sustainability

MDPI

Article

Design and Implementation of an Interworking IoT Platform and Marketplace in Cloud of Things

Faisal Mehmood, Shabir Ahmad and DoHyeun Kim *

Department of Computer Engineering, Jeju National University, Jeju 63243, Korea; faisalavan@jejunu.ac.kr (F.M.); shabir@jejunu.ac.kr (S.A.)

* Correspondence: kimdh@jejunu.ac.kr; Tel.: +82-10-5267-3263

Received: 30 August 2019; Accepted: 17 October 2019; Published: 25 October 2019



Abstract: An internet of things (IoT) platform is a multi-layer technology that enables automation of connected devices within IoT. IoT platforms serve as a middle-ware solution and act as supporting software that is able to connect different hardware devices, access points, and networks to other parts of the value chain. Virtual objects have become a vital component in every IoT platform. Virtual objects are the digital representation of a physical entity. In this paper, we design and implement a cloud-centric IoT platform that serves a purpose for registration and initialization of virtual objects so that technology tinkerers can consume them via the IoT marketplace and integrate them to build IoT applications. The proposed IoT platform differs from existing IoT platforms in the sense that they provide hardware and software services on the same platform that users can plug and play. The proposed IoT platform is separate from the IoT marketplace where users can consume virtual objects to build IoT applications. Experiments are conducted for IoT platform and interworking IoT marketplace based on virtual objects in CoT. The proposed IoT platform provides a user-friendly interface and is secure and reliable. An IoT testbed is developed and a case study is performed for a domestic environment to reuse virtual objects on the IoT marketplace. It also provides the discovery and sharing of virtual objects. IoT devices can be monitored and controlled via virtual objects. We have conducted a comparative analysis of the proposed IoT platform with FIWARE. Results conclude that the proposed system performs marginally better than FIWARE.

Keywords: internet of things (IoT); cloud of things (CoT); Amazon elastic compute cloud (EC2); IoT marketplace; IoT platform; Raspberry Pi; virtualization

1. Introduction

Internet of things (IoT) is the network of physical objects embedded with software, hardware that collects and exchanges data with each other. IoT plays an important role in different domains e.g., transport, education, healthcare, and banking [1,2]. IoT is a promising example that combines heterogeneous and versatile objects with different communication and computational power to gain insight into the physical world of a network. With the advent of IoT, devices can interact with each other remotely using portable devices like smartphones, tablets or computers which has an Internet connection either wired or wireless. Constrained-resource embedded IoT devices are used to monitor and control the smart environment [3]. Network bandwidth and power consumption should not be ignored. Constrained-resource IoT devices and different protocols are introduced that consume less power and low network bandwidth respectively.

Nowadays, developers are building applications to integrate the software with physical devices on a large scale, known as the IoT network [4]. Such applications include smart grids and transportation systems. These applications require real-time controls, large scale data centers, and powerful processing

capabilities. These requirements can be fulfilled by using the cloud of things (CoT). CoT provides the platform for integrating IoT devices with CoT [5,6].

Intelligent service provisioning in the IoT infrastructure [7,8] can be achieved by discovering and composing the most relevant virtual objects. Context-aware systems require learning of the system behavior through the steady association between the virtual object and the real-world object. To meet these challenges, the web of objects (WoO) enables the objects to be deployed, maintained and operated in the IoT through virtualizing the real-world objects with the use of semantic ontology.

In the past developers of scientific applications [9] were limited to choose a platform to run their applications such as local workstations, clusters, supercomputers, and networks. Each of these options offers different compromises in terms of availability, performance, and cost. Recently, cloud computing has become another promising solution for scientific applications and is quickly gaining interest in the scientific community.

Nowadays, IoT is facing many challenges in the virtual world. The current interaction model is based on human–object interaction. One major challenge is to shift the human–object interaction model to the object-object interaction model. In this model, objects will autonomously interact with each other to provide composite services. For this purpose, objects need to be aware of the context in which they operate so that they can execute applications and interact with the surrounding environment. One of the main goals in computing is to integrate the digital world with the physical world. The latest development in the era of integrated devices has led to smart things. Smart things are capable of communicating between digital and physical objects.

A virtual object is the major element of the IoT platforms. It supports a mash-up of different services which leads to the formation of complex applications. Virtual object addresses heterogeneous and scalability issues in the IoT world. IoT encourages physical objects to integrate and communicate with each other and humans for tasks automation. In most cases, the structure of digital objects makes these objects to interact only with certain applications in a particular format. On the basis of issues discussed above, Nitti et al. [10] have proposed a framework that is capable of creating virtual objects regardless of their integration with other applications and business logic.

The IoT platform is referred to as IoT middle-ware, which acts as a mediator between the application layer and the hardware. There are many challenges faced by the IoT platforms such as compatibility and interoperability of different IoT systems, identification, and authentication of technologies, integration of IoT products with IoT platforms, connectivity, handling unstructured data, data security and privacy issues. In this study, we have focused on the integration of IoT products with IoT platforms by introducing the IoT marketplace that provides services to the end-users by interworking with the IoT platform. The main objective of the IoT platform is to overcome the obstacles between layers and combine them to ensure efficient and seamless cooperation. The IoT platform connects hardware to the cloud by using a flexible connection and provides a security mechanism and broad data processing powers. Existing IoT platforms manage all the interaction between the hardware and the application layers. This puts a lot of load on the IoT platform. In the proposed system, the IoT platform is solely used for validation, authentication, registration of physical devices, and initialization of virtual objects. The services provided to the end-users by the IoT platform are separated, which is referred to as the IoT marketplace. The IoT marketplace provides services to the end-users by interworking with the IoT platform. The major contribution of this paper is as follows:

- Design and development of IoT platform and interworking IoT marketplace based on the virtual object in CoT to provide semantic-based discovery and sharing of virtual objects. The details are explained in Section 4.
- Virtual objects on the IoT Marketplace exchange data with the physical devices and it does not have any dependency on systems using different types of operating systems and application languages. The details are explained in Section 4.5.

The rest of this paper is organized as follows. In Section 2, the related work is described briefly. Existing research work in the relevant field is highlighted. Different methodologies and their pros and cons are explained in detail. Section 3 presents our proposed system design and architecture with a detailed description of each component. Configuration of the system is explained in Section 4. The proposed systems implementation and testing are discussed in Section 5. We have performed a case-study and discussed it in Section 6. Section 7 describes the experimental environment of the proposed system. Section 8 presents performance evaluation of the proposed system. The comparison and significance of the proposed system are discussed in Section 9. Finally, the paper is concluded in Section 10 with an outlook on our future work.

2. Literature Review

The number of IoT application deployment has grown in industries like automotive, utility, health, logistics, and home automation at an unprecedented rate in recent years [11]. Due to the increasing number of devices, there is a need to develop cloud middle-ware for managing a large number of sensors and actuators. There are IoT-Frameworks built on open source components to disseminate and process data [12].

A lot of research has been conducted on the interplay of IoT and cloud computing [13]. The main focus of the research is to obtain comprehensive knowledge on the integration of cloud computing and IoT models to fulfill the gap identified during analysis in the existing approaches. Due to the heterogeneity of different objects, IoT and cloud computing emerged to solve the issues [14]. CoT is capable to solve challenging issues such as management, aggregation, and storage for large scale IoT platforms [15,16].

Researchers are moving from cloud computing to edge computing. Edge computing meet the delay requirements of IoT applications. Edge computing is a distributed computing paradigm in which computation is largely or completely performed on distributed device nodes known as smart devices or edge devices, as opposed to primarily taking place in a centralized cloud environment [17,18]. The fog computing architecture network (FOCAN) is a multi-tier architecture in which applications are running on things jointly compute, route, and communicate with each other via smart city environment [19].

Kelaidonis et al. present a framework for the virtualization of real-world objects (RWO) and the management of virtual objects [20]. The framework has three levels of functionality. Each level provides a learning module and self-management which allows smart and flexible applications. The framework summarizes the heterogeneity of a large number of different devices or objects while increasing reliability and making it easier to consider the views of different users. Integrated virtualization has become an effective way to reduce costs, improve software quality and reduce development time [21,22].

Intelligent provisioning of services in the IoT infrastructure can be achieved by identifying and combining the most relevant virtual objects and stand-alone changes to learn the context information of the system behavior through a stable association between the virtual objects and the real objects [7]. To meet these challenges, the web-of-objects (WoO) enables the provision, maintenance, and manipulation of objects in IoT via the virtualization of real objects. The learning model can be represented by a composite virtual object formed by combining several functions of the virtual object [23,24].

IoT research is exploring ways to connect these networks and strengthen intelligence [25]. To facilitate these connections, many low-power network protocols have been developed in recent years in the fields of research and industry [10,26]. However, with more and more connections, it is a difficult task to integrate smarts things into composite applications. To solve this problem, many platforms have emerged in recent years, suggesting an integrated architecture. Some of them are implemented on some devices and are largely incompatible with each other. In addition, due to their complexity and lack of known tools, they are accessible to relatively small expert developer communities, so their use in applications is very limited [27]. The authors proposed an event-driven framework that supports

the collaboration of heterogeneous distributed production resources, according to the principles of Industry 4.0 paradigm by combining various technologies such as industrial IoT, semantic web, and multi-agent-based systems. The proposed system allows the resources to be updated about changes occurred in their context [28].

Computing devices have significant importance in the current environment but researchers and engineers are aiming to exploit the functions of pervasive systems [29]. Pervasive systems provide different types of services that are capable of solving real-time and complex problems. Such systems lack interoperable solutions whereas human-to-machine and machine-to-machine are necessary for ubiquitous computing. Integrating resource-constraint devices into the smart environment and heterogeneous hardware, the network can provide a solution to interoperability issues.

Abdelwahab et al. have proposed CoT for sensing-as-a-service [30]. In traditional CoT, there has been tremendous growth in various domains of IoT in the past. The developers are making IoT based applications to virtualize physical objects into virtual objects. In heterogeneous IoT networks, too many devices and services can be a hurdle for managing the tasks. The authors have presented the concept of service and virtual object management (SVOM) to monitor and control physical devices and their corresponding virtual objects and services in the IoT environment [31].

Normal user lacks the technical knowledge required for the development of IoT applications. Do-it-yourself (DIY) vision has made it possible for a normal user to design smart and customizable IoT applications [32]. The authors in this paper [33] have designed and developed an application that provides a user-friendly interface and visualizes the resources in the form of virtual objects. Basic operations, like drag-and-drop and settings of properties, can be done using virtual objects to create different services.

