HTTP (Transport Layer Security (TLS) and Cross-origin resource sharing (CORS)), MQTT (MQTT-S), and WebSocket. This versatility allows ACME to communicate with a variety of devices and systems. ACME supports various serialization types, including JSON and CBOR, enabling data exchange in different formats. ACME can run on different runtime environments, such as Generic Linux, Mac OS, MS Windows, and Jupyter Notebooks. This adaptability enhances its deployment flexibility. ACME primarily serves as an educational tool for learning IoT and M2M communication concepts, which is valuable for students, researchers, and developers. They can use ACME to experiment with IoT and M2M scenarios, test plugins, and explore resource management techniques in a controlled environment. Hence, for the very same reason, ACME comes with the following limitations:

**1. Educational Focus:** ACME is primarily designed for educational and experimental purposes, lacking production-grade requirements.

**2. Security Considerations:** While suitable for learning and experimentation, ACME may not incorporate advanced security features required for secure IoT applications.

**3. Limited Protocol Support:** ACME's protocol support is limited to what Flask offers and does not cover all communication protocols used in IoT and M2M scenarios.

ACME's architecture is designed to provide a lightweight, flexible, and extensible framework for IoT resource management and experimentation. It is best suited for educational and experimental purposes and may not meet production-grade IoT deployments' security and performance requirements.

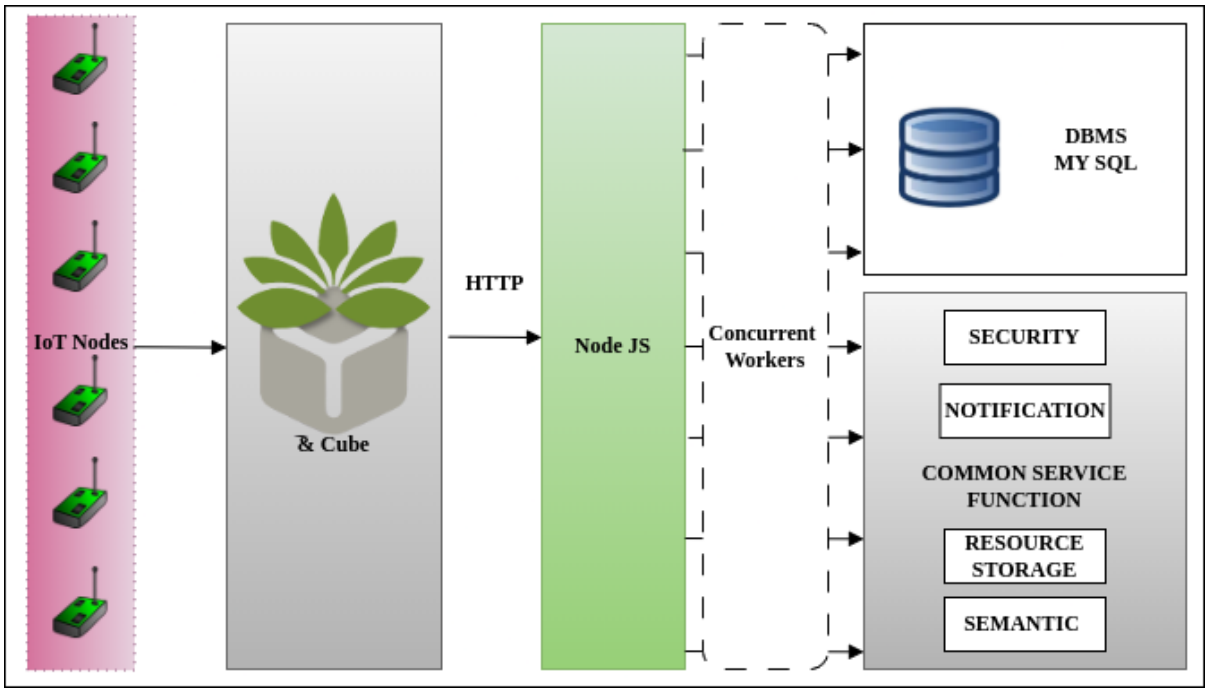



**Figure 5.4** Mobius Architecture

#### 5.3.4.3 Mobius Architecture

Mobius is a server software platform designed to manage and store data from IoT devices. It is developed using JavaScript language based on Node JS [42], which is a high-performance network server that supports asynchronous I/O. Mobius supports MySQL as the database and HTTP, MQTT, CoAP, and WebSocket as communication protocols. The software architecture of Mobius is based on the concept of functional components, which allows for easy customization and scalability. The components include the Mobius server, which manages the data from IoT devices, and the Mobius API server, which provides a RESTful API for accessing the data. Mobius also includes a web-based management console for configuring and monitoring the system. Something that mobius handles differently is that it spawns itself into multiple clusters, parallelly processing different requests, as seen in fig 5.4, which allows for a much higher concurrent performance boost. With its flexible architecture and support for multiple communication protocols, Mobius is a powerful platform for managing IoT data. One difference comes with the introduction of &Cube [43], which acts as a middleware between nodes and interoperability; &Cube, together with Mobius, forms the oneM2M software system.

## 5.4 Results

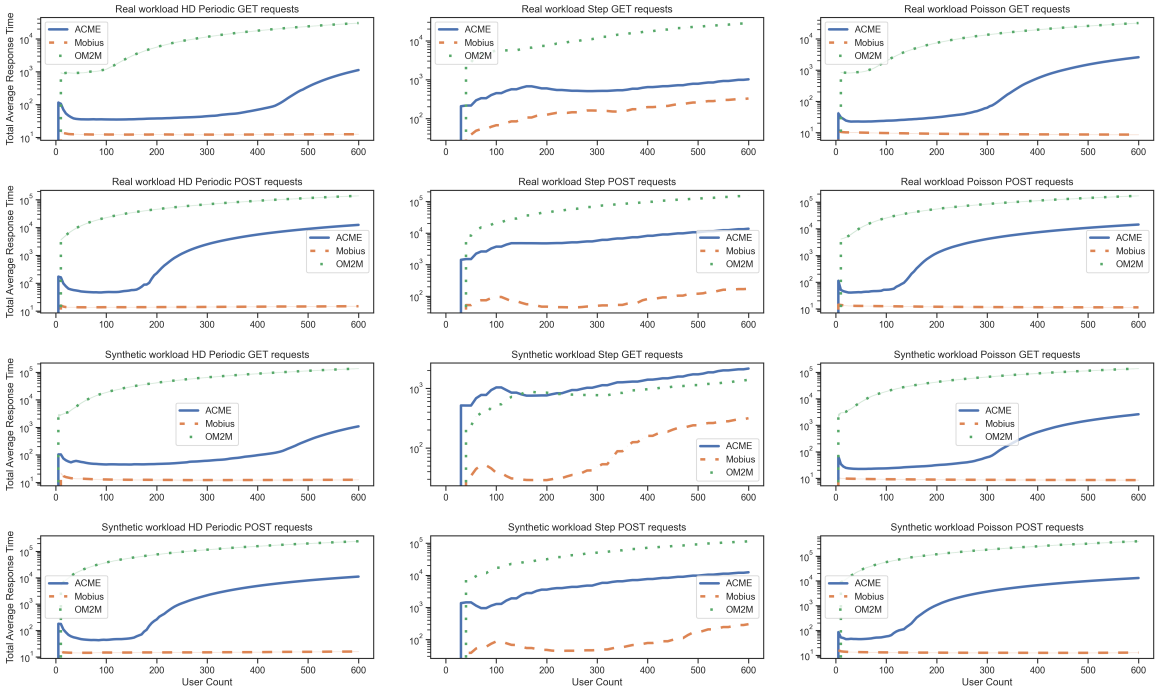



**Figure 5.5** Different request pattern comparison

This section presents the results of our comprehensive evaluation comparing the three systems: OM2M, Mobius, and ACME. The evaluation encompasses various aspects of performance and efficiency, as detailed below. We applied a logarithmic scale to the graphs to facilitate a comprehensive comparison of response times that spanned a wide range. Therefore, even seemingly small differences on the graphs can represent significant variations in actual response times. All the results are available online and can be directly accessed[4].

### 5.4.1 Synthetic Workload

#### 5.4.1.1 Response Time and Latency

In fig 5.5, we present a plot comparing the response times of the three middleware platforms as we increase user load, contributing to the evaluation of latency as part of RQ2. We conducted these comparisons for both get and post requests, resulting in a total of six graphs. These visualizations show Mobius consistently showcasing the lowest response times, followed by ACME and OM2M. One small outlier to note is the worse performance of ACME in the POST request step scenario where response

[4]https://vjspranav.github.io/om2m-comparison/

time crosses that of OM2M, deviating from all the other results we have. We attribute this to other variables and believe that ACME itself does not lead to an edge case behaviour in the given situation, given its performance across all the other scenarios.

#### 5.4.1.2 CPU Stats

Fig. 5.7 explores CPU usage trends in relation to the number of concurrent users. While our testing focused on a synthetic workload, the findings will likely generalize to real-world scenarios. Our investigation addressed resource utilization (RQ2), incorporating three distinct usage patterns with both GET and POST requests for each middleware system. Mobius consistently demonstrated superior efficiency, exhibiting lower CPU usage than ACME and OM2M across all patterns.

