electronics

MDPI

Article

Resource Management Based on OCF for Device Self-Registration and Status Detection in IoT Networks

Wenquan Jin and Dohyeun Kim *

Department of Computer Engineering, Jeju National University, Jeju 63243, Korea; wenquan.jin@jejunu.ac.kr
* Correspondence: kimdh@jejunu.ac.kr

Received: 31 December 2018; Accepted: 6 March 2019; Published: 11 March 2019

Abstract: Recently, there are heterogeneous devices that connect to the Internet to provide ubiquitous and intelligent services based on sensors and actuators in the network of the Internet of Things (IoT). The resources of IoT represent the physical entities on the Internet to expose functions through services. Resource management is necessary to enable a massive amount of IoT-connected devices to be discoverable and accessible in the network of IoT. In this paper, we propose an IoT resource management to provide schemes of device self-registration and status detection for devices based on the Open Connectivity Foundation (OCF) standard. This device self-registration scheme is based on an agent that is proposed for registering devices itself which deployed in the OCF network. The devices host the OCF resources to provide IoT services such as sensing and controlling through the sensors and actuators. For a group of devices, an agent-based self-registration is proposed to register the resources. Through the proposed self-registration, the information of IoT devices is published using profile and saved in the management platform that enables the clients to discover the resources and access the services. For accessing the IoT resources in the OCF network, an interworking proxy is proposed to support the communications between web clients and devices over Hypertext Transfer Protocol (HTTP) and Constrained Application Protocol (CoAP) based on OCF. Furthermore, through the interoperability of the resources using the registered information, a real-time monitoring scheme is proposed based on periodic request and response for the status detection of deployed devices.

Keywords: Internet of Things (IoT); resource registration; resource discovery; resource management; Open Connectivity Foundation (OCF); constrained network; status detection

1. Introduction

The Internet of Thing (IoT) is comprised of a massive amount of connected devices that provide smart and ubiquitous services based on sensors and actuators in the heterogeneous network environment. Recently, with the development of IoT technologies including networking, computation, and storage, the number of connected devices is increasing rapidly and will reach 20.4 billion by 2020 [1]. Many IoT devices are developed for the constrained environment using limited power supply, processing and storage capability, and network communication range [2]. In order to interact with these devices through a local network or the Internet, the services need to be available, which are provided by the devices. Through these services, clients can get the sensing data of the environment using sensors and control the actuators to change the environment where the devices are deployed. Although, these devices shall provide seamless services to the users through the system interoperability in IoT networks, such as smart homes, hospitals, and other smart spaces. Nevertheless, the management functionalities are required for the good interoperability of devices using these services, such as monitoring the environment and devices, registering information of devices for discovery, and proxying various protocols for interworking with various networks [3,4]. These functionalities enable the devices to be monitored and configured by users based on the represented cyber resource information.

IoT devices are equipped with sensors and actuators to provide services which expose the functions of devices based on the hosted resources [5]. Using the Application Programming Interfaces (APIs), the client application accesses the resources to get sensing data and control the actuator. A resource can have one or more handlers to handle the requests from the client and respond to the results to the client. The development of IoT applications requires simple and extendable architecture support for proving services on constrained hardware [6]. Therefore, for delivering the sensing data to clients and receiving the command to control actuators, the service-oriented architecture (SOA) shall be one of the best development architectures for the IoT applications [7]. The SOA based systems can be implemented by Representational State Transfer (REST) architecture that supports the APIs to present the service models on the Internet [8]. A service is identified and encapsulated using the Uniform Resource Identifier (URI) through the REST architecture-based application on the Web [9]. REST APIs enable the resources of an IoT device to be accessible through communication protocols using URIs [10]. Application layer protocols, such as Constrained Application Protocol (CoAP) and the Hypertext Transfer Protocol (HTTP), are used for implementing the IoT services based on the REST architecture to support reliable and efficient communication [11]. For proper maintenance of the increasing number of IoT devices through the interactions using the REST APIs in IoT networks, it is essential to manage these devices including configuration and monitoring.

The management generally can be comprised of several domains including application, network and system to enable the configurations and monitoring of the resource in an IoT network [12]. The components of device management include fault management, configuration management, firmware management and other management for configuring the entities in the system where the devices are deployed [13]. Through functionalities of management, the devices are monitored and remotely detected the status to achieve reducing operating expenses. The IoT devices are based on the communication protocols to transfer the data including the information of devices, sensing data, and commands to actuators and their acknowledgment. The interactions based on network communication enable the configuration and monitoring for the management of IoT. The Open Connectivity Foundation (OCF) is a global standard that provides specifications for implementation of IoT [14]. The interactions between entities for the management in IoT networks, the OCF enables the resources to provide fundamental communications such as request and response.

In this paper, we propose resource management based on OCF for device self-registration and status detection through the interactions between IoT server, web server, web client, IoT agent, and IoT device in the IoT network. Based on the OCF IoTivity framework, the communication functions are provided to the IoT devices and agents which publish the information to the platform for registering the resources. In the proposed IoT network, the devices are deployed to provide IoT services such as sensing and controlling through the sensors and actuators. The OCF resources are hosted on the devices to provide the IoT services. For the appearance of these resources in the IoT network, a configuration scheme based on self-registration is proposed for registering the information in the system through OCF communication. Through the registration, the published information of IoT devices are saved in the management platform that enables clients to discover the resources and access the services for monitoring the IoT devices as well as the environment where the devices are deployed. For accessing the IoT resources in the OCF network, a proxy is included in the IoT platform to support the interworking between HTTP and CoAP based on OCF. A client can monitor the IoT resources in real-time through a polling scheme based on the loop process to request the resources. Nevertheless, we propose a scheme based on the IoT platform for collecting the data from the IoT network to achieve real-time monitoring. Furthermore, through the interoperability between the IoT platform and devices in the OCF network, the status detection can be implemented for the IoT management and diagnosis.

The rest of the paper is structured as follows. Section 2 introduces the related works regarding the management schemes in the standard IoT frameworks. Section 3 presents the proposed IoT resource management including self-registration, discovery and access, and status detection scheme. Section 4 presents the implementation results for the proposed IoT network based on OCF. Section 5 presents the performance evaluation based on the implementations for the registration, discovery, service access, and interworking. Finally, this paper is concluded in Section 6.

2. Related Works

The need for IoT management plays an integral role in the IoT paradigm due to the fact that networks are becoming more and more distributed and thus the deployment of numerous heterogeneous nodes needs to happen ubiquitously. Recently, based on IoT networks, devices have been deployed to provide sensing and actuating services in various industries such as manufacturing, supply chains, energy, healthcare, and the automotive industry [15–17]. To manage these IoT devices in the above industries, many platforms are published to standardize the development environment [18]. The traditional management solutions are used for supporting remote control to the specific type of electronic devices, such as surveillance cameras, printers, personal computers, and other network-connected devices [19]. However, for IoT networks, the solutions are not sufficient to support the feature of IoT such as heterogeneity, constraints, and data collection from the environment. In order to support the IoT environment, a number of standards have been published to provide guidance and implementations such as Open Mobile Alliance (OMA), oneM2M, and OCF [20–22].

The OMA is an international standard organization for mobile communications that provides a Device Management (DM) specification for managing mobile devices [23]. The implementation of OMA is the Lightweight M2M (LWM2M), which supports the client and server communicate with each other through the LWM2M protocol based on the CoAP for low power and constrained devices [24]. OMA DM is used for management of devices using the request and response transaction model through the client and server in the OMA entity that enables the management commands to be executed on OMA nodes. In the implementation of OMA DM based on LWM2M, the main functions are bootstrap, device discovery and registration, device management and service enablement, and information reporting [25]. International IoT standard organizations, such as the European Telecommunications Standards Institute (ETSI) and oneM2M, have adopted the use of LWM2M as part of their IoT frameworks for the management of the IoT nodes [26,27]. The ETSI M2M functional architecture defines a remote entity management service for the device management based on the OMA DM which supports configuration, performance monitoring, and fault management. The fault management in the ETSI M2M that provides fault-related statistics such as memory failures and unreachable communications. However, the LWM2M is not optimized for a diversity of IoT applications such as limited resources of wireless sensor devices, distributed network environment, massive data collected from a variety of applications based on multiple communication protocols, etc.

The oneM2M is a scalable and interoperable IoT platform that provides functions for data exchange among IoT application in various industries [28]. The discovery function is a fundamental requirement in the oneM2M standard which has proposed discovery as a common service function (CSF) residing in the Common Service Entity (CSE) [29]. The CSE is a component of oneM2M that provides CSFs including communication management, data management, device management, group management, discovery, registration and security. The entities of the oneM2M-based IoT network can request to the CSE for registration. Once the information is registered, the information can be searched including the attributes and resources based on text-matching mechanisms. For interworking with other frameworks, oneM2M enables the features with other local network technologies including legacy deployments of IoT systems. An IoT gateway centric architecture is proposed to provide novel M2M services based on the oneM2M for discovery and interworking proxy with devices [30].

OCF supports a standard framework for developing an IoT network on various software platforms such as Linux, Tizen, Android, Windows, and iOS [31]. OCF is expandable to adopt any transport protocols such as Wi-Fi, Bluetooth, Bluetooth Low Energy (BLE), ZigBee, and Ethernet. According to the core specification, CoAP is the default protocol to provide services based on the REST APIs for communications between OCF entities in an OCF-based IoT network. CoAP is an application protocol based on the User Datagram Protocol (UDP) that is used for Machine to Machine (M2M) communications to support the constrained environment [32]. The IoTivity is an open source framework for the OCF specifications, which can be built as a library to be included in the IoT device application for implementing the role of client and server [33,34]. The functionalities of OCF which are supported through the interactions including messaging, discovery, monitoring, and maintenance based on the fundamental communication ability over the OCF protocol. Device management in OCF that includes functionalities of diagnostics and maintenance. Once diagnostics and maintenance are supported by an OCF device, the resource "/oic/mnt" shall be supported to provide factory reset, reboot, and response of error code. Instead of the resource "/oic/mnt", we present a scheme to enable real-time interaction with a resource to detect the status of an OCF device.

For service management in the SOA-based IoT systems, the Arrowhead framework is proposed to support efficient development and operation of interconnected, cooperative systems [35]. The core services of Arrowhead framework are divided by information assurance services, information infrastructure services, and system management services. Through these services, the framework provides service registration, service discovery, security, authentication, and orchestration of complex services in a local cloud [36,37]. In an IoT network, sensing and actuating services are provided resources within a server of a physical device that is accessible through a wired or wired network. For IoT devices, the Arrowhead framework can provide the device registration through its service registry to enable accessibility of devices [38]. The Arrowhead framework requires every service to be registered to the service registry in a local cloud. The implementation of the service registry is based on a DNS-SD [39]. However, REST APIs are the communication architecture for accessing services including registration, discovery, and interoperability [40]. The interoperability scheme of the Arrowhead framework provides a protocol translation service to support different protocols for information exchange technologies.

