A Proposed Framework for Context-Aware Semantic Service Provisioning

Wael Haider[1], Hatem Abdelkader[2], Amira Abdelwahab[3]
Department of Management Information Systems, Higher Institute of Qualitative Studies, Heliopolis, Cairo 11757, Egypt[1]
Department of Information Systems-College of Computers and Information,
Menoufia University, Shibin Al Kawm 32511, Menoufia, Egypt[1, 2, 3]
Department of Information Systems-College of Computer Sciences and Information Technology,
King Faisal University, P.O. Box 400, Al-Ahsa 31982, Saudi Arabia[3]

***Abstract*—Web-hosted Internet of Things (IoT) applications are the next logical step in the recent endeavor by academia and industry to design and standardize new communication protocols for smart objects. Context Awareness is defined as the property of a system that employs context to provide related information or services to the user, where the relationship is based on the user's task. Therefore, context-aware service discovery can be defined as utilizing context information to discover the most relevant services for the user. Merging context-aware concepts with the IoT facilitates IoT system developments that depend on complex environments with many sensors and actuators, user, and their environment. The main objective of this study is to design an abstract framework for provisioning smart objects as a service based on context-aware concepts while considering constraints of bandwidth, scalability, and performance. The proposed framework building blocks include data acquisition and management service and data aggregation, and rules reasoning. The proposed framework is validated and evaluated by constructing an IoT network simulation and testing accessing the service in the traditional method and according to the proposed framework and comparing the results.**

***Keywords*—Internet of Things (IoT); Web of Things (WoT); Web of Objects (WoOs); context-awareness; service provisioning; interoperability; ontology; OWL**

I. Introduction

Every object in our environment, including chairs, gas meters, electricity meters, curtains, lights, office equipment, and home appliances, should be transformed into Internet-connected smart objects to improve a variety of application domains (e.g., building automation, healthcare services, smart grids, transportation, and environmental monitoring).

A smart object is defined as an entity that is provided with a sensor or actuator, a microprocessor, memory, a communication module, and a power source. The Lowpower Wireless Personal Area Network (LoWPAN) is a crucial component of the IoT due to its advantageous features such as energy efficiency, widespread accessibility, and the ability to integrate smart objects with the Internet [1].

The IoT has emerged as a transformative force in recent years, connecting billions of devices worldwide and generating massive amounts of data. These connected devices have the potential to drive numerous applications, from smart homes and healthcare to transportation and industry automation. However, the sheer volume and diversity of IoT data often makes it challenging to extract meaningful insights and enhance user experiences.

One way to address this challenge is by incorporating context awareness into IoT systems. Context-awareness refers to the ability of a system to understand and respond to its environment by considering various contextual factors such as time, location, and user preferences. By incorporating context awareness, IoT devices can adapt to their surroundings and provide personalized experiences to users.

The main contribution in this paper, that we propose a novel Context Awareness IoT framework that combines the Context Awareness concept with the IoT concept, considering performance problems related to limited smart objects resources.

A. Internet of Things

In 1999, Kevin Ashton introduced the concept of the IoT while working at the Auto-ID Center at MIT. The research conducted at this center focused on network radio frequency identification (RFID) and sensor technologies [2], in which People and things could provide information about their current state and their surroundings in a much more efficient manner [3]

IoT comprises wireless systems that are compact in size and interconnected with each other. These systems are equipped with computational capabilities and can transfer data over a network without the need for human interaction. These smart objects are identifiable, can be accessed, and can be programmed locally or remotely via the Internet. They are designed to monitor or control smart spaces. The underutilization of the potential of IoT devices in various domains can be attributed to the limited expressivity and high heterogeneity of the commonly employed scenario programming paradigms [4]. IoT plays a vital role in many domain areas, such as home automation [4], elderly care [5], [6], home safety [7], energy efficiency, and preservation [8], [9].

Recently, research has focused on the connectivity of all physical objects and information environments to the Internet to enable a user to access and control things from anywhere and at any time, which coined the term IoT. IoTs enable

smaller, less complex devices to do complex tasks by enhancing their intelligence and connectivity.

IoT is converting smart spaces from hype into reality. IoT lacks standardization at the application level. IoT devices have limited computational power and memory [10].

The number of devices connected to the Internet has grown significantly in recent years. Perceiving reality through digitizing some parameters of interest can provide a massive amount of data. This data is then shared across the network with other devices, applications, and infrastructures. The IoT paradigm is based on this dynamic and ever-changing world. To date, countless IoT-based applications have been developed considering smart Cities, smart roads, and smart industries [9].

IoT systems face many challenges which make them exploit the intrinsic potential of IoT devices, such as high heterogeneity of the devices and protocols, which results in limited interoperability.

The IoT paradigm has the potential to merge the boundaries between physical objects and computational devices through their interconnectivity via the Internet. This interconnectivity holds the promise of delivering user-centric services that consider both the user's context and profile information [10].

For example, traditional regulation of classroom temperature consists of power on the air conditioner set to the desired temperature and letting the internal thermostat do the rest. However, a smart IoT system could find the best way to cool down the classroom temperature based on the integration of context awareness, ontology, and IoT. For instance, by combining data from internal thermometers, current time, online weather services, and the current number of students, it can decide to just open the window instead of simply turning on the air conditioner, thus saving energy. Also, the system will increase and decrease temperature based on the number of students. Additionally, the same system could automatically close the air conditioner when there is no person in the classroom. Therefore, it is required to manage all forms of data and events that are collected from sensors and devices. For example, simple events (such as the door being opened) and activities (such as the professors and students going out of the classroom) can be translated into context information. It could be accessed by a service that monitors the classroom (e.g., the lecture ended) and a notification sent to the person responsible for this classroom.

B. Context Awareness

Context is defined as information that can be used to characterize the situation of an entity in context-aware computing literature. An entity is a person, location, or thing that is related to the interaction between a user and an application, including the user and the application [11], [12].