The Internet is used in many applications to convey time-sensitive messages because of its expanding Internet coverage and easy accessibility [34]. Embedded devices are gaining attention due to the immense use of IoT applications. The main challenge in the development of these applications is the limited capabilities of embedded devices such as power, storage, and memory [35,36]. In Real-Time task scheduling [37], many algorithms like Rate Monotonic and Earliest Deadline First (EDF) compute the hyper-period of input tasks which result in more CPU cycles and power consumption. Many solutions are provided like a resource-aware approach to minimize the hyper-period of input tasks.

Open Intelligent communication platforms are on the rise [38]. The main purpose of the IoT platform is to provide semantically analyzed and enriched data to users or devices according to their needs and context. The semantic events notifier (SEN) enables the involved smart objects to cooperate synergistically based on the shared semantic model [39]. It also supports the fusion of a large amount of data. There are many popular open-source IoT platforms such as OpenMTC [40], FIWARE [41], SiteWhere, and Amazon IoT platform [42]. Middleware for cooperative interaction of things (MinT) is an IoT middle-ware that supports thread pooling that efficiently processes IoT device requests [43].

OpenMTC provides an implementation of the oneM2M standard. It allows the user to create IoT/M2M infrastructure and develop new IoT/M2M applications. In an OpenMTC system, all the entities are represented as resources. OpenMTC provides a mechanism to create, change, update and delete resources. FIWARE is an open-source initiative defining a universal set of standards for context information management which facilitates the development of smart solutions. The FIWARE platform provides a set of APIs and also combines components enabling the connection to the IoT with context information management and big data services on the cloud.

Mineraud et al. have evaluated different proprietary and open-source IoT platform. Evaluation is based on support for heterogeneous interconnected devices, data security and privacy, data processing and sharing. This gap analysis aims to highlight the issues and provide solutions to future IoT platforms [44,45]. Lysis is a cloud-based IoT platform to deploy IoT applications [46]. The main features of Lysis are that each object is an autonomous social agent, platform as a service (PaaS) is fully exploited, data is fully controlled by the users. Cloud-based IoT platforms should have features

like low latency interaction between physical objects and virtual objects, autonomous virtual agents, flexible dynamic services, mobility management to provide services [47].

3. Proposed Architecture of IoT Platform and Control Service Based on Interworking IoT Marketplace

In this section of the paper, we describe the proposed architecture of the IoT platform and the interworking IoT marketplace. Figure 1 shows the proposed architecture of the system. The proposed system consists of three layers: the physical layer, repository layer, and application service layer. The physical layer includes sensors and actuators or any embedded IoT device like Raspberry Pi, Arduino. Sensors are used to detect events from the surrounding environment e.g., BME280 sensor is used to sense temperature, humidity, and pressure from the environment. Actuators are used to control a device based on a command or a signal. Actuators are used to control devices like light, fan, or motor etc. The MQTT protocol is used for communication between the physical and repository layer. IoT devices in the physical layer can publish and subscribe to messages. Sensors publish data to the MQTT broker via MQTT protocol, whereas actuators are subscribed to the message against a unique topic.

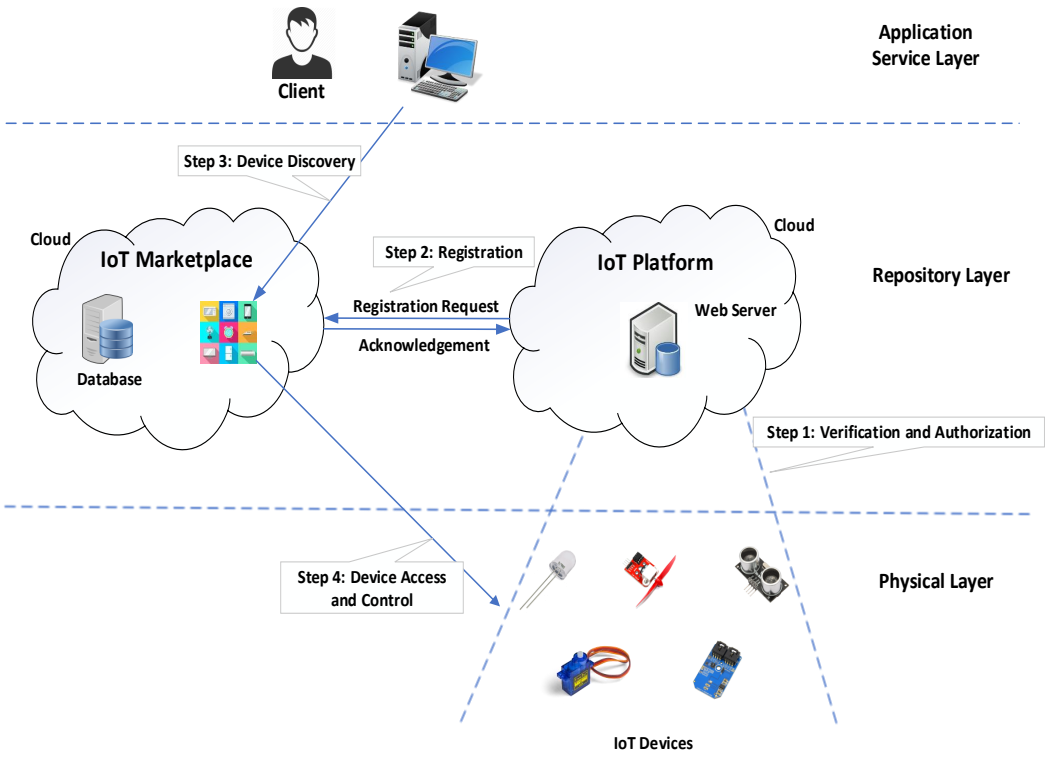


Figure 1. Proposed Internet of Things (IoT) platform and control service mechanism.

Configuration, installation, and deployment of IoT platforms are performed in the repository layer. We have used Amazon web services (AWS) to deploy our application. Web server is running on Ubuntu operating system. MySQL database is used to store the information regarding IoT device registration. Only authorized users are allowed to register the IoT device on the IoT marketplace. The registration process involves the verification of an IoT device. After registration, acknowledgment is sent back to the IoT platform. The visibility mode of the IoT device can be set to the public or private mode by the user. Http protocol is used for communication between the application service and the repository layer.

The application service layer includes a user-friendly web interface to access IoT Marketplace. The application service layer provides a control service mechanism. Users can easily monitor and control registered IoT devices using the IoT marketplace.

Proposed System Sequence Diagram

Figure 2 represents the sequence diagram for the registration of an IoT device on the marketplace. The user accesses the client application and enters the credentials i.e., username and password for authorization. Authorization is required so that the user can retrieve data from the server. After the user is authorized, the server generates access token in response. The user enters IoT device metadata like device name, device type, IP address, MAC address. IoT device meta-data along with access token is sent to the webserver via a POST request for verification. The web server processes the request. If the IoT device is verified it sends back the acknowledgment to the client application. After authorization of user and verification of IoT device, it sends POST request to the IoT Marketplace for registration via RESTful API. At this stage, the visibility mode of the IoT devices can be set public or private. IoT Marketplace processes the request and stores the information in the database. It sends an acknowledgment back to the webserver. Notification is sent back to the client application for successful registration.

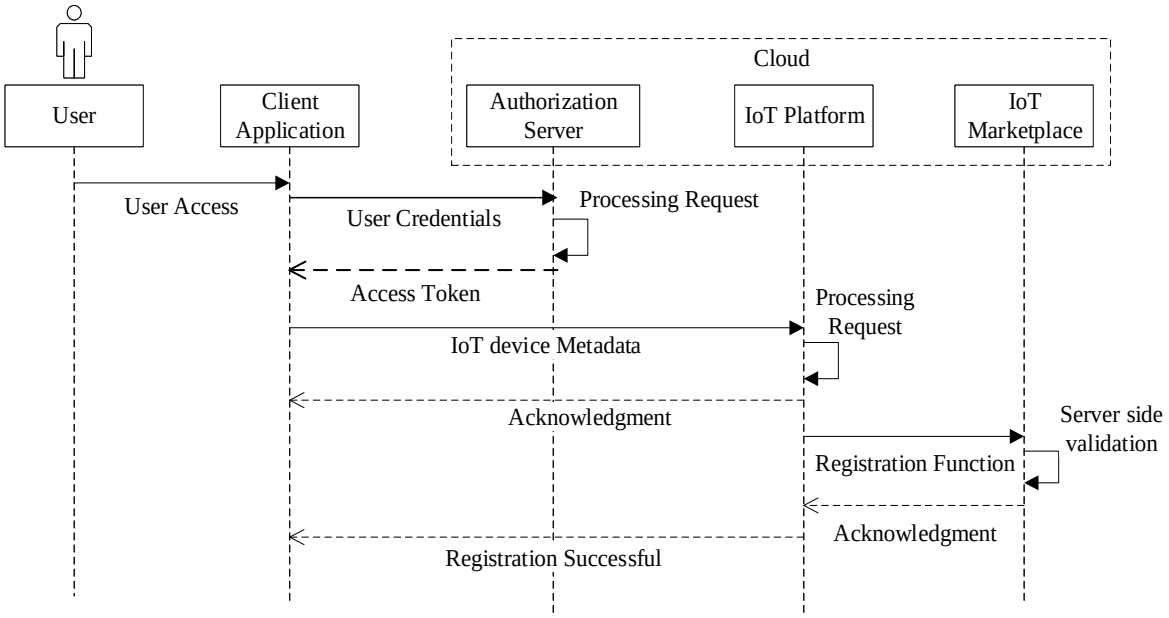


Figure 2. Sequence diagram for the registration of a virtual object.

Figure 3 represents the sequence diagram for the discovery and accessibility of an IoT device. There are two types of visibility mode for IoT devices. Either the IoT device is made public for everyone to use or private to be used only by the owner. In the first step, the user gets access to the IoT marketplace. The user enters its credentials i.e., user name and password to log in. If credentials are invalid it means that the user has entered the wrong username or password. The user is notified with invalid credentials notification. If the user has successfully logged in, a session is created for that user. The user gets a list of the registered IoT devices. IoT devices can be monitored and controlled after discovery. If the user selects a sensor, then it retrieves the values of the corresponding sensor e.g., if the user selects BME280 sensor, it will send temperature, humidity, and pressure values to the IoT marketplace. If the user selects actuator, it will require some action to be performed e.g., turn On or turn Off the actuator. It will publish the command via MQTT protocol to the corresponding IoT device. The IoT device will be subscribed to that message against a unique topic. It will perform the required action and notify the user.