Providing interoperability to heterogeneous IoT devices in physical networks is important. In constrained IoT environment, trust management requires keeping the trust information for a huge number of heterogeneous devices [41]. The resource of IoT devices in a network, P2P technology can be used for distributed IoT resource management [42]. In SOA-based IoT systems, each device is a service consumer as well as a service provider through compatible service APIs. OCF is implemented based on CoAP and provides services through RESTful APIs for an SOA-based system. Therefore, interoperability with other solutions can be flexible [43]. For the registration and discovery of OCF devices, a resource directory can be used to provide a consistent interface [44]. A consistent registration scheme enables heterogeneous IoT devices can be registered to the resource directory and discovered by clients for being accessed [45]. The proposed self-registration scheme is used for OCF devices. Nevertheless, the scheme can be applied to other protocol-based devices in a constrained IoT network. Therefore, the resource directory can be a service registry in an SOA based IoT system to provide storage for registering the device and resource information. For the interoperability with the IoT network, Internet clients need the proxy to access these constrained devices [46]. A proxy-based middleware can be used for enabling communications between different protocols [47]. Integrating the proxy to a resource directory can be a suitable solution to manage IoT devices in constrained networks.

For constrained IoT networks, supplying sufficient energy to fulfill the service quality of IoT devices is important. Mostly, the IoT devices are battery powered with limited resources. Therefore, improving the energy efficiency of IoT devices is mandatory. Several methods have been proposed including communication-based energy saving using novel protocols [48,49], applying a sleep–wake schedule to achieve energy saving [50,51] and energy harvesting from ambient sources to increase battery life [52]. In order to enable IoT device sustainably except battery replacement, energy harvesting solutions support sufficient power based on renewable energy resources [53]. Radio frequency is a signal that carries information as well as delivers energy to a device for providing controllable and sustainable power supply [54]. For managing IoT devices, status detection through real-time monitoring is necessary to acquire information about device status in a constrained environment. A management server can be used for detecting the status including power and faults, and based on the detection to apply solutions it can perform other functions such as energy delivering, issue reports, and commands.

3. Proposed IoT Resource Management Based on OCF

An IoT network is comprised of heterogeneous entities that provide services based on transferring the data through communications. The IoT devices in the network provide services based on sensors and actuators for interacting with the environment where the devices are deployed [55]. Users can monitor the environment through the presented information of resources which are hosted on the devices to provide services. These devices are developed for the constrained environment in the edge of the IoT network to update the environment ubiquitously [56]. In this network, for sufficient computation and storage, the high-performance servers are required to support these constrained devices. The management functionalities shall be implemented on the server to provide the sufficient hardware environment for storing information and presenting information to the mass of users.

Figure 1 presents the IoT architecture that includes clients, IoT platform, IoT device, and IoT agent for the proposed resource management. The IoT device and IoT agent are deployed in the local network where can be the smart space for providing user-specific services based on the constrained environment. In this network, IoT devices and IoT agent host the resources to provide IoT services. The IoT devices can be assumed to have the sensors and actuators on the board directly. The IoT agent can be assumed to interact with sensor devices through a network protocol for bridging the sensing data from the sensor devices to clients. From the perspective of clients in the local network, the resources of the IoT device and IoT agent provide transparent access to the sensors and actuators. In order to be discovered in the IoT network, the information of resources needs to be uploaded to the IoT platform. Once the resource information of IoT nodes for the entities of the IoT device and IoT agent from the local network are sent to the IoT platform and saved in the database, the clients can retrieve the information to discover the resources using consistent interface in the Internet. Through the information of discovered resources, clients can access the services not only accessing directly in the same network but also accessing based on the proxy in the IoT platform. The proxy-based request to the IoT nodes is supported by the IoT platform as a function through forwarding the request in this model. For supporting interworking between different networks, the message converter is an important module to translate packets of different protocols.

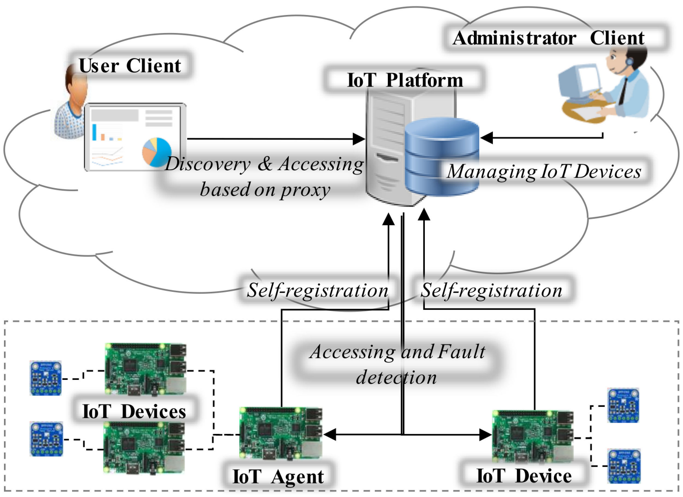


Figure 1. Internet of Things (IoT) architecture for proposed resource management.

3.1. IoT Resource Self-Registration Using Profile

The proposed IoT network is based on an OCF framework that supports solutions for IoT networks through the integrated specification. As an OCF entity, the IoT device and IoT agent include the OCF client to communicate with the OCF server of IoT server in the IoT platform for publishing the information of devices in the local network. A self-registration scheme is proposed to publish the information of devices for registering to the IoT network.

Figure 2 shows the self-registration scheme for registering the IoT devices using profiles which include the information of devices. The IoT platform is a server that provides services to enable the registration of IoT device and IoT agent. The registration service of the IoT platform enables IoT devices to be represented in the IoT network for being discovered by clients through the discovery service. In the IoT network, the devices and agent send the data to the IoT platform. Through the registration interface, the data is uploaded and saved in the storage of the IoT platform.

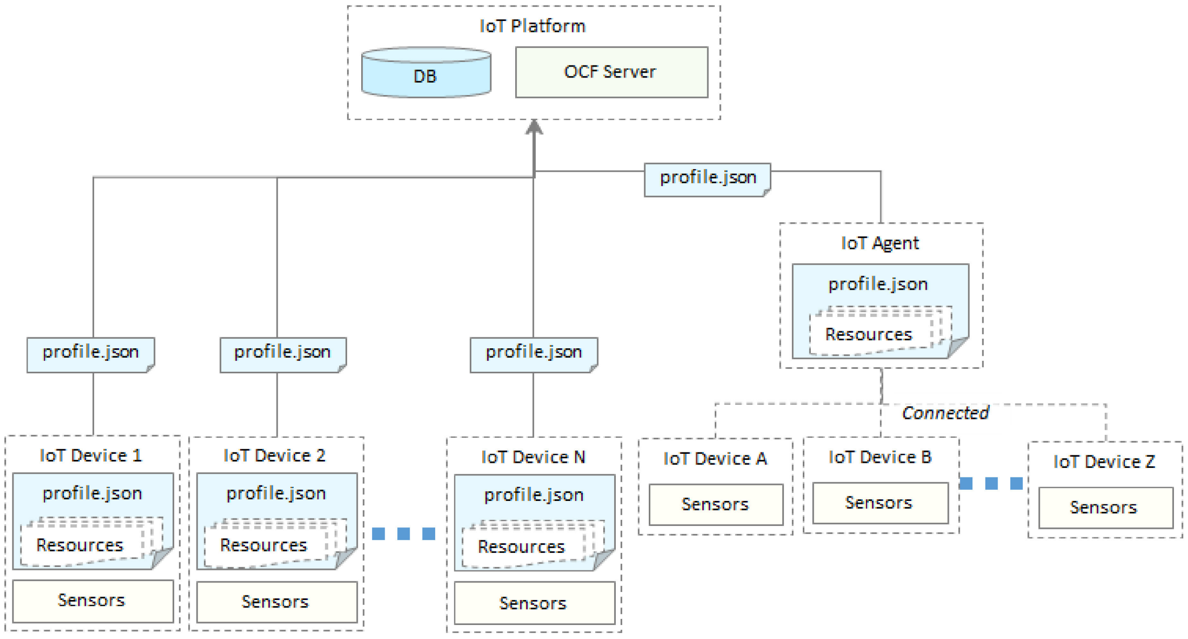


Figure 2. Self-registration using the profile in the IoT network.

According to the self-registration process, once an IoT device is started, the profile is configured and published to the IoT platform. A profile involves the information of a device including a list of resource details. The profile is formatted by JSON to be stored in a JSON file that can be easily parsed by the software in the IoT server. The profile includes the information of each IoT device that presents the information of resources as well as the device information. Based on the profile, the IoT platform exposes the accessible RESTful APIs to represent the OCF resources in the Internet. The IoT agent is used for registering the devices which only provide services without the ability of registration using OCF client. For example, in a constrained environment, some devices are equipped with sensors and communicate through the BLE. The main task for these devices is collecting sensing data. Therefore, the IoT agent as a bridge to provide the registration function for supporting functionalities of management.

Figure 3 presents the model of IoT configuration based on self-registration through an OCF network using the OCF framework. The IoT device is comprised of the IoT device application, profile, and sensors. The sensors are connected to the main board of the IoT device for supporting the sensing data. The application collects the sensing data through the sensor driver to use the data for providing sensing services by the OCF resources in the OCF server. The profile is used for publishing to the IoT server by the OCF client to register the information on the IoT platform. The registered information exposes the specification to the Internet for enabling the clients to access the resources in the IoT device. Additionally, the IoT agent is the device that is used for bridging the sensor devices to the system. The sensor devices can be deployed in the private network; therefore, the accessing to the network is allowed by requesting the OCF resources in the IoT agent. The resources of the IoT agent are mapped to the services which are provided by the sensor devices. Therefore, the profile of the IoT agent includes the information of resources in the IoT agent. However, requesting resources can acquire the sensing data that is provided by the sensor network where the sensor devices are deployed.

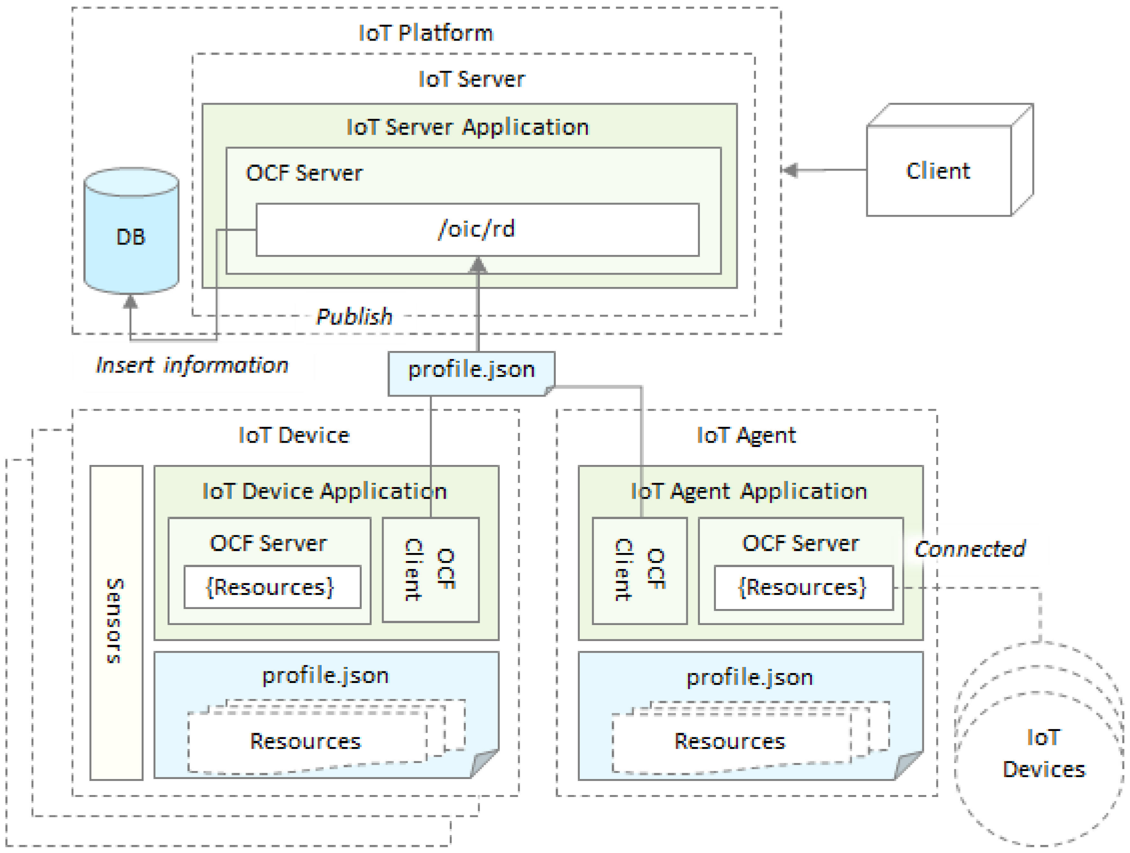


Figure 3. IoT configuration based on self-registration through an Open Connectivity Foundation (OCF) network.