On the other hand, context awareness is defined as the property of a system that employs context to provide related information or services to the user, where the relationship is based on the user's task. Therefore, context-aware service discovery can be defined as utilizing context information to discover the most relevant services for the user [11], [13].

Proposing a service-oriented architecture SOA to abstract the complexity in the access to smart devices so that the devices can be viewed as a service (thing as a service). This will enable the smart system development team to focus only on their functional requirements instead of device-specific technical details. Context from the web service perspective:

- From the service requester's perspective, context is defined as the surrounding environment affecting the requester's Web services discovery and access.
- From the perspective of the services, context is defined as the surrounding environment affecting Web services delivery and execution.

To achieve successful IoT systems, there is a need to integrate a context awareness system with IoT.

Context-aware solutions face numerous challenges, including managing the heterogeneous and massive amount of data generated by IoT devices. Second, how to store and handle events, as well as infer higher-level activities from a set of simple events. Finally, use the development framework to implement applications across multiple domains [14].

The growth of the IoT created a fragmented landscape with many devices, technologies, and platforms, creating interoperability issues on many system deployments [15].

Interoperability is one of the most significant barriers to promoting IoT adoption and innovation.

C. Interoperability

Nowadays the ecosystem of the IoT is currently facing a lack in terms of interoperability among the various competing platforms that are presently accessible [16].

Semantic Web (SW) technologies, which consist of an open set of recommendations for associating data with their formal meaning, have been demonstrated to be a suitable means of achieving data interoperability in IoT systems [17]. The reason for using SW technologies is its inference capabilities over semantically annotated data.

Similar to Semantic Web's vision for the Web of Linked Data, the literature on deploying Semantic Web technologies to IoT focuses on semantically annotating data from smart objects. The most prevalent method for representing semantics is Resource Description Framework (RDF), which represents knowledge as triples (subject, predicate, object) (for example, [TempSensor105, Value, 25] and [TempSensor105, Location, Lab2]). A set of triples constitutes a graph consisting of subjects, objects, and predicates. The benefit of RDF and graph data models is that new knowledge can be inferred from an existing graph. Using domain knowledge, a system can comprehend, for instance, that the temperature in Lab 2 is 25 degrees Fahrenheit, which is a transitive property. Web Ontology Language (OWL), one of the primary languages (with RDF schema) used to define ontologies on the web, is frequently used to express domain knowledge to perform the annotation on intelligent [18]. Recently, there has been a lot of research into how SW technologies, in particular OWL ontologies, can be used to improve the IoT field's poor interoperability [4].

Several solutions exist to facilitate interoperability among diverse IoT platforms and application domains. One such solution is the Web of Things (WoT) architecture, recently introduced by the W3C consortium [19].

WoT is a paradigm devised by the World Wide Web Consortium (W3C) on top of the IoT concept. It provides standard mechanisms to interact with any type of device from any automatic system using a descriptive JSON file called Thing Description [20].

Rule-based programming approaches are suitable for IoT automation systems due to their simplicity and intuitive use. One of the most common tools used in programming IoT scenarios using trigger action rules is IFTTT which has many limitations, such as low-level abstraction and low generalization [4], [21].

This paper is organized as follows: an introduction is presented in Section I. Section II illustrates some of the related works. Section III presents the general architecture of the proposed framework. Section IV shows the experiments and results. Section V presents the conclusion and future works.

D. Web of Objects

Web of Objects (WoOs) objectifies and virtualizes real-world objects to support intelligent features and provide Realtime data about the physical world by representing them as Web resources, which can be accessed using the lightweight REST-based APIs principles rather than the heavyweight SOAP-based architecture. Any object with a sensor or actuator, CPU, memory, communication, and power source is considered smart. The web is an ideal universal platform for IoT applications because it uses open standards and can be accessed from any device. In the web environment, sensors or actuators can offer their capabilities via a REST-based API (e.g., URI/lightON and URI/light OFF), which enables objects to interact dynamically. Service provisioning is the process of providing smart object services to the web, similar to traditional web services available on the web. Any web application that communicates with smart objects via communication networks and Web standards is referred to as an IoT application on the web [22].

The smart environment comprises sensors, actuators, interfaces, and appliances networked together to provide localized and remote control of the environment. Sensing and monitoring the environment include temperature, humidity, light, and motion. Environment control such as heater and fan ON/OFF control is provided by the actuator having dedicated hardware interfaces and computing capabilities. Localized control is provided by Bluetooth and remote access through WiFi. The RESTful architecture enables interoperability in Smart space WoOs Architecture.

Semantic ontology helps ubiquitous environments address key issues like knowledge representation, semantic interoperability, and service discovery and provides an efficient platform for building highly responsive and context-aware interactive applications.

According to the following reasons, information systems' ability to communicate with smart objects has become more complicated:

- Many hardware devices rely on proprietary protocols to perform their functions.
- Many devices have embedded software that remains constant over their entire lifespan.
- Semantic annotation for the sensors and the services.
- Service discovery and subscription.
- Simultaneous requests.
- Service authorization.
- Web API generation.

II. LITERATURE REVIEW

There have been several studies focused on how to apply Web paradigms and protocols to service provisioning, such as Service-Oriented Architecture (SOA), RESTful service (REST), Semantic-based provisioning, and the WoT.

Sciullo et al. [15] propose a WoT Store, a centralized repository for managing resources and applications on the WoT. While the proposed system has several potential benefits, there are also some limitations and disadvantages to consider. The WoT Store system may rely heavily on centralized infrastructure, which can lead to scalability, reliability, and security challenges. Additionally, the system may require developers to use a specific set of APIs and communication protocols, which could limit the flexibility and interoperability of IoT devices and applications. Nonetheless, the paper presents a prototype implementation of the WoT Store system and evaluates its performance in terms of resource discovery time and application deployment time, demonstrating the effectiveness of the proposed system for managing IoT resources and applications on the WoT.