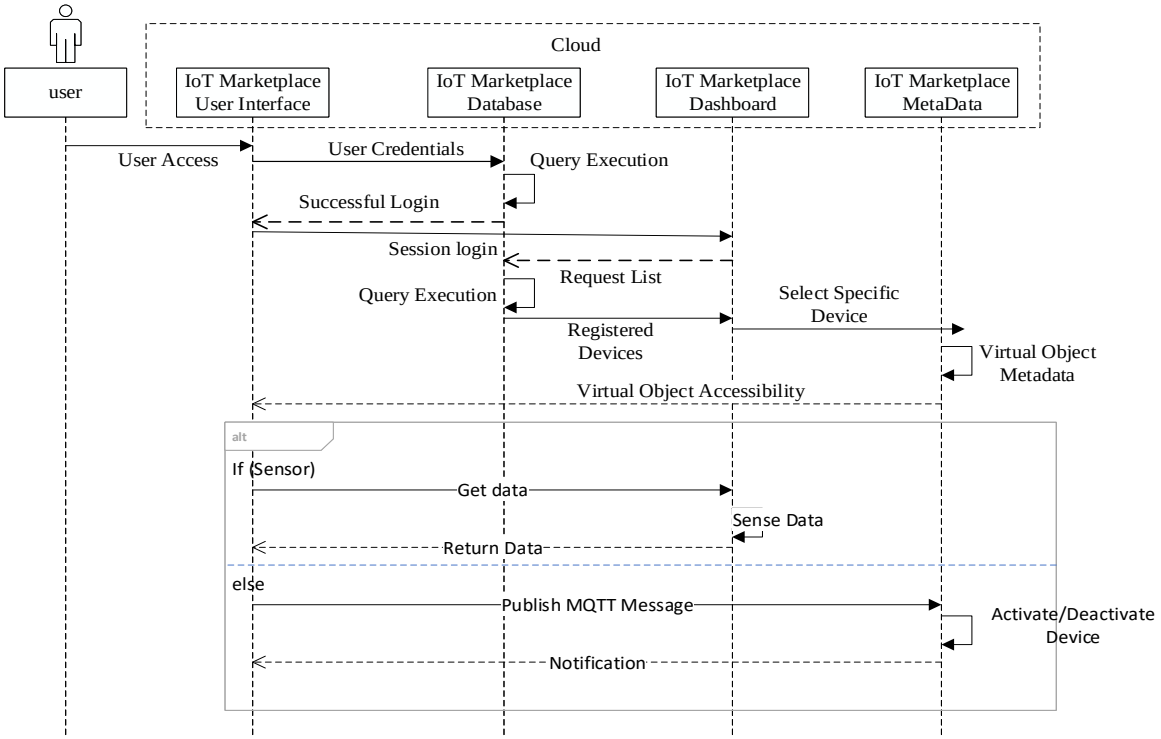


Figure 3. The sequence diagram for discovery and accessibility of a virtual object.

4. Configuration of the Proposed System

The proposed system works in five steps: (a) verification and authorization of embedded IoT devices. (b) Registration of the IoT device on the IoT marketplace. (c) Discovery of IoT device. (d) Access to IoT device via virtual object. (e) Data exchange. Figure 4 shows the working of first four steps. Each step is described in the forthcoming subsections.

4.1. Authorization and Verification

Step 1 includes authorization and verification of the users and IoT devices respectively. Web services could be easily accessed by any user if the URL and input parameter are disclosed to get an output. To keep unauthorized people away from accessing the APIs, the proposed system uses the OAuth 1.0 authentication method for the authenticity of the users at a checkpoint. OAuth 1.0 authenticates the client in three steps. First, the user enters its credentials i.e., username and password, and sends a POST request to acquire a request token. In the next phase, the user will pass the token and token secret as a query parameter to the authorization URL provided by the IoT platform. If the user authorizes, it will generate an access token. The user is granted access to retrieve data from the server with the access token. After the authorization, the user can enter device information like device name, device type, device IP, device MAC address and send a POST request to the web server for verification. The web server validates by checking weather the device information is valid or not.

4.2. Registration of IoT Device

Step 2 includes registration of the IoT device to the IoT marketplace via the IoT platform. The registration request is sent to the IoT Marketplace for a verified IoT device. At this stage, the visibility of the IoT devices can be set to public or private mode. Private mode is for IoT devices used within the home, office for personal usage like monitoring or controlling any appliance. The public mode is for IoT devices to be used anywhere. In public mode, IoT devices can be monitored by the public but not controlled by the public. Only the owner of the IoT device can control the device.

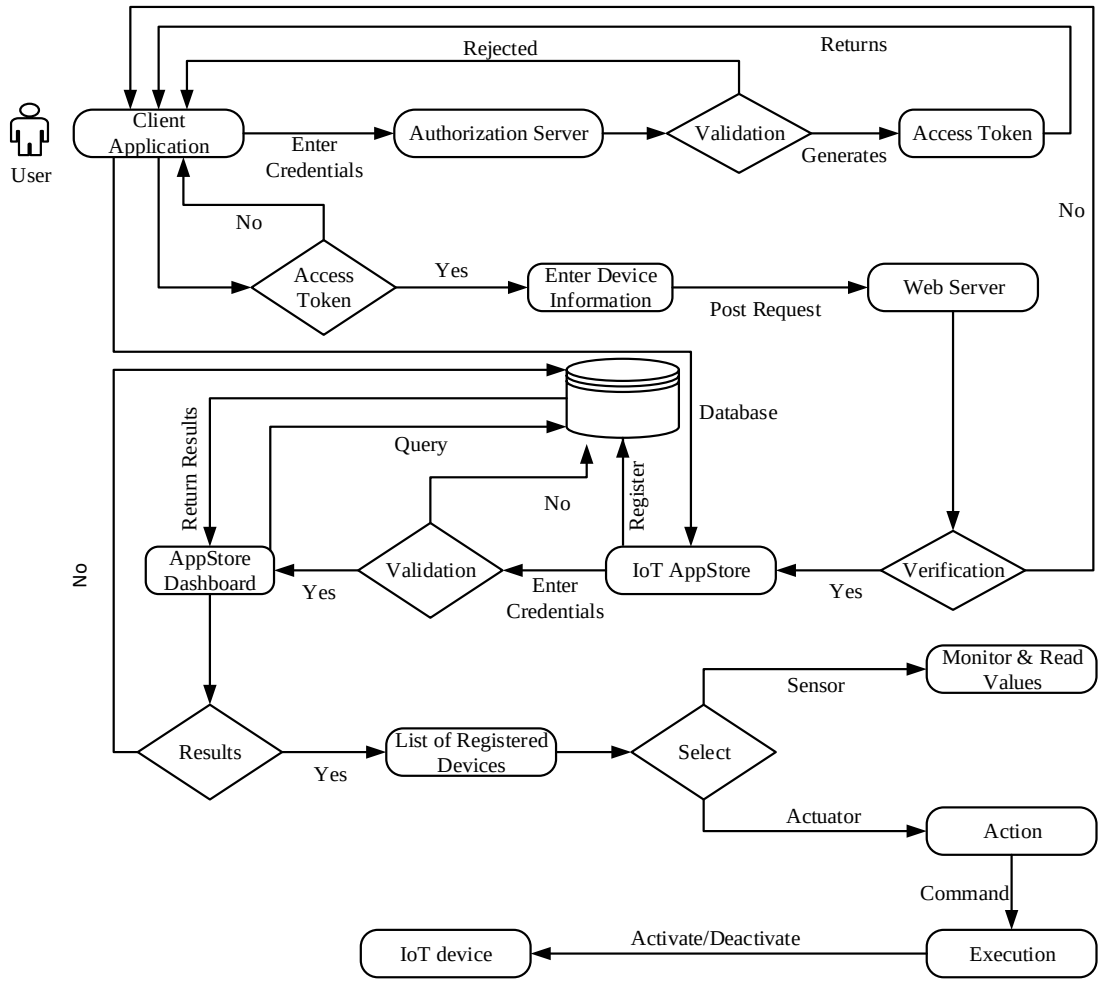


Figure 4. Flow chart of device registration and discovery. of the proposed system.

4.3. Discovery of IoT Device

Step 3 includes the discovery of IoT devices. Users can have access to public mode IoT devices without logging into the IoT Marketplace. To discover private mode IoT devices, the user has to log in to the IoT marketplace. After successful login user is redirected to the IoT marketplace dashboard where users can view registered private and public IoT devices. Users can discover IoT devices by performing a search request. IoT marketplace will return the list of registered IoT devices.

4.4. Access to IoT Device

Step 4 includes the IoT platform control service mechanism. It provides accessibility to the IoT device. Public IoT devices can only be monitored by the public. Public IoT devices do not provide access to the public except the owner. Private IoT devices can be monitored and controlled by the owner after logged In to the IoT marketplace dashboard. There are two types of IoT devices i.e., sensors and actuators. The sensor is a device whose main purpose is to sense events in its surrounding environment and send the information to other devices where it can be used for some purpose. Sensors are used in everyday objects such as temperature sensors, and motion sensors. An actuator is a component of a machine that is responsible for controlling a mechanism or system. An actuator requires a control signal or command and a source of energy to operate. An actuator is a mechanism by which a control system acts upon an environment for a specific purpose. Users can monitor and read the values of the sensor e.g., user can read the temperature from the sensor installed in some environment. Users can also perform specific actions to actuators e.g., turn off the fan, turn on the LED. The command is sent

to the actuator via MQTT protocol for execution. The actuator interprets the command and performs the required action.

4.5. Data Exchange

The resource description framework (RDF) is a framework for describing resources on the web [48]. RDF is designed to be read and understood by computers. RDF is a standard model for data exchange on the web. The main purpose of using RDF in the proposed system is that information can easily be exchanged between different types of computers using different types of operating systems and application languages. The proposed system is capable of resource discovery, resource sharing, and resource data. so we used RDF for discovery, sharing, and displaying the data of the resources on the IoT marketplace. RDF is a terminology used to encode, exchange, and reuse meta-data on the web. It includes information about the physical device such as device ID, device type, device title. The meta-data about the resource can be decoded and displayed in the IoT marketplace. The data can be easily exchanged between servers and clients using RDF/XML via the IoT marketplace. Figure 5 presents the RDF/XML format of BMP280 temperature sensor data. This example shows data being retrieved in RDF/XML format. Data gathered from the surrounding environment is transformed into RDF/XML format. This device has the ID as 'tempbmp28001'. The unit set to retrieve the sensor value is in Fahrenheit and the value of the temperature sensor is 59 Fahrenheit. The data is then decoded and displayed on the IoT marketplace.

```
<?xml version="1.0" encoding="UTF-8"?>
<RDF>
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:i="http://13.209.74.51/jnuappstore#" xmlns:iotdevice="http://13.209.74.51/jnuappstore
/device" xml:base="http://13.209.74.51/jnuappstore">
      <i:SensingNode rdf:ID="tempbmp28001">
         <i:temperature>59</i:temperature>
         <iotdevice:units rdf:resource="http://13.209.74.51/jnuappstore/units/Farenheit" />
      </i:SensingNode>
   </rdf:RDF>
</RDF>
```

Figure 5. The resource description framework (RDF)/XML representation of BMP280 temperature sensor.

