Peer-to-peer Smart Lighting System Based on OneM2M Standard

Carlos Costa[a], Daniel Carreira[a] and Fábio Gaspar[a]

[a]School of Technology and Management, Polytechnic University of Leiria, 2411-901 Leiria, Portugal

ARTICLE INFO

Keywords:
Auto Discovery
Lighting system
OneM2M
Peer-to-Peer

ABSTRACT

Smart home automation systems have the potential to revolutionize the way we interact with our homes. However, the widespread adoption of these systems has been limited by challenges such as interoperability, security, and ease of use. To address these challenges, we propose a peer-to-peer network-based smart lighting system that utilizes the OneM2M standard for seamless communication among devices. Our proposed approach eliminates the need for a central hub, reducing potential points of failure and enhancing scalability. It also supports a diverse range of devices and protocols, leading to a more comprehensive and flexible smart home automation system. We demonstrate the feasibility and effectiveness of the proposed approach through a proof-of-concept implementation of a peer-to-peer smart lighting system operating on Wifi. Our paper includes a detailed description of the system architecture, and communication flow and also presents a thorough evaluation of the proposed system's performance and efficiency, highlighting its scalability. Our contributions include the design and implementation of a lighting system based on the OneM2M standard, which offers a user-friendly scalable solution to the challenges faced by traditional smart home automation systems.

1. Introduction

As the world becomes more connected and the number of IoT (Internet of things) devices increases rapidly [1], the potential applications for these devices in our daily lives continue to expand [2, 3]. One of the most promising applications of the IoT is in the realm of smart home automation systems [4, 5, 6], which have the potential to transform the way we interact with our homes. With the ability to remotely control and monitor various aspects of our homes, smart home automation systems offer unparalleled convenience and efficiency [7]. The widespread integration of smart home automation systems has fundamentally transformed our home interactions, offering the convenience of remote access from anywhere and at any time. These systems have the capacity to learn from repetitive patterns through artificial intelligence algorithms [8, 9, 10, 11], further enhancing their capabilities.

When idealizing a solution, there are two main architectural approaches, centralized and peer-to-peer (P2P). The predominant architecture in IoT networks is often centralized [12], where a designated device, commonly referred to as a thing controller or gateway, is responsible for managing the network. P2P [13] follows a decentralized architecture, and it is becoming more popular in the setting of smart homes. This P2P strategy has a number of benefits, including high scalability and modularity, which completely eliminates the single point of failure (SPOF) issue. This flexibility makes the smart home ecosystem more adaptable to changing customer demands by enabling the seamless integration of additional devices and capabilities.

The lack of interoperability is a significant obstacle in IoT environments, as it hinders seamless communication, data exchange, and effective collaboration among different devices, systems, and platforms. The adoption of a standardized communication protocol is an important aspect of achieving this integration. OneM2M [14] has emerged as a pivotal solution that addresses the challenges associated with the machine-to-machine (M2M) communication protocol [15], providing a common language for IoT devices to exchange information and interact with each other. Its primary objective is to enable collaboration between heterogeneous devices and platforms, regardless of the underlying communication technologies and network infrastructures. One of the significant advantages of the OneM2M standard is its ability to support multiple transport protocols, including HTTP, CoAP (Constrained Application Protocol), MQTT (Message Queuing Telemetry Transport), and WebSocket. This flexibility allows IoT devices to utilize the most appropriate protocol for their specific communication needs. The OneM2M standard encompasses various aspects, but three key components to highlight are application entities (AEs), containers, and content instances. AEs represent logical entities responsible for providing services within the IoT system, while containers act as virtual repositories that organize related content instances. Content instances, on the other hand, encapsulate individual data units within containers and hold valuable information such as sensor data. This hierarchical structure facilitates efficient data management, and manipulation in the IoT ecosystem, promoting seamless integration and interoperability within the OneM2M framework.

Our proposed approach centers around the development of a P2P network specifically designed for a smart lighting system. This network architecture enables seamless communication among the various components without the need for centralization. In our P2P network, each component, including the smart switches and bulbs, assumes the roles of both clients and servers, allowing them to directly communicate with one another. The adoption of the OneM2M standard

2220662@my.ipleiria.pt (C. Costa); 2220680@my.ipleiria.pt (D. Carreira); 2220660@my.ipleiria.pt (F. Gaspar)
ORCID(s): 0000-0000-0000-0000 (C. Costa); 0000-0001-8785-6001 (D. Carreira); 0000-0000-0000-0000 (F. Gaspar)

serves as a fundamental aspect of our approach, establishing a unified platform for devices to interact and exchange information. In terms of the communication medium, we have chosen to leverage Wi-Fi technology due to its inherent resilience and extensive coverage, which ensures reliable connectivity within the system. To validate the feasibility and effectiveness of our proposed approach, we have implemented a proof-of-concept smart lighting system. This implementation consists of two bulbs and one switch, with each component operating its own dedicated instance of the OneM2M standard. This design enables coordination among the components, demonstrating the practicality of our proposed approach in a real-world scenario.