The IoT platform includes the IoT server and database for the registration of IoT nodes, such as the IoT device and IoT agent which are deployed in the IoT network. The IoT server is a high-performance device which has sufficient computing ability to handle the requests from the clients and IoT nodes. The database is used for saving the information of IoT nodes and sensing data that is collected from the environment. The IoT server application includes the OCF server to handle the OCF request by the resource /oic/rd. The resource handles the request for the registration of the IoT node that involves the profile data in the payload of OCF request.

Figure 4 shows the sequence diagram for the proposed self-registration process based on the interaction of IoT node and IoT server. The IoT node has a file for its profile using a JSON format. The file is used for describing the information of endpoints to access by the client in the IoT network based on the OCF framework. In the proposed OCF network, the IoT node is an OCF client for the registration process; the node is required to do self-configuration in the onboard step. The IoT node can be deployed a space and powered by a battery to enable the tasks without human touch. The registration process can also be done by itself through preconfigured rules and parameters in the application.

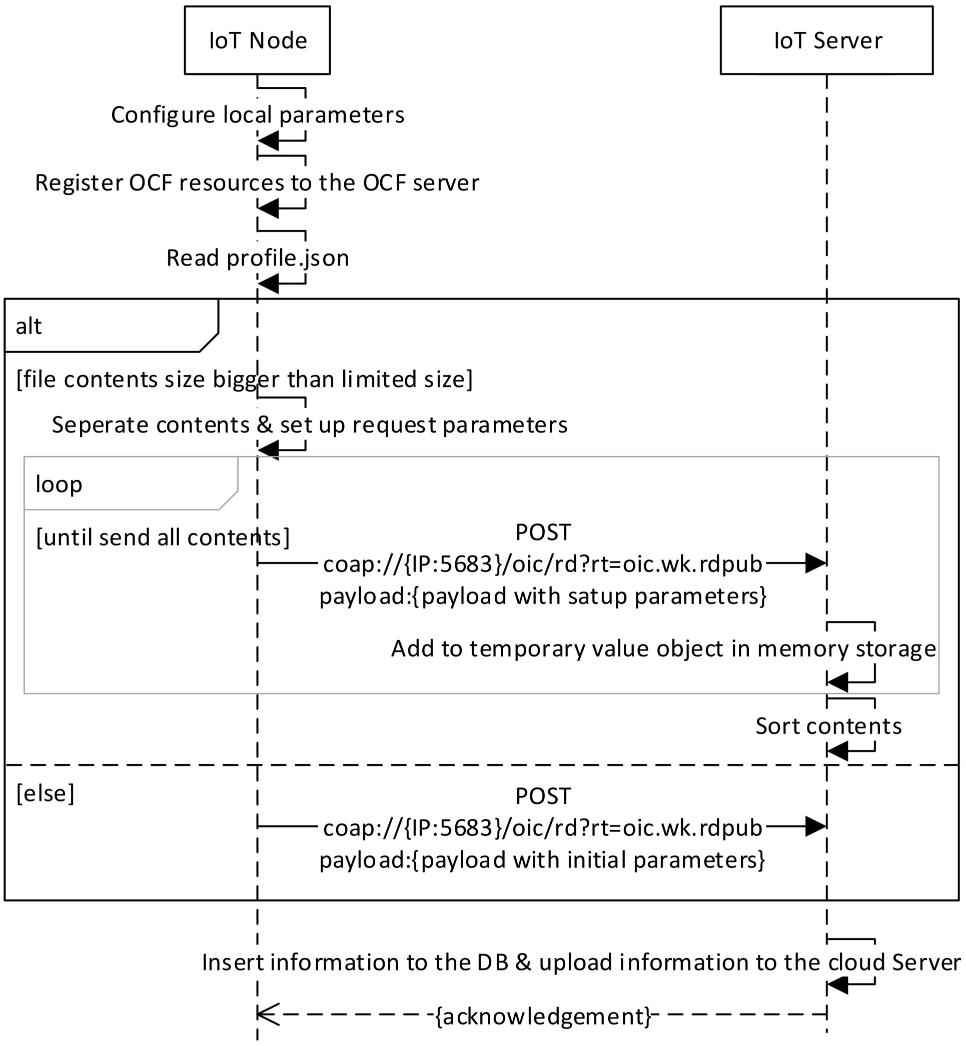
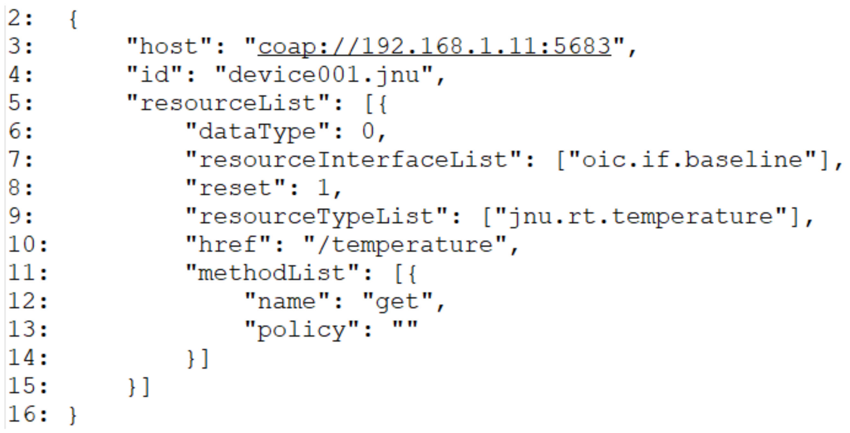

IoT Node
IoT Server
Configure local parameters
Register OCF resources to the OCF server
Read profile.json
alt
[file contents size bigger than limited size]
Seperate contents & set up request parameters
loop
[until send all contents]
POST
coap://{IP:5683}/oic/rd?rt=oic.wk.rdpub
payload:{payload with satup parameters}
Add to temporary value object in memory storage
Sort contents
[else]
POST
coap://{IP:5683}/oic/rd?rt=oic.wk.rdpub
payload:{payload with initial parameters}
Insert information to the DB & upload information to the cloud Server
{acknowledgement}

Figure 4. Sequence diagram for the proposed self-registration.

The first step, the IoT node configures the local parameters for the communications and status to understand the current environment. Through the parameters, the node prepares the OCF communication setups, including registering the OCF resource to the OCF server to provide OCF servers and generate the OCF client for requesting the IoT server for publishing profile data. The profile data is saved in profile.json, which is a file in the IoT node. The administrator can manage this file to configure the IoT node in the system. Once the IoT node is started, the node reads the file to generate the OCF payload for publishing to the IoT server. The contents of the profile that may have a big size of data which depends on the count of resources in the OCF server to provide services. The request to the IoT server cannot include contents in the payload that are too big. Therefore, The OCF client needs to send profile data separately. In the proposed registration process, if the size of the contents is bigger than the limited size, then the contents need to be separated and put in the payload with parameters. The publishing request is comprised of the POST method with the OCF message based on CoAP. The separated contents are sent by multiple requests and saved in the temporary value objects in the memory storage until all contents are sent. Once the contents are sent to the IoT server, the IoT server sorts the contents by sequence number and combines the multiple contents to a single profile. If the size is sufficient for the payload, then the contents can be sent by message. The publishing request is comprised of URI coap://{ip:5683}/oic/rd?rt =oic.wk.rdpub, which refers to the OCF specification document. The payload of the request includes the setup parameters which are generated for the separated contents or includes the initial parameters which are generated for the profile data that is sufficient to the limited size. The published profile data shall be inserted to the database and uploaded to the cloud server. Finally, the IoT node gets the acknowledgment for the successful registration process.

Figure 5 shows an example of a registration profile that is sent by an IoT node to an IoT server. The profile involves information that is used for representing a physical IoT node to the Internet to be discovered by clients. The structure of profile is based on the OCF core specification that defines a format for registering an OCF node including values of device URI, resource URI, resource type, and resource interface. Through these values, an OCF client can access an OCF resource to acquire sensing data or control an actuator by a handler of the method such as GET, PUT, POST, and DELETE. However, the information of the method is not included in the OCF-suggested profile. The proposed profile includes information which is a minimum requirement for a client to access an IoT node. In the profile, the value of "host" is a device URI that is used with a value of "href" in the JSON list "resourceList". Each item in "resourceList" is used for building an OCF request. Based on the parser, a client application presents the information and functions to a user.



```
{
    "host": "coap://192.168.1.11:5683",
    "id": "device001.jnu",
    "resourceList": [{
        "dataType": 0,
        "resourceInterfaceList": ["oic.if.baseline"],
        "reset": 1,
        "resourceTypeList": ["jnu.rt.temperature"],
        "href": "/temperature",
        "methodList": [{
            "name": "get",
            "policy": ""
        }]
    }]
}
```

Figure 5. Registration profile example.

3.2. IoT Resource Discovery and Accessing

The IoT resources are registered to the IoT platform that includes the IoT server and database for providing the registration service. The published information of IoT resources is saved in the database that enables the client to discover the IoT nodes and access the services which are provided by these IoT nodes. Through the published information, the IoT server registers the information to the IoT platform for configuring these devices in the system.

The uploaded data from the IoT nodes to the IoT platform, that not only the resource information for discovery but also the monitoring data for the resources and environment where the resources are deployed. Through the monitoring data of devices, the status of devices can be detected. Based on the data, the IoT platform provides services to the administrator clients for supporting the management functions. Also, providing services to the user clients who have the right to access the services in this system through user authentication. For supporting the synchronization of the data on the cloud, the IoT platform has a function to upload the data to the cloud; then the users can assess the information from the cloud.

Figure 6 shows the discovery model in the proposed IoT network. For the discovery functionality, the IoT platform includes the database to save the information of IoT nodes such as the IoT device and IoT agent, which are deployed in this network to provide IoT services. The database includes tables for saving information of IoT nodes as well as saving status and sending data. For providing the discovery service to the client in the Internet, the web server includes HTTP-based resources in the HTTP server to provide HTTP services to the HTTP client.