Figure 6 shows the graphical representation of BMP280 sensor. It is the graphical representation of RDF/XML used in Figure 5. As explained earlier RDF/XML is used to encode, decode, and reuse the meta-data to display the data on the web or exchange data between server and clients. In this example, there is a root node. The root node is the parent node and all other nodes are children of the parent node. From Figures 5 and 6, we can see that root node is BMP280. All other nodes are the children nodes such as temperature, pressure, and humidity. Each child node has a unique ID. The temperature sensor has an ID of 01, pressure has an ID of 02, and humidity has an ID of 03. Each child node has two child nodes that show the unit and value of that specific sensor.

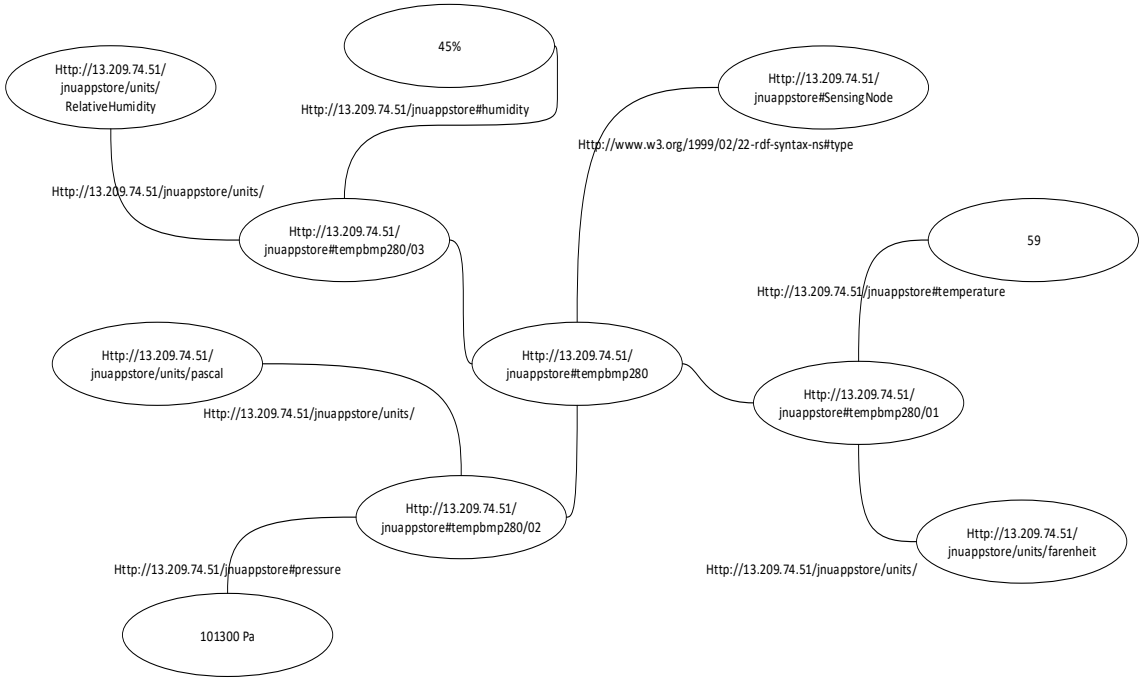


Figure 6. RDF graph of the BMP280 sensor.

5. System Development and Testing

Figure 7 represents the development model of the proposed system. The whole system is divided into two sub-systems i.e., the IoT platform and IoT marketplace. Both systems are configured, installed, and deployed on the cloud. We have used AWS for cloud computing. We have used the Amazon elastic compute cloud (EC2) for the deployment of the IoT platform and IoT Marketplace. Ubuntu 16.04 operating system is used to run a web server and host applications.

There are three layers in the proposed system i.e., application layer, repository layer, and context acquisition and control layer. In context of the acquisition and control layer, we have used two IoT devices as actuators i.e., LED and Fan and one Raspberry Pi. LED and fan are connected with Raspberry Pi on specific GPIO pins. MQTT broker is configured and installed on the ubuntu 16.04 web server. MQTT broker is used to publish and subscribe to messages against a unique topic. There are three types of topics: content-based, topic-based, and type-based. In our proposed system we have used topic-based. In topic-based there is a set of topics that are specified before. A broker is responsible for publishing and subscribing messages. We have used MQTT protocol for communication between the repository and the control layer. As there is a large number of messages being published and subscribed, we have used the concept of a distributed broker to overcome the load on the broker. The distributed broker minimizes the load on the central broker. Each distributed broker handles the request for different IoT devices.

In the context acquisition and control layer, we have used Raspberry Pi 3 Model B in the development model. LEDs and fans are acting as actuators. LED is attached to GPIO pin 23 and the fan is attached to GPIO pin 24. IoT devices are first registered to the IoT Marketplace. The registration process includes authorization and verification methods. Authorization is required to allow only to authorize people. The user requires credentials i.e., username and password for authorization. The authorization server sends an access token in response. The user enters device information along with access token for registration. Each IoT device is subscribed to a specific topic. Actuators receive commands from IoT Marketplace via MQTT protocol. The command could be to turn on or turn off the IoT device.

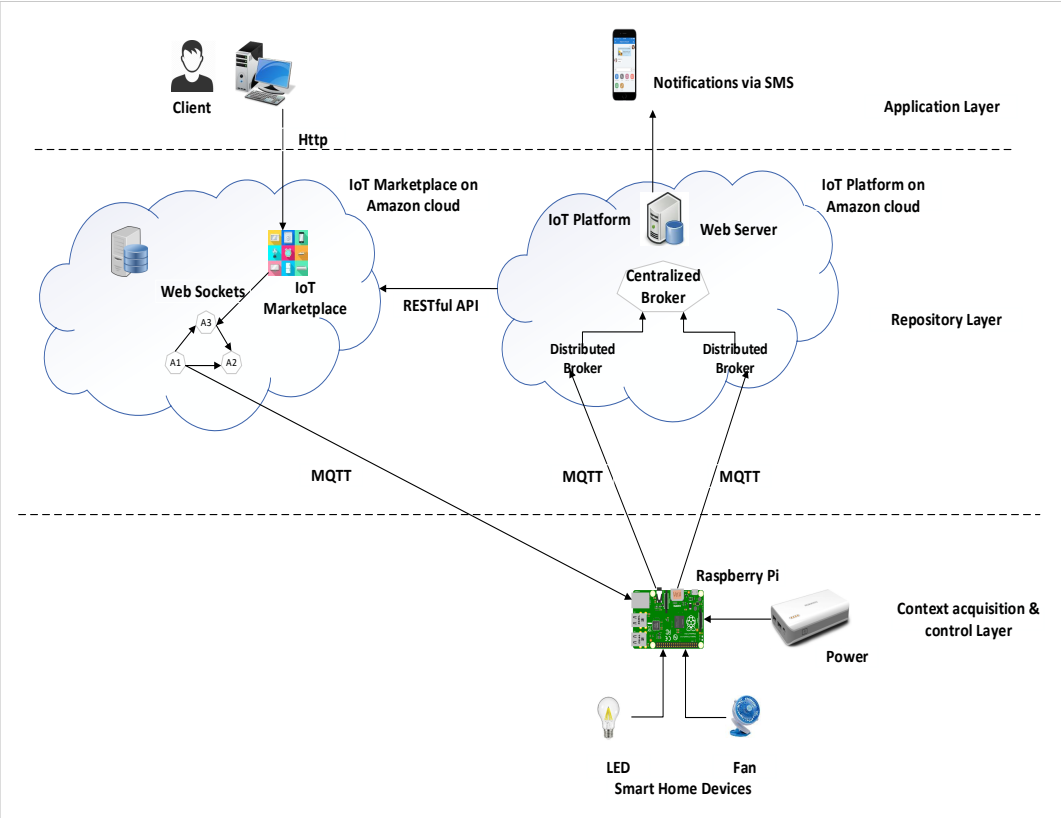


Figure 7. Development model of IoT platform and control service based on the interworking IoT marketplace.

The repository layer includes the configuration, installation of a web server and deployment of IoT platform and IoT Marketplace. The repository layer is used for IoT devices' metadata storage. MQTT protocol is used for communication between the physical layer and repository layer. RESTful APIs are used for the registration of IoT devices.

The application service layer has a user-friendly web interface to provide access to IoT Marketplace. With the IoT marketplace, users can monitor and control IoT devices. A case study is provided in the next section to understand the working of the IoT platform for a domestic environment.

6. Case-Study: Use of Proposed IoT Platform for Domestic Environments

In this case study, we provide the use of a proposed IoT platform for domestic environments. The proposed IoT platform provides component-based service to the end-users. In Figure 9, we can see that the user has registered three IoT devices i.e., light, fan, and temperature sensor in a smart home.

Figure 8 shows the user registration form on IoT Marketplace. User registration is necessary to provide secure and reliable services. It will keep a record of registered users and avoid unauthorized users. To create a new account on the IoT marketplace, it requires a valid email address, available user name, strong password, Role of the user, domain, and picture (optional), and time zone. There are two types of Role i.e., consumer and provider. The consumer is any user that wants to monitor public and private registered IoT devices e.g., the user wants to see the temperature of some specific location. The provider is any user that provides services to the users and can monitor and control IoT devices e.g., the user wants to monitor and control the camera. The domain can be e.g., home, or office. In local settings, the user can set the time zone of the current location. The time zone will help to discover the nearest sensors or actuators. In Figure 8, left navigation pane, there is a tab for sensors and actuators. Users can see the list of public sensors and actuators that are registered.