The main contributions of this paper include the development of a smart home automation system based on the OneM2M standard. This innovative approach provides a user-friendly, secure, and scalable solution to the challenges that have limited the adoption of traditional smart home systems. The proposed system's performance and efficiency are thoroughly evaluated, and the paper also discusses the limitations and potential future directions of this approach.

To provide a comprehensive understanding of our work, we have structured the remaining paper as follows: Firstly, in Section 2, we will present the current state-of-the-art for the type of problem addressed in this study. Next, in Section 3, we will introduce the system architecture and the communication flow that we have implemented. We will then detail the system implementation 4. Moving on to Section 5, we will describe the experiments carried out and present the results obtained. Finally, we will summarize our contributions and insights in Section 6.

2. Related Work

In this section, we discuss the relevant literature and previous work that addresses the integration of the OneM2M standard and P2P capabilities in the context of machine-to-machine (M2M) communication highlighting the advantages of leveraging OneM2M and P2P technologies to overcome challenges in various domains.

The work in [16] presents an implementation of a Fog Node architecture that is built upon the OneM2M standard. By leveraging the Open Machine Type Communication (OpenMTC) implementation, this architecture enables efficient M2M communication with enhanced P2P capabilities. The main objective of the research is to tackle the challenges faced in collecting sensor data and controlling actuators within Fog Nodes deployed in smart factory environments and Industrial Internet applications. In order to achieve that, they have adapted the OneM2M standard to provide the necessary P2P capabilities, ensuring seamless and effective communication resulting in improved efficiency and reliable communication and performance within the Fog Node architecture.

The article in [17] presents a full M2M application idea based on a decentralized energy management system and P2P networking. Its major goal is to facilitate information interchange and to improve services across several domains. The concept highlights the benefits of a common infrastructure that enables multiple services across distinct M2M application domains while utilizing cost-effective hardware and removing dependency on a central service provider. The suggested paradigm includes both the second generation of M2M applications, which entail interaction between end-users, devices, and the environment, as well as the third generation, which includes social M2M, integration, and networking among people.

The article in [18] proposes a decentralized framework for M2M service platforms, utilizing a distributed system architecture and a P2P network for communication. It introduces graphical application design and a formal language for modeling M2M applications. The architecture eliminates central components and stakeholders, allowing for autonomous and flexible service provision. A proof-of-concept implementation is included to validate the framework.

The paper in [19] explores trust evaluation in decentralized machine-to-machine (M2M) communities, where multiple end-users independently provide or consume M2M application services. While various trust management systems exist in the M2M application field, this paper highlights a common weakness: the lack of a secure method to store computed trust values within the community. To tackle these challenges, the study proposes a novel approach that integrates blockchain technology into the trust evaluation process and introduces the combination of a P2P overlay network to verify trust data storage and for trust verification.

In 2018, Kim *et al.* [20] introduced two oneM2M-compliant M2M/IoT software platforms, Mobius and &Cube which are IoT software platforms that comply with oneM2M standards. These platforms provide a set of tools and APIs for building IoT applications that can communicate with different types of devices and sensors. Since the release of oneM2M multiple implementations have been created for the use of this standard in smart homes, providing interoperability decentralization on the management of devices.

In [21] a new proposal of a smart home deployment that utilizes the oneM2M standard is presented, where the public service entity (CSE) is managed in layers and MN-CSE as the core node for threshold detection. The values are forwarded to IN-CSE for in-AE data operation. The solution reduces device requirements for the home core node MN-CSE and lowers the cost of intelligent home deployment maintenance.

Upon reviewing the related studies presented in Table 1, it becomes evident that none of the prior research has explored the implementation of MQTT subscriptions within a P2P context. Recognizing this gap in the existing literature, our study aims to address this limitation by introducing our innovative approach, followed by the implementation of a proof of concept.