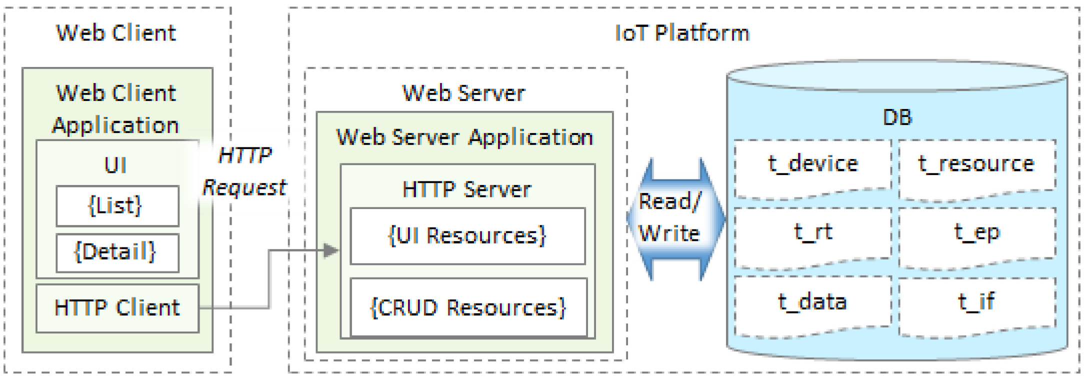


Figure 6. IoT resource discovery model.

The presented IoT network is based on the OCF communication between the IoT platform and IoT nodes. However, most of the clients in the Internet that consume the services are the HTTP-based. Therefore, in detail, a web client with a HTTP client is presented in the proposed discovery model. To provide the HTTP-based service, the IoT platform includes the web server that implements the UI resources and Create, Read, Update, Delete (CRUD) resources. The UI resources provide the interface resources to the client application to display the UI to the user. The interface resources can be an interpretable programing language such as HTML, or a data structure using JSON or XML to present the client interface through the parser of a client application. The CRUD resources provide services for supporting the interaction between clients and the database through the reading and writing the data of the database. Therefore, through the interaction with the database, the clients can access the data that is provided by the IoT nodes.

For supporting the interworking between heterogeneous communication protocol-based networks an interworking proxy is required to be included in the proposed IoT platform. The interworking proxy in the IoT platform that provides the interface to access transparently the IoT nodes in the OCF network for the clients in the Internet. For this purpose, the proxy is required to include the HTTP server and the OCF client.

Figure 7 shows the interworking model of the proxy in the IoT platform for accessing the IoT nodes to get real-time sensing data. For providing the interfaces to the HTTP client, the IoT server includes an HTTP server. The HTTP server includes an HTTP resource that needs to support the handler for the method of GET, POST, PUT, and DELETE. The handler receives the request message from an HTTP client and sends the response message to the HTTP client. Once the HTTP request message is received by the IoT server to acquire the sensing data from the OCF network, the message converter converts the HTTP message to an OCF message. For converting to the OCF message, the query parameters, and payload of the HTTP message are required to be handled. Through the converter, the proxy generates an OCF message and sends the message using the OCF client to an IoT node which in the OCF network. The IoT node returns the sensing data to the response handler of OCF client. Through the HTTP payload generator, the HTTP server can get the payload for the HTTP response message and returns the response to the HTTP client. The HTTP request message shall be URI http:// {server URI}/ocf?rt={rt}&if={if}&ocf_uri=coap:// {IP}:5683/{resource}{?query} with a payload. The HTTP handler of IoT platform parses the message and gets the URI for requesting the OCF server and generating the OCF payload. The URI of OCF request shall be coap:// {IP}:5683/{resource}{?query} and the payload is an OCF payload.

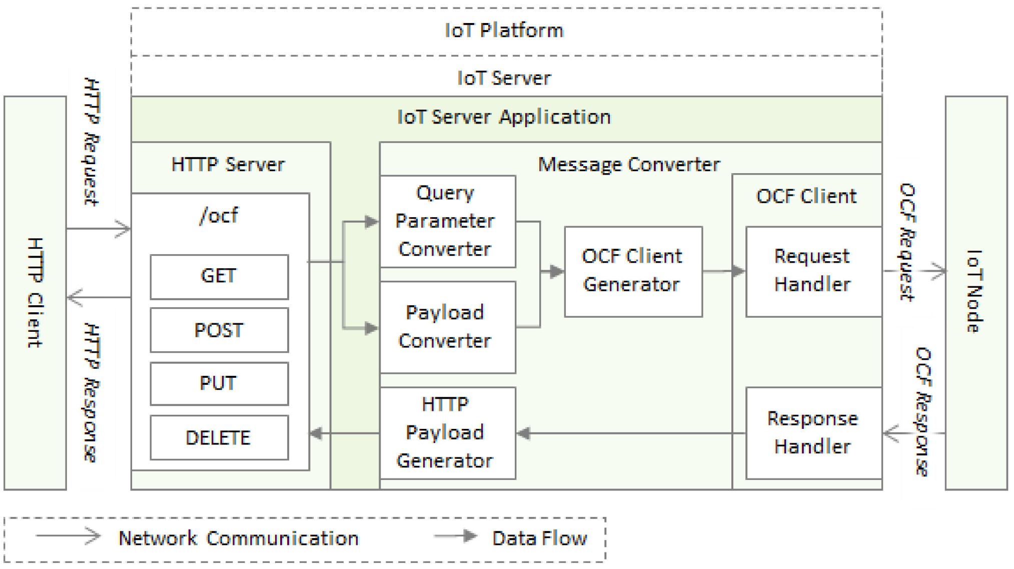


Figure 7. IoT access model based on interworking proxy.

Figure 8 shows the Entity–Relationship Diagram (ERD) for the database in the proposed platform. For storing the information of IoT nodes in the OCF network, the status of these objects and the collected sensing data, the tables t_node, t_resource, t_rt, t_ep, t_if, and t_data are designed and the database is deployed in the IoT platform regarding the designed ERD. Table t_node is used for storing the basic information of IoT nodes. This table has one too many relationships with table t_resource. Table t_resource is used for storing resource information. This table has one too many relationships with tables t_rt, t_ep, t_if, and t_data. Table t_rt is used for storing the resource type information. Table t_if is used for storing the resource interface information. Table t_ep is used for storing endpoint information. Table t_data is used for storing the sensing data. The attributes of tables t_node and t_resource are used for detecting the operating status of the device and its resources. Device information includes ID, expire time, device name, and multiple resource information; this is a resource which is used for providing a service from an IoT node. For the OCF devices, resource information includes its URI, resource type, interface, policy, and endpoint information. For presenting the device and its resources, the properties di, ttl, name, and links with properties anchor, href, rt, if, p, and eps with properties ep and pri are included in an OCF device information.

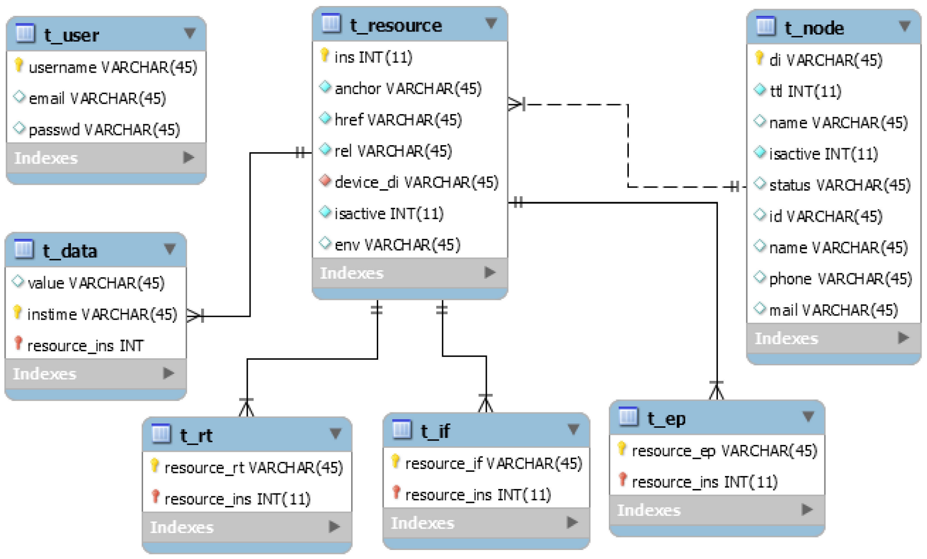


Figure 8. Entity–relationship diagram for the database in the IoT platform.

3.3. Real-Time Status Detection

Using the registered information of devices and information of its resources, the IoT server can access the services of resources. The process can be programmed to run the function periodically and proceeds the data which are returned from the IoT networks. Through the interoperability between the IoT nodes and platform, the IoT server can detect the status of these devices such as low battery, sensors, and network problems. Using the returned information, the IoT server can make real-time status detection. For example, the continuous sensing data presents useless, then, based on the preconfigured rules, the handler can make the diagnostic decision for the sensor that is related to the resource.

Figure 9 shows the real-time IoT resources access for the status detection based on a periodic sensing scheme that collects sensing data and status information from the IoT device and agent in the OCF base IoT network. The IoT server interacts with the database in the IoT platform to retrieve information of registered resources and update the status of devices. The thread pool generates the process to handle the request for accessing the resources of the devices which are electronic devices which interact with the environment and provide the services based on the resources. The thread pool retrieves the information of resources from the database and generates the OCF client to access these resources. Through the OCF client in the process that is generated by the thread pool, the handler of the OCF request gets the sensing data and status information from the devices. Once the data is returned from the OCF network, the handler of the OCF client inserts the data to the database.

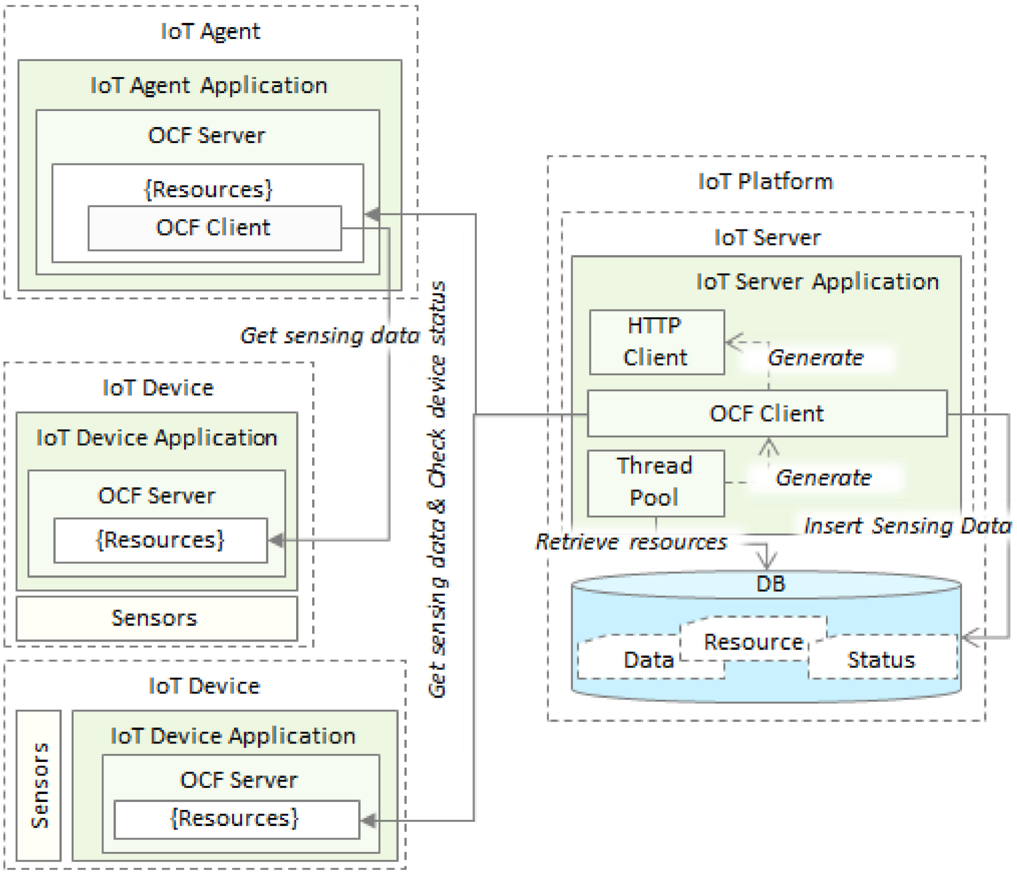


Figure 9. Real-time IoT resource access for status detection.