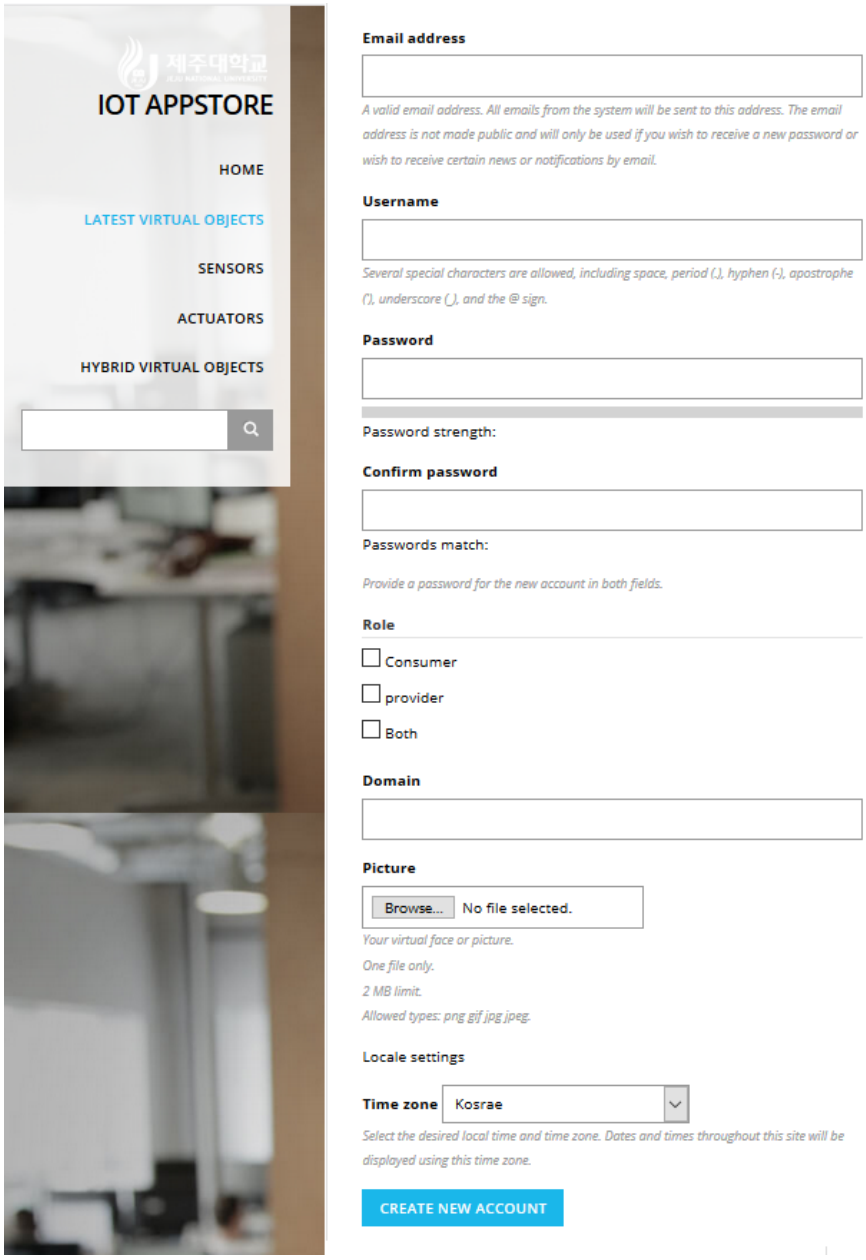


Figure 8. User registration on the IoT marketplace.

6.1. Step 1: Registration and Connectivity of IoT Device

Figure 9 shows the web interface of the IoT platform in which a registered user can add IoT devices to which they can monitor and control. From this figure, we can depict that the user is adding IoT devices for a smart home environment. Users can add multiple rooms in a smart home. The user can create a bedroom and add multiple IoT devices. The user can add a light, fan, temperature sensor to which it can be monitored and controlled.

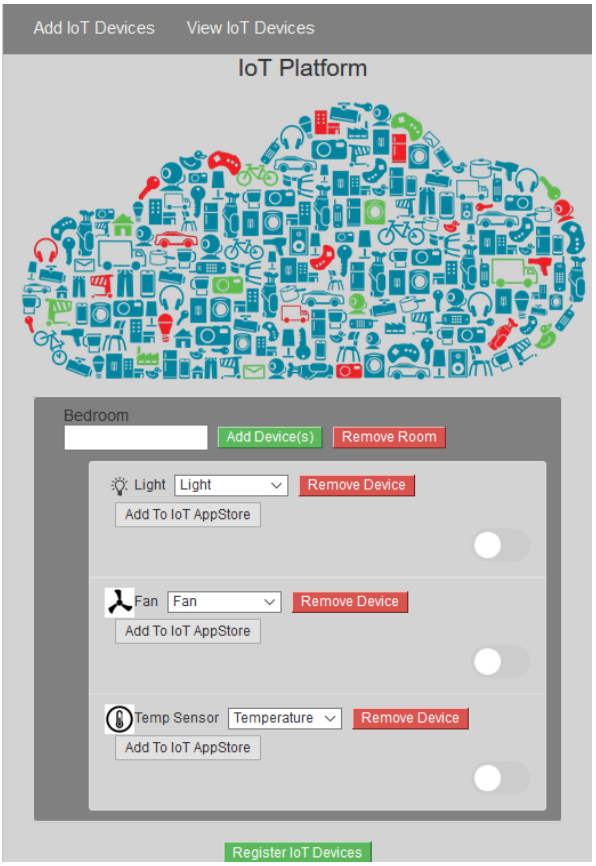


Figure 9. Web Interface for IoT device Registration on IoT Platform.

6.2. Step 2: Discovery of Virtual Object

Figure 10 shows the discovery of the IoT device. Users have access to public and private registered IoT devices. In this figure, the user can search for any IoT device in the search bar and choose the category type. It will return the list of registered IoT devices. Users can select any IoT device i.e., sensor or actuator. In this case, the user has searched for an LED. Users can perform some specific actions like turn On the LED or turn Off the LED. Users can download JSON or XML format of the IoT device to send commands to the corresponding IoT device.

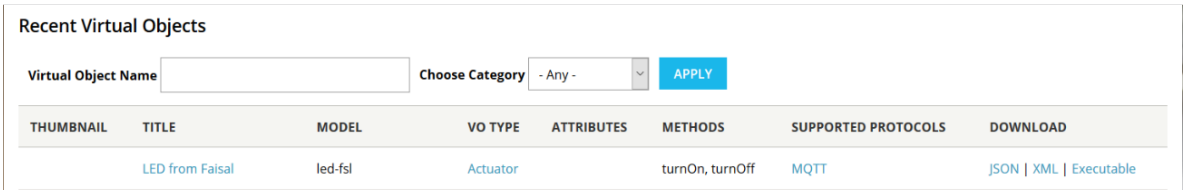


Figure 10. Discovery of a virtual object.

Figure 11 represents the JSON format of the virtual object of IoT device on the Marketplace. user can download the virtual object in two formats i.e., JSON or XML. Virtual object meta data includes virtual object title, model, type, attributes, and supported protocols.

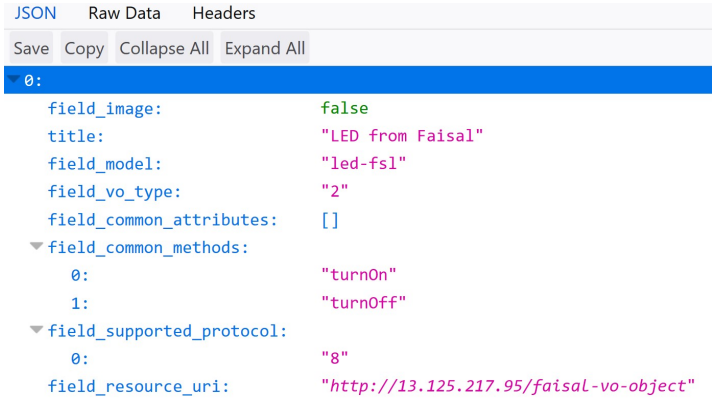


Figure 11. JSON format of the LED virtual object.

6.3. Step 3: Access of IoT Device

Figure 12 shows access to the IoT device. In this example, a user can turn on and turn off the registered LED device. Users can easily select required IoT devices and perform specific actions. Commands are sent from server to IoT device via MQTT protocol.



Figure 12. Access to the LED virtual object.

7. Experimental Environment

This section explains the experimentation performed in this study. Figure 13 shows IoT devices connected with Raspberry Pi. We have specified GPIO pins for each IoT device. LED is attached with GPIO pin 23 and Fan is attached with GPIO pin 24. The devices are connected to a cloud-based web server. AWS is used for cloud purposes. Our proposed system is deployed on Amazon elastic cloud compute (EC2). Ubuntu 16.04 server is used to configure, install, and deploy IoT platform and IoT AppStore. MySQL database is used for data storage. Data storage includes registered users' information, registered IoT devices and their metadata.

Table 1 summarizes the hardware services being used in experimentation. Raspberry Pi 3 Model B is used for the experiment. Raspberry Pi is a small size embedded IoT device that has all the capabilities to perform like a computer. It consumes less power. Raspbian Operating System is configured and installed on the Raspberry Pi for practical implementation. The hardware specification of Raspberry Pi is explained in Table 1. 16 GB SD Card is used for memory. Table 1 also explains the software tools used in our experiment. WebStorm is a software tool used for software development. WebStorm is specifically used for client-side applications in the JavaScript language. Sublime Editor is used for designing web pages for IoT platform and IoT Marketplace. In Raspberry Pi, Python programming language is used for registration, connectivity, and accessibility of IoT devices. Thonny is a software tool used to program in python language. FileZila is a software application used to transfer files from client to server. FileZila is used to upload the source code from the local system to the Amazon cloud. PuTTY is a software tool used to access devices remotely.

Different frameworks are used in this experiment. Our IoT platform is based on MVC model. The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components is built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects. Drupal is

an open-source software that can be used by an individual or group of users; even those who lack technical skills. It is used to create and manage dynamic websites. It can be used to develop applications like content management platform. The IoT marketplace is based on the Drupal framework.

Table 1. Detailed configuration of hardware and software used in experimental setup.

	Component	**Description**
Raspberry Pi	Model	Raspberry Pi 3 Model B
	Operating System	Raspbian
	Network	802.11n Wi-Fi
	CPU	1.2 GHz quad-core ARM Cortex A53
	GPU	Broadcom VideoCore IV @ 400 MHz
	SD Card	16 GB
	RAM	1 GB of LPDDR2-900 SDRAM
Software Tools	WebStorm	WebStorm 2018.1.3 for Javascript development
	Sublime Editor	Sublime Text 3 for web application design
	Thonny	Thonny for python development
	File Zila	FileZila for uploading client application.
	PuTTY	PuTTY to access Raspberry Pi remotely.
Framework(s)	MVC	Model-View-Controller for IoT Platform
	Drupal	Drupal Framework for IoT Marketplace
Programming Languages	Python	Python 3.7
	Node.js	Node.js 4.8.1
	Angular JS	AngularJS 1.1
	HTML	Html is used for designing web application interface
	CSS	CSS is used for styling the web interface
Libraries and API's	Paho-mqtt	MQTT version 3.1
Database	MySQL	MySQL 5.5 is used to store Metadata of IoT virtual objects
Cloud Services	Amazon Web Service	Elastic Compute Cloud (EC2)
	Operating System	Ubuntu Server 16.04 LTS (HVM)
	CPU	1 vCPUs 2.5 GHz Intel Xeon Family
	Memory	1 GB

Table 2 presents brief summary of various protocols used in this experiment. We have used four different protocols in the proposed system i.e., HTTP, MQTT, web sockets, SSH.