3. Architecture

This section will provide an in-depth exploration of the architecture of our P2P smart lighting implementation, elu-

Table 1
Related work comparison to our study

Paper	OneM2M	P2P	MQTT	Proof of Concept
Santos De Brito et al. [16]	✓	✓		✓
Steinheimer et al. [17]	✓	✓		
Steinheimer [18]	✓	✓		✓
Shala et al. [19]	✓	✓		✓
Kim et al. [20]	✓		✓	✓
Sun et al. [21]	✓	✓		✓
Our study	✓	✓	✓	✓

cidating the operation of the entire system. The existing modules are smart switches and lightbulbs and they are integrated with services and protocols in order to establish connections with one another. The architecture of this solution illustrated in Figure 2, uses the local Wifi network to integrate the modules, each module has its own OneM2M instance running on port 8000, alongside an MQTT Broker on port 1883, has a mean of notifications. Additionally, we added HTTP and WS services, aiming to present a web resource with live data on port 8080.

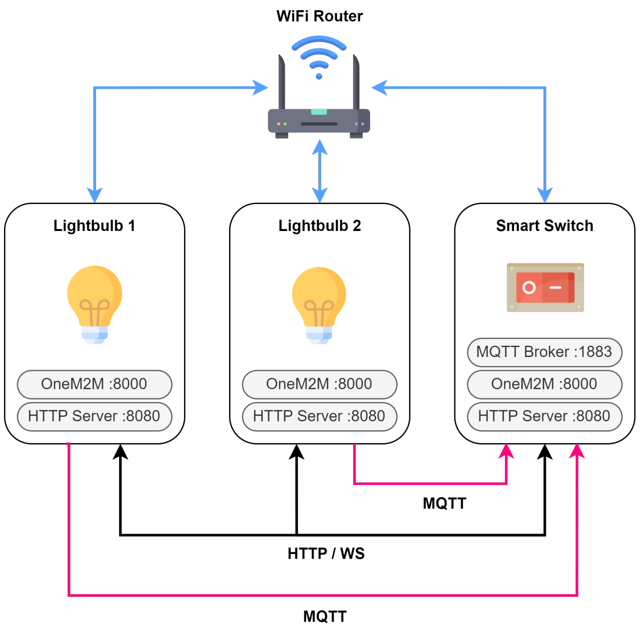

Figure 1: Abstract solution's architecture

3.1. Smart switch

The smart switch plays a vital role in controlling the lightbulbs, serving two primary functions: changing the lightbulb that is being controlled and modifying the state of the associated lightbulb (ON/OFF).

Once a smart switch is turned on, it employs a discovery process, explain in subsection 4.1 to find potential lightbulbs to be controlled. If there are no lightbulbs to be controlled then the switches remain static for potential lightbulbs that try to establish a "connection" with it.

3.2. Lightbulb

The lightbulb serves as an illumination device that receives commands from a controller, the smart switch, which changes the lightbulb illumination state based on the user input. When the lightbulb initializes, a discovery process is started looking for potential smart switches that can control it. If there are no smart switches available, the lightbulb will not associate with any and it will remain in a static state until a new switch identifies and tries to "establish a connection" with it.

4. System Implementation

During our implementation, we discovered that running the OneM2M standard alongside our code base posed a challenge due to the resource requirements. It became evident that microcontrollers with good processing power were necessary as the existing implementations seemed heavy or incompatible in terms of technology or in monetary terms considering the type of system.

To overcome these limitations, we decided to adopt an alternative approach for our implementation. Rather than relying on microcontrollers, we opted to utilize computers solving the challenge as these devices offer significantly greater efficiency and capabilities in comparison. By increasing the computational power, we successfully simulated and evaluated our system's functionality as a fully operational lighting system. This proof of concept demonstrated the feasibility of our approach, although we are aware that such systems are not suitable for this implementation type.

The system aims to establish a peer-to-peer network using the OneM2M standard for each device that makes part of the illumination system (smart switches and lightbulbs). The design allows each switch to exert control over every lightbulb within the network, eliminating the notion of switches being exclusively associated with specific lightbulbs. It is essential that each instance adheres to the standard, for instance, we have employed the OpenMTC [1] implementation and tinyOneM2M [2] implementation (C programming language implementation of the standard).

The solution has been developed using Python[3] programming language version 3.11 along with several libraries, in-

[1]https://github.com/OpenMTC/OpenMTC
[2]http://20.224.164.15/tinyOneM2M
[3]https://www.python.org/

cluding Flask[4] for web page development, backend functionality, and web sockets implementation. For communication and subscriptions with MQTT, we utilized the Paho[5] library and employed NMAP[6] library for the discovery process.