Iqbal et al. [22] propose an interoperable IoT platform that can be utilized in a smart home system. The proposed platform employs WoO and cloud architecture. The platform under consideration offers the capability of achieving interoperability among a range of legacy home appliances, diverse communication technologies, and protocols. The platform facilitates remote control of household appliances and enables the storage of home data in the cloud for use by diverse service providers' applications and analytical purposes. The article proposes potential areas for further research, including the incorporation of machine learning algorithms, the deployment of a mobile application, and the creation of a security framework for the smart home system. The proposed architecture is extensible to a variety of smart building use cases, including factories, offices, smart infrastructure, etc.

Ibaseta et al. [23] propose a new methodology for the monitoring and controlling of energy usage in constructions through the utilization of WoT technology. The proposed methodology incorporates diverse building systems and devices, such as lighting, occupancy sensors, and smart plugs, through a cloud-based platform that employs web protocols

and standards. The present study showcases a suggested methodology through a case study of a retrofit project undertaken in an office building. The results indicate that the proposed approach is both cost-effective and energy-efficient while also being interoperable and scalable. Furthermore, the study suggests that this approach can be implemented in real-world settings.

Reda et al. [4] propose a knowledge-based approach for home automation systems using IoT devices. The proposed approach aims to achieve greater expressivity and a higher level of abstraction needed to build knowledge-enabled and reasoning-capable home automation systems so that the potential offered by IoT devices can be fully exploited. The paper demonstrates the feasibility and efficiency of the proposed approach in a simulated house environment, and it contributes to the development of home automation systems by providing a new approach that uses web standards and public ontologies to implement well-defined reasoning without the need for ad hoc control programs or ontologies. The paper also suggests several future works, including investigating the scalability of the proposed approach, evaluating it in a real-life setting, comparing it with other existing approaches, extending it to support more advanced reasoning capabilities, and developing a user-friendly interface for configuring and managing the proposed approach.

Gochhayat et al. [10] propose LISA Lightweight Context-Aware IoT Service Architecture for managing and efficiently managing and delivering push-based services in an IoT environment designed to minimize the overhead of communication and processing while using context information such as device capabilities, user preferences, and environmental conditions to provide personalized and efficient IoT services. LISA filters and forwards the most important and relevant services to the users by understanding their context. The paper evaluates the scalability of LISA through simulations that test its ability to handle many IoT devices and services. The results show that LISA can scale efficiently to support many devices and services while maintaining acceptable levels of performance. However, the paper also acknowledges some limitations of LISA, such as limited support for complex services and limited scalability with centralized deployment.

Krati et al. [24] discuss the challenges of maintaining excellent air quality in indoor environments. It proposes a context aware IoT system that collects data, predicts ventilation conditions, and provides the end user with alerts and recommendations. Through a smartphone application, the system notifies the end-user of contextual information regarding the indoor environment and current ventilation conditions. The system can also provide the end-user with recommendations on improving ventilation and reducing indoor pollutant levels, such as opening windows, installing air purifiers, and modifying the HVAC system. Using the ventilation rate calculated with the aid of interior CO2 concentration, multilevel logistic regression is used to define indoor ventilation states using ventilation rate. K-NN classification technique to predict ambient ventilation.

Xue et al. [25] examine the advancement of context-aware information fusion technology in the context of smart libraries, employing IoT situational awareness. This paper presents a comprehensive examination of the present status of smart libraries and context-aware technology while also conducting an analysis of the implementation of context-aware technology within smart libraries. This paper presents a conceptual framework for implementing a smart library, which leverages context awareness to enhance its services. It further proposes integrating context-aware services within smart libraries to optimize user experiences and improve overall functionality. The aforementioned findings propose potential avenues for future research in this field. These include the advancement of more precise algorithms for context-aware information fusion, the investigation of alternative IoT technologies, and the resolution of ethical and privacy issues. In general, the paper emphasizes the potential of the IoT situational awareness technology in improving the efficacy and efficiency of intelligent libraries.

Kim et al. [26] present a middleware architecture for a context-aware system in a smart home environment. To infer high-level contexts from available low-level contexts, the proposed architecture incorporates a profile-applied improved rule-based reasoning algorithm. The context is modeled using OWL and ontology. The experimental result demonstrates that the middleware provides more accurate and quicker reasoning results than the conventional rule-based method. In addition, the context-aware service is also selected using a rule-based algorithm, allowing the service to be readily expanded by adding new service rules to the service rule base.

In this paper, a Service-Oriented Context-Aware Middleware (SOCAM) architecture is proposed for developing and prototyping context-aware services in pervasive computing environments. To develop context-aware services, architecture provides efficient support for acquiring, discovering, interpreting, and accessing diverse contexts. In addition, the paper proposes a formal context model based on Web Ontology Language and ontology to address issues such as semantic representation, context reasoning, context classification, and dependency. The context model and middleware architecture are described, as well as the prototype's performance in a smart home environment [27].

III. PROPOSED FRAMEWORK

IoT service provisioning techniques must take into consideration how to provide service consumers with complete data, including sensor data as well as its context and don't overwhelm the consumer with repeated information.

In addition, the rapid growth of IoT services available to users will result in a significant increase in information overload and network resource consumption. The exponential growth in the number of services available presents a challenge in selecting and providing the appropriate service from the vast array of required services. Therefore, to locate the most pertinent service, it is essential to construct an accurate query that considers the consumer's context.

In this section, we present our proposed service provisioning framework called CSSP, Fig. 1 presents the proposed framework, and we will discuss how to overcome the problems related to integrating IoT and Context-aware systems in a scalable manner which will be described in detail in the following sections.

A. Data Acquisition and Management Service

This layer is the core layer of the framework which is responsible for all operations related to collecting sensor data and storing it locally. This layer has many data collection mechanisms to overcome the limitation of IoT sensors. The purpose of this layer is to hide the details of data collection and protocols used from users; user may be a web developer or mobile developer who interacts with sensors as a virtual object. Our methodology for collecting sensors data is based on three levels:

- Collect sensor data using the suitable protocol and according to the user role (Fig. 2 shows accessing sensor value through CoAP protocol in the Cooja simulator from a web browser).
- Annotate sensor data with context using ontology. For example, Fig. 3 shows how to model the value of temperature using context aware. There are many methods to represent RDF, such as Triples, Shorthand notation, and XML notation, showing the use of Shorthand notation code to model sensor values.

