Digital Communications and Networks(DCN)

HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/dcan

Standardised Interworking and Deployment of IoT and Edge Computing Platforms

Jieun Lee[a], JooSung Kim[a], Seong Ki Yoo[b], Tarik Taleb[*ac], JaeSeung Song[*a]

[a]*Sejong University, Seoul, 05006, Korea*
[b]*Coventry University, Coventry, U.K.*
[c]*Ruhr University Bochum, Bochum, Germany*

Abstract

Edge computing is swiftly gaining traction and is being standardised by the European Telecommunications Standards Institute (ETSI) as Multi-access Edge Computing (MEC). Simultaneously, oneM2M has been actively developing standards for dynamic data management and IoT services at the edge, particularly for applications that require real-time support and security. Integrating MEC and oneM2M offers a unique opportunity to maximize their individual strengths. Therefore, this article proposes a framework that integrates MEC and oneM2M standard platforms for IoT applications, demonstrating how the synergy of these architectures can leverage the geographically distributed computing resources at base stations, enabling efficient deployment and added value for time-sensitive IoT applications. In addition, this study offers a concept of potential interworking models between oneM2M and the MEC architectures. The adoption of these standard architectures can enable various IoT edge services, such as smart city mobility and real-time analytics functions, by leveraging the oneM2M common service layer instantiated on the MEC host.

KEYWORDS:
Internet of things, Multi-access edge computing, oneM2M, Interworking, Standards.

1. Introduction

In the rapidly advancing digital era, the Internet of Things (IoT) and edge computing have emerged as leading technological trends, garnering increasing interest across various industries. The combined potential of these two technologies to bring computational capabilities closer to data sources, or edge nodes, is poised to reshape future communication systems [1]. However, the growing demand for edge computing and IoT technologies often encounters an impediment: a lack of standardised interworking between the two.

The advent of IoT standardization, such as oneM2M (a global IoT standards initiative), has heralded enabling diverse applications across a wide range of services. However, the traditional IoT platforms on the cloud struggle with latency, especially for real-time services. This delay, caused by the back-and-forth data transmission to the cloud, underscores the importance of edge computing as a solution. Also, integrating these standardized IoT platforms with edge computing has remained a challenge.

While edge computing enhances processing efficiency, existing approaches often suffer from fragmented solutions and lack of seamless interoperability with standardized IoT platforms. Previous works have primarily focused on theoretical or simulation-based models, which are insufficient for real-time, latency-sensitive applications. Moreover, dynamic resource offloading to edge nodes, crucial for ensuring low latency, has not been addressed comprehensively in practical deployments. Hence, there is a growing discussion in the industry advocating for IoT cloud services to be executed in edge computing environments [2]. This shift could significantly reduce system latency. Compared to previous works, which either rely on simulations or lack standard-based methodologies, our study bridges this gap by demonstrating practical feasibility and enhanced performance through edge-cloud synergy.

In this article, we explore the integration and deployment strategies of standardised edge computing and IoT platforms, specifically focusing on oneM2M and Multi-access Edge Computing (MEC) developed by the European Telecommunications Standards Institute (ETSI). We selected these two platforms owing to their robust standardisation and growing prominence in their respective domains [3]. Our approach, combining oneM2M with MEC, addresses this gap by providing a standardized, interoperable platform that leverages the strengths of both technologies. This integration facilitates lower latency, greater data processing efficiency, and improved scalability, surpassing the capabilities of traditional edge computing models. By focusing on this synergy, we propose a unique solution that significantly enhances IoT applications' responsiveness and efficiency.

First, we conducted a comprehensive review and analysis of the existing standard technologies related to IoT and edge computing, performing a gap analysis to identify and address the disparities and limitations in the current landscape. Our objective is to pinpoint methods that en-

*JaeSeung Song and Tarik Taleb are co-corresponding authors.
[1]E-mail addresses: je.lee@sejong.ac.kr (J. Lee); sametim17@gmail.com (J. Kim); ad3869@coventry.ac.uk (S. K. Yoo); tarik.taleb@ruhr-uni-bochum.de (T. Taleb); jssong@sejong.ac.kr (J. Song).

able a harmonious collaboration between these platforms, addressing existing challenges and promoting broader deployment. We assess a lightweight iteration of edge computing within the oneM2M system, illustrating the potential of merging edge computing and IoT to enhance applications across different sectors. Ultimately, it is crucial to establish standardised techniques for IoT-focused edge computing to ensure consistency, reliability, and efficiency in deployment. This integrated approach can maximise the capabilities of both edge computing and IoT, laying the groundwork for more agile and efficient applications in diverse industries.

In summary, the main contributions of this study are as follows:

1. In-Depth Analysis of current standards: Provide comprehensive analysis and gap analysis of existing standard technologies in IoT and edge computing, providing a foundation for further research and development based on real-world standard applications.
2. Innovative Deployment Mechanism based on actual standards: Develop and introduce a novel mechanism for deploying standardized IoT services within edge computing environments, ensuring seamless interoperability based on actual interworking scenarios rather than hypothetical models.
3. Robust performance evaluation with real data: Showcase the capabilities of a newly developed, lightweight IoT server optimized for edge computing through comprehensive performance evaluations and experimental results using actual experimental data.

The remainder of this paper is organised as follows. Section 2 delves into edge computing technologies and surveys the trends in various standardization efforts, including the oneM2M system, MEC, and the 3rd Generation Partnership Project (3GPP) edge computing. Section 3 introduces specific use cases that are applicable in both the IoT and edge computing landscapes. Section 4 provides a detailed discussion on the architecture of IoT and edge computing as well as the interoperability architecture of standardisations. Section 5 elaborates on the integrated deployment strategies and procedural intricacies for edge computing, explaining various interworking options in detail. Section 6 presents the evaluation results concerning performance, using a lightweight IoT server in an edge computing environment. Finally, we conclude the article in Section 7.

2. Technical Standards for Edge Computing

Numerous standardisation bodies in the domain of edge computing have significantly contributed to its advancements in this field. As edge computing continues to evolve, standards are expected to adapt and grow to meet emerging technological advancements. In this section, we explore the current standard trends and potential related to edge computing.

2.1. oneM2M with edge computing

oneM2M is a global standards initiative developing technical specifications for a common IoT and machine-to-machine (M2M) service layer [4]. The oneM2M IoT service layer platform constitutes a critical set of standards aimed at facilitating communication and interoperability among a diverse array of IoT devices and applications. Established by multiple international standardisation bodies, oneM2M addresses the need for a universal IoT service layer that can be seamlessly embedded within a variety of hardware and software solutions. Such a capability can ensure uninterrupted connectivity and interoperability among IoT devices and applications.

A key attribute of oneM2M is its role as an enabler for cloud-based IoT service platforms, enabling it to interface with a wide range of IoT devices/networks to provide seamless interoperability, irrespective of the underlying technologies or communication protocols. Additionally, oneM2M offers an essential framework for bolstering security and privacy management in IoT applications. The scope of the oneM2M is expanding to include edge computing technologies. This expansion aims to reduce the load on data centers and core networks and improve communication latency through localised data acquisition, processing, and storage. Notably, oneM2M has already defined edge computing use cases, requirements and architectural design in its Release 4 specifications [5].