Table 2. Protocols used for communication between different system modules.

Protocol(s)	**Usage Description in Proposed System**
HTTP	Hyper-text Transfer Protocol runs at port 80 and it is used for accessing web applications online.
MQTT	Message Queuing Telemetry Protocol is a light weight protocol for publishing message. The default port is 1883.
Web Sockets	Web socket provides full-duplex communication on a single TCP channel. It is a communication protocol and works over http protocol.
SSH	Secure Shell Protocol runs on port 22 by default. We used to access the Ubuntu instance on Amazon cloud.

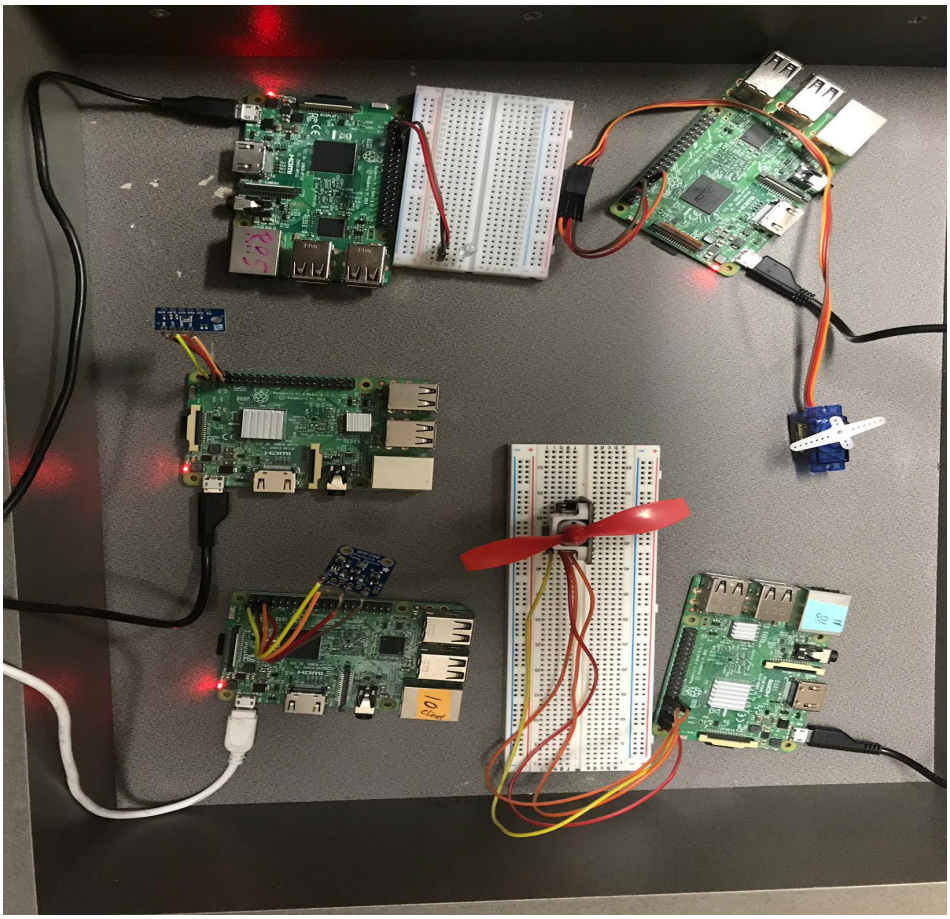


Figure 13. Experimental environment of IoT platform.

8. Performance Evaluation

In this section, we will evaluate and discuss the performance of the proposed work. Figure 14 shows a comparison between FIWARE and the proposed IoT platform. Figure 14 represents the number of responses per second for OAuth authentication requests between FIWARE and the proposed IoT platform. Results indicate that the proposed IoT platform performs marginally better than FIWARE. The minimum number of responses per second is 103 and 104 for FIWARE and proposed IoT platform respectively. The maximum number of responses per second is 157 and 199 for FIWARE and proposed IoT platform respectively. The average number of responses per second is 111 (stdev. 12.42) and 120 (stdev. 21.8) for FIWARE and proposed IoT platform respectively.

Figure 15 represents the number of registered devices per second between FIWARE and the proposed IoT platform. Results indicate that the proposed IoT platform performs marginally better than FIWARE. The minimum number of registered IoT devices per second is 15 and 14 for FIWARE and proposed IoT platform respectively. The maximum number of registered devices per second is 28 and 36 for FIWARE and proposed IoT platform respectively. The average number of registered devices per second is 18 (stdev. 2.96) and 19 (stdev. 3.2) for FIWARE and proposed IoT platform respectively.

Figure 16 represents the number of devices per second to transmit data to FIWARE and the proposed IoT platform. Results indicate that the proposed IoT platform performs marginally better than FIWARE. The minimum number of devices transmitting data per second is 10 and 11 for FIWARE and proposed IoT platform respectively. The maximum number of devices transmitting data per second is 22 and 26 for FIWARE and proposed IoT platform respectively. The average number of activated devices transmitting data per second is 14 (stdev. 2.73) and 16 (stdev. 2.95) for FIWARE and proposed IoT platform respectively.

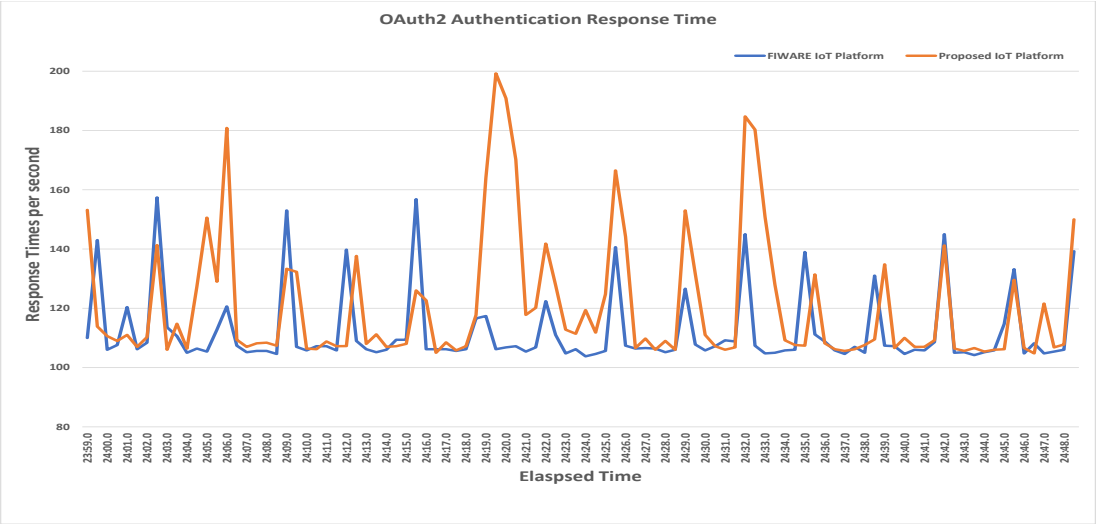


Figure 14. OAuth2 authentication response time.

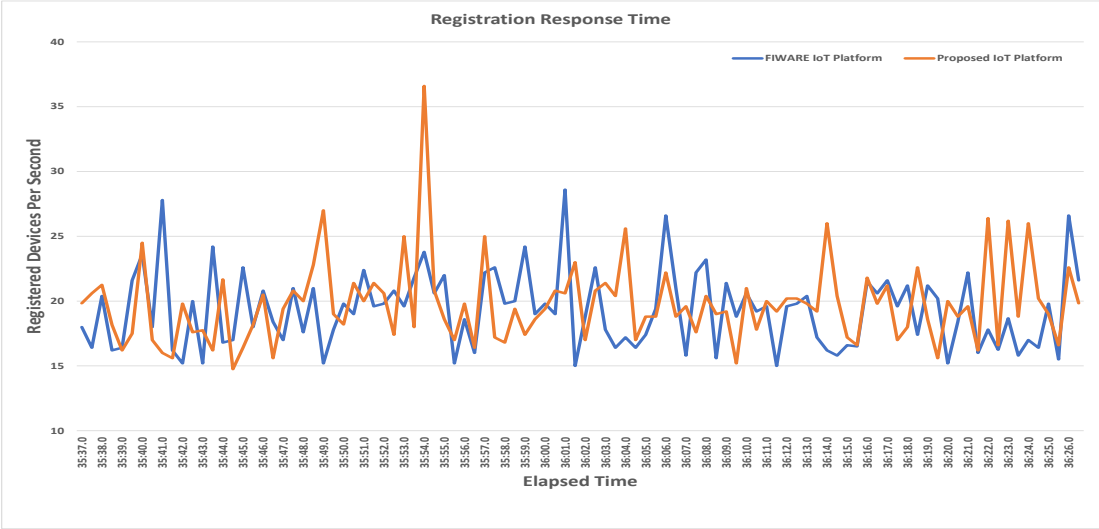


Figure 15. IoT device registration response time.

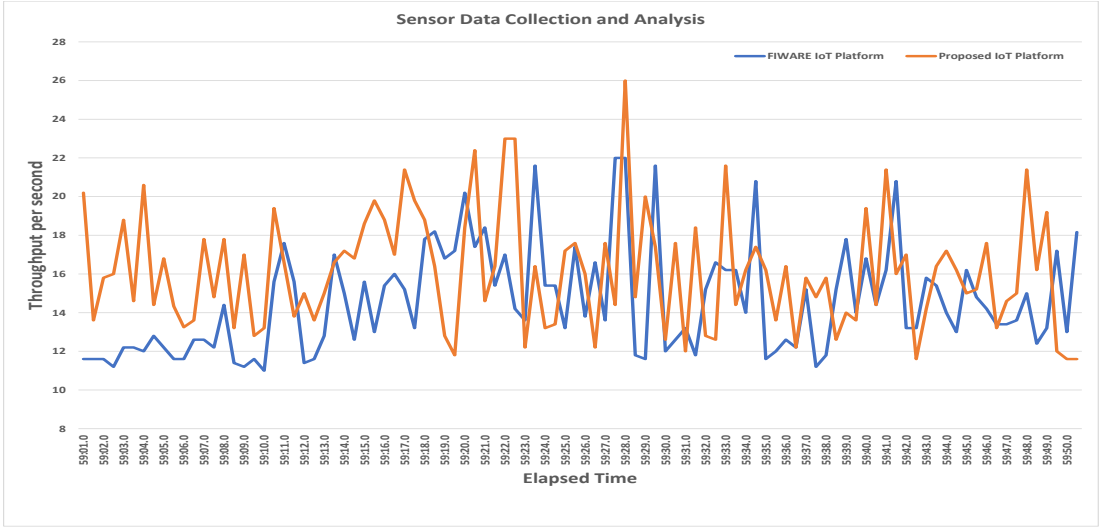


Figure 16. Sensor data response time.