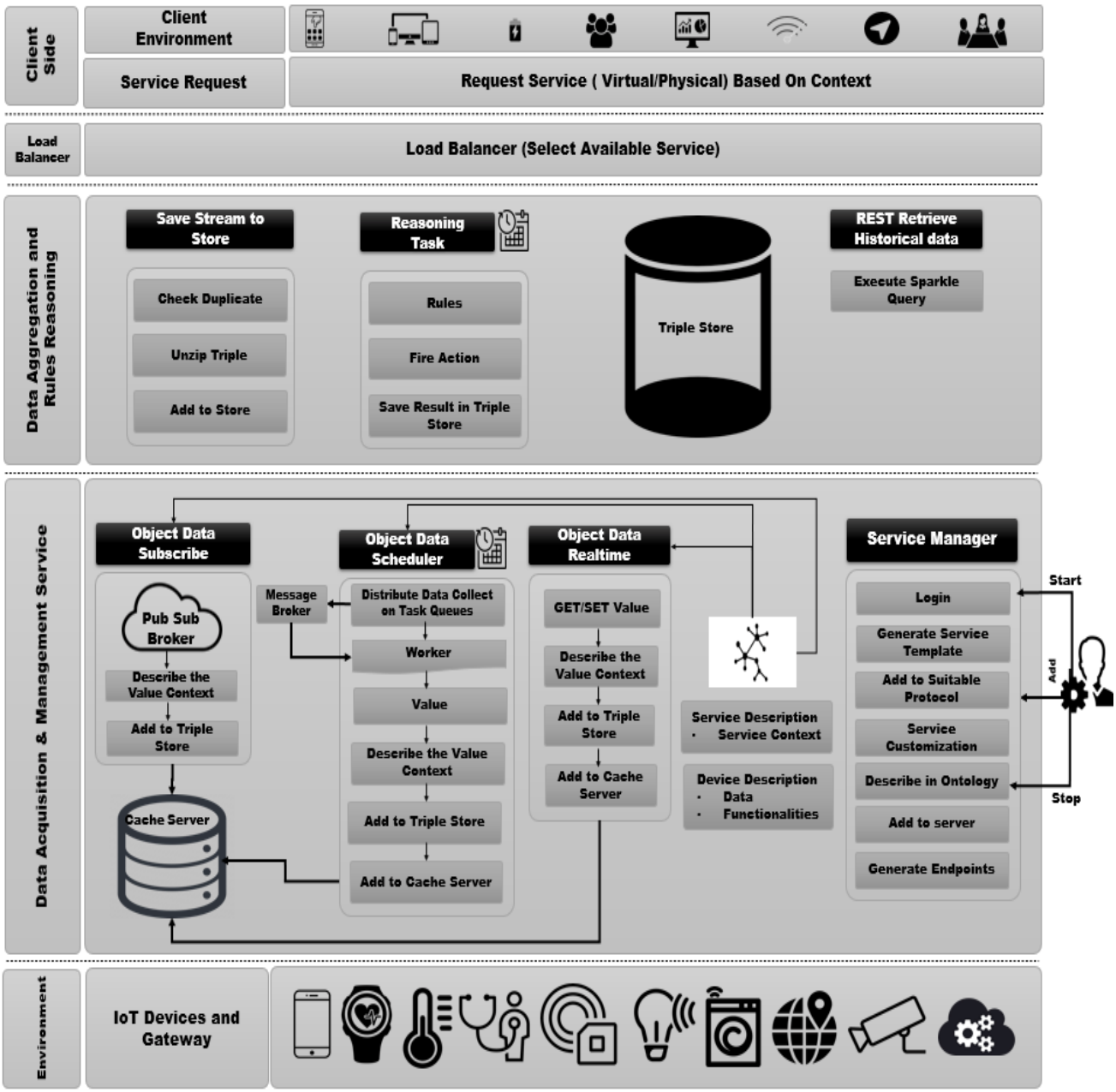

Fig. 1. The proposed framework (CSSP).