### 5.4.2 Real Workload

#### 5.4.2.1 Response Time and Latency

Similar to the synthetic workload, we see Mobius consistently performing better, followed by ACME and OM2M. These results can be seen in fig 5.5

#### 5.4.2.2 Emulation of Real System

Fig 5.6 illustrates the response time as a function of the user count in an emulation of real-world

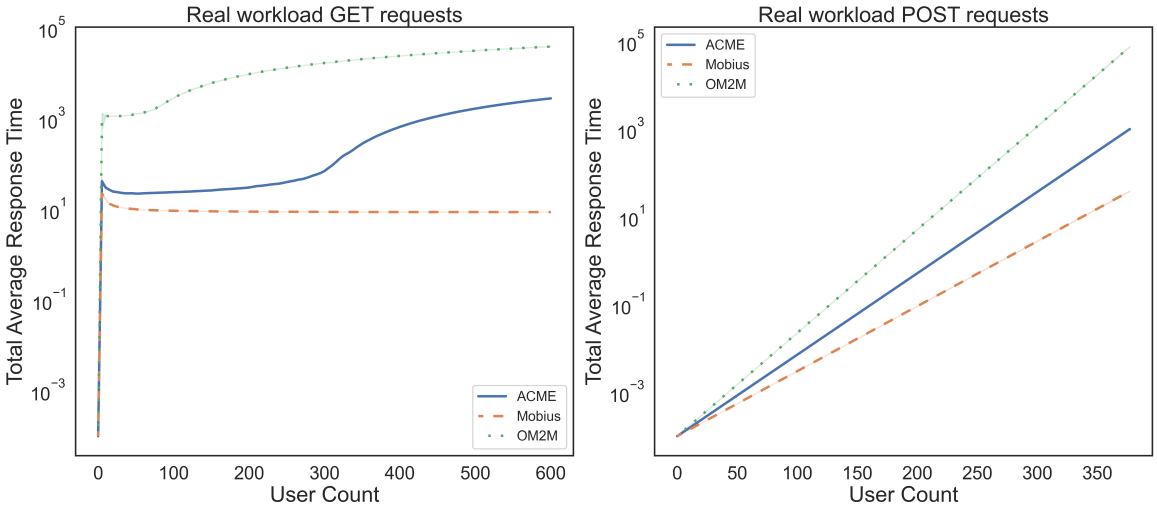



**Figure 5.6** Emulation of the real workload

IoT workload conditions. We conducted post requests at a frequency equivalent to real-life nodes, using actual production deployment replicas. Notably, across different user counts, Mobius consistently demonstrated lower response times compared to both ACME and OM2M, addressing the latency aspect of RQ2.

#### 5.4.2.3 Stress Test

| System | Peak RPS | 95th Avg Latency(ms) |
|---|---|---|
| OM2M | 1 | 432000 |
| ACME | 14 | 95000 |
| Mobius | 568 | 1600 |

**Table 5.1** Stress Testing POST Requests on Real Workload

| System | Peak RPS | 95th Avg Latency(ms) |
|---|---|---|
| OM2M | 3 | 207000 |
| ACME | 43 | 42000 |
| Mobius | 124 | 12 |

**Table 5.2** Stress Testing GET Requests on Real Workload

In addition to our comprehensive evaluation of latency, throughput, and resource utilization, we conducted stress testing to assess the robustness and scalability of the three IoT middleware platforms: OM2M, ACME, and Mobius. Stress testing involved incrementally increasing the number of concurrent users to determine the maximum requests per second (RPS) each system could handle without crashing. We also measured the 95th percentile average latency (95th average latency) at these peak RPS values. The results, presented in Tables 5.1 and 5.2. In the stress testing of POST requests, Mobius demonstrated exceptional scalability, with a peak RPS of 568, representing a remarkable 56,700% increase in RPS compared to OM2M and a 3,942% increase compared to ACME. Additionally, Mobius achieved an impressively low 95th average latency of 1600 ms under this extreme load, highlighting its efficiency even in high-stress situations. Similarly, in the stress testing of GET requests, Mobius achieved a peak RPS of 124, showing a significant 4,033% increase compared to OM2M and a 188% increase compared to ACME. Furthermore, Mobius achieved an exceptionally low 95th Avg Latency of 12 ms.
This extensive evaluation consistently positions Mobius as the leading middleware interoperability platform across various performance metrics, including latency and resource utilization. Its efficiency, especially in terms of response time and CPU usage, underscores Mobius as a robust and high-performing interoperable middleware solution for IoT deployments, effectively addressing the metrics associated with **RQ2** and highlighting its suitability for a wide range of IoT use cases.

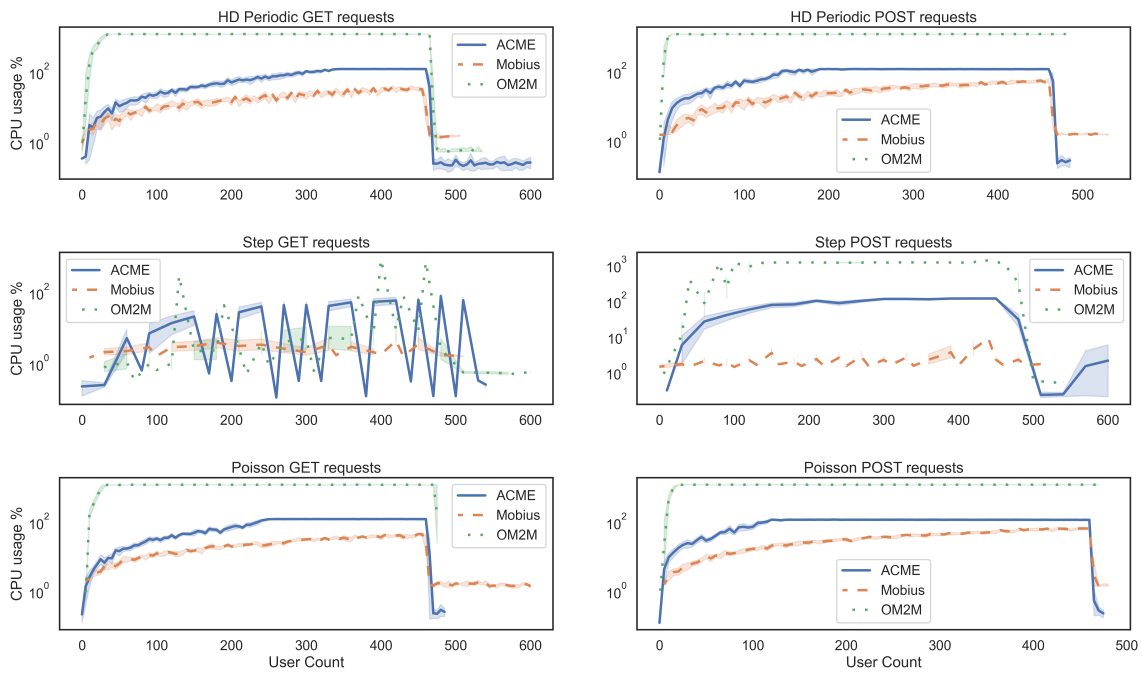



**Figure 5.7** CPU usage statistics with respect to number of concurrent users

## 5.5 Discussion

In this section, we will address the three research questions that guided our evaluation of IoT middleware platforms: RQ1, RQ2, and RQ3. These questions delve into the key characteristics, performance metrics, and scalability aspects of the interoperability architectures. In our comprehensive evaluation of these platforms, Mobius consistently emerged as the clear winner across all performance metrics. Its superior responsiveness, scalability, and efficiency underscore its potential as a robust middleware solution for IoT applications. We take an in-depth look at the implementational differences between OM2M, ACME, and Mobius and how these differences may influence the software's performance.

**What are the key characteristics and components of each interoperability architecture based on OneM2M standards? (RQ1)**

From our study of the architectures of the systems in Section 5.3.4, we delve into the distinctive characteristics and components of interoperability architectures based on the OneM2M standards. By examining the architectural details of these systems, we gain valuable insights into their key attributes and functionalities.

**OM2M**: The OM2M architecture is characterized by its exceptional modularity and extensibility, allowing the seamless integration of specialized functionalities through plugins without requiring system-wide reboots. In the context of our research, OM2M's codebase exhibits a high degree of modularity and extensibility, following good design patterns [44] and aligning with established software engineer-

ing principles. This architectural attribute aligns with the concept of a ”modular monolith” [45] within the realm of Software Engineering, highlighting its structural soundness and versatility.

**ACME**: ACME, designed primarily for educational and experimental purposes, adopts a lightweight and modular approach. It utilizes Flask and TinyDB to offer flexibility and extensibility. The codebase itself was quite straightforward, leveraging the power Python provides.

**Mobius**: The Mobius architecture integrates with &Cube, enhancing its capabilities as part of a comprehensive oneM2M software system. It is worth noting, however, that the Mobius codebase exhibits a certain level of complexity and limited extensibility. The presence of substantial hardcoding within the code, while not inherently erroneous, deviates from established design patterns and best coding practices.