Despite oneM2M expanding its activities related to edge computing, technological development in this field is still in its infancy. Therefore, further research is required to strengthen the integration of oneM2M and edge computing technologies.

2.2. ETSI ISG MEC

The ETSI Industry Specification Group for Multi-access Edge Computing (ISG MEC) has established an array of standards for edge computing of mobile networks. MEC provides an IT service environment and cloud-computing capabilities at the edge of the network, within proximity to mobile subscribers. MEC technology aims at reducing latency, providing high bandwidth, improving the real-time performance of applications, and enabling efficient service delivery [6, 7, 8]. Its ability to support high-frequency data transfer makes it an ideal choice for applications such as IoT, augmented/virtual reality (AR/VR), autonomous vehicles, and 5G services.

MEC has evolved through several phases of development, each with its focus [9, 10]. Phase 1 provided the groundwork, setting up architectures and application programming interfaces (APIs) for managing and enabling applications in edge data centers. Phase 2 expanded the scope to "multi-access" edge computing, incorporating other networks such as Wi-Fi and Fixed Access, and beginning to address vertical market needs. Phase 3 extended traditional cloud and Network Functions Virtualisation (NFV) approaches to accommodate the heterogeneity of MEC, introducing the MEC IoT API and exploring MEC deployments in various enterprises. Several studies have explored the potential of MEC. For instance, [11] proposed a novel architecture for the efficient deployment of services in MEC environments. MEC has also been considered for enabling ultra-reliable low-latency communication in 5G networks [12].

However, the research and standardisation work on IoT requires to be further strengthened to unlock the true potential of this technology. Standardised research and protocols are required to ensure that MEC can integrate seamlessly and operate effectively with technologies such as IoT. In particular, it is important to develop scalable architectures and frameworks to ensure interoperability between different devices and platforms, effectively address security and data protection issues, and support a variety of IoT services.

Therefore, the research and development activities of ETSI ISG MEC must be further strengthened through close integration and collaboration with IoT, enabling a robust and standardised MEC platform to effectively address all future use cases and applications. This standardized approach will be a key element in ensuring that MEC and IoT integrate and interact with each other while ensuring reliable and efficient service delivery across industries.

2.3. 3GPP and Others

3GPP, a global body for mobile communication standards, covers edge computing due to its ties to 5G. Notably, "5G System; System Architecture (TS 23.501)" details edge computing support via user plane separation and network slicing, while "5G System; Procedure descriptions (TS 23.502)" outlines its practical implementation [13].

The OpenFog Consortium, now under the Industrial Internet Consortium (IIC), introduced Fog Computing to tackle edge computing deployment challenges. It extends cloud principles to the network's edge, reducing latency and boosting efficiency. The approach supports diverse applications, fosters industry standardization, and addresses issues like bandwidth and latency.

EdgeX Foundry, by the Linux Foundation, offers a common IoT edge computing framework. This open-source initiative provides a versatile platform for designing and launching diverse industrial solutions. It ensures device and application interoperability through APIs and nurtures an ecosystem of compatible components.

Table 1
Comparison of ETSI MEC, oneM2M, and Our Work

Features	ETSI MEC	oneM2M	Our Work
Standards-based Integration	Partial (MEC-specific)	IoT-focused only	Full MEC-oneM2M integration
Interworking Architecture	Defined APIs but limited	Focus on IoT Cloud	Real-world interworking
Edge Service Offloading	Limited offloading	Not supported for edge	Fully supported
Latency Optimization	Localized at MEC nodes	High latency at cloud	Cloud-edge synergy results
Performance Evaluation	Conceptual/Simulations	Theoretical focus	Extensive real-world data

In [14], the authors introduced an interworking between MEC and the IoT. While positioning the IoT platform within the MEC architecture shares similarities, the actual data processing and handling do not occur at the edge, preventing the realization of the full benefits. Furthermore, considerations regarding the migration of the IoT platform and the continuity of services have not been addressed, rendering it challenging to predict the extent of benefits that can be derived from utilizing edge computing.

The [15] introduces the concept of offloading in the context of the IoT utilizing MEC. The study primarily focuses on the performance of algorithms, overlooking the practical applicability of the proposed procedures and platforms based on simulations and proprietary solutions. This approach limits the usability of these concepts in real-world environments that adhere to universal standards. Additionally, the research does not address considerations such as the migration of tasks during object movement or the instantiation of the IoT platform, further constraining its applicability in dynamic and standard-based settings.

In summary, the research and development related to service provision utilizing the IoT and MEC, has been a notable deficiency in studies grounded in standard-based approaches. Additionally, research involving tangible implementations remains underdeveloped.

Table 1 highlights the differences and novel contributions of this work. Compared to ETSI MEC and oneM2M, this study fully integrates MEC and oneM2M platforms, enabling seamless edge service offloading and real-world interworking. This integrated approach addresses the limitations of existing solutions and offers a robust framework for real-time, low-latency IoT applications in edge computing environments.

Unlike prior research that often relies on hypothetical models or simulation-based approaches, our study employs a standard-based methodology leveraging both oneM2M and ETSI MEC standards to demonstrate practical edge computing applications. While ETSI MEC provides localized edge capabilities and oneM2M focuses primarily on IoT cloud solutions, a gap exists in achieving full interworking between these platforms for efficient edge service delivery.

3. Use Cases for Edge Computing

In this section, we explore some use cases that exemplify the synergistic potential of IoT and edge computing.

3.1. Automotives

Fig. 1a illustrates a scenario where a cloud IoT server delegates tasks and resources to edge computing capabilities for executing a vulnerable road user (VRU) detection service, which uses accurate positioning information shared by VRUs (e.g., pedestrians and cyclists) via their mobile devices to determine their positions on the road. When a host vehicle (HV) registered with an offloading service enters a zone covered by an edge node, such as a roadside unit, the offloading process begins, shifting the VRU detection tasks and resources associated with the HV to the edge node. In this process, the IoT cloud signals the gateway to perform resource offloading, enabling the gateway to retrieve all relevant resources and services related to the HV. Due to its proximity to the HV, the edge gateway can promptly dispatch a warning notification to the HV upon detecting a VRU on the road.

Specifically, these technologies can pre-emptively identify VRUs in real time because the capabilities originally associated with the cloud are now being available at the edge. Particularly advantageous is that the peripheral information from the HV can be processed directly at the edge nodes, eliminating the need for data transmission to and from a cloud-based IoT platform. This enables significantly faster data processing and command delivery compared to the traditional cloud-based IoT services. The concept of offloading within oneM2M enables the IN-CSE, a centralized IoT server platform, to migrate relevant resources and tasks to a target Edge MN-CSE node. Consequently, the MN-CSE node can directly cater to the needs of nearby IoT end devices with the offloaded service.