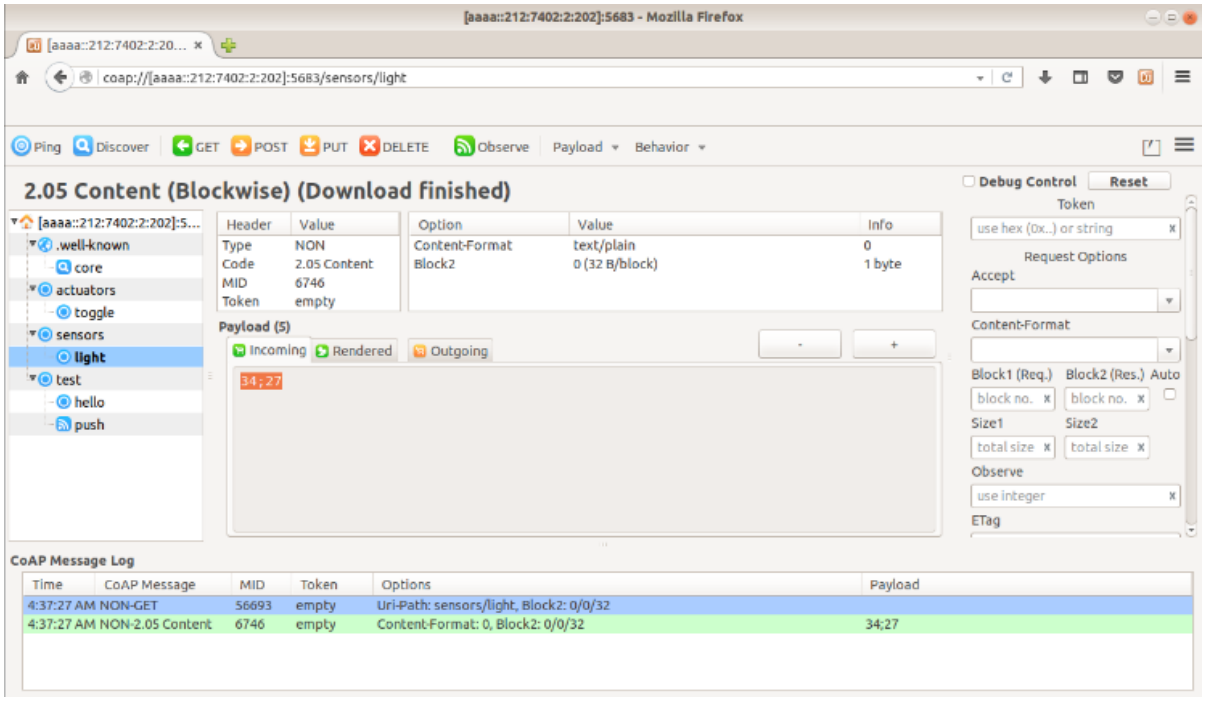

Fig. 2. Accessing CoAP sensor value.

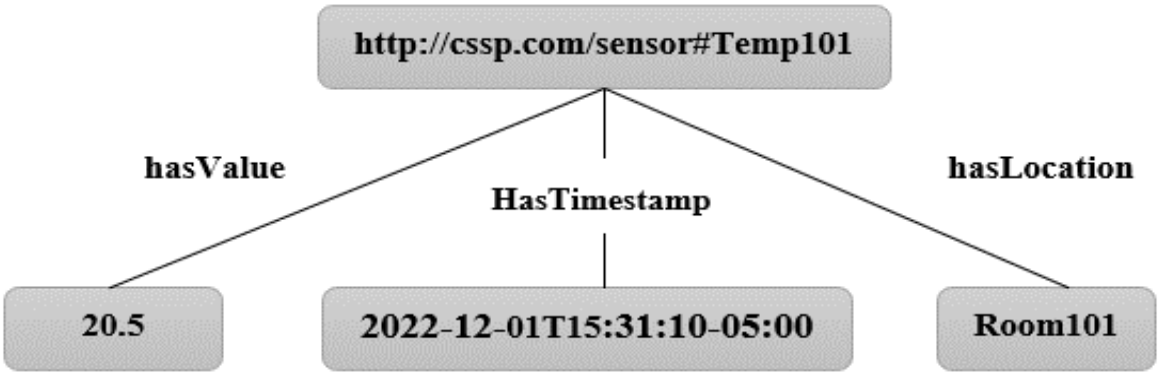

Fig. 3. Temperature sensor value representation using RDF Graph.

The example in Listing. 1 show how to use shorthand notation to model sensor data.

```
@prefix so: <http://cssp.com/sensor#> .
so:Temp101  hasValue   20
so:Temp101 hasTimestamp 2022-12-10T15:31:00Z"^^xsd:dateTime
so:Temp101  hasLocation  Room101
```

Listing. 1. Sensors reading representation using RDF

- Save the generated data in a triple store. This component starts working through the system admin when the admin wants to add a sensor. The admin fills in some data about the sensor, and the component generates an abstract service template according to the specified protocol. The admin customizes the service, such as customizing the result format or any metadata about the result, and how to read the data from pin id, etc.

The service itself is described using an ontology. Every detail about the service is described, such as URL, Endpoints, data format, category, location, etc. Admin can start, stop, and remove any service in the framework at any time. After the admin creates, customizes, and describes the service, the service will be added to the server to allow access to sensor data based on identified mechanism.

In our view of data Acquisition of sensors, it depends on the used protocol, the role of the user, and the status of data (Realtime data or cashed instant data).

We can view the data generated from IoT sensors as the following:

- Object Data Realtime
- Object Data Scheduler
- Object Data Subscribe

All these components depend on ontology to describe IoT devices.

1) Object data realtime: This is the simplest and most direct way to access sensor data which is used by Realtime role users. Based on the type of IoT device, it offers GET operation in case of getting data from the device, else if the device is an actuator, it will have GET and POST operations, Also Mapping of CoAP [28] URI requests to standard web requests.

Data generated from sensors or written to actuators are not meaningful without a description, so we use ontology to describe the context of sensor values such as location, time, and available people.

After the description of sensor data, we add this data to triple store locally and the most recent value to a cache server which can be used by other users who are not concerned with instant data from sensors.

This role is very important in cases of emergencies where milliseconds can be important for patients or the elderly, so we must decrease the request count on sensors for any user.

2) Object data scheduler: This is one of the most important components of the framework and is commonly used by many types of users. This mechanism works with the normal users of the system.

There are many task queues to distribute task requests and categorize task queues for every type of sensor. First, it checks the cash server to see if available recent data is present. If ok, it will return and delete the request. Else, the request will be added to the suitable task queue. After the worker gets the value, the component describes the value, and it is to tribble-store and adds the value to the cache server Scheduler threshold settings are configured using system admin.

3) Object data subscribe: This mechanic is frequently used for monitoring purposes, and it uses different protocols, such as Message Queuing Telemetry Transport MQTT [28] to allow the user to subscribe to a collection of sensor value changes. The layer also adds context data to the value returned and adds it to the tribble store and cache server.

B. Data Aggregation and Rules Reasoning

In the Data Acquisition and Management Service layer, the data is distributed on edges and have duplicates in the data aggregation layer. The framework will do the following:

- Save Stream to Store
- Reasoning Task
- Retrieve Historical Data

1) Save stream to store: In this phase, First, the system checks sensors data and stores only incoming new unique data and ignores repeated values. The service receives only new values to reduce the size of the data. Secondly, extracted common data from ontology was done on the local edges. In this phase, we complete the data of the sensor. We extract metadata or minimize to reduce used bandwidth used to transfer sensor data from edges to a data store. Finally, the IoT value will be stored in the triple store database.