A key insight from this study is that well-structured, best-practice code does not always translate to optimal performance. It suggests that OM2M's architectural complexity may be excessive for specific tasks, highlighting the balance between code quality and performance.

**How do different interoperability frameworks compare with respect to metrics such as latency, throughput and resource utilization? (RQ2)**

Mobius exhibits a clear and substantial performance advantage within comparative interoperability framework benchmarks as can be seen fin Fig. 5.4. In the handling of POST requests, Mobius achieves a peak rate of 568 requests per second, representing a 56,700% improvement over OM2M. This exceptional throughput, coupled with a 95th percentile average latency of 1600ms for data insertion during stress testing, demonstrates the framework's robust scalability. Mobius consistently maintains low latency for data retrieval (under 12ms), ensuring rapid information access. These empirical results underscore Mobius's leadership in terms of latency, throughput, and resource utilization. Such performance characteristics are indispensable within the context of IoT applications, where scalability and real-time responsiveness are crucial.

**What are the scalability and adaptability aspects of each architecture to accommodate large-scale IoT deployments? (RQ3)**

RQ3 delves into the scalability and adaptability of each interoperability framework to effectively accommodate large-scale IoT deployments. OM2M, despite its support for threading, exhibits a substantial CPU usage spike even under relatively light loads, suggesting the presence of potential bottlenecks. In contrast, ACME showcases well-structured code and leverages Python's threading capabilities, but it falls short due to incomplete adherence to the oneM2M specification. Moreover, the use of tinyDB [46] appears to contribute to its relatively better performance compared to OM2M. Mobius stands out with its highly efficient concurrency control mechanism, employing multiple replicas, precisely equivalent to the number of CPU cores, with fault tolerance. This parallel execution approach provides a substantial advantage over the other two systems, particularly in scenarios involving extensive concurrency.

### 5.5.1 Threats to Validity

While our study provides valuable insights, it is essential to acknowledge potential validity considerations. Internally, we conducted our evaluation under specific test conditions and workloads to ensure robustness. However, it is worth noting that these conditions may only partially encapsulate the richness of all real-world scenarios. Variations in workloads and use cases could yield diverse outcomes. We rigorously managed system configurations throughout our experiments to maintain consistency and minimize biases. Externally, we aimed to generalize our findings across different IoT systems, yet the chosen metrics may not cover every possible performance aspect. Additionally, our study primarily focused on performance metrics, with limited exploration of security and fault tolerance aspects. Furthermore, we conducted a fair comparison, but it is important to recognize that inherent differences in the platforms themselves may have influenced our results.

## 5.6 Summary

In this study, we extensively evaluated three prominent IoT middleware platforms, OM2M, ACME, and Mobius, with a focus on metrics such as latency, throughput, and resource utilization. Key findings from our study highlight the importance of careful consideration in architectural design. While modular code adhering to good coding standards is essential for maintainability and codebase organization, our results demonstrate that it does not always guarantee superior performance. Proper parallelism and concurrency management emerge as crucial factors, as evidenced by Mobius consistently outperforming its counterparts.

These findings hold significant implications for IoT developers and researchers. They highlight the need to balance modularity with performance considerations in architectural design. Developers should carefully assess the scalability and concurrency handling capabilities of middleware platforms, taking into account the specific requirements of their applications. Researchers can use these insights to refine existing IoT architectures and design more efficient systems.

*Chapter 6*

# Real World Deployment

## 6.1 Introduction

This chapter introduces the City IoT Operating Platform (ctOP) platform, a major contribution to interoperability and as an IoT platform for large-scale deployments such as actual cities. The ctOP platform, meticulously engineered to serve as a lightweight oneM2M wrapper, is an innovation designed to enhance interoperability and streamline IoT device integration within smart city deployments. Its development is not merely a technical feat but a strategic endeavour to bridge the gap between the complexity of IoT ecosystems, especially in the realm of oneM2M standards, and the pragmatic needs of urban infrastructures. Through the lens of the ctOP platform, this chapter utilizes the research impacts of concepts explored in earlier chapters, translating them into actionable insights and scalable solutions for smart cities worldwide.

## 6.2 Motivation and Design Philosophy

The development of the ctOP platform is fundamentally motivated by the imperative need for standardization in large-scale IoT deployments, particularly within smart city infrastructures. The Government of India's impending regulation to adopt oneM2M as the interoperability standard underscores the criticality of this initiative. Systems not adhering to or integrating with this standard risk exclusion from certification as part of the IoT ecosystem, emphasizing the urgency for compliant solutions. The Smart City Living Lab, through its endeavours, seeks to mirror a microcosm of a smart city, testing and validating various solutions in a controlled yet scalable environment. The lab's commitment to adhering to the oneM2M standard from the very beginning shows a strategic choice aimed at fostering a standardized, interoperable smart city framework. This choice has allowed us to propose a data standard while being compliant.

While the oneM2M standard offers a robust framework for interoperability in smart city deployments, its adoption has been hindered by the complexity and lack of user-friendliness of existing solutions. Open-source platforms are scarce, and those that exist often require extensive technical knowledge to use. This presents a significant barrier for city planners and IoT developers who seek to effectively

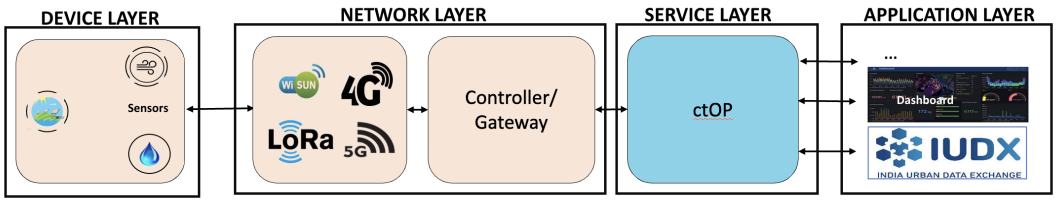



**Figure 6.1** The ctOP architecture

deploy and manage smart city technologies. This gap in the ecosystem is particularly critical in the context of India's Smart City Mission, where diverse stakeholders need to collaborate and contribute to the development of efficient and sustainable urban environments. Recognizing this gap, a strategic partnership was formed between the Smart City Mission (SCM), Ministry of Housing and Urban Affairs (MoHUA), IUDX, and our team, leading to the conceptualization of ctOP.

The ctOP platform directly addresses this challenge by providing an accessible and user-centric solution that encapsulates the oneM2M standard within an easy-to-use interface. By lowering the barrier to entry for smart city projects and enhancing interoperability across different IoT devices and systems, ctOP aims to accelerate the development and adoption of smart city technologies, ultimately contributing to improved urban living.

## 6.3 ctOP Architecture

The ctOP platform is strategically positioned within the Service Layer of a comprehensive four-layered smart city application architecture, which encompasses Device, Network, Service, and Application layers as seen in Fig 6.1. Central to its design is a middleware component that serves as an intermediary between the Network and Application layers. This middleware facilitates communication with oneM2M standards, effectively bridging the gap between low-level network operations and high-level application services. By interfacing directly with both oneM2M for standard-based interoperability and a database for data persistence, ctOP abstracts the complexities of direct oneM2M interactions. This abstraction not only simplifies the integration and management of IoT devices across urban ecosystems but also ensures adherence to the oneM2M standard, underpinning the platform's commitment to fostering scalable, interoperable smart city solutions. Within the ctOP platform, a hierarchical data model is meticulously implemented to streamline the organization and management of IoT devices. At the apex of this hierarchy lies the concept of 'Domains,' which categorizes the IoT ecosystem into distinct areas of application. Subsequent to Domains are 'Sensor Types,' defining the variety of sensors within each domain based on their Key Performance Indicator (KPI) (parameters) characteristics. At the base of the hierarchy, 'Nodes' represent individual instances of these sensor types, serving as the physical embodiment of the ctOP data model in the real world. This structured approach not only enhances the clarity and manageability of IoT devices across the smart city landscape but also lays the foundation for