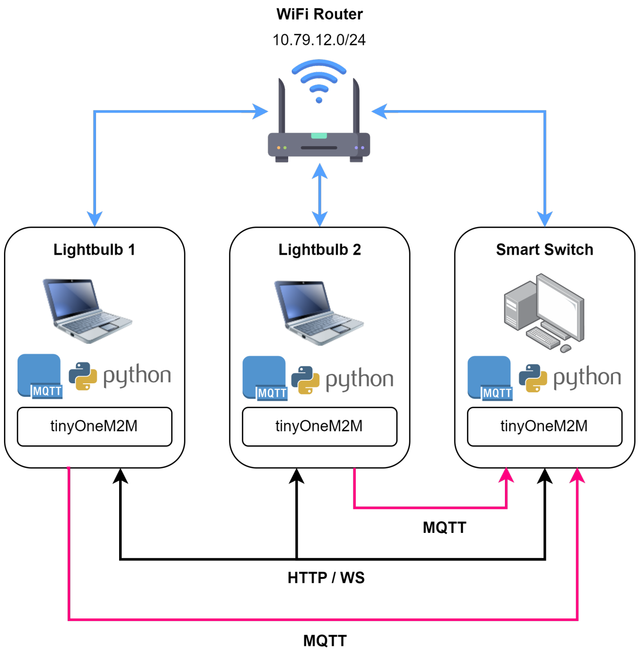

Figure 2: Abstract solution's architecture

4.1. Smart switch

During the initialization process, the default setup for smart switches in their OneM2M implementation involves the creation of an Application Entity (AE) named "switch." Within this AE, two nested containers (CNT) are created: "state" and "lightbulbs."

When a new smart switch is powered on, it initiates a network search using NMAP, enabling the smart switch to detect and identify all lightbulbs present in the network. Once the switch completes the discovery process and detects the lightbulbs, the "lightbulbs" container is used to store their corresponding IP addresses. This information is crucial for future control operations.

On the other hand, the "state" container is utilized to store the currently controlled lightbulb. Since only one lightbulb can be controlled at any given time, the container is specifically designed with a "Maximum Number Instance" (mni) set to 1. This configuration ensures that multiple Content Instances (CINs) of lightbulbs are not simultaneously present in the container, thereby avoiding potential conflicts in the implementation.

To enable our solution to detect changes in its OneM2M resource, we implemented a self-subscription mechanism on each container. By creating a self-subscription, any updates or modifications made to the container's resource within the OneM2M architecture trigger notifications that are received by the solution itself. This allows the web application to stay up-to-date with the latest changes and ensures that the user interface accurately reflects the current state of the OneM2M instance.

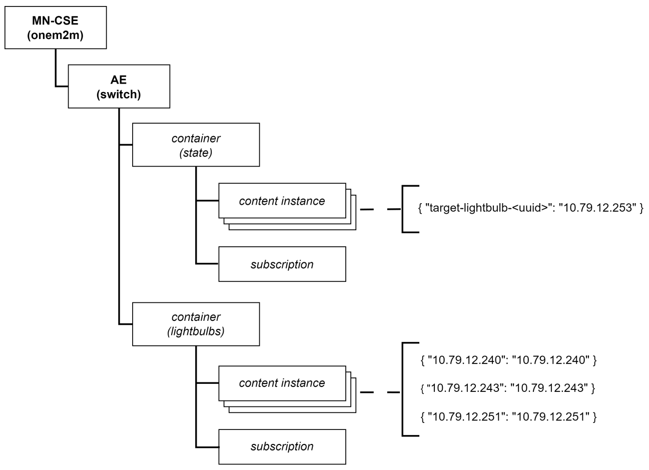

Figure 3: Smart switch OneM2M diagram

4.2. Lightbulb

For the lightbulb initialization process, the setup starts by creating the `AE` "lightbulb" followed by the `CNT` "state". The state `CNT` is designed to store the last fifty state changes (instances when the illumination was turned ON or OFF). This is made possible through the use of the Maximum Number of Instances `mni` field declared on the container. When the number of instances reaches the limit of 50, the oldest instance is automatically deleted to make room for the most recent one, the same happens when the storage size of the instances exceeds 10,000 megabytes thanks to the field Max Byte Size (`mbs`).

In cases where a lightbulb is already functioning before the smart switch, our NMAP scan will successfully locate the lightbulb. However, when a lightbulb is added to the network after the smart switch, it becomes crucial to notify all active smart switches about its presence. To address this requirement, we have implemented a mechanism where the lightbulb itself creates a Content Instance (CIN) in each active smart switch. This CIN is created within the "lightbulbs" container and includes the IP address of the newly added lightbulb. By creating a CIN in each smart switch, the lightbulb effectively notifies all active switches about its existence. This allows the smart switches to subscribe to the newly added lightbulb, enabling them to receive updates and control their state effectively. This approach ensures that all smart switches are aware of the presence of the lightbulb and can establish the necessary connections to facilitate seamless communication and control within the network.