3.2. Smart Factories

Smart factory environments employ programmable logic controllers, which generate substantial volumes of data through the continuous monitoring of production lines. Particularly, in a standard smart factory environment, a diverse range of machines—including robots and monitoring equipment—produce substantial volumes of data, which are impractical to transfer to the cloud. Notably, functions such as real-time defect monitoring necessitate swift data processing and minimal response times. Edge computing can aid in minimising latency by locally processing these data. By making and implementing decisions that affect individual assembly lines or workstations at their respective locations, operational efficiency can be enhanced (Fig. 1b. Furthermore, local data processing minimises privacy risks and reduces reliance on central clouds.

Interworking between Cloud IoT and edge platforms enhances smart factory capabilities. oneM2M and similar IoT platforms ensure interoperability among diverse devices and systems, enabling dynamic task and resource allocation between edge and cloud computing, thereby optimising speed while reducing latency. This interoperability ensures harmonious function between machines from different manufacturers. For predictive maintenance, machine learning forecasts potential machine failures, preventing unscheduled downtime and distributing analytical tasks between edge devices and centralised resources for quicker detection and response, with enhanced geospatial specificity.

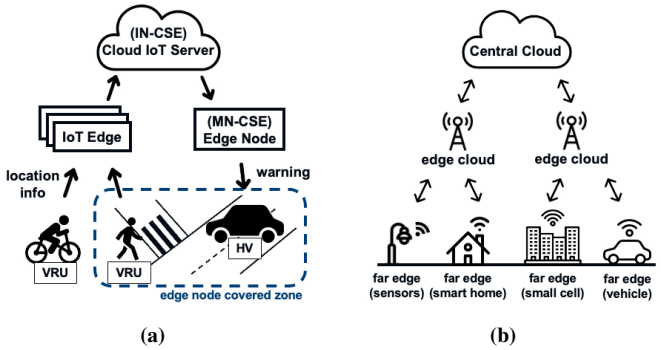

Fig. 1. Scenario overview of use cases: (a) VRU collision detection utilizing IoT and edge computing, and (b) robots and machines in a smart factory are controlled in real-time through edge computing and IoT platforms.

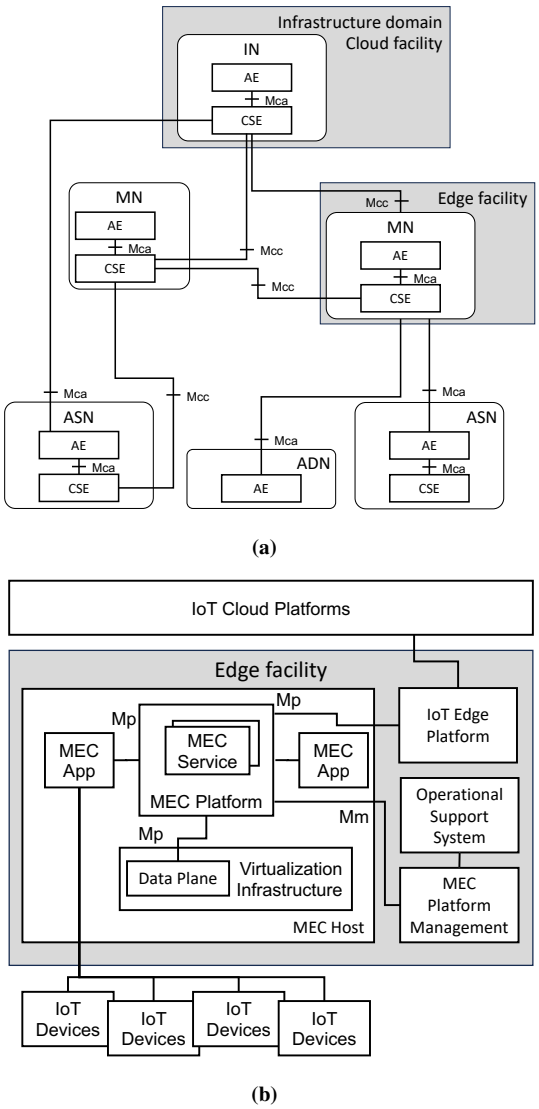

Fig. 2. Simplified oneM2M and MEC reference architectures: (a) oneM2M reference architecture connecting the cloud, gateway, edge, and devices to enable IoT services, and (b) ETSI MEC platform architecture interconnecting IoT devices and cloud IoT platforms.

4. Standardized IoT and Edge Platforms

In this section, we first analyse the architectures of the ETSI ISG MEC and oneM2M, which are leading global standards for edge computing and IoT, respectively. Next, we explore how these two architectures can be synergistically deployed to enable an efficient and interoperable implementation of common IoT services within edge computing environments.

4.1. oneM2M Architecture

The oneM2M consolidates all components within the IoT solution stack, establishing a distributed software/middleware layer that serves as a bridge between applications and the fundamental communication networking hardware/software [16]. It can be incorporated into various IoT devices, gateways, and cloud servers. As shown in Fig. 2a, the architecture of oneM2M is designed for flexibility and universal applicability. It comprises the following key logical entities:

1. Application Entity (AE): This entity encompasses the application-specific logic, for instance, software operating on an IoT device such as a temperature sensor or connected car.
2. Common Service Entity (CSE): This entity is responsible for delivering common services required by the AEs. It can handle tasks like data management, device management, and security services. It can be not only located in various places, including an IoT device, a gateway, or a network server but also be an MEC application.
3. Network Services Entity (NSE): The NSE offers services to the CSEs beyond just data transport. It may include network resources such as DNS services or NTP servers.

In the oneM2M standard for IoT device management and communication, these logical entities can be deployed in various types of nodes to play distinct roles within the architecture. Application Service Nodes (ASNs) host application services and instances are responsible for handling application-level data. They can interact with other ASNs or Middle Nodes (MNs). Application Dedicated Nodes (ADNs) are specialised nodes tailored for specific applications or services, usually with a more limited scope than that of ASNs. MNs host CSEs and facilitate communication between ASNs and Infrastructure Nodes (INs), playing a crucial role in routing and service discovery. Especially, MNs can be deployed not only in gateways but also in edge nodes. This enables them to substitute for some of the functions, data, and services of the cloud IoT platform, providing these directly to devices. Furthermore, by pre-processing data collected from numerous IoT devices and then delivering it to the cloud IoT platform, the MN plays a crucial role in edge computing by reducing factors such as the size of the transmitted data. IN, which is oneM2M terminology for the cloud IoT platform, provides common services and functionalities to support IoT applications. These services and functionalities may include registry, discovery, data management, security, and communication (among other things). INs can be deployed at various locations based on the requirements of the system and the specific use. They can be located in cloud servers to provide centralised services. INs and MNs can be deployed at the edge of the network.