2) Reasoning task: In his phase, we define rules which can be used to infer new knowledge from sensor data. Any rules engine base can be used, such as SWRL rules. When any activity is detected based on rules, sensors, and context data, action is taken and also described and saved in the RDF triple store. The following example in Listing. 2 demonstrates an example of the SWRL rule to regulate air conditioners in a classroom based on current readings from temperature sensors, the number of persons in the same room, and the state of the window.

In this example, hasNumberOfStudents, hasTemperature, and hasWindowState are individual properties that represent the number of students in the classroom, the temperature in the room, and the state of the window (open or closed), respectively. hasAirConditionerState and hasTemperatureSetting are individual properties that represent the state of the air conditioning system (on or off) and the temperature setting, respectively. The SWRL rules in this example take into account the number of students in the room, the temperature in the room, and the state of the window to determine the appropriate state and temperature setting for the air conditioning system. For example, if there are less than 10 students in the room and the temperature is above 25°C, the air conditioning system will be turned on and set to cool the room down to 24°C. On the other hand, if the window is open, the air conditioning system will be turned off regardless of the number of students or temperature.

3) Retrieve historical data: Cause we have a triple store, we can run the system in two methods:

- REST API: this is a simple method and can be used by any developer to call API for getting device data by its id and during a specific period.
- SPARQL Query: this is the standard language used to query from triple store TDB and which will be important in the case of a query for historical data in the TDB.

C. Client Side

The client-side layer is the layer responsible for collecting context data and sending this context with the service request, for example, when the user in a smart home environment wants to turn on the air conditioner, and there are many types of contexts information that can be sent with requests such as the room of the user (location), current temperature from internet service, number of users in the rooms, and preferred temperature from user profile.

D. Load Balancer

The load balancer layer is an optional layer in our framework according to the application size and number of users. This layer is responsible for receiving a request, then finding an available server and routes the request to this server. Load balancers, including physical appliances, software instances, or a combination of the two.

```
hasNumberOfStudents(?classroom, ?numStudents) ∧ hasTemperature(?classroom, ?temp) ∧ hasWindowState(?window, ?state)
        → hasAirConditionerState(?aircon, ?airconState)
∧ lessThan(?numStudents, 10) ∧ greaterThan(?temp, 25) ∧ equal(?state, closed)
        → hasAirConditionerState(?aircon, on)
∧ hasTemperatureSetting(?aircon, 24) ∧ greaterThanOrEqualTo(?numStudents, 10)
∧ lessThanOrEqualTo(?numStudents, 20) ∧ greaterThan(?temp, 25) ∧ equal(?state, closed)
        → hasAirConditionerState(?aircon, on)
∧ hasTemperatureSetting(?aircon, 23) ∧ greaterThan(?numStudents, 20) ∧ greaterThan(?temp, 25) ∧ equal(?state, closed)
        → hasAirConditionerState(?aircon, on) ∧ hasTemperatureSetting(?aircon, 22)
∧ equal(?state, open)
        → hasAirConditionerState(?aircon, off)
```

Listing. 2. the SWRL rule example to regulate air conditioners in classroom

E. Environment

This layer is responsible for the IoT environment we want to monitor and control locally or through the Internet, which has sensors and actuators. This layer needs to use a gateway such as Arduino and Raspberry Pi to control various digital or analog devices.

IV. EXPERIMENTS

To confirm the proposed framework's effectiveness, an experiment has been applied to evaluate the response time in the case of traditional requests and in the case of our framework. We build a simple IoT network in a Cooja simulator [29] and test simultaneous requests for the sensors, then measure and compare the performance of requests for the traditional request and our framework.

1) Dataset: The environment used to generate these datasets was Contiki 3.0 OS (Ubuntu 18.04 Based) in a virtualization environment (VMware application). The device used for generating datasets is a laptop with a Core I5 processor and 8 GB Ram (2 GB for the virtual machine). The Cooja simulator was used to build a simulation of an IoT network.

We built a Python application to simulate the request of the huge simulation for a sensor in case of a CoAP request and proposed HTTP request (500 requests simultaneously) and then saved the details of requests and responses in a CSV file. In the case of a CoAP request, the log file will contain the following attributes: request id, start time, end time, response time, status, and value. In the case of the proposed HTTP, we added an attribute to indicate the source of data, which will be physical sensor data or from a cache server. Two data sets for CoAP and HTTP are generated.

2) Results and discussion: Based on the simulation results, it is observed that an increased number of CoAP requests to the sensor resulted in a significant exponential increase in response time. This effect can be attributed to the absence of scheduling or caching mechanisms, as the direct access of sensor values contributed to longer response times. A sample of requests and their corresponding response times are illustrated in Table *I* (Sample of CoAP Requests Dataset) and Table II (Sample of Proposed HTTP Requests Dataset).

TABLE I. SAMPLE OF COAP REQUESTS DATASET

Request Number	Response Time
145	1.9
404	2.6
147	3.8
474	4.6
48	6.9
281	7.9
70	8.9
442	9.9
419	10.5
41	14.9
112	16.9
78	17.9
126	18.8
289	19.9
204	20.9
346	21.9
475	22.2
98	31.9
250	32.8

TABLE II. SAMPLE OF PROPOSED HTTP REQUESTS DATASET

* C: Cached		* R: Realtime	
Request ID	**Response Time**	**Value**	**Type**
73	0.11	240	C
132	0.15	240	C
347	0.14	76	C
1	0.72	240	R
38	2.16	240	C
364	3.37	122	C
252	4.74	122	C
95	7.13	122	C
10	15.53	54	L

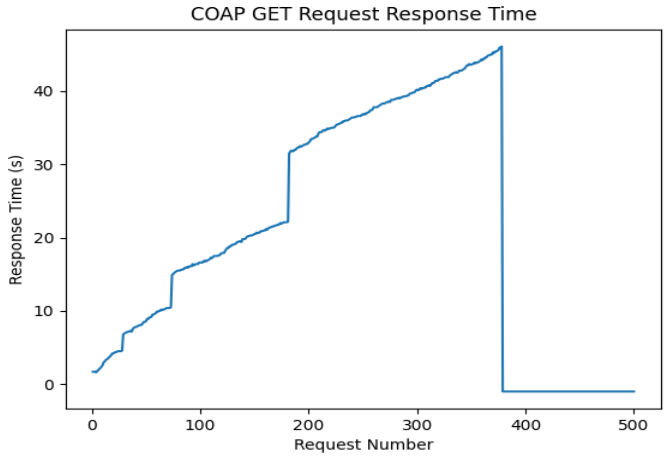

Fig. 4. CoAP response time.