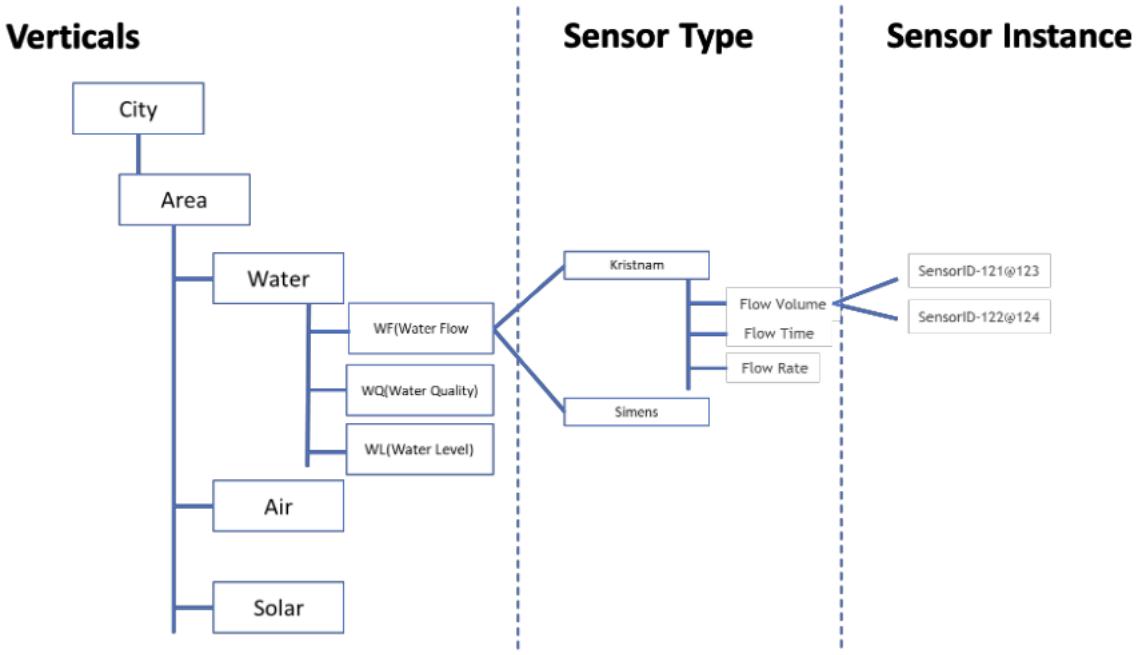



**Figure 6.2** The ctOP data model

sophisticated data analysis and application development. An example of the hierarchy can be seen in Fig 6.2. The intricacies of this data model and its implications for IoT device management and application integration within the ctOP platform are further explored in subsequent sections, providing a deeper understanding of its operational framework and strategic advantages.

## 6.4 Features and Functionalities

The ctOP platform is meticulously designed with an array of features aimed at simplifying the management and operation of IoT devices within smart city ecosystems. Each functionality plays a critical role in enhancing the platform's efficiency, scalability, and user-friendliness. Here, we delve into the core features of ctOP and illustrate how they collectively fulfil the platform's overarching goals.

- **User Management:** This foundational feature ensures robust access control and authentication mechanisms, enabling secure user registration, login, and management. By distinguishing between different user roles and permissions, ctOP facilitates a tailored experience for each stakeholder, from administrators to end-users, ensuring secure and efficient platform access.

- **CRUD for Verticals:** Vertical management allows for the categorization of IoT devices and services into distinct sectors such as transportation, energy, water, and air quality. The platform provides capabilities to create, read, update, and delete (CRUD) these verticals, offering flexi-

bility and adaptability to accommodate evolving urban needs and integrating new technologies seamlessly.

- **CRUD for Sensor Types:** Sensor types represent a higher level of abstraction, grouping nodes based on their functionalities, such as temperature sensors, air quality monitors, or water level detectors. This feature facilitates the management of diverse sensor types, enabling city planners and IoT developers to efficiently categorize and oversee a multitude of sensors deployed across different city verticals.

- **CRUD for Nodes:** At the heart of ctOP is the node management system, which allows for the comprehensive handling of individual sensor nodes — including creation, monitoring, updating, and deletion. This ensures that each node within the smart city network is accurately configured, functional, and up-to-date, thereby enhancing the reliability and effectiveness of data collection and analysis.

- **CR for Node Data:** Data management is a critical aspect of ctOP, enabling the creation and retrieval (CR) of data generated by the sensor nodes. This feature ensures the efficient collection, storage, and processing of real-time data, which is essential for monitoring urban environments and making data-driven decisions.

- **Vendor User Assigning to Node for Security:** Security is paramount in IoT deployments. ctOP addresses this by allowing specific vendor users to be assigned to nodes and ensuring that data posting and node management are securely controlled. This feature minimizes vulnerabilities and unauthorized access, bolstering the integrity and confidentiality of sensitive data.

- **Subscribing to a Node:** To facilitate real-time alerts and updates, ctOP incorporates a subscription mechanism. Users can subscribe to specific nodes to receive timely notifications about data changes or events, enhancing situational awareness and enabling prompt responses to emerging urban challenges.

Together, these features not only streamline the management of IoT deployments but also reinforce the ctOP platform's commitment to fostering scalable, interoperable, and user-friendly smart city infrastructures. By addressing the intricacies of device and data management, ctOP paves the way for smarter, more responsive urban environments that can adapt to the needs of their inhabitants. Through its comprehensive suite of features, ctOP demonstrates a tangible step forward in realizing the full potential of IoT technologies in enhancing urban living.

## 6.5 Implementation and Deployment

This section outlines the technical underpinnings, the implementation strategy for key features, and the deployment process of ctOP, illustrating its applicability in real-world smart city applications.

### 6.5.1 Technical Overview

The ctOP platform leverages a robust technology stack designed for scalability, flexibility, and ease of use:

- **OneM2M v1.4.1 with MongoDB:** Adopts the latest OneM2M standards for IoT device interoperability, ensuring seamless communication across various devices and platforms.
- **Frontend:** Developed using React and Material UI (MUI), the ctOP frontend offers a user-friendly interface for managing IoT deployments.
- **Backend:** FastAPI with Gunicorn and PostgreSQL provides a high-performance, easy-to-scale server side, capable of handling extensive IoT data securely and efficiently.

All components are dockerized for ease of deployment, with Docker Compose enabling a straightforward setup process detailed on the ctOP GitHub repository.

### 6.5.2 Implementation Strategy and Deployment Process

Key features of ctOP—ranging from user management to node data management—are designed to support complex smart city infrastructures. The platform emphasizes security and efficiency through JWT authentication and a clear separation of user roles (Admin, Vendor, and Normal Users), ensuring that each stakeholder can interact with the system effectively.

Deployment within a city-wide context begins with the installation and initial setup, followed by the detailed configuration of Domains (Verticals), Sensor Types, and Nodes. Administrators play a crucial role in orchestrating this process, from user creation to node assignment and data management.

### 6.5.3 Naming Convention for Nodes

An integral aspect of ctOP's implementation in city-wide deployments is its systematic naming convention for nodes, designed to facilitate easy identification, categorization, and management of IoT devices across diverse urban areas. The convention follows the pattern: *VerticalTypeSensorType-PinCode-Number (ABXX-YYYY-ZZZZ)*:

- **VerticalType:** A two-letter code representing the vertical (e.g., Water Monitoring – WM, Energy Monitoring – EM).
- **SensorType:** A unique ID within a vertical, identifying the sensor type (e.g., 01 for 'Kristnam').
- **PinCode:** The last four digits of the area's postal code where the node is deployed.
- **Number:** A four-digit unique identifier for the node that increments automatically.

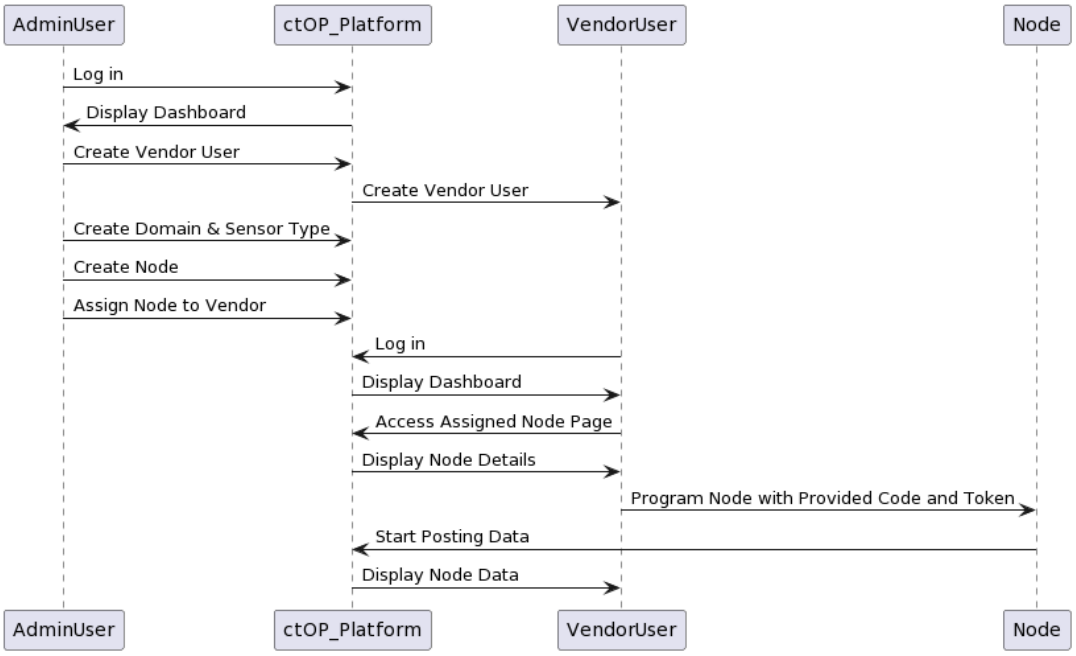



**Figure 6.3** High-Level Overview of ctOP Workflow

For example, a water quality sensor by Kristnam in the Gachibowli area would be named WQ01-0032-0001, where WQ stands for Water Quality, 01 for the Kristnam sensor type, 0032 for the area's pin code, and 0001 is the unique node identifier.