4.2. MEC Architecture

The MEC architecture, a simplified network architectural concept illustrated in Fig. 2b, introduces cloud computing capabilities at the edge of a mobile network. The fundamental components of the MEC architecture are the following:

1. Host: This represents the edge server tasked with running the MEC applications and providing it with the required resources.
2. Platform: The MEC Platform is in charge of managing application lifecycles, endowing them with crucial capabilities such as radio network information and supervising their interaction with the network.
3. Applications: These refer to the applications that operate on the MEC Host and exploit the capabilities of the MEC Platform. MEC applications can vary from content caching and delivery to IoT applications and mission-critical services.
4. System Level Management: This component is accountable for managing MEC system resources, services, and application rules.
5. Virtualisation Infrastructure: This provides the hardware and virtualisation layer on which the MEC applications, platform, and host operate.

The MEC architecture promotes a collaborative ecosystem among telecom operators, service providers, vendors, and third-party developers. By relocating processing capabilities to the network edge, MEC enables real-time, high-bandwidth, low-latency applications and services to be situated closer to the end-user, thereby generating countless opportunities for service and application innovation.

To support IoT services, ETSI MEC has outlined a series of IoT-specific APIs that facilitate the deployment and operation of IoT devices in need of additional support within an MEC environment. This additional support may be necessitated by various factors, such as security constraints or limited power, computing, and communication capabilities. As shown in Fig. 2b, the introduced MEC architecture offers a

framework for integrating heterogeneous IoT platforms into the MEC system and exposes IoT APIs to enable the configuration of the various system components. ETSI MEC aims to host multiple IoT services across different platforms and, therefore, defines functions such as the discovery of IoT platforms, provisioning of IoT devices, and the routing of messages between devices and platforms.

4.3. Synergised oneM2M-MEC Interworking Architecture

In Section 4.2, we described the architecture of the ETSI MEC system, where MEC applications can both consume and produce services. These services are then made available by the MEC platform to other applications. In this context, building on the oneM2M architecture depicted in Section 4.1, the following mapping of oneM2M entities with ETSI MEC entities can be considered:

1. The oneM2M MN, acting as an instance of oneM2M IN specifically designed for edge computing, can function as an MEC service and/or as a service-producing MEC App instance. This service can handle oneM2M common service functions and data offloaded from the cloud-based oneM2M platform. Furthermore, this oneM2M MEC service can be exposed by the MEC platform for connection to consumer IoT applications.
2. oneM2M IoT devices equipped with 3GPP capabilities can interface with the MEC platform as MEC App.

In essence, architectural interworking between oneM2M and ETSI MEC is achieved by accommodating the oneM2M MN instance for edge computing within the MEC platform. From the perspective of the ETSI MEC system, these instances act as specific MEC services and applications. The oneM2M IoT cloud platform can trigger the instantiation of its data and service functions to a target MEC platform. This is particularly useful when serving IoT applications and devices located close to that target MEC platform. Future work can focus on exploring specific integration points to enhance the synergy between these two systems, potentially including specialized mechanisms for application onboarding, instantiation, additional MEC service APIs, and discovery of MEC platforms.

Fig. 3 provides a conceptual interworking architecture, where the oneM2M edge instance is accommodated in the MEC platform as an MEC IoT service. This oneM2M MN in the MEC host performs the necessary IoT service functions offloaded from the cloud-based IoT platform to support its subscribed IoT devices and services. The MEC platform enables these oneM2M IoT devices to communicate with their corresponding IoT platform services within the MEC host. Therefore, the MEC platform should facilitate IoT platform discovery, device provisioning, and message transport configuration. Architecturally, oneM2M and ETSI MEC are already significantly compatible, enabling joint deployments to support end-to-end IoT services.

In summary, building upon the individual strengths of oneM2M and MEC, we propose an interworking architecture that aligns oneM2M's distributed middleware layer with MEC's edge computing capabilities. This architecture (Fig. 3) allows for dynamic offloading of IoT services from the cloud to the edge. We introduce specialized algorithms and protocols to facilitate seamless data exchange and service migration between cloud-based oneM2M instances and MEC-hosted edge nodes. This integration not only reduces latency but also enhances data processing efficiency, making it particularly suitable for real-time IoT applications.

5. Deployment Scenarios and Interworking Architecture between oneM2M and ETSI MEC

In this section, present various deployment scenarios demonstrating the practical implementation of our integrated oneM2M-MEC architecture. Several deployment options are available for this integration, each carrying distinct technical and business impacts. Each scenario is characterized by unique deployment configurations, ranging from cloud-dominated setups to edge-centric models. We provide mathematical models to calculate the IoT service provisioning time in each scenario, considering factors like processing time at the cloud and edge and data transmission times.

5.1. Deployment Scenarios

Edge computing offers various deployment options for integrating these platforms, each carrying distinct technical and business impacts. These scenarios envision a wide range of deployment configurations, blending cloud and edge elements to optimize IoT service delivery. The deployment scenarios discussed in this section assume seamless interworking between the oneM2M and MEC platforms, enabling IoT devices to benefit from IoT edge computing support via both systems. Furthermore, an equation is provided for each option to calculate the IoT service provisioning time, which refers to the duration from the initiation of IoT services for users or devices to the point where these services are fully operational.

We introduce a novel approach for computing IoT service provisioning time in each scenario, incorporating variables such as processing times at cloud and edge, and data transmission durations. T_{poc} represents the time for processing and service provisioning in the cloud platform, whereas T_{pme} and T_{poe} denote the processing time expended by the MEC and oneM2M edge platforms at the edge nodes, respectively. Because T_{poc} in the cloud deals with storing and processing large volumes of data, it naturally consumes more time compared with T_{pme} and T_{poe}, which handle processing for smaller data sets, thereby requiring relatively less time for data and service processing. T_{dce}, T_{dee} and T_{ded} symbolise the message transmission time between the cloud and edge, between edge nodes, and between edge nodes and devices, respectively. Given the deployment structures of the cloud, edge, and devices, T_{dce} generally has the highest value, whereas T_{dee} and T_{ded} have comparatively shorter message transmission times. Fig. 4 depicts four different deployment scenarios of integrating oneM2M and MEC platforms.

For Option (A), which deploys the oneM2M IoT platform on the cloud and MEC at the edge, the IoT platform resides mainly on the cloud side. This represents a widely used deployment that utilises cloud-based IoT platforms alongside edge computing. Although this setup gains some edge computing advantages through the network and processing, it does not leverage the full benefits of a 100% edge computing environment, as the cloud remains the ultimate data storage and management point. The IoT service provisioning time for Option (A) can be calculated as follows:

$$T_{\text{total\_}a} = T_{\text{poc}} + T_{\text{dce}} + T_{\text{pme}} + T_{\text{ded}} \tag{1}$$

This option requires processing at both the cloud (oneM2M) and the edge (MEC), incorporating data travel times between the cloud, edge, and device. Factors such as network congestion, distance, and others that affect data transmission time must be considered. Consequently, Option (A) must consider the processing time for cloud, T_{poc}, unlike the other options.