Figure 10 illustrates a flowchart of status detection based on real-time IoT resource access. Once the IoT server is started in the IoT platform, the server configures the OCF parameters for enabling the OCF communication with the OCF-based local network. Then the thread pool is enabled that includes functions to access the database and OCF devices. Firstly, the thread pool accesses the database to retrieve the registered device list and based on the device list, to retrieve the resource list. The resource information provides values of parameters to construct the OCF resource object for sending a request to an OCF device. For status detection, the thread pool sends the request to the status resource of an OCF device that returns the status information. If the response body is empty, meaning that the resource of the device is unreachable, the status of the device in the database is updated to be 0, otherwise, the status is updated to 1.

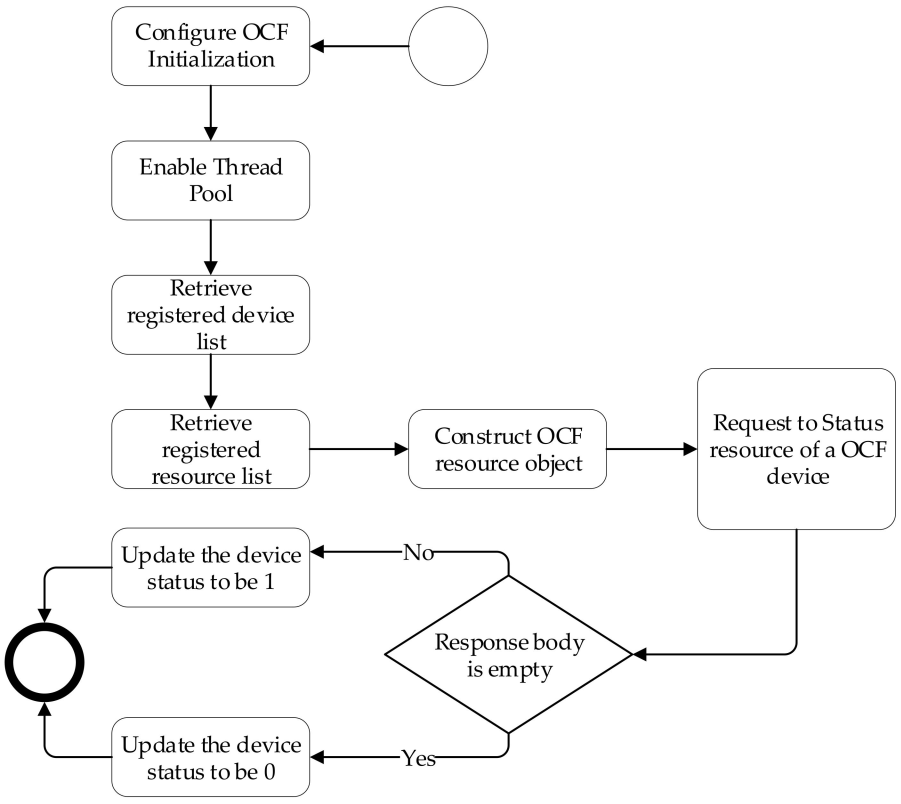

Configure OCF Initialization
Enable Thread Pool
Retrieve registered device list
Retrieve registered resource list
Construct OCF resource object
Request to Status resource of a OCF device
Response body is empty
No
Yes
Update the device status to be 1
Update the device status to be 0

Figure 10. Status detection based on real-time IoT resource access.

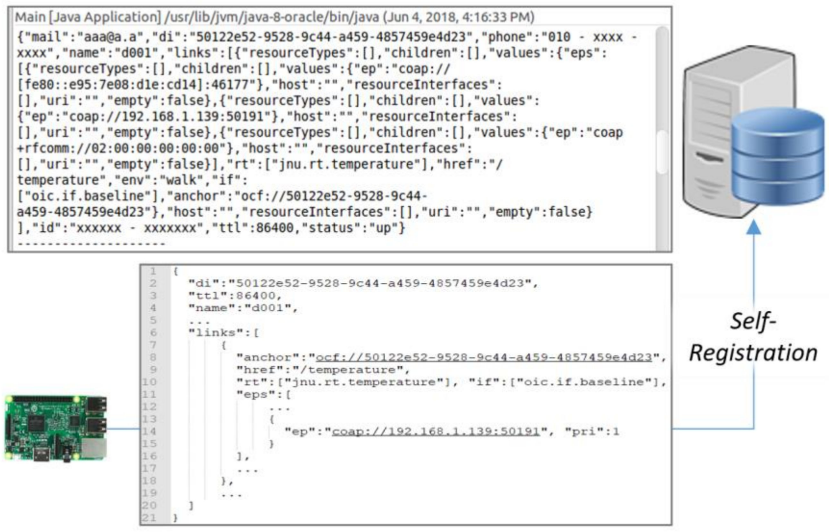
4. Implementation Results

For implementing the proposed system comprised of an IoT server, web server, device, and agent, we configured the development environment as shown in Table 1. The IoT platform is deployed based on an Ubuntu platform that involves the IoT server and web server to provide services to the clients and nodes from the Internet and the OCF-based IoT network. Both servers are deployed in Ubuntu 14.04 64-bit on a high-performance PC based on sufficient storage and stable networking. The HTTP server is implemented based on the Spring Framework 4.3 using the Spring Tool Suit (STS) 3.9.5 development tool. For providing the OCF-based service interfaces, the IoTivity 1.3.1 x86 Java version is built for the Ubuntu operating system. The user interfaces on the client, which are implemented based on jQuery 3.3.1.min. The IoT device is developed on the Raspberry Pi 3 Model B that is installed the Android Things base on Android OS. For providing sensing services, the BMP 280 is connected that is used for collecting temperature and humidity sensing data from the environment. Body temperature sensor is also supported through the Libelium e-Health Kit. The IoTivity version is 1.3.1 for the armeabi architecture and Android OS that is included in the Android Studio to develop the IoT device. The IoT agent is a smartphone that runs Android 7.0 OS on the Samsung Galaxy S6. The IoTivity version is 1.3.1 for the armeabi architecture and Android OS which is the same library with IoT device.

Table 1. Development environment of proposed entities.

Component	IoT Server	Web Server	IoT Device	IoT Agent
Hardware	PC (CPU: Intel i5-8400)		Raspberry Pi 3 Model B, BME 280, Libelium e-Health Kit	Samsung Galaxy S6
Operating Platform	Ubuntu 14.04 64bit		Android Things 0.4	Android 7.0
Development Tool	STS 3.9.5		Android Studio 3.1.3	Android Studio 3.1.3
Library and Frameworks	Spring Framework 4.3, IoTivity 1.3.1 (x86, Java), JQuery 3.3.1.min		IoTivity 1.3.1 (armeabi, Android)	IoTivity 1.3.1 (armeabi, Android)

Figure 11 shows the interaction result for self-registration using the profile. The profile is sent by an OCF device which can be an agent as well as an IoT device. The agent and device are developed based on Android that runs the Android application. The profile is a file based on a JSON format that is deployed with the application in the Android platform. The context of the file is inputted by the administrator who manages the IoT environment. Depending on the implementation of OCF resources and functions in the device, the size of the profile is increased or decreased. The formatted structure supports the parser of IoT server to read the information and configure the resources to be registered in the IoT environment.


Main [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Jun 4, 2018, 4:16:33 PM)

```
{"mail":"aaa@a.a","di":"50122e52-9528-9c44-a459-4857459e4d23","phone":"010 - xxxx -
xxxx","name":"d001","links":[{"resourceTypes":[],"children":[],"values":{"eps":
[{"resourceTypes":[],"children":[],"values":{"ep":"coap://
[fe80::e95:7e08:d1e:cd14]:46177"},"host":"","resourceInterfaces":
[],"uri":"","empty":false},{"resourceTypes":[],"children":[],"values":
{"ep":"coap://192.168.1.139:50191"},"host":"","resourceInterfaces":
[],"uri":"","empty":false},{"resourceTypes":[],"children":[],"values":{"ep":"coap
+rfcomm://02:00:00:00:00:00"},"host":"","resourceInterfaces":
[],"uri":"","empty":false}],"rt":["jnu.rt.temperature"],"href":"/
temperature","env":"walk","if":
["oic.if.baseline"],"anchor":"ocf://50122e52-9528-9c44-
a459-4857459e4d23"},"host":"","resourceInterfaces":[],"uri":"","empty":false}
],"id":"xxxxxx - xxxxxxx","ttl":86400,"status":"up"}
--------------------
```

```
{
  "di":"50122e52-9528-9c44-a459-4857459e4d23",
  "ttl":86400,
  "name":"d001",
  ...
  "links":[
      {
        "anchor":"ocf://50122e52-9528-9c44-a459-4857459e4d23",
        "href":"/temperature",
        "rt":["jnu.rt.temperature"], "if":["oic.if.baseline"],
        "eps":[
            ...
            {
              "ep":"coap://192.168.1.139:50191", "pri":1
            }
        ],
        ...
      },
      ...
  ]
}
```