[4]https://flask.palletsprojects.com/en/2.3.x/
[5]https://www.eclipse.org/paho/index.php?page=clients/python/index.php
[6]https://nmap.org

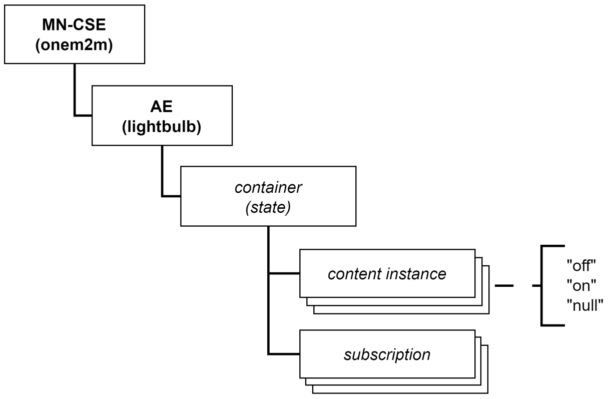

Figure 4: Lightbulb OneM2M diagram

4.3. Web application

Since the implementation didn't rely on physical lightbulbs and switches as mentioned before, we implemented a web application that provides users with a comparable level of control and system visualization.

Each smart switch is associated with its dedicated web page, allowing users to visualize and control the connected lightbulbs. They have the flexibility to switch between different lightbulbs, selecting the one they want to control at any given moment and adjusting its illumination state accordingly. In the images below Figure 5 it is possible to see an example of a smart switch web application.

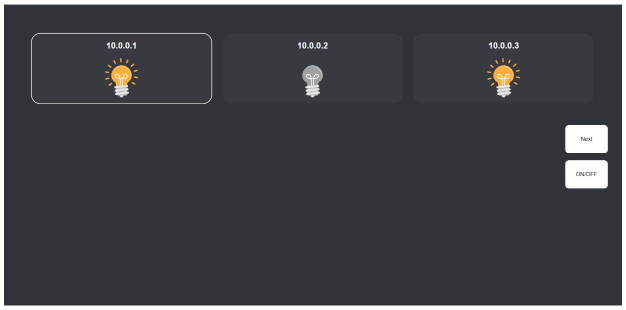

Figure 5: Smart Switch Web application Controller

Furthermore, individual lightbulbs have their own dedicated web pages too, with a straightforward representation displaying the current illumination state, whether it is ON or OFF. In the images below Figure 6 it is possible to see an example of a lightbulb web application.

4.4. Subscriptions

As mentioned previously, the switch needs to be aware of the state of the lightbulbs it controls. To achieve this, the creation of subscriptions was implemented along with MQTT protocol with a mongoose[7] implementation.

After the switch discovers the lightbulbs and saves their information in the container, it sends a request to each lightbulb, instructing it to include the switch's notifiable URL (NU) in its notifications. The NU corresponds to the MQTT server address where the switch is listening for event changes in the topic of the lightbulb's namespace `/onem2m/lightbulb/state`.

Figure 6: Bulb Web application Controller

In the proposed smart lighting system, every message published to the common topic follows a specific format. This format includes three fields: "IP", which represents the IP address of the corresponding lightbulb, "UUID", which is a universally unique identifier for each publication, and "CON," which indicates the state of the lightbulb ("on" or "off").

For example, if a lightbulb is in with an IP address of 192.168.0.100 and a UUID of 123456789, the corresponding resource name would be `lightbulb_192.168.0.100_123456789`.

By using this approach, whenever a lightbulb changes its state, it promptly notifies the switch (which is subscribed to the topic) about the action ensuring the state change.

To ensure that the switch is aware when a lightbulb is removed from the network and is no longer controllable, we implemented signals in our solution. When the user presses CTRL+C to stop the application, the lightbulbs transition to a null state, replacing the on and off states. This triggers a subscription that informs the switch to remove the lightbulb from its list of controlled lightbulbs. It's important to note that this implementation, while effective for testing purposes, may not reflect a real-world scenario where a lightbulb loses power and cannot send such a notification. In such cases, a heartbeat mechanism would be more suitable.

4.5. OneM2M Instances

For the purpose of learning and testing our application, we integrated three distinct reference implementations of the OneM2M standard: ACME[8], OpenMTC[9], and tinyOneM2M[10]. In the following subsections, we will delve into these implementations and discuss the reasons behind our selection.