Option (B) requires deploying oneM2M and MEC as edge nodes at different physical locations. Unlike Option A, in this setup, all data and information exchanges occur locally. Even though oneM2M and MEC serve different entities, this arrangement is viable in the emerging edge computing market. Similar to the previous option, the provisioning time in this case can be expressed as follows:

$$T_{\text{total\_}b} = T_{\text{poe}} + T_{\text{dee}} + T_{\text{pme}} + T_{\text{ded}} \tag{2}$$

Given that all data and information exchanges are localised, T_{poc} is not a factor, because the processing occurs at the edge. Both oneM2M and MEC conduct processes locally at the edge, with travel time between the two edge locations and from the edge to the device being vital. Travel time is dependent on the distance between the edge or cloud and the device. Consideration should also be given to the network infrastructure and the proximity between various physical edge locations.

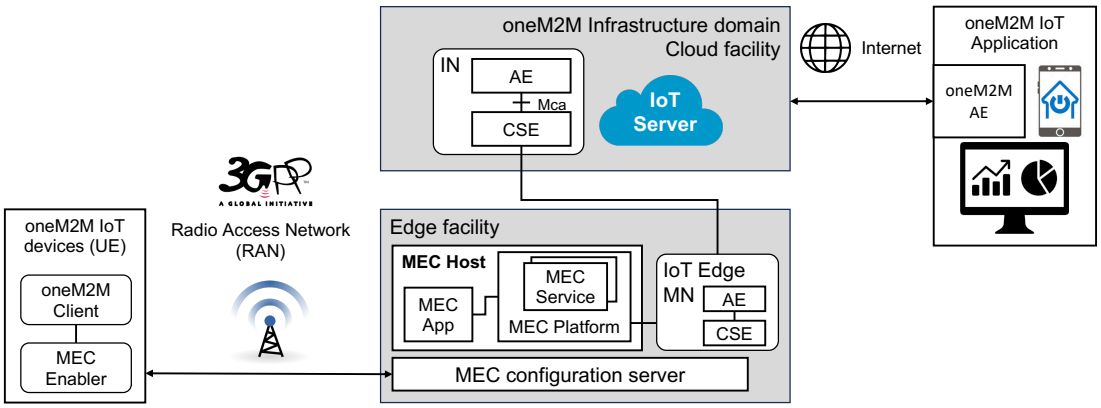

Fig. 3. High-level conceptual architecture to support IoT services via edge computing with oneM2M and MEC.

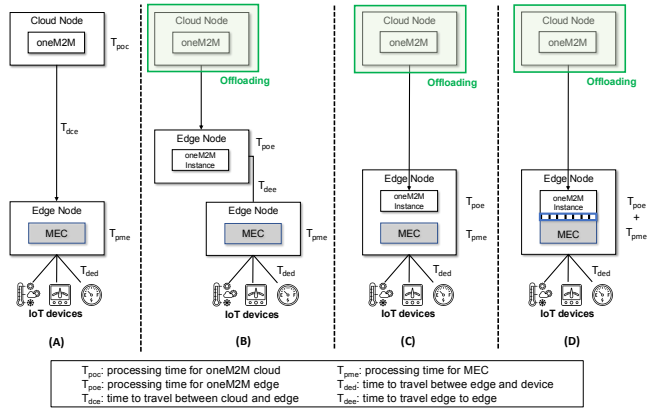

Fig. 4. Four deployment scenarios integrating oneM2M and MEC platforms to support IoT services using edge computing.

Option (C) positions both oneM2M and MEC platforms on the same physical edge node. This configuration can sharply improve service by minimising extraneous data and information exchanges. A service level agreement is necessary between the platform providers, and both platforms must support dynamic deployments to various edge nodes. The time for service provisioning ($T_{\text{total\_c}}$) for this option is the summation of T_{poe}, T_{pme} and T_{ded}.

Option (D) is a tightly coupled scenario where oneM2M and MEC platforms are physically linked via APIs. This facilitates seamless interoperation between the platforms. In this setup, oneM2M functions as an MEC application, fully harnessing all capabilities provided by the MEC platform. The MEC platform can offer data sources, processing, and multi-access networking by hosting oneM2M as an integrated application. Standard activities ensuring interoperability between oneM2M and MEC are essential for this option. The service provisioning time ($T_{\text{total\_d}}$) for this option is also the summation of T_{poe}, T_{pme} and T_{ded} similar to Option (C).

Eventually, Options B, C, and D facilitate oneM2M's processing within an edge computing environment, hence T_{poc} is not relevant for these options. When offloading occurs, consideration of T_{poc} becomes unnecessary. Instead, these options rely on T_{poe}, representing the processing time at the edge, which substitutes for T_{poc}. Owing to the closely-knit nature of Options C and D, a specific function representing the interaction time between oneM2M and MEC via APIs is necessary. Options C and D are similar in terms of service provisioning time because both the oneM2M edge and MEC platform are located on the same edge node. In Option D, the oneM2M edge incorporates additional features that enable it to utilise standard APIs provided by MEC, fostering a closely coupled relationship between the two platforms. With this deployment, the oneM2M edge can leverage network status and quality of service functions through MEC, thereby effectively reducing the service provisioning time with more advanced edge computing functionalities compared with that of Option C. Consequently, we can draw the following conclusion:

$$T_{\text{total\_a}} > T_{\text{total\_b}} > T_{\text{total\_c}} \geqq T_{\text{total\_d}} \tag{3}$$

5.2. *Interworking between oneM2M and MEC*

To facilitate IoT services through interworking between oneM2M and MEC, enhancements to both these standards are imperative. The oneM2M requires improvements to enable the dynamic installation of oneM2M instances on edge nodes operating within the MEC platform. This is crucial for supporting edge computing. This integration involves several enhancements to both standards, focusing on dynamic interaction and utilization of platform capabilities.

oneM2M Enhancements: oneM2M enhancements to use MEC include capability exchange with the MEC platform, edge platform discovery, deployment of oneM2M edge instances, and platform and data migration.

- Dynamic Installation: A mechanism for dynamically installing oneM2M instances on edge nodes within the MEC platform. This includes protocols for real-time deployment and management of oneM2M services directly on edge nodes.
- API Utilization: Our approach leverages MEC-provided APIs to enhance oneM2M functionalities, enabling effective communication and data exchange between oneM2M entities and the MEC platform.

MEC Enhancements: MEC enhancements to support oneM2M edge instances involve capability exchange with the IoT platform, IoT platform discovery, message routing and transportation, IoT platform and device registrations, and platform and data migration.

- Hosting oneM2M Instances: MEC to support hosting of oneM2M edge instances, including protocols for instance management and resource allocation.
- Functionality Support: Enhancements are made for oneM2M edge instance discovery, message routing, and service migration within the MEC environment, ensuring seamless interworking and efficient service delivery.