In the traditional approach, the results from our analysis showed that approximately 32% of the requests failed due to this lack of optimization and huge simultaneous requests to the same sensor. The processing time for all requests is 47 seconds. Fig. 4 shows a graph representing the sensors direct access performance through the CoAP protocol.

The implementation of scheduling and caching mechanisms as well as a time threshold established in the system configuration affect the response time in our proposed prototype. As per this threshold, the initial request may experience a relatively longer latency, but subsequent requests are expected to have faster and more consistent response times.

In the proposed approach, the results from our analysis show that the simulation conducted with this prototype revealed a failure rate of 0%. Additionally, the data analysis in the table indicates that a significant portion of sensor data is obtained from the cache server. Consequently, most responses were delivered within an acceptable timeframe, ensuring efficient and reliable system performance. The processing time for all requests is 16 seconds. Fig. 5 shows a graph representing the sensors access performance through the proposed framework based on the HTTP protocol.

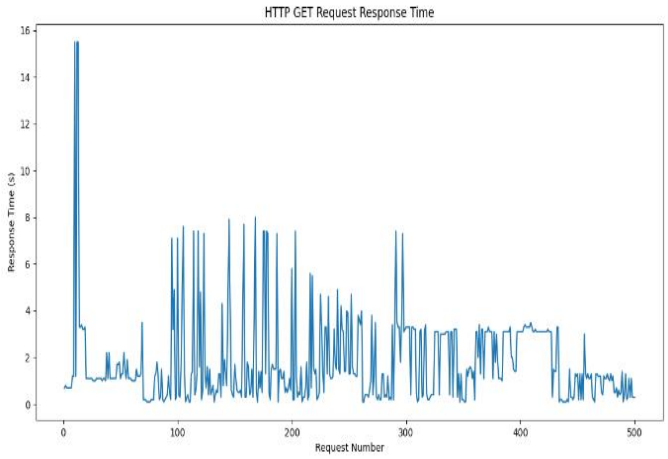

Fig. 5. Poposed HTTP response time.

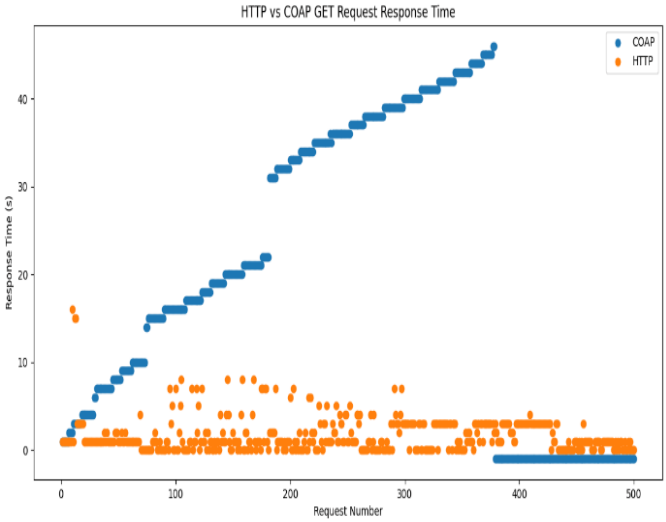

Fig. 6. Performance comparison of CoAP and proposed HTTP.

Fig. 6 shows a graph representing a comparison of performance according to response time to the CoAP protocol (traditional direct access) and HTTP protocol (Proposed framework).

V. Conclusion and Future Works

This paper proposes a context-aware semantic service provisioning framework. The framework focuses on producing solutions for many problems by providing smart devices as smart objects in standard web environments and facilitating the development of any category of IoT application.

The framework was organized to solve many network problems such as bandwidth, scalability, limited resources of smart devices, semantic annotation, and reasoning.

The proposed framework is validated and evaluated by testing accessing the service in the traditional CoAP protocol and according to the proposed framework and comparing the results. The evaluation shows that the proposed framework increases performance and decreases failed requests.

In the future, we will try to test this framework in a real environment, and we will try to increase the performance.

References

[1] S. N. Han and N. Crespi, “Semantic service provisioning for smart objects: Integrating IoT applications into the web,” Future Generation Computer Systems, vol. 76, pp. 180–197, Nov. 2017, doi: 10.1016/J.FUTURE.2016.12.037.

[2] K. Avila, P. Sanmartin, D. Jabba, and M. Jimeno, “Applications Based on Service-Oriented Architecture (SOA) in the Field of Home Healthcare,” Sensors (Basel), vol. 17, no. 8, Aug. 2017, doi: 10.3390/S17081703.

[3] Kevin Ashton, “That ‘internet of things’ thing,” RFID journal, vol. 22, no. 7, 2009.

[4] R. Reda et al., “Supporting Smart Home Scenarios Using OWL and SWRL Rules,” Sensors (Basel), vol. 22, no. 11, Jun. 2022, doi: 10.3390/S22114131.

[5] S. Y. Y. Tun, S. Madanian, and F. Mirza, “Internet of things (IoT) applications for elderly care: a reflective review,” Aging Clin Exp Res, vol. 33, no. 4, pp. 855–867, Apr. 2021, doi: 10.1007/S40520-020-01545-9/TABLES/4.