This naming convention ensures precise and scalable management of IoT nodes across extensive urban deployments, enhancing the ctOP platform's ability to support diverse and comprehensive smart city applications.

### 6.5.4 Real-World Applications

Deploying ctOP in city-wide contexts enables effective management of IoT devices for various applications, from environmental monitoring to resource management. The platform's architecture and features, supported by a logical naming convention for nodes, provide the foundation for scalable smart city initiatives, demonstrating ctOP's potential to significantly improve urban living conditions through advanced IoT solutions. By addressing the complexities of city-wide IoT deployments with a structured and user-friendly approach, ctOP stands as a pivotal tool for cities aiming to harness the power of IoT technologies for sustainable development and enhanced quality of life. This tool is planned to be launched across 200 cities in India.

## 6.6 Workflow

Here, we detail the workflow for using the platform, as can be seen in Fig 6.3. This will give a clear picture of how to use the platform. Detailed workflow images will be available in the Appendix

- Admin User Creates a Vendor User
- Admin Creates a Domain and Sensor Type for a node if not already existing
- Admin Creates a node using area details, latitude and longitude
- Admin Assigns Node to a Vendor from Node Details page (All Verticals ¿ Select Domain ¿ Select Node)
- The vendor now logs into the platform and opens the node page.
- If properly assigned, the vendor will be able to see the code to be put on a node to post data to om2m through the platform. (This code will also include a specific token)
- Node Starts posting data
- The data can be seen on the platform UI.
- The data can now further be used for any application.

## 6.7 Future Directions

As the ctOP platform evolves to meet the burgeoning demands of smart city ecosystems, strategic upgrades and innovations are imperative to maintain its relevance and effectiveness. Recognizing the dynamic nature of urban development and IoT technologies, the roadmap for ctOP's advancement encompasses several key areas of focus:

### 6.7.1 Transition to Mobius for Performance Enhancement

Informed by the comparative analysis presented in the previous chapter, a significant future direction for ctOP involves transitioning its core interoperability framework to Mobius. This shift is motivated by the observed performance gains, promising enhanced efficiency and scalability essential for processing the voluminous data generated by urban IoT networks. The transition to Mobius will be executed with meticulous planning to ensure seamless integration, minimizing disruption to existing services while unlocking new capabilities for the platform.

### 6.7.2 Clustered/Distributed Deployment Approach

To accommodate the diverse and distributed nature of smart cities, ctOP is set to adopt a clustered deployment strategy. This approach envisions setting up dedicated instances of the OM2M platform at the city level, thereby creating a network of localized IoT management hubs. Each city within the ctOP ecosystem will have its own OM2M instance tailored to its specific geographic, operational, and regulatory requirements. This decentralized architecture aims to enhance system resilience, scalability, and data sovereignty, aligning with the unique demands of each urban area.

### 6.7.3 Implementation of City-Level OM2M Instances

The rollout of city-level OM2M instances will involve close collaboration with local stakeholders, including municipal authorities and infrastructure partners. This process includes provisioning the necessary hardware, configuring network settings, and ensuring the integration of these instances with existing urban IT infrastructures. The objective is to equip each city-level instance with the capabilities to support a wide range of IoT applications, from traffic management systems to environmental sensors, thereby fostering a responsive and interconnected urban IoT ecosystem.

*Chapter 7*

# Conclusion

## 7.1 Conclusion

This thesis has embarked on an in-depth exploration of the pivotal role of IoT technologies in the realization of smart city visions, laying a particular emphasis on the essence of interoperability and scalability. Through the development and analysis of the ctOP platform, alongside a comprehensive study of multiple oneM2M systems, this work has highlighted the intricacies and challenges inherent in designing IoT infrastructures that are both robust and flexible.

The proposed ctOP architecture, comprising the Data Monitoring Layer (DML), Data Enhancement Layer (DEnL), Data Storage Layer (DSL), and Data Exchange Layer (DEL), represents a significant contribution to the field of smart city interoperability. By leveraging distributed oneM2M instances, this architecture achieves enhanced performance and scalability, addressing the critical demands of smart city ecosystems.

The comparative analysis of oneM2M implementations underscored Mobius's superior performance, particularly in handling high concurrency with its efficient concurrency control mechanism. Mobius's capacity to process a peak rate of 568 requests per second, coupled with its significantly lower latency in data insertion and retrieval, distinctly positions it as the preferred framework for IoT applications demanding scalability and real-time responsiveness. These findings not only inform the decision to transition ctOP to Mobius but also emphasize the delicate balance between architectural complexity and performance efficiency. A key takeway from this analysis was that a well written code does not always imply that it's a well performing code.

The broader implications of this study suggest a potential shift towards more adaptable and efficient architectural models in the realm of IoT-based interoperability solutions. The ctOP platform's real-world significance underscores the value of architectural innovation in meeting the complex needs of smart cities.

## 7.2 Future Work

Future research could focus on addressing potential limitations in existing oneM2M implementations. Specifically, refining the publish/subscribe mechanism to better mirror scalable models like Kafka could significantly enhance performance. Additionally, exploring ways to further modularize functionalities would promote easier customization and integration of new features and services. This ongoing refinement will ensure oneM2M architectures remain adaptable and effective in meeting the evolving needs of smart cities. Looking ahead, the architectural evolution of IoT platforms for smart cities appears poised for a paradigm shift towards microservices-based approaches.

While oneM2M provides a standardized framework for IoT interoperability, its implementations as we discussed in Chapter 4 face scalability and performance challenges. Future research could explore the potential of microservices architecture in enhancing the efficiency and adaptability of oneM2M platforms. This could involve decomposing core functionalities into independent microservices, allowing for granular scaling and deployment based on specific needs. Additionally, investigating the integration of containerization technologies and service meshes could further streamline the management and communication within these distributed architectures. Addressing challenges such as data consistency and network latency will be crucial in realizing the full potential of microservices for oneM2M-based smart city solutions. This approach not only addresses the identified limitations in current frameworks but also aligns with the evolving requisites of smart cities, fostering more resilient, efficient, and intelligent urban landscapes.

Furthermore, the exploration of adaptive and fault-tolerant mechanisms within these distributed architectures could significantly contribute to the resilience and robustness of smart city platforms. Ensuring the resilience and robustness of interoperable IoT systems is paramount for reliable smart city operations. Future research could investigate the integration of adaptive and fault-tolerant mechanisms within oneM2M frameworks. This could involve exploring techniques such as self-healing systems that automatically detect and recover from errors, dynamic load balancing to optimize resource utilization across distributed nodes, and redundancy and failover mechanisms to ensure continuous service availability. By incorporating these features, oneM2M platforms can become more dependable and resilient, supporting critical smart city. These principles could be applied to the ctOP platform to make it more fault tolerant through the integration of Artificial Intelligence (AI) and Machine Learning (ML). This holds immense potential for enhancing the intelligence and self-management capabilities of IoT platforms. Future research could explore the use of AI for predictive maintenance, anomaly detection, and automated resource optimization within smart city deployments. By enabling systems to learn from data patterns and proactively address potential issues, AI can contribute to minimizing downtime and improving overall service quality.

In conclusion, this thesis lays the groundwork for future innovations in the field of IoT and smart cities, offering a roadmap for the development of more capable, resilient, and user-friendly platforms. The journey towards fully realized smart cities is ongoing, and the insights gained from this work will undoubtedly inform and inspire the next steps in this exciting and dynamic field.