As depicted in Fig. 5, the oneM2M cloud platform can perform edge computing through interworking with MEC. Initially, to effectively utilise various IoT support functions provided by MEC, enhancements to some existing Common Service Functions (CSFs) and the introduction of additional CSFs for MEC (i.e., MEC CSF) are required. The oneM2M cloud platform creates a oneM2M edge instance for edge

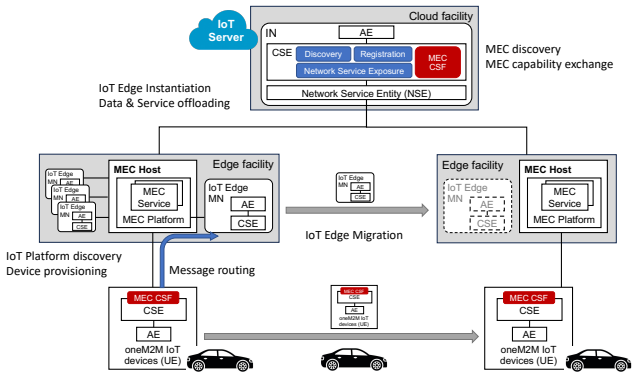

Fig. 5. Interworking architecture between oneM2M and MEC supports the deployment of oneM2M edge instances to the MEC platform for IoT edge computing. This architecture facilitates IoT service continuity through oneM2M edge instance migration services.

computing, incorporating parts of its data and service functions, and performs a discovery process for an edge facility capable of hosting it. During this process, a capability exchange is performed to select an MEC facility suitable for the created oneM2M edge instance from the discovered facilities. Next, the selected MEC platform and oneM2M cloud platform engage in Data & Service offloading through the MEC CSF, facilitating the deployment and activation of the oneM2M edge instance on the target MEC platform. This procedure ensures the efficient operation of services and allocation of resources to the oneM2M edge instance without compromising the MEC system performance or that of other running applications. This procedure ensures that messages received or transmitted from these devices are routed accordingly. With resources allocated and oneM2M devices provisioned to the MEC platform, seamless interworking between the MEC system and oneM2M edge instance is supported, ensuring high-quality service delivery to the end-user. This interworking coordinates processes and services across both platforms, effectively sharing and exchanging information to optimise their collective operation.

Interworking between oneM2M and MEC supports not only static services but also those requiring mobility such as vehicles, unmanned aerial vehicles (UAV), and delivery robots. When such devices are mobile, the initially deployed oneM2M edge instance must move correspondingly to ensure uninterrupted service delivery. In essence, oneM2M MEC interworking must support service mobility, achievable through oneM2M edge instance migration services. The MEC platform, upon detecting device movement, prepares for the migration of the oneM2M edge instance to a neighboring MEC platform. The oneM2M edge instance synchronises with the oneM2M cloud platform for migration purposes, conducting synchronisation regarding services and data executed and stored locally at the edge, and then transmitting information regarding the target MEC facility. The oneM2M cloud platform offloads the updated oneM2M edge instance to target the MEC facility through a provisioning process to complete the migration preparations. Subsequently, the target devices then connect to the new adjacent MEC and receive services through the migrated oneM2M edge instance. Next, the oneM2M cloud platform finalises the platform migration procedure by deleting the now redundant oneM2M edge instance from the previous MEC platform.

Through these enhancements and the proposed interworking architecture, we achieve a robust, efficient, and versatile framework for IoT services in edge computing, surpassing traditional cloud-centric models. This integrated approach paves the way for advanced IoT applications, leveraging the combined strengths of oneM2M and MEC. Also, our architecture offers a robust solution for future-proof IoT service deployment, addressing challenges like latency, bandwidth, and scalability.

6. Edge Computing Deployment and Performance Evaluation

As discussed in Section 5.1, the placement of cloud IoT, edge IoT, and MEC platforms in relation to edge computing deployment plays a crucial role in determining performance. This section presents the experiments on performance disparities incurred in executing services related to IoT edge computing, based on various deployment options of the oneM2M cloud IoT platform and the data it processes.

6.1. IoT Platforms for Edge Computing

In this experiment, the focus was on verifying the performance improvements achievable through edge computing under conditions closely resembling a real-life environment. IoT platforms necessitate a cloud platform designed for IoT and an edge IoT platform operable on edge nodes. There are various open-source IoT platforms developed for cloud use, based on the oneM2M standard, notably Mobius (a JavaScript-based platform developed by the Korea Electronics Technology Institute) and OM2M (a Java-based platform developed by LAAS-CNRS and distributed and managed through the Eclipse Foundation) [17, 18]. In this experiment, Mobius, which has been used in large-scale smart city projects in both Korea and Europe, was selected as the cloud-based IoT platform.

Existing open-source oneM2M platforms typically implement most functions defined by oneM2M and employ databases to store and manage substantial volumes of data. Operating these platforms on resource-constrained edge nodes can be challenging. Consequently, a separate oneM2M-based IoT platform (referred to as TINYIOT), operable on edge nodes, was developed using the C language for this experiment. The TINYIOT platform is a oneM2M-based IoT platform optimised for lightweight deployment on edge nodes. To process data and efficiently provide services from edge nodes proximate to IoT devices and applications, it is imperative to exclude unnecessary functions and enable rapid data processing through direct memory manipulation and hardware control. The core functions and logic of the TINYIOT server were crafted using pure C language, and they exclusively used C language-based open-source components. This included SQLite[2] for database management, cJSON[3] for JSON parsing, lightHTTP[4] for the HTTP server, and wolfMQTT[5] for the MQTT client. In this experiment, the developed TINYIOT was configured to provide IoT services on edge nodes, handling partial data and features from the Mobius cloud IoT platform.

6.2. Deployment Setting and Performance Evaluation

As noted in Section 5.1, the performance of IoT edge computing significantly depends on the proximity of the edge IoT nodes to the applications receiving the services. This experiment aimed to demonstrate this empirically. First, the basic performances of a cloud IoT platform and an IoT platform operating at the edge facility were examined, followed by an experiment to investigate how service provision time varied based on location. The IoT services were assumed to be operating in Seoul, South Korea, whereas the cloud oneM2M IoT platforms providing the services were run on Google Cloud Platform (GCP) located in Iowa and San Paulo. Each cloud IoT platform was allocated four virtual CPUs and 32 GB RAM memory from GCP. Iowa in the United States and San Paulo in Brazil are, respectively, 6,424 miles and 11,397 miles away from Seoul, South Korea. The TINYIOT edge IoT platform was located in Seoul and operated on hardware equivalent to Raspberry Pi 4 Model B, with a Quad-core Cortex-A72 (ARM v8) 64-bit SoC and 8 GB of LPDDR4-3200 SDRAM. For the performance evaluation of IoT edge computing, it was presupposed that a substantial amount of data and services were already operating on both the cloud IoT and edge IoT platforms. Specifically, the experiment assumed that the cloud IoT platform stored ten times more data than that of the edge IoT platform.