4.5.1. ACME

Initially, we started with ACME, which offered a straightforward installation process and a user-friendly interface for resource creation. This allowed us to gain a better understanding of how the oneM2M standard operates within a

[7]https://github.com/cesanta/mongoose/tree/master/examples/mqtt-server

[8]https://github.com/ankraft/ACME-oneM2M-CSE

[9]https://fiware-openmtc.readthedocs.io/en/latest/index.html

[10]http://20.224.164.15/tinyOneM2M

web-based user interface. ACME's web UI also included a REST interface, enabling us to send REST requests directly to resources on the CSE (Common Service Entity). However, we encountered specific limitations when using this approach. Firstly, it was necessary additional implement of headers, which introduced complexity to the integration process. Secondly, we experienced difficulties with subscriptions, where they were detected in the broker but were not successfully created within ACME. This discrepancy caused unexpected behavior within our specific solution. Therefore, with this approach, our intention was to implement the core concept of the solution by creating resources without relying on oneM2M subscriptions. Instead, we explored a simpler communication MQTT protocol that involved subscribing and publishing messages to replace the oneM2M subscriptions for testing purposes. Following the implementation and thorough testing, it became evident that a shift to a different reference implementation was necessary.

4.5.2. OpenMTC

OpenMTC solution is an alternative approach. This solution presented two installation options: a Python environment and a Docker image. Unfortunately, the Python environment version proved problematic due to its outdated packages, so we opted for the Docker image instead. Using the Docker image offered advantages such as requiring fewer fields to be filled in when creating resources and eliminating the need for headers (compared to ACME). However, during our implementation, we encountered an issue with the functionality of the created subscriptions, which was also reported by other individuals who used the docker image. So it was also necessary to find a different approach.

4.5.3. tinyOneM2M

The tinyOneM2M was employed for the last approach. Tiny-OneM2M C API is a compact library that offers a straightforward interface for engaging with the OneM2M IoT platform utilizing the C programming language. Unlike OpenMTC, this approach effectively operates with oneM2M subscriptions, making it the definitive selection for implementing our solution and meeting all the requirements.

In addition to utilizing tinyOneM2M, we obtained access to a device capable of running the OpenMTC GitHub implementation. This allowed us to conduct comprehensive testing on both platforms, enabling us to compare and evaluate their performance.

5. Experimental results

To assess the performance and reliability of the system under different workloads, we conducted a series of tests using the scenario depicted in Figure 2. In this setup, we had two smart lightbulbs running on separate laptops, while the smart switch was connected to a desktop computer. All devices were connected to the same network through Wi-Fi. To ensure consistency across different operating systems, we created a virtual environment using the conda tool. This environment isolated the Python version and its dependencies, running Python 3.11 with the requirements specified in the `requirements.txt` file.

Regarding the OneM2M instance, we found that the most widely used implementation, OpenMTC (Python-based), has been discontinued for the past four years. Therefore, we opted to use tinyOneM2M, an implementation based on the C language, in our solution.

To visualize the system, we employed a web page that depicted the switches and lightbulbs. Initially, we had planned to use a Raspberry Pi with an LED light, but due to limited access to the necessary hardware components, we decided to create a virtual scenario instead.

5.1. System on High Demand

To simulate a real-world scenario and evaluate the performance of our proposed solution, we conducted tests on a single-module OneM2M instance. During the tests, we continuously created CIN objects until we reached a predefined threshold. We monitored the system to identify any potential issues, such as crashes or significant performance degradation. The objective was to determine if the module could handle the high request rate without any major issues.

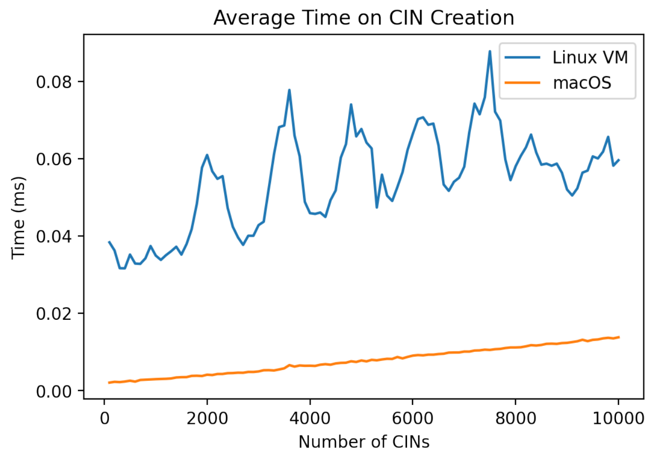

Figure 7: System on High Demand: Average time on CIN creation by device.

In the Figure 7, it is represented the data collected by the continuous creation of 10 000 CIN objects. The results tell us that the performance will degrade for applications with several thousands of CIN objects. For that reason, our solution establishes a hard limit of CIN objects defined in the parent container by a field named *mni*, max number of instances, meaning that the OneM2M instance will only keep the *mni* latest instances. Further, besides the slower performance, in some moments, the OneM2M instance couldn't keep up with the traffic flow, as seen in Table 2, where the best fail rate was achieved by our macOS device with a fail rate of 0.05%.