Figure 17 represents the number of activated devices per second between FIWARE and the proposed IoT platform. Results indicate that the proposed IoT platform performs marginally better than FIWARE. The minimum number of activated IoT devices per second is 8 for both FIWARE and proposed IoT platform. The maximum number of activated devices per second is 12 and 13 for FIWARE and proposed IoT platform respectively. The average number of activated devices per second is 9 (stdev. 0.72) and 10 (stdev. 1.04) for FIWARE and the proposed IoT platform, respectively.

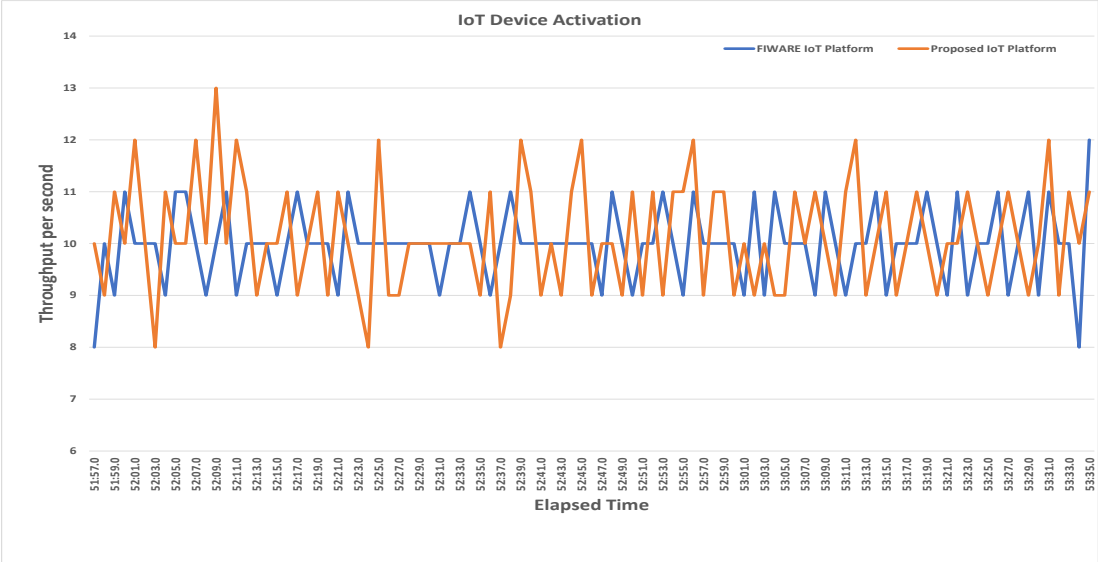


Figure 17. IoT device activation response time.

Figure 18 represents the number of deactivated devices per second between FIWARE and the proposed IoT platform. Results indicate that the proposed IoT platform performs marginally better than FIWARE. The minimum number of deactivated IoT devices per second is 8 for both FIWARE and proposed IoT platform. The maximum number of deactivated devices per second is 18 and 19 for FIWARE and proposed IoT platform respectively. The average number of activated devicecs per second is 11 (stdev. 2.3) and 12 (stdev. 2.63) for FIWARE and proposed IoT platform respectively.

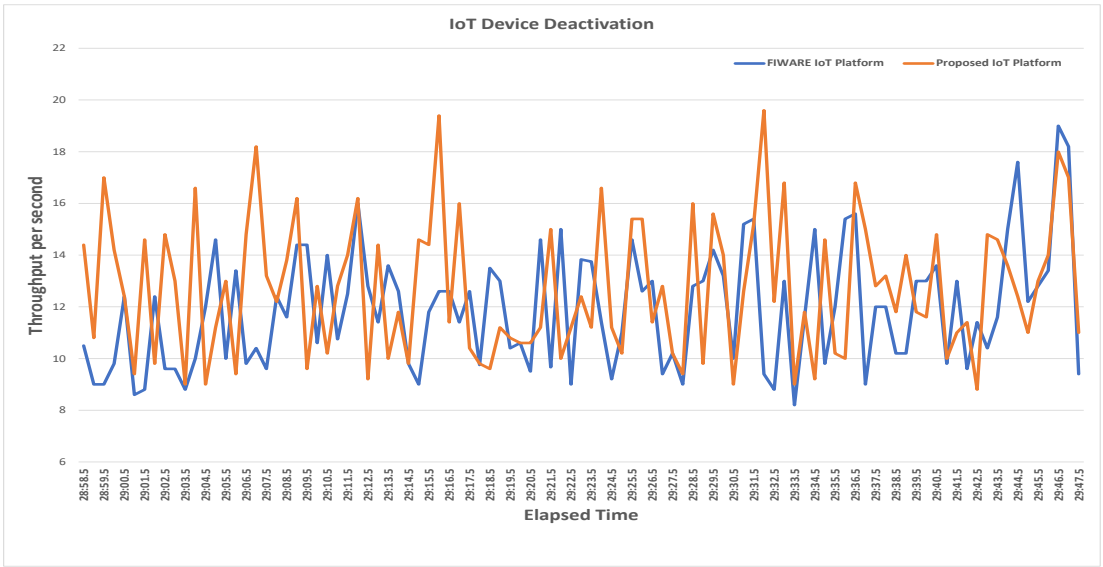


Figure 18. IoT device deactivation response time.

9. Comparison and Significance

In this section, we will emphasize the significance of the proposed work and provide a comparative analysis with the existing state-of-the-art tools. We have conducted a comparative analysis of the proposed IoT platform with existing IoT platforms. The IoT platform has become an integral part of the IoT deployment, with many types and suppliers having their own focus and go-to-market strategy. IoT platform market is complex as different IoT projects and software applications are developed in different architectures, their mode of connectivity and device management, capabilities to build applications and options to leverage IoT in a meaningful way for different IoT use cases in any context i.e., consumer applications, enterprise IoT applications, etc.

Table 3 summarizes various tools and their comparison with the proposed work. As described in the earlier sections, the proposed IoT platform performs marginally better than FIWARE. We have selected the attributes that are necessary for any open-source tools in general to compare the performance of the IoT platforms. There are many open-source and paid IoT platforms such as FIWARE and Kaa. These platforms provide cloud services for computation and data management to monitor and control IoT devices. Each IoT platform differs in providing services and is used for some specific purpose. The proposed IoT platform is able to register any IoT device i.e., sensor or actuator. The IoT device can be monitored or controlled using MQTT protocol which is lightweight and consumes less power. Due to load on IoT platforms, the proposed system separated the marketplace from the platform to provide services to the end-users.

Table 3. Comparative analysis of proposed tool with existing state-of-the-art tools.

S.No	Name	Cloud Support	Open Source	Major Domain	Maintenance Status	IoT Services	Protocols
1	Salesforce	Yes	Yes	Real-time event processing engine.	Yes	Same Platform	MQTT
2	ThingWorx	Yes	Yes	Industrial Internet of Things.	Yes	Same Platform	MQTT, CoAP
3	Kaa	Yes	No	Multi-purpose middleware platform for end-to-end IoT solutions.	Yes	Same Platform	MQTT, CoAP
4	Zetta	Yes	Yes	Assembling IoT devices into data-intensive, real-time applications.	No	Same Platform	MQTT
5	FIWARE	Yes	Yes	IoT with context Information Management and Big Data Services.	No	Same Platform	MQTT
6	Proposed IoT Platform	Yes	Yes	Monitor and Control Service of IoT devices.	Yes	Separate Marketplace	MQTT

10. Conclusions and Future Directions

This paper aims to provide an IoT platform and control service mechanism based on virtual objects in CoT. This paper outlines the procedure for the design and implementation of an architecture for enabling the general public to register, discover, and access IoT devices. In this study, we have developed an IoT platform that is secure and reliable for registration of different IoT devices. The registration process requires authorization and verification for security purposes. The authorization process will only allow authorized persons to register IoT devices and avoid unnecessary usage of the IoT platform. Verification of IoT devices will validate the existence of an IoT device by checking the model, IP address, MAC address. The proposed system also provides the discovery and sharing of registered IoT devices. The visibility mode of IoT devices can be set to public or private based on the user requirement. Users can discover required IoT devices using IoT Marketplace. Users are able to monitor and control the IoT device from the IoT Marketplace. The significance of the proposed work has been tested during experimentation

with software tools, and it has been found that the proposed system performs better with respect to FIWARE. Results conclude that the proposed IoT platform performs marginally better than the FIWARE. In future work, we can use other lightweight communication protocols such as CoAP and compare the performance with the proposed system.

Author Contributions: F.M. conceived the idea for this paper, designed the experiments and wrote the paper; S.A. assisted in model designing and experiments. D.K. conceived the overall idea of Interworking IoT platform and marketplace, and proof-read the manuscript.