[2]https://www.sqlite.org
[3]https://github.com/DaveGamble/cJSON
[4]https://www.lighttpd.net
[5]https://github.com/wolfSSL/wolfMQTT

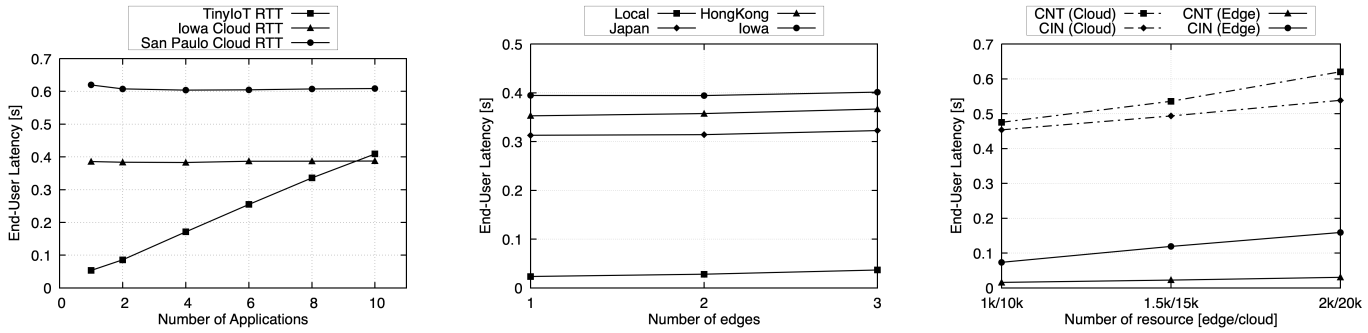

Fig. 6. Experimental results. (a) End-user latency for IoT service request processing: impact of the geographical location of IoT service platforms (cloud and edge) and increased number of supported IoT applications, (b) Impact of the geographical location of IoT edge platforms and increased number of supported edges, and (c) performance evaluation of cloud and edge IoT platforms in executing application and data DISCOVERY requests.

6.2.1. Evaluation based on geography cloud configuration

Fig. 6a presents the results of an experiment conducted to examine the impact of geographical distance on IoT services and to observe the performance benefits obtainable through IoT edge computing. In this experiment, three IoT platforms (two cloud platforms and one edge platform) were configured to receive 120 sensor measurements per second each. The round trip time (RTT) for processing these requests was measured and used as the end-user latency. During this process, the number of applications managed within the platform was increased incrementally to observe the performance variations in IoT services corresponding to the increase in applications.

The experiment was designed to compare two scenarios: one involving a single device sending 120 measurements for processing, and another involving 12 devices, each sending 10 measurements for processing. As illustrated in Fig. 6a, for cloud IoT platforms, the geographical distance from the service provision location proved to be the most significant factor affecting performance, irrespective of the number of supported applications. Because cloud platforms possess sufficient computing power and memory for data processing, an increase in the number of managed applications did not significantly impact end-user latency. In contrast, the TINYIOT edge platform exhibited performance benefits owing to its proximity to the service provision location when managing eight or fewer applications. However, when the number of applications increased to 10 or more (thereby requiring more computing power and memory), the end-user latency surpassed the processing time of cloud computing located in Iowa, USA. This indicates that to maximise the advantages of edge computing, the number of data and services being processed and managed at the edge node must be maintained at a certain level or below.

In addition to the geographical distance of cloud servers, the latency between two devices (device-to-device distance) can also contribute significantly to overall performance. Further investigation is required to analyze how these distances influence latency, particularly in edge computing environments.

6.2.2. Evaluation based on geography edge configuration and increased number of edges

Fig. 6b examines how the geographical positioning of IoT edge platforms and the scale of edge deployments influence the performance of IoT services. It is crucial to analyze how the increase in the number of edge nodes, each supporting multiple IoT applications, affects the overall system latency and data processing efficiency. In this experiment, each edge is connected to three devices, each transmitting 120 sensor measurements. For example, the result with only one edge received 360 measurements, and the three edges received 1080 measurements.

In Fig. 6b, we measure and compare the response times of IoT services when varying the number of edge nodes from a minimal deployment in Figure 6a into a more extensive setup. This is crucial for understanding the scalability of the edge computing model, especially when considering the deployment in expansive geographical areas or scenarios with dense IoT device distributions.

Fig. 6b demonstrate the scalability and robustness of the edge computing model in handling increased loads, with a particular focus on how effectively the oneM2M-MEC interworking architecture with the TINYIOT server manages the growing demand without significantly compromising service quality. This insight is essential for planning and deploying IoT services in diverse environments, ensuring that the edge computing infrastructure can adapt to various scales and densities of IoT applications.

6.2.3. Evaluation based on resource discovery

Discovery operation, alongside IoT data measurement, is a principal task supported by IoT platforms. Fig. 6c depicts the RTT for processing of discovery requests on both edge and cloud platforms, which is a key metric for assessing end-user latency. The amount of data stored on each platform significantly influences the respective service performance during the discovery request processing phase.

The oneM2M platform stores both application-related data (denoted as `Container - CNT`) and distinct data generated by applications (referred to as `contentInstance - CIN`) in the form of resources. During the experiment, both cloud and edge platforms stored an equivalent number of `CNTs`; however, the cloud platform maintained ten times the number of `CIN` resources that on the edge platform. Under these conditions, the experiment measured the time taken to receive responses to specific resource discovery requests. The oneM2M discovery requests are exemplified below:

- `CNT` discovery:
  "Discover Container resource with filtering including filter usage (fu) 1, discovery resource type (ty) is 3 which means container resource, filter operation (fo) 1, 20 of result limit (lim), created before (crb) 20230908T053623, cnt label (lbl), offset (ofst) 0, and 101 of stateTag smaller (sts) which means every container is included in the scope (each container has 100 contentInstance)."
  `GET /TinyIoT?fu=1&ty=3&fo=1&lim=20&ofst=0`
  `&crb=20230908T053623&lbl=cnt&sts=101 HTTP/1.1`
- `CIN` discovery:
  "Discover contentInstance resource with filtering including discovery resource type (ty) is 4 which means contentInstance resource, 20 of result limit (lim), created before (crb) 20230908T053623, and offset (ofst) 0"
  `GET /TinyIoT?fu=1&ty=4&fo=1&lim=20&ofst=0`
  `&crb=20230908T053623 HTTP/1.1`

Fig. 6c suggests that the edge IoT platform exhibited superior discovery performance compared with that of the cloud platform — between 25 to 30 times more efficient for individual data discovery requests and approximately five times more efficient for application discovery requests. The experiment also monitored the processing speed of

discovery requests while progressively increasing resources (1k/10k to 1.5k/15k and eventually 2k/20k) stored on both platforms. An increase in stored data on both cloud and edge platforms caused an increase in end-user latency owing to the increased search space.

Fig. 6c also shows divergent outcomes in the discovery of resources such as Containers (`CNT`) and ContentInstances (`CIN`) across edge and cloud IoT platforms. Especially, the Mobius cloud IoT platform and the TINYIOT edge platform, utilized in the experiments, were developed by different manufacturers but referenced the same standards, specifically oneM2M. Typically, as observed with TINYIOT, discovery should be faster for a smaller number of resource types like `CNT`. However, the cloud platform exhibits quicker discovery times for `CIN` due to internal logic designed to reduce response times for frequently searched applications. Consequently, Fig. 6c demonstrates that even when adhering to the same standards, variations in internal implementation logic can result in performance discrepancies.