*Appendix A*

# Low Level Workflow Diagrams for ctOP

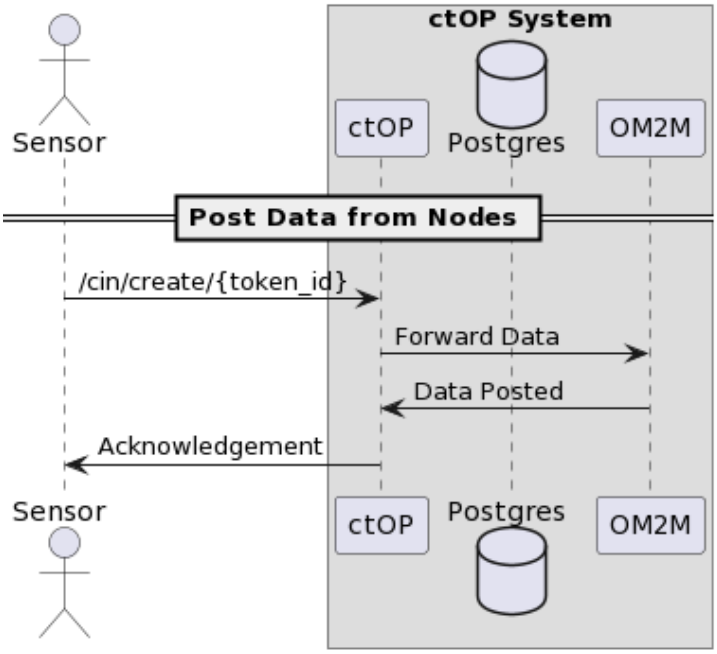



**Figure A.1** Data Posting

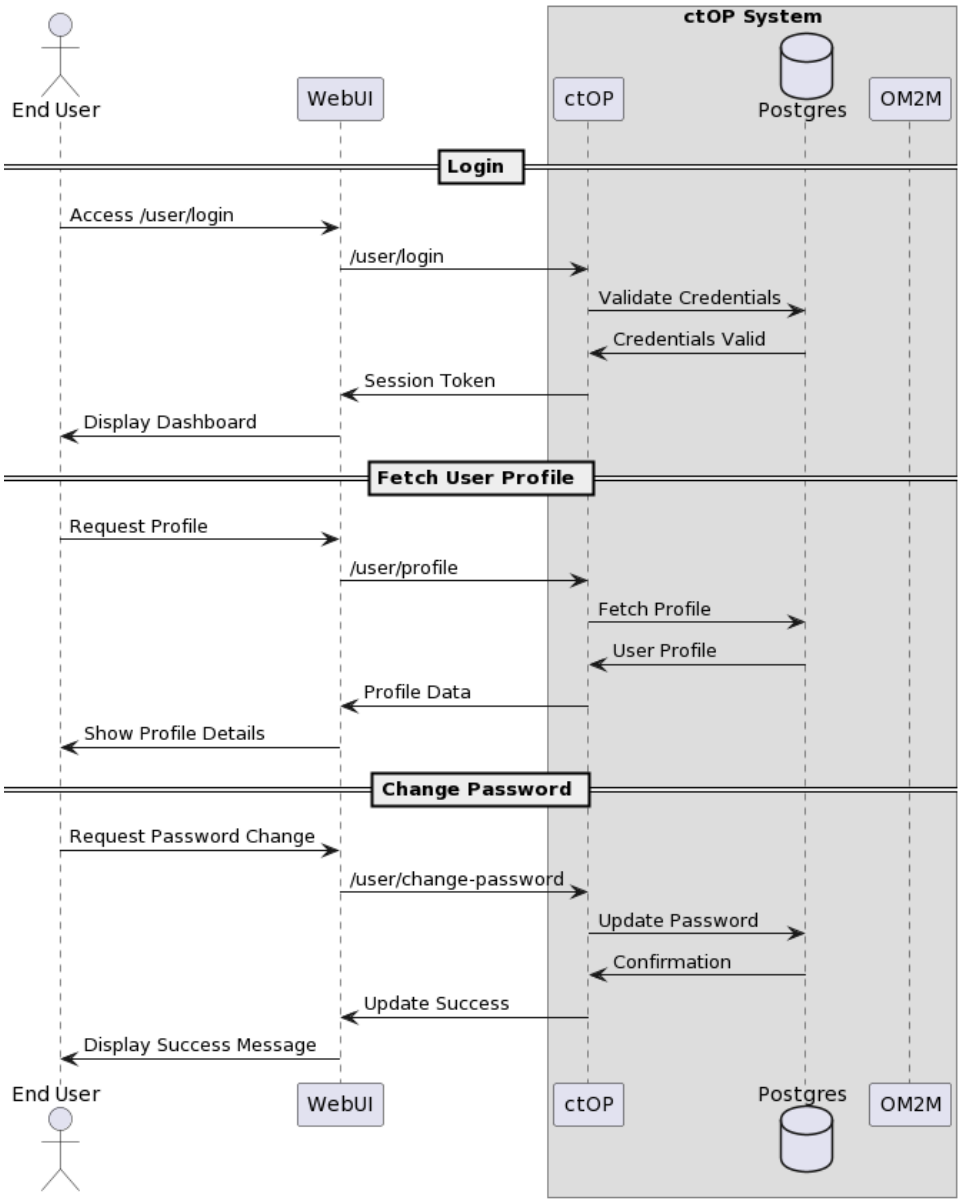



**Figure A.2** User Auth

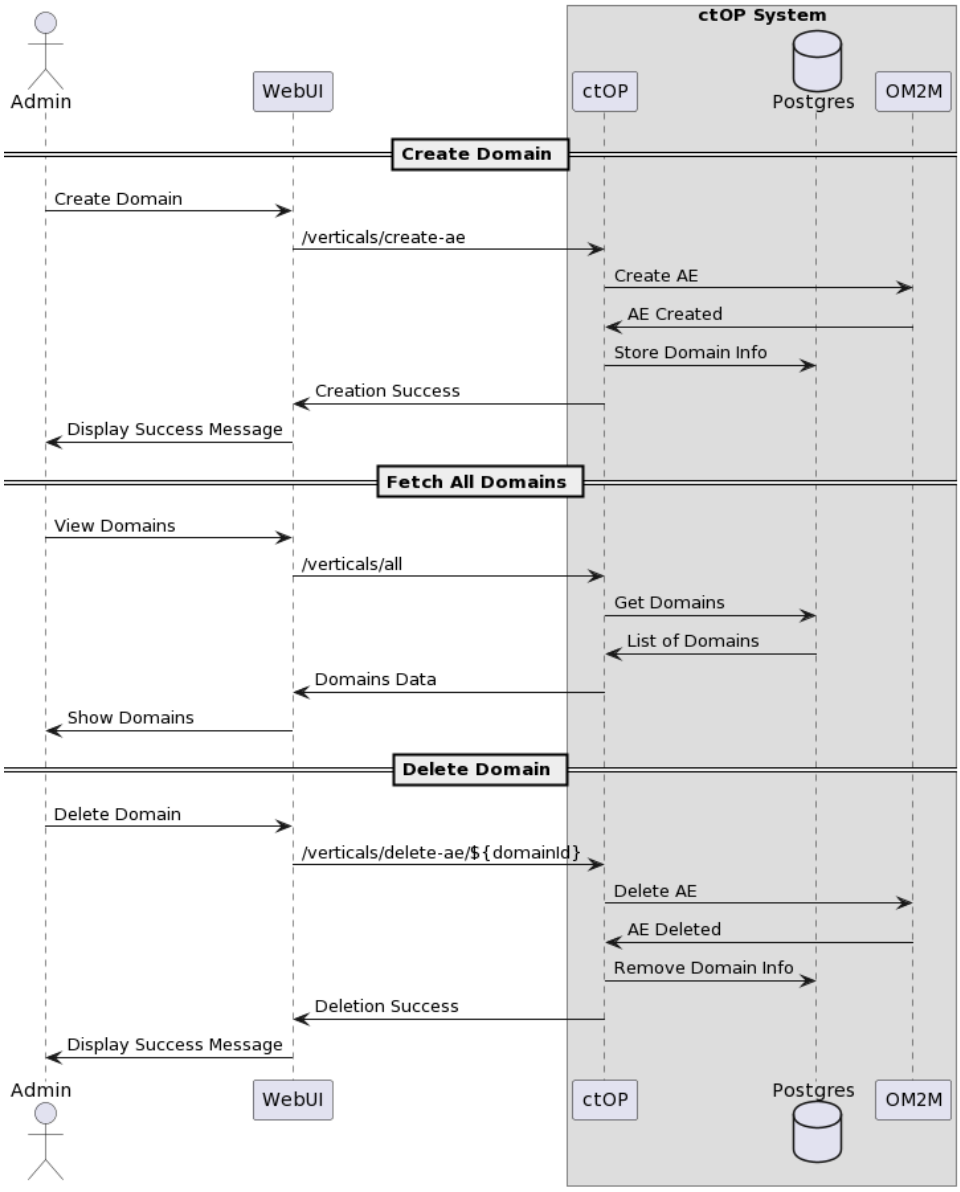