5.2. Auto-Discover

In a peer-to-peer (P2P) approach, it is essential to have a mechanism that enables each module to locate one another

Table 2
Experiment results of the fail rate on different devices.

Device	Fail Rate (%)
macOS	0.05
Linux VM	0.15

Table 3
Experiment results of the NMAP scan on Class C network.

Device	Avg Time per Scan (s)
macOS	36.63
Linux VM	12.58

and initiate communication. Our solution utilizes NMAP as the primary discovery system, where you specify the IP address and network mask. We employed this approach to locate an instance of OneM2M within the network, specifically targeting port 8000.

Since this process takes place at the beginning of each module, a quick auto-discovery is good to inspire confidence in the system in its user. The time it takes can be given theoretically by Equation 1.

$$\text{Time} = \frac{\text{Hosts}}{\text{Hosts per Second}} \times \text{Scan Duration} \tag{1}$$

The results of the NMAP scan on various hosts are presented in Table 3. We conducted the scan 20 times, and on average, it took between 10 to 30 seconds to complete. This scan was performed on 256 hosts, which is the typical number of hosts in a household scenario with a Class C (/24) network.

It is important to note that when scanning larger networks, such as a Class B (/16) network, the scanning time is directly proportional to the number of hosts. For example, a Class B network encompasses 65 536 hosts, which is 256 times more than a Class C network. Consequently, a full scan of a Class B network would take approximately 1 hour and a half.

Considering the time required for scanning, it becomes evident that the scalability of the auto-discovery process depends on the size of the network. It is crucial to consider the trade-off between the depth of the scan and the time it takes to complete when dealing with larger networks.

5.3. Subscriptions

In a typical setup, each lightbulb maintains its state, and any changes to this state must be communicated to all the smart switches on the network. To assess this functionality, we conducted a test scenario where we examined the impact of creating subscriptions for lightbulbs. Specifically, we focused on evaluating the performance when the lightbulb had 50 subscriptions.

In addition to the subscription evaluation, we expanded the experiment to include the creation of 1000 Content Instance (CIN) objects. We aimed to verify the successful delivery of messages to all the specified notifiable URLs, which, in this case, was an MQTT Broker. Considering the extreme frequency of CIN object creation during the experiment, it is important to note that such rapid content generation would not occur in a real-world scenario.

During the test, we expected a total of 50 000 messages to be delivered. However, only 1 381 messages were successfully transmitted. This result highlights the limitations and challenges that can arise when dealing with a large number of subscriptions and a high volume of data generation and delivery.

It is crucial to consider these findings when designing and implementing systems that involve numerous subscriptions and frequent data updates. Real-world scenarios may require optimization and fine-tuning to ensure efficient and reliable message delivery to all intended recipients.

6. Conclusion

In conclusion, this article presents an approach to the development of a peer-to-peer network-based smart lighting system based on the OneM2M standard. The testing findings demonstrated the system's efficiency and reliability under various workloads, giving important views into the system's capabilities and potential areas for development.

The peer-to-peer architecture adopted offers numerous benefits, including high scalability, modularity, and the elimination of single points of failure by allowing simple integration of devices and ensuring adaptability to changing customer demands.

One notable aspect of the proposed system is the discovery approach implemented for smart switches and lightbulbs which ensures that smart switches and lightbulbs do not solely rely on their own efforts to establish connections by implementing a discovery process both in smart switches and lightbulbs. Thus, this approach prevents unnecessary continuous discovery and advertisement, which is particularly beneficial for battery-operated equipment since it allows the devices to remain in a static state until a new connection attempt is made, optimizing energy efficiency and overall system performance.

Overall, this peer-to-peer smart lighting system offers exciting possibilities for example for smart homes, making them more interconnected and convenient.

For future work, we plan to conduct tests in a more variable environment by incorporating a larger number of devices. Unfortunately, due to limitations with the current available instances of OneM2M and the unavailability of minor devices such as Raspberry Pis and other microcontrollers, we were unable to perform these tests in the current implementation.

Additionally, we intend to implement a heartbeat mechanism to enhance the system's functionality. This mechanism will prevent unnecessary notifications from being sent to devices that are temporarily unavailable, such as during a power outage affecting the lightbulbs and smart switches.

References

[1] D. Paraskevopoulos, Iot software adoption report 2023, 2023.