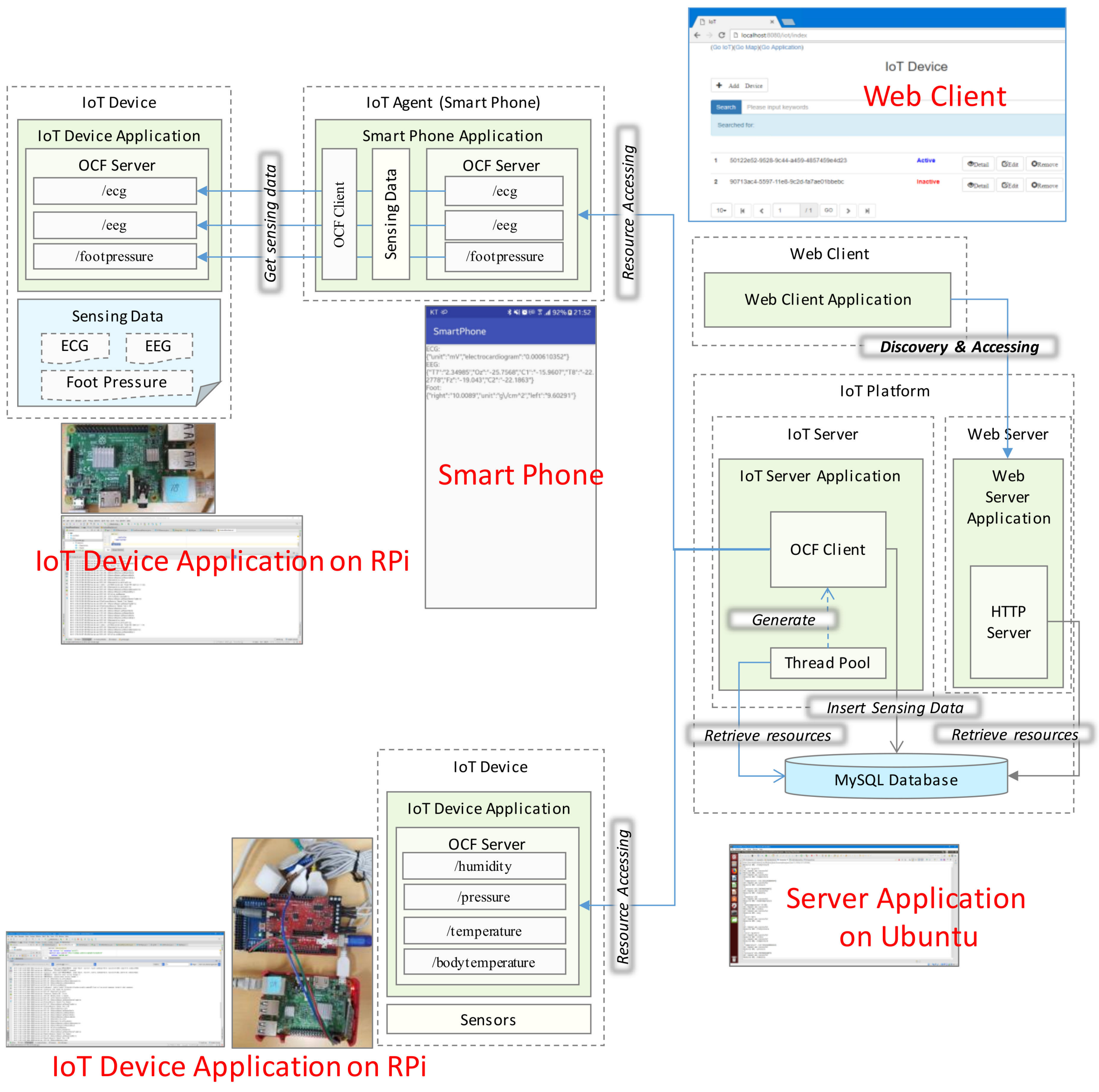
Figure 11. Interaction result for self-registration.

Figure 12 presents the experimental environment for interactions in the proposed IoT network that is comprised of IoT device, agent, web client, and IoT platform including IoT server and web server. The IoT device is implemented on Raspberry Pi running an Android Things OS to provide connections with sensors, storage, and communications through wireless and wired modes. In this experiment, Ethernet is used to provide network communication for the IoT devices. The smartphone is an Android phone that communicates with IoT devices and IoT server through the Wi-Fi. The network is provided through the Ethernet and Wi-Fi, which are provided by a network router; therefore, these entities are deployed in the same network. The IoT platform is also deployed in the network; however, the port forwarding is configured to expose services of the web server to the Internet. Therefore, the web client can access the IoT services through the web server.

Figure 13 shows network packet capture of device registration and periodic sensing data collecting. The URI for registration is /oic/rd and is shown in Figure 13. For accessing the registration service of the server, the device sends the information using OCF over the CoAP. The CoAP is based on

UDP, therefore, the figure shows the communication is done by CoAP and UDP. If the registration information is bigger than a limited size, then the message is sent separately to the server as shown in the figure. The rest part shows the process of periodic sensing data collecting. The communications of collecting sensing data are based on OCF, which are used for accessing services of OCF devices.

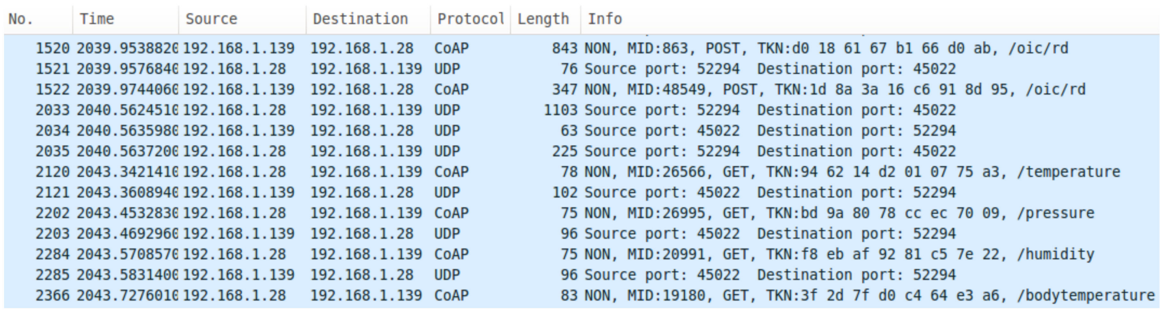
Figure 14 shows the IoT resource discovery and accessing results on the web client. The web client is provided by the web server that is deployed in the IoT platform. Through the network configuration, the services of web server are exposed in the Internet. The web client provides the user-friendly interface to users. The retrieved device list is displayed in the interface that includes the status of devices in the list. The user can retrieve the resource list of a device by clicking the presented button on the web client. The presented information supports the parameters for the request to the web server for retrieving the resource list of the device. For gathering sensing data, the user can send the request through a button that presented on the web client. The information for gathering a sensing data from a sensor resource, the web client provides the parameters of the request. The information is provided by the discovery interface, and is displayed through the web client in a user-friendly manner.



```
ECG:
{"unit":"mV","electrocardiogram":"0.000610352"}
EEG:
{"T7":"2.34985","Oz":"-25.7568","C1":"-15.9607","T8":"-22.
2778","Fz":"-19.043","C2":"-22.1863"}
Foot:
{"right":"10.0089","unit":"g\/cm^2","left":"9.60291"}
```

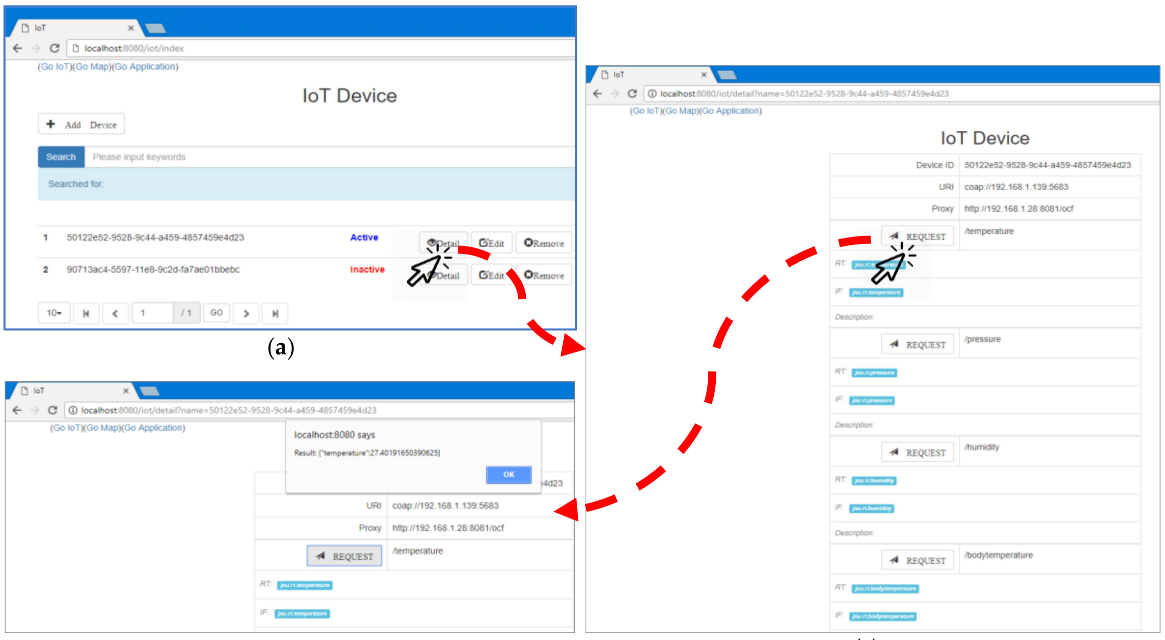
1	50122e52-9628-9c44-a459-4857459e4d23	Active	Detail	Edit	Remove
2	90713ac4-5597-11e8-9c2d-fa7ae01bbebc	Inactive	Detail	Edit	Remove

Figure 12. Experiment environment for interactions in the proposed IoT network.



```
No.   Time        Source         Destination    Protocol Length Info
 1520 2039.953882 192.168.1.139  192.168.1.28   CoAP      843 NON, MID:863, POST, TKN:d0 18 61 67 b1 66 d0 ab, /oic/rd
 1521 2039.957684 192.168.1.28   192.168.1.139  UDP        76 Source port: 52294  Destination port: 45022
 1522 2039.974406 192.168.1.139  192.168.1.28   CoAP      347 NON, MID:48549, POST, TKN:1d 8a 3a 16 c6 91 8d 95, /oic/rd
 2033 2040.562451 192.168.1.28   192.168.1.139  UDP      1103 Source port: 52294  Destination port: 45022
 2034 2040.563598 192.168.1.139  192.168.1.28   UDP        63 Source port: 45022  Destination port: 52294
 2035 2040.563720 192.168.1.28   192.168.1.139  UDP       225 Source port: 52294  Destination port: 45022
 2120 2043.342141 192.168.1.28   192.168.1.139  CoAP       78 NON, MID:26566, GET, TKN:94 62 14 d2 01 07 75 a3, /temperature
 2121 2043.360894 192.168.1.139  192.168.1.28   UDP       102 Source port: 45022  Destination port: 52294
 2202 2043.453283 192.168.1.28   192.168.1.139  CoAP       75 NON, MID:26995, GET, TKN:bd 9a 80 78 cc ec 70 09, /pressure
 2203 2043.469296 192.168.1.139  192.168.1.28   UDP        96 Source port: 45022  Destination port: 52294
 2284 2043.570857 192.168.1.28   192.168.1.139  CoAP       75 NON, MID:20991, GET, TKN:f8 eb af 92 81 c5 7e 22, /humidity
 2285 2043.583140 192.168.1.139  192.168.1.28   UDP        96 Source port: 45022  Destination port: 52294
 2366 2043.727601 192.168.1.28   192.168.1.139  CoAP       83 NON, MID:19180, GET, TKN:3f 2d 7f d0 c4 64 e3 a6, /bodytemperature
```

Figure 13. Network packet capture of registration and periodic sensing data collecting.