Acknowledgments: This research was supported by Energy Cloud R&D Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT (2019M3F2A1073387), and this research was supported by Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2018-0-01456, AutoMaTa: Autonomous Management framework based on artificial intelligent Technology for adaptive and disposable IoT). Any correspondence related to this paper should be addressed to Dohyeun Kim.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jamil, F.; Iqbal, M.; Amin, R.; Kim, D. Adaptive Thermal-Aware Routing Protocol for Wireless Body Area Network. *Electronics* **2019**, *8*, 47. [CrossRef]
2. Wortmann, F.; Flüchter, K. Internet of things. *Bus. Inf. Syst. Eng.* **2015**, *57*, 221–224. [CrossRef]
3. Kodali, R.K.; Soratkal, S. MQTT based home automation system using ESP8266. In Proceedings of the 2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Agra, India, 21–23 December 2016; pp. 1–5.
4. Al-Jaroodi, J.; Mohamed, N.; Jawhar, I.; Mahmoud, S. CoTWare: A Cloud of Things Middleware. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), Atlanta, GA, USA, 5–8 June 2017; pp. 214–219.
5. Truong, H.L.; Dustdar, S. Principles for engineering IoT cloud systems. *IEEE Cloud Comput.* **2015**, *2*, 68–76. [CrossRef]
6. Dehury, C.K.; Sahoo, P.K. Design and implementation of a novel service management framework for IoT devices in cloud. *J. Syst. Softw.* **2016**, *119*, 149–161. [CrossRef]
7. Kibria, M.G.; Kim, H.S.; Chong, I. IoT learning model based on virtual object cognition. In Proceedings of the 2016 International Conference on Information Networking (ICOIN), Kota Kinabalu, Malaysia, 13–15 January 2016; pp. 369–371.
8. Xia, F.; Yang, L.T.; Wang, L.; Vinel, A. Internet of things. *Int. J. Commun. Syst.* **2012**, *25*, 1101–1102. [CrossRef]
9. Juve, G.; Deelman, E.; Vahi, K.; Mehta, G.; Berriman, B.; Berman, B.P.; Maechling, P. Scientific workflow applications on Amazon EC2. *arXiv* **2010**, arXiv:1005.2718.
10. Nitti, M.; Pilloni, V.; Colistra, G.; Atzori, L. The virtual object as a major element of the internet of things: A survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1228–1240. [CrossRef]
11. Vandikas, K.; Tsiatsis, V. Performance evaluation of an IoT platform. In Proceedings of the 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, Oxford, UK, 10–12 September 2014; pp. 141–146.
12. Ahmad, S.; Mehmood, F.; Kim, D.H. A DIY Approach for the Design of Mission-Planning Architecture Using Autonomous Task–Object Mapping and the Deployment Model in Mission-Critical IoT Systems. *Sustainability* **2019**, *11*, 3647. [CrossRef]
13. Cavalcante, E.; Pereira, J.; Alves, M.P.; Maia, P.; Moura, R.; Batista, T.; Delicato, F.C.; Pires, P.F. On the interplay of Internet of Things and Cloud Computing: A systematic mapping study. *Comput. Commun.* **2016**, *89*, 17–33. [CrossRef]
14. Mehmood, F.; Ullah, I.; Ahmad, S.; Kim, D. Object detection mechanism based on deep learning algorithm using embedded IoT devices for smart home appliances control in CoT. *J. Ambient Intell. Humaniz. Comput.* **2019**, 1–17. [CrossRef]
15. Farahzadi, A.; Shams, P.; Rezazadeh, J.; Farahbakhsh, R. Middleware technologies for cloud of things: A survey. *Digit. Commun. Netw.* **2018**, *4*, 176–188. [CrossRef]

16. Khudoyberdiev, A.; Jin, W.; Kim, D. A Novel Approach towards Resource Auto-Registration and Discovery of Embedded Systems Based on DNS. *Electronics* **2019**, *8*, 442. [CrossRef]
17. Ren, J.; Guo, H.; Xu, C.; Zhang, Y. Serving at the edge: A scalable iot architecture based on transparent computing. *IEEE Netw.* **2017**, *31*, 96–105. [CrossRef]
18. Morabito, R.; Cozzolino, V.; Ding, A.Y.; Beijar, N.; Ott, J. Consolidate IoT edge computing with lightweight virtualization. *IEEE Netw.* **2018**, *32*, 102–111. [CrossRef]
19. Naranjo, P.G.V.; Pooranian, Z.; Shojafar, M.; Conti, M.; Buyya, R. FOCAN: A Fog-supported smart city network architecture for management of applications in the Internet of Everything environments. *J. Parallel Distrib. Comput.* **2019**, *132*, 274–283. [CrossRef]
20. Kelaidonis, D.; Giaffreda, R.; Somov, A.; Foteinos, V.; Poulios, G.; Stavroulaki, V.; Vlacheas, P.; Demestichas, P.; Baranov, A.; Biswas, A.R. Virtualization and cognitive management of real world objects in the internet of things. In Proceedings of the 2012 IEEE International Conference on Green Computing and Communications, Besancon, France, 20–23 November 2012; pp. 187–194.
21. Moratelli, C.; Johann, S.; Neves, M.; Hessel, F. Embedded virtualization for the design of secure IoT applications. In Proceedings of the 27th International Symposium on Rapid System Prototyping: Shortening the Path from Specification to Prototype, Pittsburgh, PA, USA, 1–7 October 2016; pp. 2–6.
22. Ullah, I.; Ahmad, S.; Mehmood, F.; Kim, D. Cloud Based IoT Network Virtualization for Supporting Dynamic Connectivity among Connected Devices. *Electronics* **2019**, *8*, 742. [CrossRef]
23. Nitti, M.; Pilloni, V.; Atzori, L. Real and Virtual Objects in the IoT: Two Distinct Worlds or Two Sides of the Same Coin? *Newsletter* **2014**, *2014*.
24. Guinard, D. A Web of Things Application Architecture: Integrating the Real-World into the Web. Ph.D. Thesis, ETH Zurich, Zürich, Switzerland, 2011.
25. Borgia, E. The Internet of Things vision: Key features, applications and open issues. *Comput. Commun.* **2014**, *54*, 1–31. [CrossRef]
26. Prabaharan, J.; Swamy, A.; Sharma, A.; Bharath, K.N.; Mundra, P.R.; Mohammed, K.J. Wireless home automation and security system using MQTT protocol. In Proceedings of the 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 19–20 May 2017; pp. 2043–2045.
27. Stastny, S. UbiBazaar: App Store for the Internet of Things. Master's Thesis, NTNU, Trondheim, Norway, 2015.
28. Modoni, G.E.; Trombetta, A.; Veniero, M.; Sacco, M.; Mourtzis, D. An event-driven integrative framework enabling information notification among manufacturing resources. *Int. J. Comput. Integr. Manuf.* **2019**, *32*, 241–252. [CrossRef]
29. Fysarakis, K.; Askoxylakis, I.; Soultatos, O.; Papaefstathiou, I.; Manifavas, C.; Katos, V. Which IoT protocol? Comparing standardized approaches over a common M2M application. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–7.
30. Abdelwahab, S.; Hamdaoui, B.; Guizani, M.; Znati, T. Cloud of things for sensing-as-a-service: Architecture, algorithms, and use case. *IEEE Internet Things J.* **2016**, *3*, 1099–1112. [CrossRef]
31. Ullah, I.; Sohail Khan, M.; Kim, D. IoT Services and Virtual Objects Management in Hyperconnected Things Network. *Mob. Inf. Syst.* **2018**, *2018*, 2516972. [CrossRef]
32. Espada, J.P.; Martínez, O.S.; Bustelo, B.C.P.G.; Lovelle, J.M.C. Virtual objects on the internet of things. *IJIMAI* **2011**, *1*, 23–29. [CrossRef]
33. Ahmad, S.; Hang, L.; Kim, D.H. Design and Implementation of Cloud-Centric Configuration Repository for DIY IoT Applications. *Sensors* **2018**, *18*, 474. [CrossRef] [PubMed]
34. Ahmad, S.; Hussain, I.; Fayaz, M.; Kim, D.H. A Distributed Approach towards Improved Dissemination Protocol for Smooth Handover in MediaSense IoT Platform. *Processes* **2018**, *6*, 46. [CrossRef]
35. Ahmad, S.; Malik, S.; Ullah, I.; Fayaz, M.; Park, D.H.; Kim, K.; Kim, D. An Adaptive Approach Based on Resource-Awareness Towards Power-Efficient Real-Time Periodic Task Modeling on Embedded IoT Devices. *Processes* **2018**, *6*, 90. [CrossRef]
36. Ahmad, S.; Malik, S.; Ullah, I.; Park, D.H.; Kim, K.; Kim, D. Towards the Design of a Formal Verification and Evaluation Tool of Real-Time Tasks Scheduling of IoT Applications. *Sustainability* **2019**, *11*, 204. [CrossRef]

37. Ahmad, S.; Malik, S.; Kim, D.H. Comparative Analysis of Simulation Tools with Visualization based on Real-time Task Scheduling Algorithms for IoT Embedded Applications. *Int. J. Grid Distrib. Comput.* **2018**, *11*, 1–10. [CrossRef]
38. Kos, A.; Pristov, D.; Sedlar, U.; Sterle, J.; Volk, M.; Vidonja, T.; Bajec, M.; Bokal, D.; Bešter, J. Open and scalable IoT platform and its applications for real time access line monitoring and alarm correlation. In *Internet of Things, Smart Spaces, and Next Generation Networking*; Springer: New York, NY, USA, 2012; pp. 27–38.
39. Modoni, G.E.; Veniero, M.; Trombetta, A.; Sacco, M.; Clemente, S. Semantic based events signaling for AAL systems. *J. Ambient Intell. Humaniz. Comput.* **2018**, *9*, 1311–1325. [CrossRef]
40. Wahle, S.; Magedanz, T.; Schulze, F. Demonstration of OpenMTC M2M Solutions for Smart Cities and the Internet of Things. In Proceedings of the 37th IEEE Local Computer Networks Conference, Clearwater Beach, FL, USA, 22–25 October 2012.
41. Zahariadis, T.; Papadakis, A.; Alvarez, F.; Gonzalez, J.; Lopez, F.; Facca, F.; Al-Hazmi, Y. FIWARE lab: Managing resources and services in a cloud federation supporting future internet applications. In Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, London, UK, 8–11 December 2014; pp. 792–799.
42. Guth, J.; Breitenbücher, U.; Falkenthal, M.; Leymann, F.; Reinfurt, L. Comparison of IoT platform architectures: A field study based on a reference architecture. In Proceedings of the 2016 Cloudification of the Internet of Things (CIoT), Paris, France, 23–25 November 2016; pp. 1–6.
43. Jeon, S.; Jung, I. Experimental evaluation of improved IoT middleware for flexible performance and efficient connectivity. *Ad Hoc Netw.* **2018**, *70*, 61–72. [CrossRef]
44. Mineraud, J.; Mazhelis, O.; Su, X.; Tarkoma, S. A gap analysis of Internet-of-Things platforms. *Comput. Commun.* **2016**, *89*, 5–16. [CrossRef]
45. Ray, P.P. A survey of IoT cloud platforms. *Future Comput. Inform. J.* **2016**, *1*, 35–46. [CrossRef]
46. Girau, R.; Martis, S.; Atzori, L. Lysis: A platform for IoT distributed applications over socially connected objects. *IEEE Internet Things J.* **2017**, *4*, 40–51. [CrossRef]
47. Farris, I.; Girau, R.; Militano, L.; Nitti, M.; Atzori, L.; Iera, A.; Morabito, G. Social virtual objects in the edge cloud. *IEEE Cloud Comput.* **2015**, *2*, 20–28. [CrossRef]
48. RDF. Available online: https://www.w3.org/RDF/ (accessed on 7 October 2019).



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).