[2] J. Li, W. Liang, W. Xu, Z. Xu, Y. Li, X. Jia, Service home identification of multiple-source iot applications in edge computing, IEEE Transactions on Services Computing 16 (2023) 1417 – 1430. Cited by: 1.

[3] H. Korala, D. Georgakopoulos, P. P. Jayaraman, A. Yavari, Managing time-sensitive iot applications via dynamic application task distribution and adaptation, Remote Sensing 13 (2021). Cited by: 4; All Open Access, Gold Open Access.

[4] A. Al-Ali, I. A. Zualkernan, M. Rashid, R. Gupta, M. Alikarar, A smart home energy management system using iot and big data analytics approach, IEEE Transactions on Consumer Electronics 63 (2017) 426 – 434. Cited by: 375.

[5] B. L. Risteska Stojkoska, K. V. Trivodaliev, A review of internet of things for smart home: Challenges and solutions, Journal of Cleaner Production 140 (2017) 1454 – 1464. Cited by: 864.

[6] M. Alaa, A. Zaidan, B. Zaidan, M. Talal, M. Kiah, A review of smart home applications based on internet of things, Journal of Network and Computer Applications 97 (2017) 48 – 65. Cited by: 368.

[7] G. P. Filho, L. A. Villas, V. P. Gonçalves, G. Pessin, A. A. Loureiro, J. Ueyama, Energy-efficient smart home systems: Infrastructure and decision-making process, Internet of Things 5 (2019) 153–167.

[8] A. Nasif, Z. A. Othman, N. S. Sani, The deep learning solutions on lossless compression methods for alleviating data load on iot nodes in smart cities, Sensors 2021, Vol. 21, Page 4223 21 (2021) 4223.

[9] G. Varma, R. Chauhan, E. Yafi, Artycul: A privacy-preserving ml-driven framework to determine the popularity of a cultural exhibit on display, Sensors 2021, Vol. 21, Page 1527 21 (2021) 1527.

[10] A. Zgank, Iot-based bee swarm activity acoustic classification using deep neural networks, Sensors 2021, Vol. 21, Page 676 21 (2021) 676.

[11] A. Churcher, R. Ullah, J. Ahmad, S. U. Rehman, F. Masood, M. Gogate, F. Alqahtani, B. Nour, W. J. Buchanan, An experimental analysis of attack classification using machine learning in iot networks, Sensors 2021, Vol. 21, Page 446 21 (2021) 446.

[12] M. G. dos Santos, D. Ameyed, F. Petrillo, F. Jaafar, M. Cheriet, Internet of things architectures: A comparative study (2020).

[13] L. M. Vaquero, L. Rodero-Merino, Finding your way in the fog: Towards a comprehensive definition of fog computing, Computer Communication Review 44 (2014) 27–32.

[14] C. Gezer, E. Taskin, onem2m standardina genel bir bakis, 2016 24th Signal Processing and Communication Application Conference, SIU 2016 - Proceedings (2016) 1705–1708.

[15] M. Beale, Y. Morioka, Wireless machine-to-machine communication, in: 2011 41st European Microwave Conference, 2011, pp. 115–118. doi:10.23919/EuMC.2011.6102011.

[16] M. Santos De Brito, S. Hoque, R. Steinke, A. Willner, Towards programmable fog nodes in smart factories (2016) 236–241.

[17] M. Steinheimer, U. Trick, W. Fuhrmann, M. Steinheimer, B. Ghita, P2p-based community concept for m2m applications, in: Second International Conference on Future Generation Communication Technologies (FGCT 2013), 2013, pp. 114–119. doi:10.1109/FGCT.2013.6767198.

[18] M. Steinheimer, Autonomous decentralised M2M Application Service Provision, Ph.D. thesis, University of Plymouth, 2018. URL: http://hdl.handle.net/10026.1/11957. doi:http://dx.doi.org/10.24382/1243.

[19] B. Shala, U. Trick, A. Lehmann, B. Ghita, S. Shiaeles, Blockchain-based trust communities for decentralized m2m application services, in: F. Xhafa, F.-Y. Leu, M. Ficco, C.-T. Yang (Eds.), Advances on P2P, Parallel, Grid, Cloud and Internet Computing, Springer International Publishing, Cham, 2019, pp. 62–73.

[20] J. Kim, S. Choi, J. Yun, J.-W. Lee, Towards the onem2m standards for building iot ecosystem: Analysis, implementation and lessons, Peer-to-Peer Networking and Applications 11 (2018).

[21] H. Sun, C. Zhang, J. Si, W. Xu, Smart home system design and implementation based on onem2m, in: 2021 2nd International Conference on E-Commerce and Internet Technology (ECIT), 2021, pp. 344–347. doi:10.1109/ECIT52743.2021.00078.