IoT
localhost:8080/iot/index
(Go IoT)(Go Map)(Go Application)
IoT Device
Add Device
Search
Please input keywords
Searched for:
1 50122e52-9528-9c44-a459-4857459e4d23 Active Detail Edit Remove
2 90713ac4-5597-11e8-9c2d-fa7ae01bbebc Inactive Detail Edit Remove
localhost:8080/iot/detail?name=50122e52-9528-9c44-a459-4857459e4d23
localhost:8080 says
Result: {"temperature":27.40191650390625}
OK
Device ID 50122e52-9528-9c44-a459-4857459e4d23
URI coap://192.168.1.139:5683
Proxy http://192.168.1.28:8081/ocf
REQUEST /temperature
REQUEST /pressure
REQUEST /humidity
REQUEST /bodytemperature
RT
IF
Description

(**a**) (**b**) (**c**)

Figure 14. IoT resource discovery and accessing results on web client. (**a**) IoT device discovery. (**b**) IoT resource discovery. (**c**) IoT resource accessing.

5. Performance Evaluation

For the performance evaluation of the proposed management model, the processor, memory, and network of the IoT device are monitored using the Android Profiler of the Android Studio to check the performance of monitoring and registration processes. Moreover, the network packet is captured to present the performance for the interaction between the IoT platform and resources in the OCF-based IoT network. CoAP-based communication shows that the packet size is decreased by 87% compared to the size in the case of HTTP-based communication.

Figure 15 shows a monitor of registration and sensing data collecting in the OCF device. Once the device sends the registration message to the registration server, the server needs to save the registration information to the database. Then the client can access the server to retrieve the device. The server includes a function that is used for collecting the sensing data from the registered resources of devices in the OCF network. In Figure 15, first, the network data consumption shows the registration process and, second, it shows the recursive sensing data collecting process. The registration process gives rise to a relatively increased cost that by the sensing data collecting process because in the earlier case the packet size is more than the one in a later case. In the process of monitoring, the sensing data collection almost does not consume network cost because the communication is based on the CoAP for the constrained environment.

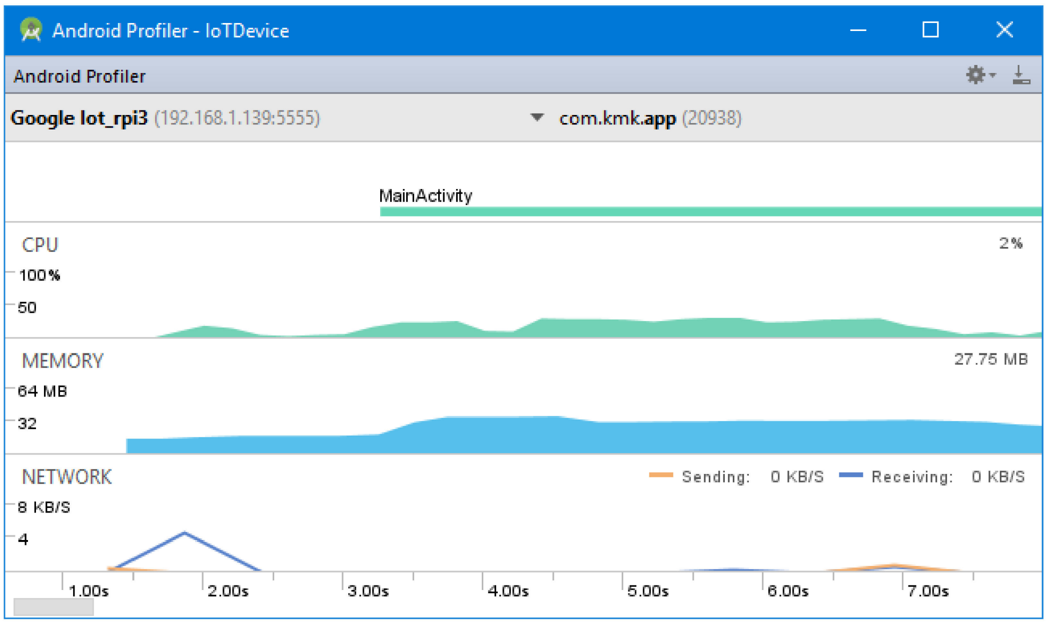


Figure 15. OCF device monitoring of registration and recursive sensing data collecting.

Figure 16 shows the interaction model of the experiment for requesting the OCF device through the IoT server. The client and IoT server communicate through the HTTP network, and the IoT server and IoT device communicate through the OCF network that is based on the CoAP. In this interaction, we had measured the message size is 589 bytes for the request from the client to the IoT server, and the message size is 233 bytes for the response from the IoT server to the client. In the OCF network, the message size is much smaller than the HTTP network. After translation from the HTTP to OCF, the request message size is 75 bytes. The response message size is 96 bytes from the IoT device.

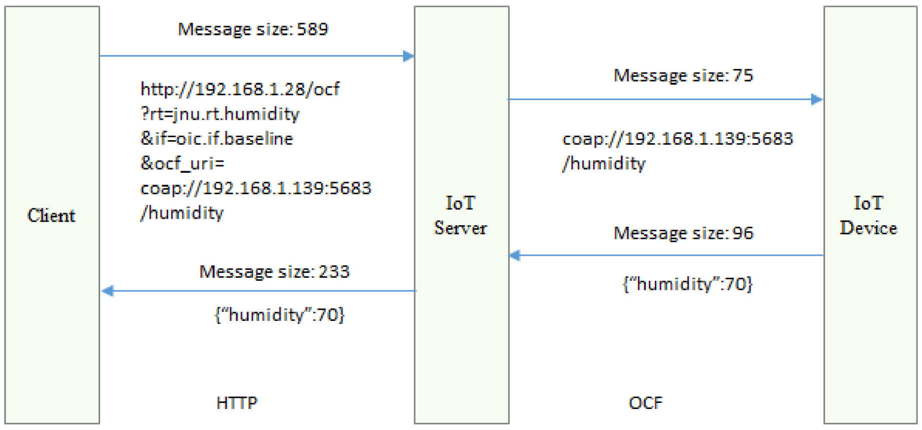
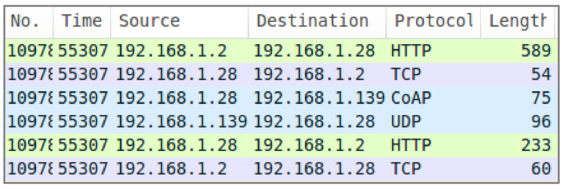


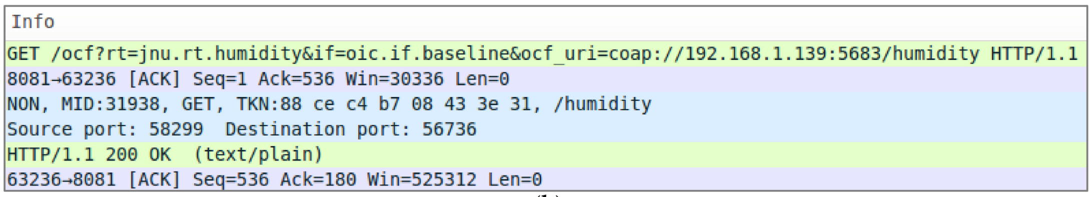
Figure 16. Interaction model for requesting an OCF device through the IoT server.

Figure 17 shows the network packet capture of requesting the OCF device through the IoT server. The capture is separated into (a), (b), and (c). Figure 17a shows the network communication specification for service accessing time, resource, destination, protocol, and length. The message is sent by the HTTP client that is a web browser for gathering the sensing data from the OCF device through the IoT gateway. Therefore, the communication network specification shows HTTP and CoAP in this communication. Figure 17b shows communication information that includes the request method, URI, and HTTP version of the HTTP request, request type, method, ID, token, and URI for the CoAP request. The size of HTTP request message from the client is 589 bytes, and the size of OCF request message is 75 bytes that is converted from the HTTP request message. The size of OCF response message is 96 bytes, and the size of HTTP response message is 233 bytes that is delivered to the client after converted from the OCF payload to the HTTP payload.



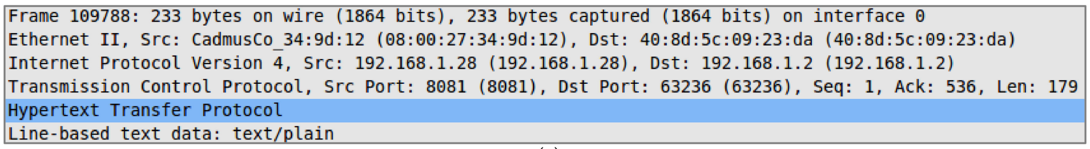
```
No.   Time  Source        Destination   Protocol Length
10978 55307 192.168.1.2   192.168.1.28  HTTP       589
10978 55307 192.168.1.28  192.168.1.2   TCP         54
10978 55307 192.168.1.28  192.168.1.139 CoAP        75
10978 55307 192.168.1.139 192.168.1.28  UDP         96
10978 55307 192.168.1.28  192.168.1.2   HTTP       233
10978 55307 192.168.1.2   192.168.1.28  TCP         60
```

(**a**)



```
Info
GET /ocf?rt=jnu.rt.humidity&if=oic.if.baseline&ocf_uri=coap://192.168.1.139:5683/humidity HTTP/1.1
8081→63236 [ACK] Seq=1 Ack=536 Win=30336 Len=0
NON, MID:31938, GET, TKN:88 ce c4 b7 08 43 3e 31, /humidity
Source port: 58299  Destination port: 56736
HTTP/1.1 200 OK  (text/plain)
63236→8081 [ACK] Seq=536 Ack=180 Win=525312 Len=0
```

(**b**)



```
Frame 109788: 233 bytes on wire (1864 bits), 233 bytes captured (1864 bits) on interface 0
Ethernet II, Src: CadmusCo_34:9d:12 (08:00:27:34:9d:12), Dst: 40:8d:5c:09:23:da (40:8d:5c:09:23:da)
Internet Protocol Version 4, Src: 192.168.1.28 (192.168.1.28), Dst: 192.168.1.2 (192.168.1.2)
Transmission Control Protocol, Src Port: 8081 (8081), Dst Port: 63236 (63236), Seq: 1, Ack: 536, Len: 179
Hypertext Transfer Protocol
Line-based text data: text/plain
```

(**c**)

Figure 17. Network packet capture of requesting OCF device through IoT server. (**a**) Network request and response packet list. (**b**) Packet details. (**c**) Hypertext Transfer Protocol (HTTP) response message.

The network packet capture of HTTP response message that is captured in the IoT server when the HTTP client sends the request message for gathering sensing data from the OCF device through IoT server. The response message size is 233 bytes, and the other information is displayed in the capture. The HTTP is based on TCP, IP, and Ethernet; therefore, the capture shows the details of those protocols.

Figure 18 shows the evaluation results for the IoT device registration Round Trip Time (RTT). In this experiment, the profile size of the IoT device is 1605 bytes. The profile is sent to the IoT server separately three times because the file size is too big. We present the evaluation results through the collection of RTT 20 times. The RTTs for each registration is done by the same hardware and network environment. The RTT is collected in the IoT device through the time difference between the time for sending the first request and the time for receiving the last response. The results illustrate the RTTs are taken by between 368 ms and 580 ms, the average is 443 ms, and the standard deviation is 61 ms. The results are great because of the process of packet combination. In the experiment, a device sends its profile by three separate packets. Therefore, the process time also includes the communication time of packet transfer. However, the proposed self-registration scheme enables a great size of the profile to be published by the constrained devices.