**Figure A.3** Domain Management

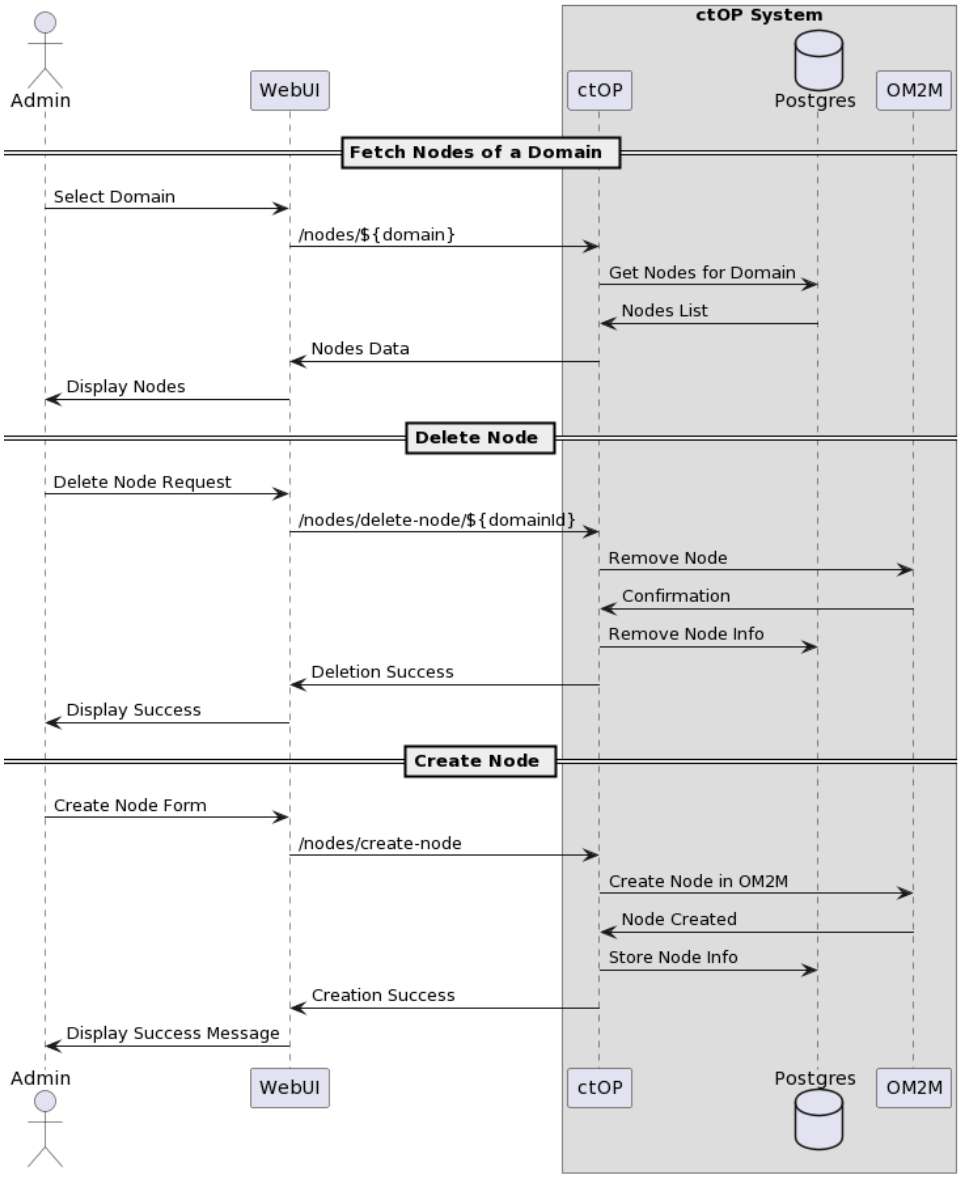



**Figure A.4** Nodes Management

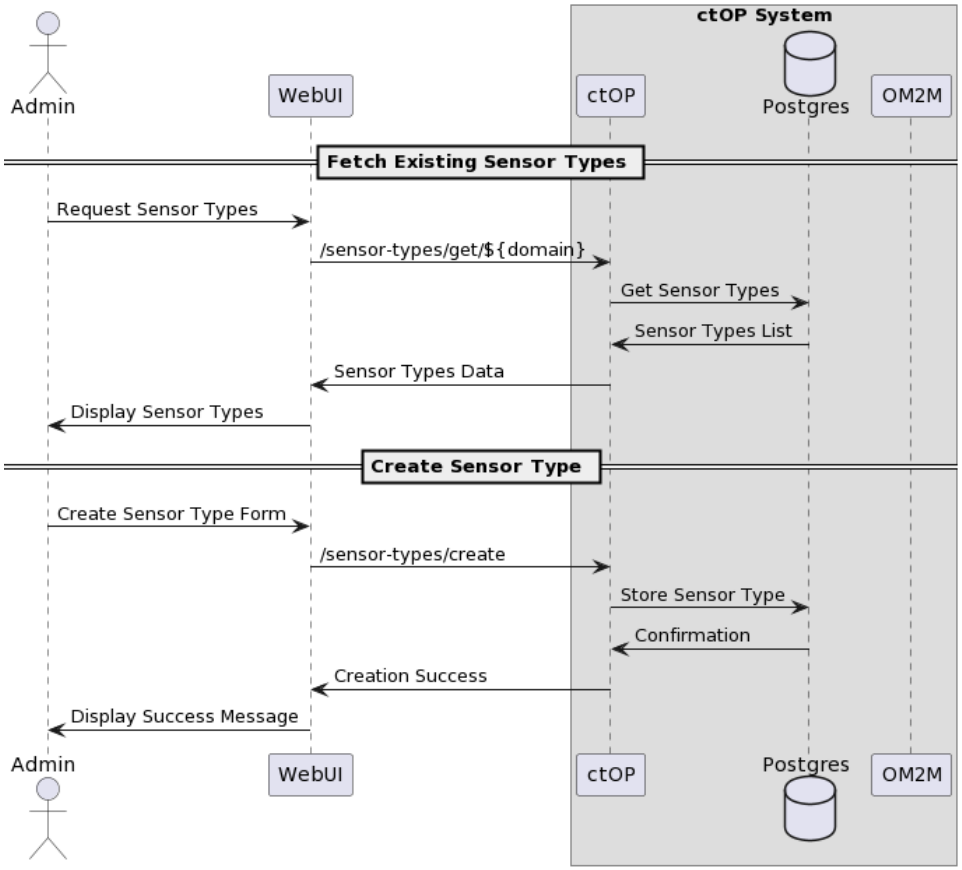
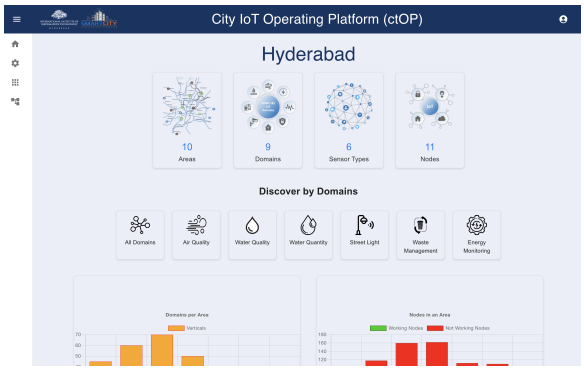