In conclusion, implementing edge computing for IoT services delivery markedly enhances performance in terms of data storage, management, and retrieval in comparison with traditional cloud computing approaches. With the incorporation of additional considerations such as network status, access network-specific quality of service, and current processing capabilities through oneM2M-MEC interworking, edge computing is poised to offer even more performance gains, surpassing those observed in this experiment.

7. Conclusion

Conventional IoT platforms provide various functionalities necessary for delivering IoT services from the cloud. However, they are not suitable for services requiring real-time responses or rapid data processing owing to significant delays occurring while communicating with remote cloud servers for IoT service support. Therefore, active research is underway to offer ultra-low-latency IoT services by reducing network load resulting from traffic increases through the application of edge computing technology.

oneM2M, which develops international standards for the IoT, has developed standards for functionalities to enable the design of an edge oneM2M platform (originally intended to operate from the cloud) to support edge computing. This development includes the capability to move some functions and data from the cloud to edge nodes for remote service provision and the ability to dynamically install the oneM2M platform. This study explored various approaches to providing interworking between oneM2M and MEC platforms and examined the performance improvements achievable through these approaches via diverse experiments.

By devising a methodology grounded in real-world standards and demonstrating through experiments the tangible benefits of such an approach—achieving up to a 20-fold improvement in RTT for specific discovery scenarios—our work underscores the critical advantage of standard-based interworking between oneM2M and MEC platforms. This synergy not only enhances resource efficiency but also significantly reduces energy consumption and operational costs, marking a substantial leap toward more responsive and efficient IoT services.

Further studies are required to refine these integration processes and fully understand and exploit the potential of this technological fusion. It is also necessary to explore and address potential security and privacy issues that could emerge from deploying these edge computing systems. Edge nodes, due to their distributed and often less physically secure nature, are susceptible to unauthorized access and data tampering. Additionally, processing sensitive data at the edge increases privacy risks, such as unintended data exposure or misuse. These concerns highlight the need for robust encryption protocols, secure authentication mechanisms, and privacy-preserving techniques like federated learning and homomorphic encryption.

In future work, we plan to address these challenges by implementing and evaluating advanced security measures to further strengthen the resilience of edge computing systems. This will ensure that our proposed framework not only improves performance but also adheres to stringent security and privacy requirements necessary for real-world deployments. Furthermore, as real-world applications and deployments of these technologies increase, we anticipate new challenges and opportunities that could lead to further improvements and innovations.

Acknowledgements

This work was supported by the Institute of Information & Coummunications Technology Planning & Evaluation (IITP)-Information Technology Research Center (ITRC) grant funded by the Korea government(IITP-2025-RS-2021-II211816) and supported by the National Research Foundation of Korea(NRF) grant (NRF-2023R1A2C1004453). In addition, this work is partly conducted at ICTFICIAL Oy, Finland. It is partly funded by the European Union's HORIZON-JUSNS-2023 HE research and innovation program (6G-Path project, Grant No. 101139172) and the Horizon 2020 Research and Innovation Program (aerOS project, Grant No. 101069732). The views expressed are solely those of the authors, and the European Commission is not responsible for any use of this information.

References

[1] N. Hassan, K. A. Yau, C. Wu, Edge computing in 5g: A review, IEEE Access 7 (2019) 127276–127289.

[2] F. Giust, X. Costa-Perez, A. Reznik, Multi-access edge computing: An overview of etsi mec isg, IEEE 5G Tech Focus 1 (4) (2017) 4.

[3] E. Coronado, Z. Yousaf, R. Riggio, Lightedge: mapping the evolution of multi-access edge computing in cellular networks, IEEE Communications Magazine 58 (4) (2020) 24–30.

[4] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, J. Song, Toward a standardized common m2m service layer platform: Introduction to oneM2M, IEEE Wireless Communications 21 (3) (2014) 20–26.

[5] oneM2M, Study on edge and fog computing in oneM2M system. `https://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=32633`, 2020 (accessed 23 august 2024).

[6] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, F. H. Fitzek, Device-enhanced mec: Multi-access edge computing (mec) aided by end device computation and caching: A survey, IEEE Access 7 (2019) 166079–166108.

[7] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, T. Taleb, Survey on multi-access edge computing for internet of things realization, IEEE Communications Surveys & Tutorials 20 (4) (2018) 2961–2991.

[8] T. Taleb, P. A. Frangoudis, I. Benkacem, A. Ksentini, Cdn slicing over a multi-domain edge cloud, IEEE Transactions on Mobile Computing 19 (9) (2020) 2010–2027.

[9] E. G. M. 038, Multi-access edge computing (MEC); MEC in park enterprises deployment. `https://www.etsi.org/deliver/etsi_gr/MEC/001_099/038/03.01.01_60/gr_MEC038v030101p.pdf`, 2022 (accessed 23 august 2024).

[10] X. Wang, R. Li, C. Wang, X. Li, T. Taleb, V. C. M. Leung, Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching, IEEE Journal on Selected Areas in Communications 39 (1) (2021) 154–169.

[11] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration, IEEE Communications Surveys & Tutorials 19 (3) (2017) 1657–1681.

[12] M. Liyanage, P. Porambage, A. Y. Ding, A. Kalla, Driving forces for multi-access edge computing (mec) iot integration in 5g, ICT Express 7 (2) (2021) 127–137.

[13] H. Yu, M. Shokrnezhad, T. Taleb, R. Li, J. Song, Toward 6g-based metaverse: Supporting highly-dynamic deterministic multi-user extended reality services, IEEE Network 37 (4) (2023) 30–38.

[14] L. Zanzi, F. Cirillo, V. Sciancalepore, F. Giust, X. Costa-Perez, S. Mangiante, G. Klas, Evolving multi-access edge computing to support enhanced iot deployments, IEEE Communications Standards Magazine 3 (2) (2019) 26–34.

[15] C. Chen, Y. Zeng, H. Li, Y. Liu, S. Wan, A multihop task offloading decision model in mec-enabled internet of vehicles, IEEE Internet of Things Journal 10 (4) (2022) 3215–3230.

[16] S. Husain, A. Kunz, J. Song, T. Koshimizu, Interworking architecture between oneM2M service layer and underlying networks, in: 2014 IEEE Globecom Workshops (GC Wkshps), IEEE, 2014, pp. 636–642.
[17] J. Yun, I.-Y. Ahn, J. Song, J. Kim, Implementation of sensing and actuation capabilities for iot devices using onem2m platforms, Sensors 19 (20) (2019) 4567.
[18] G. Ihita, V. K. Acharya, L. Kanigolla, S. Chaudhari, T. Monteil, Security for onem2m-based smart city network: An om2m implementation, in: 2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS), IEEE, 2023, pp. 808–813.