[6] A. S. Salama and A. M. Eassa, "IOT AND CLOUD BASED BLOCKCHAIN MODEL FOR COVID-19 INFECTION SPREAD CONTROL," J Theor Appl Inf Technol, vol. 15, no. 1, 2022, Accessed: Aug. 22, 2023. [Online]. Available: www.jatit.org

[7] Taryudi, D. B. Adriano, and W. A. Ciptoning Budi, "Iot-based Integrated Home Security and Monitoring System," J Phys Conf Ser, vol. 1140, no. 1, p. 012006, Dec. 2018, doi: 10.1088/1742-6596/1140/1/012006.

[8] V. Marinakis and H. Doukas, "An Advanced IoT-based System for Intelligent Energy Management in Buildings," Sensors (Basel), vol. 18, no. 2, Feb. 2018, doi: 10.3390/S18020610.

[9] M. Lombardi, F. Pascale, and D. Santaniello, "Internet of Things: A General Overview between Architectures, Protocols and Applications," Information 2021, Vol. 12, Page 87, vol. 12, no. 2, p. 87, Feb. 2021, doi: 10.3390/INFO12020087.

[10] S. P. Gochhayat et al., "LISA: Lightweight context-aware IoT service architecture," J Clean Prod, vol. 212, pp. 1345–1356, Mar. 2019, doi: 10.1016/J.JCLEPRO.2018.12.096.

[11] V. Ponce and B. Abdulrazak, "Context-Aware End-User Development Review," Applied Sciences 2022, Vol. 12, Page 479, vol. 12, no. 1, p. 479, Jan. 2022, doi: 10.3390/APP12010479.

[12] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 1707, pp. 304–307, 1999, doi: 10.1007/3-540-48157-5_29/COVER.

[13] S. A. Z. Hassan and A. M. Eassa, "A Proposed Architecture for Smart Home Systems Based on IoT, Context-awareness and Cloud Computing," International Journal of Advanced Computer Science and Applications, vol. 13, no. 6, pp. 89–96, Autumn 2022, doi: 10.14569/IJACSA.2022.0130612.

[14] M. Elkady, A. Elkorany, and A. Allam, "ACAIOT: A framework for adaptable context-aware IoT applications," International Journal of Intelligent Engineering and Systems, vol. 13, no. 4, pp. 271–282, 2020, doi: 10.22266/IJIES2020.0831.24.

[15] L. Sciullo, L. Gigli, A. Trotta, and M. Di Felice, "WoT Store: Managing resources and applications on the web of things," Internet of Things, vol. 9, p. 100164, Mar. 2020, doi: 10.1016/J.IOT.2020.100164.

[16] J. Lanza, L. Sánchez, D. Gómez, J. R. Santana, and P. Sotres, "A Semantic-Enabled Platform for Realizing an Interoperable Web of Things," Sensors (Basel), vol. 19, no. 4, Feb. 2019, doi: 10.3390/S19040869.

[17] D. Andročec, M. Novak, and D. Oreški, "Using Semantic Web for Internet of Things Interoperability," Int J Semant Web Inf Syst, vol. 14, no. 4, pp. 147–171, Oct. 2018, doi: 10.4018/IJSWIS.2018100108.

[18] F. Z. Amara, M. Hemam, M. Djezzar, and M. Maimour, "Semantic Web Technologies for Internet of Things Semantic Interoperability," Lecture Notes in Networks and Systems, vol. 357 LNNS, pp. 133–143, 2022, doi: 10.1007/978-3-030-91738-8_13/COVER.

[19] "Solution for IoT Interoperability - W3C Web of Things (WoT)." https://www.w3.org/2020/04/pressrelease-wot-rec.html.en (accessed Apr. 01, 2023).

[20] S. Murawat et al., "WoT Communication Protocol Security and Privacy Issues," International Journal of Advanced Computer Science and Applications, vol. 11, no. 3, pp. 155–161, 2020, doi: 10.14569/IJACSA.2020.0110319.

[21] "IFTTT - Connect Your Apps." https://ifttt.com/ (accessed Dec. 18, 2022).

[22] A. Iqbal et al., "Interoperable Internet-of-Things platform for smart home system using Web-of-Objects and cloud," Sustain Cities Soc, vol. 38, pp. 636–646, Apr. 2018, doi: 10.1016/J.SCS.2018.01.044.

[23] D. Ibaseta et al., "Monitoring and control of energy consumption in buildings using WoT: A novel approach for smart retrofit," Sustain Cities Soc, vol. 65, p. 102637, Feb. 2021, doi: 10.1016/J.SCS.2020.102637.

[24] K. Rastogi, D. Lohani, and D. Acharya, "Context-Aware Monitoring and Control of Ventilation Rate in Indoor Environments Using Internet of Things," IEEE Internet Things J, vol. 8, no. 11, pp. 9257–9267, Jun. 2021, doi: 10.1109/JIOT.2021.3057919.

[25] X. Chen and Q. Hao, "Research on Internet of Things Context-Aware Information Fusion Technology for Smart Libraries," Sci Program, vol. 2022, 2022, doi: 10.1155/2022/5282932.

[26] H. W. Kim, M. R. Hoque, H. Seo, and S. H. Yang, "Development of middleware architecture to realize context-aware service in smart home environment," Computer Science and Information Systems, vol. 13, no. 2, pp. 427–452, Jun. 2016, doi: 10.2298/CSIS150701010H.

[27] T. Gu, H. K. Pung, and D. Q. Zhang, "A service‐oriented middleware for building context‐aware services," Journal of Network and Computer Applications, vol. 28, no. 1, pp. 1–18, Jan. 2005, doi: 10.1016/J.JNCA.2004.06.002.

[28] S. Bansal and D. Kumar, "IoT Ecosystem: A Survey on Devices, Gateways, Operating Systems, Middleware and Communication," Int J Wirel Inf Netw, vol. 27, no. 3, pp. 340–364, Sep. 2020, doi: 10.1007/S10776-020-00483-7/FIGURES/7.

[29] G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, and N. Tsiftes, "The Contiki-NG open source operating system for next generation IoT devices," SoftwareX, vol. 18, p. 101089, Jun. 2022, doi: 10.1016/J.SOFTX.2022.101089.