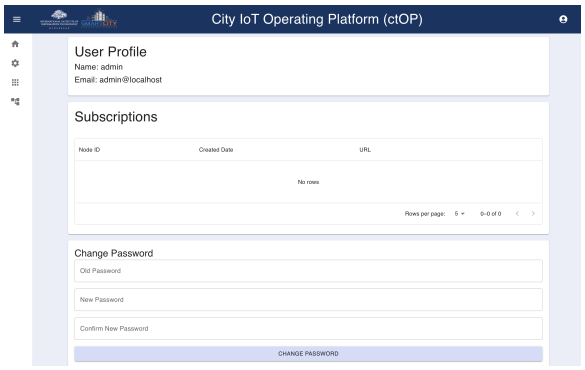

**Figure A.5** Sensor Type Management

*Appendix B*

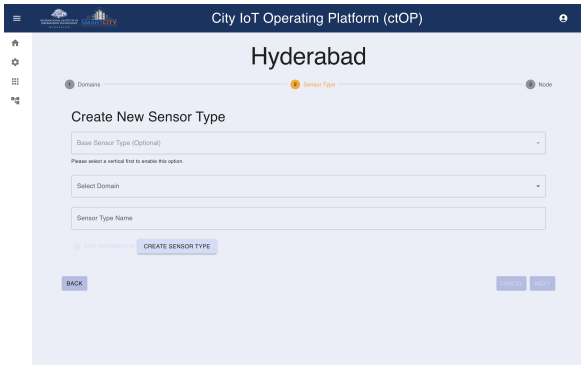
# ctOP Pages



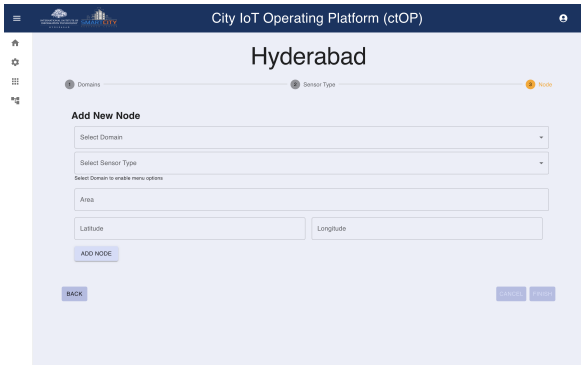

(a) Homepage



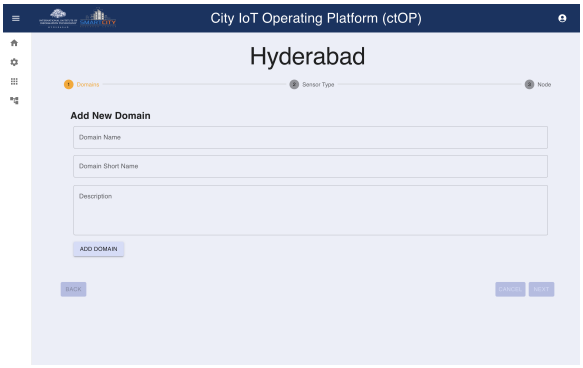

(b) Account Management



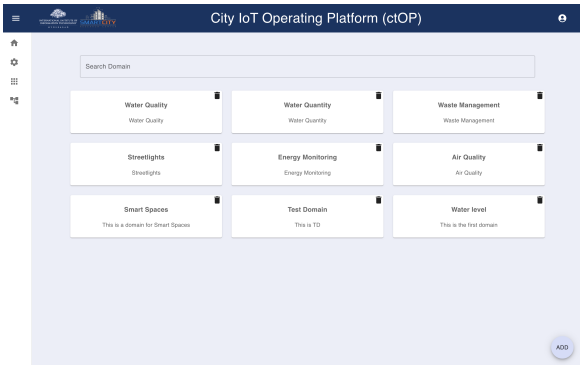

(c) Create Sensor Type



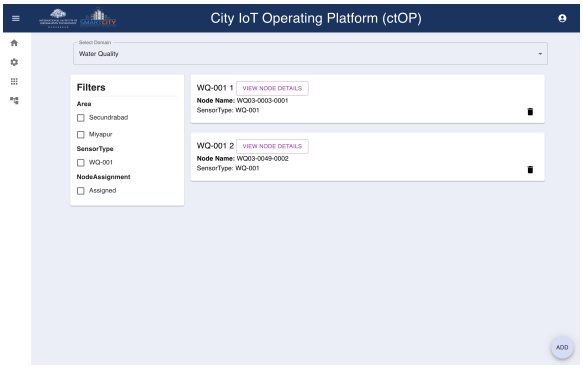

(d) Create Node



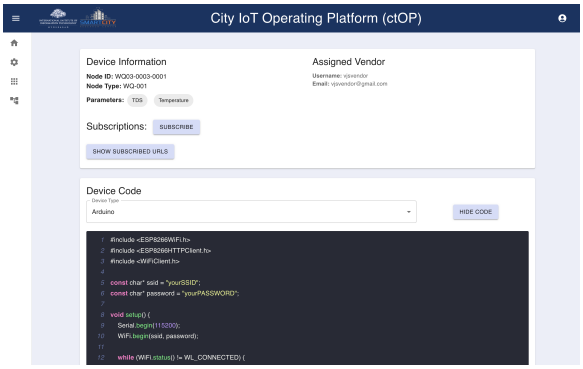

(e) Create Domain




(f) Domains




(g) Nodes




(h) Node Detail


(i) Create User


(j) Login

**Figure B.1** ctOP Deployment Screenshots

# Bibliography


[1] "Smart city living lab," 2024. [Online]. Available: https://smartcityresearch.iiit.ac.in/

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

[3] R. Kamal, *Internet of Things: Architecture and Design Principles*. McGraw Hill Education, 2017.

[4] A. Bahga and V. Madisetti, *Internet of Things: A hands-on approach*. Vpt, 2014.

[5] oneM2M, "oneM2M," 2024. [Online]. Available: https://www.oneM2M.org/

[6] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128610001568

[7] D. Navani, S. Jain, and M. S. Nehra, "The internet of things (iot): A study of architectural elements," in *2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, 2017, pp. 473–478.

[8] S. Mante, S. S. S. Vaddhiparthy, M. Ruthwik *et al.*, "A multi layer data platform architecture for smart cities using onem2m and iudx," in *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)*, 2022, pp. 1–6.

[9] A. Taivalsaari and T. Mikkonen, "A roadmap to the programmable world: Software challenges in the iot era," *IEEE Software*, vol. 34, no. 1, pp. 72–80, 2017.

[10] A. Gaur, B. Scotney, G. Parr, and S. McClean, "Smart city architecture and its applications based on iot," *Procedia Computer Science*, vol. 52, pp. 1089–1094, 2015, the 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915009229

[11] oneM2M, "oneM2M Common Service Entities," 2024. [Online]. Available: https://onem2m.org/using-onem2m/developers/basics

[12] P. Sethi and S. Sarangi, "Internet of things: Architectures, protocols, and applications," *Journal of Electrical and Computer Engineering*, vol. 2017, pp. 1–25, 01 2017.

[13] I. Yaqoob, E. Ahmed, I. A. T. Hashem *et al.*, "Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges," *IEEE Wireless Communications*, vol. 24, no. 3, pp. 10–16, 2017.

[14] S. Pal, V. G. Diaz, and D.-N. Le, "Aws iot core," 2022. [Online]. Available: https://docs.aws.amazon.com/iot/

[15] Philmea, "Azure iot." [Online]. Available: https://learn.microsoft.com/en-us/azure/iot/

[16] M. B. Alaya, S. Medjiah, T. Monteil *et al.*, "Toward semantic interoperability in onem2m architecture," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 35–41, 2015.

[17] A. Souza, J. Pereira, J. Oliveira *et al.*, "A data integration approach for smart cities: The case of natal," in *2017 International Smart Cities Conference (ISC2)*, 09 2017, pp. 1–6.

[18] J. Guth, U. Breitenbücher, M. Falkenthal *et al.*, "A detailed analysis of iot platform architectures: Concepts, similarities, and differences," in *Internet of Everything*, 2018.

[19] S. Jeong, S. Kim, and J. Kim, "City data hub: Implementation of standard-based smart city data platform for interoperability," *Sensors*, vol. 20, no. 23, p. 7000, Dec 2020. [Online]. Available: http://dx.doi.org/10.3390/s20237000

[20] X. Liu, A. Heller, and P. S. Nielsen, "Citiesdata: a smart city data management framework," *Knowledge and Information Systems*, vol. 53, no. 3, pp. 699–722, Dec 2017. [Online]. Available: https://doi.org/10.1007/s10115-017-1051-3

[21] S. Krčo, B. Pokrić, and F. Carrez, "Designing iot architecture(s): A european perspective," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 79–84.

[22] V. Pranavasri, L. Francis, U. Mogadali, G. Pal, S. S. S. Vaddhiparthy, A. Vattem, K. Vaidhyanathan, and D. Gangadharan, "Scalable and interoperable distributed architecture for iot in smart cities," *Authorea Preprints*, 2023.

[23] B. Mohanty, *Strategic Investments Towards Resource Efficient Cities*. United Nations Environment Programme (UNEP), 01 2015, pp. 140–155.

[24] M. S. Camaren Pete, *Sustainable, Resource efficient cities - Making it happen!* United Nations Environment Programme, 2012.

[25] T. Nam and T. Pardo, "Conceptualizing smart city with dimensions of technology, people, and institutions," in *Value Co-Creation Practices in Smart City Ecosystem*, 06 2011, pp. 282–291.

[26] R. Giffinger, C. Fertner, H. Kramar *et al.*, *Smart cities - Ranking of European medium-sized cities*. Vienna University of Technology, 01 2007.

[27] N. D. K. R. Chukka, A. Arivumangai, S. Kumar *et al.*, "Environmental impact and carbon footprint assessment of sustainable buildings: An experimental investigation," *Adsorption Science & Technology*, Mar 2022.

[28] L. Farhan, S. Shukur, A. E. Alissa *et al.*, "A survey on the challenges and opportunities of the internet of things (iot)," in *2017 Eleventh International Conference on Sensing Technology (ICST)*, 12 2017, pp. 1–5.

[29] M. B. Alaya, Y. Banouar, T. Monteil *et al.*, "Om2m: Extensible etsi-compliant m2m service platform with self-configuration capability," *Procedia Computer Science*, vol. 32, pp. 1079–1086, 2014, the 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050914007364

[30] R. Kuc and M. Rogozinski, *Elasticsearch Server*, ser. Community experience distilled. Packt Publishing, 2013. [Online]. Available: https://books.google.co.in/books?id=PEFK3MuwBsIC

[31] "Kafka connect — confluent documentation." [Online]. Available: https://docs.confluent.io/platform/current/connect/index.html

[32] U. Thacker, M. Pandey, and S. Rautaray, *Review of Elasticsearch Performance Variating the Indexing Methods*. Springer Singapore, 01 2018, pp. 3–8.

[33] K. Vandikas and V. Tsiatsis, "Performance evaluation of an iot platform," in *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, 2014, pp. 141–146.

[34] IoTKETI, 2023. [Online]. Available: http://developers.iotocean.org/archives/module/mobius

[35] A. Kraft, "Acme," https://github.com/ankraft/ACME-oneM2M-CSE.

[36] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.

[37] S. Pradeep and Y. K. Sharma, "A pragmatic evaluation of stress and performance testing technologies for web based applications," in *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 399–403.

[38] J. Heyman, L. Holmberg, C. Byström *et al.*, "Locust." [Online]. Available: https://docs.locust.io/

[39] O. W. Group, "Osgi docs home page." [Online]. Available: https://docs.osgi.org/

[40] Apache, “Maven,” https://maven.apache.org/, 2023.

[41] E. Web, “Eclipse tycho,” Jan 2013. [Online]. Available: https://projects.eclipse.org/projects/technology.tycho

[42] Node.js, “Node.js,” 2023. [Online]. Available: https://nodejs.org/en

[43] J. Yun, I.-Y. Ahn, S. Choi *et al.*, “Tteo (things talk to each other): Programming smart spaces based on iot systems,” *Sensors*, vol. 16, p. 467, 04 2016.

[44] C. Zhang and D. Budgen, “What do we know about the effectiveness of software design patterns?” *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 1213–1231, 2012.

[45] N. Gonçalves, D. Faustino, A. R. Silva *et al.*, “Monolith modularization towards microservices: Refactoring and performance trade-offs,” in *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, 2021, pp. 1–8.

[46] M. Siemens, “Tinydb,” https://tinydb.readthedocs.io/, 2023.