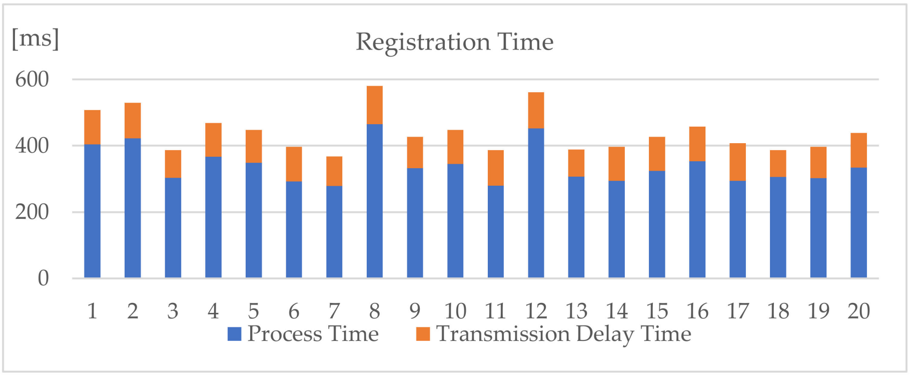


Figure 18. RTTs of IoT device registration.

6. Conclusions

We proposed an IoT resource management to provide device self-registration and status detection based on OCF in this paper. The device self-registration scheme enables devices to be registered to the IoT server by itself to be discovered by web clients from the Internet. Based on the discovered resources, a web client accesses the services which are provided by an IoT device from the OCF-based local network. For accessing the OCF service from the Internet, an interworking proxy is proposed to support the communications between web clients and OCF devices over HTTP and CoAP based on OCF. Through the interoperability with the resources using the proposed management model, real-time status detection is proposed for the awareness device status in the constrained network. According to the performance evaluation based on our implementations, the interactions are monitored for the self-registration and accessing. For self-registration, the RTTs take between 368 ms and 580 ms. The size of the profile is 1650 bytes, which is used for publishing to the IoT server. In the process of registration, the performance result presents the android application consumes more network cost compare to periodic interaction for real-time status detection; other hardware performances are approximately the same. Moreover, using the interworking proxy, the packet of CoAP-based communication is decreased by 87% compared to HTTP-based communication.

Author Contributions: W.J. and D.K. designed the overall system. W.J. implemented the overall system and performed experiments. W.J. and D.K. wrote this paper.

Funding: This research was funded by Korea government (MSIT) grant number 2018-0-01456.

Acknowledgments: This work was supported by Institute for Information & communications Technology Promotion (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), and this research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program(IITP-2017-2016-0-00313) supervised by the IITP (Institute for Information & communications Technology Promotion). Any correspondence related to this paper should be addressed to Dohyeun Kim.

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

References

1. Gartner. Available online: https://www.gartner.com/newsroom/id/3598917 (accessed on 2 May 2018).
2. Want, R.; Schilit, B.N.; Jenson, S. Enabling the internet of things. *Computer* **2015**, *48*, 28–35. [CrossRef]
3. Sheng, Z.; Mahapatra, C.; Zhu, C.; Leung, V.C.M. Recent advances in industrial wireless sensor networks toward efficient management in IoT. *IEEE Access* **2015**, *3*, 622–637. [CrossRef]
4. Jin, W.; Kim, D. Development of Virtual Resource Based IoT Proxy for Bridging Heterogeneous Web Services in IoT Networks. *Sensors* **2018**, *18*, 1721. [CrossRef] [PubMed]
5. De, S.; Barnaghi, P.; Bauer, M.; Meissner, S. Service modelling for the Internet of Things. In Proceedings of the 2011 Federated Conference on Computer Science and Information Systems (FedCSIS), Szczecin, Poland, 18–21 September 2011.
6. Shang, W.; Yu, Y.; Droms, R.; Zhang, L. *Challenges in IoT Networking Via TCP/IP Architecture*, Technical Report NDN-0038. NDN Project. 2016.
7. Guinard, D.; Trifa, V.; Karnouskos, S.; Spiess, P.; Savio, D. Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. *IEEE Trans. Serv. Comput.* **2010**, *3*, 223–235. [CrossRef]
8. Pautasso, C. RESTful web services: Principles, patterns, emerging technologies. In *Web Services Foundations*; Springer: New York, NY, USA, 2014; pp. 31–51.
9. Guinard, D.; Trifa, V.; Wilde, E. A resource oriented architecture for the web of things. In Proceedings of the Internet of Things (IOT), Tokyo, Japan, 29 November–1 December 2010; pp. 1–8.
10. Collina, M.; Corazza, G.E.; Vanelli-Coralli, A. Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST. In Proceedings of the 2012 IEEE 23rd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), Sydney, NSW, Australia, 9–12 September 2012.

11. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
12. Musaddiq, A.; Zikria, Y.B.; Hahm, O.; Yu, H.; Bashir, A.K.; Kim, S.W. A survey on resource management in IoT operating systems. *IEEE Access* **2018**, *6*, 8459–8482. [CrossRef]
13. Lu, G.; Seed, D.; Starsinic, M.; Wang, C.; Russell, P. Enabling smart grid with ETSI M2M standards. In Proceedings of the 2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Paris, France, 1 April 2012.
14. Open Connectivity Foundation (OCF). *OCF Core Specification, Version 2.0*; Open Connectivity Foundation: Beaverton, OR, USA, 22 June 2018.
15. Legner, C.; Thiesse, F. RFID-based maintenance at Frankfurt airport. *IEEE Pervasive Comput.* **2006**, *5*, 34–39. [CrossRef]
16. Tan, L.; Wang, N. Future internet: The internet of things. In Proceedings of the 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), Chengdu, China, 20–22 August 2010; Volume 5.
17. Bhatia, S.; Chauhan, A.; Nigam, V.K. The Internet of Things: A Survey on Technology and Trends. *Inf. Syst. Front.* **2016**, *17*, 261–274.
18. Sethi, P.; Sarangi, S.R. Internet of things: Architectures, protocols, and applications. *J. Electr. Comput. Eng.* **2017**, *2017*. [CrossRef]
19. Gürgen, L.; Honiden, S. Management of networked sensing devices. In Proceedings of the IEEE MDM'09 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, Taipei, Taiwan, 18–20 May 2009.
20. OMA. Available online: www.openmobilealliance.org (accessed on 12 December 2018).
21. oneM2M. Available online: www.onem2m.org (accessed on 12 December 2018).
22. OCF. Available online: https://openconnectivity.org (accessed on 12 December 2018).
23. Putera, C.A.L.; Lin, F.J. Incorporating OMA Lightweight M2M protocol in IoT/M2M standard architecture. In Proceedings of the 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, Italy, 14–16 December 2015.
24. Rao, S.; Chendanda, D.; Deshpande, C.; Lakkundi, V. Implementing LWM2M in constrained IoT devices. In Proceedings of the 2015 IEEE Conference on Wireless Sensors (ICWiSe), Melaka, Malaysia, 24–26 August 2015.
25. Open Mobile Alliance. *Lightweight Machine to Machine Architecture*; Draft Version 1; OMA: San Diego, CA, USA, 2012; pp. 1–12.
26. Daradkeh, Y.; Namiot, D.; Sneps-Sneppe, M. M2M standards: Possible extensions for open API from ETSI. *Eur. J. Sci. Res.* **2012**, *72*, 628–637.
27. Datta, S.K.; Bonnet, C. A lightweight framework for efficient M2M device management in oneM2M architecture. In Proceedings of the 2015 International Conference on Recent Advances in Internet of Things (RIoT), Singapore, 7–9 April 2015.
28. Park, H.; Kim, H.; Joo, H.; Song, J. Recent advancements in the Internet-of-Things related standards: A oneM2M perspective. *ICT Express* **2016**, *2*, 126–129. [CrossRef]
29. Datta, S.K.; Bonnet, C. Search engine based resource discovery framework for Internet of Things. In Proceedings of the 2015 IEEE 4th Global Conference on Consumer Electronics (GCCE), Osaka, Japan, 27–30 October 2015.
30. Datta, S.K.; Bonnet, C.; Nikaein, N. An IoT gateway centric architecture to provide novel M2M services. In Proceedings of the 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, Korea, 6–8 March 2014.
31. Park, S. OCF: A new open IoT consortium. In Proceedings of the 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), Taipei, Taiwan, 27–29 March 2017.
32. Bormann, C.; Castellani, A.P.; Shelby, Z. Coap: An application protocol for billions of tiny internet nodes. *IEEE Internet Comput.* **2012**, *16*, 62–67. [CrossRef]
33. Lee, J.-C.; Jeon, J.; Kim, S. Design and implementation of healthcare resource model on IoTivity platform. In Proceedings of the 2016 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 19–21 October 2016.
34. IoTivity. Available online: https://iotivity.org (accessed on 2 May 2018).

35. Varga, P.; Blomstedt, F.; Ferreira, L.L.; Eliasson, J.; Johansson, M.; Delsing, J.; de Soria, I.M. Making system of systems interoperable—The core components of the arrowhead framework. *J. Netw. Comput. Appl.* **2017**, *81*, 85–95. [CrossRef]
36. Albano, M.; Barbosa, P.M.; Silva, J.; Duarte, R.; Ferreira, L.L.; Delsing, J. Quality of service on the Arrowhead Framework. In Proceedings of the 2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS), Trondheim, Norway, 31 May–2 June 2017.
37. Varga, P.; Hegedus, C. Service interaction through gateways for inter-cloud collaboration within the arrowhead framework. In Proceedings of the 5th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (Wireless VITAE), Hyderabad, India, 13–16 December 2015.
38. Bicaku, A.; Maksuti, S.; Hegedűs, C.; Tauber, M.; Delsing, J.; Eliasson, J. Interacting with the arrowhead local cloud: On-boarding procedure. In Proceedings of the 2018 IEEE Industrial Cyber-Physical Systems (ICPS), St. Petersburg, Russia, 15–18 May 2018; pp. 743–748.
39. Zabasta, A.; Kondratijevs, K.; Kunicina, N.; Peksa, J.; Ribickis, L.; Caiko, J. Smart municipal systems and services platform development. In Proceedings of the 2016 17th International Conference on Mechatronics-Mechatronika (ME), Prague, Czech Republic, 7–9 December 2016.
40. Hegedűs, C.; Kozma, D.; Soós, G.; Varga, P. Enhancements of the arrowhead framework to refine inter-cloud service interactions. In Proceedings of the IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 23–26 October 2016.
41. Chen, R.; Guo, J.; Bao, F. Trust management for SOA-based IoT and its application to service composition. *IEEE Trans. Serv. Comput.* **2016**, *9*, 482–495. [CrossRef]
42. Li, Y. An Integrated Platform for the Internet of Things Based on an Open Source Ecosystem. *Future Internet* **2018**, *10*, 105. [CrossRef]
43. Cavalieri, S.; Salafia, M.G.; Scroppo, M.S. Realising Interoperability Between OPC UA and OCF. *IEEE Access* **2018**, *6*, 69342–69357. [CrossRef]
44. Jin, W.; Kim, D. Consistent Registration and Discovery Scheme for Devices and Web Service Providers Based on RAML Using Embedded RD in OCF IoT Network. *Sustainability* **2018**, *10*, 4706. [CrossRef]
45. Kafle, V.P.; Fukushima, Y.; Martinez-Julia, P.; Harai, H. Scalable Directory Service for IoT Applications. *IEEE Commun. Stand. Mag.* **2017**, *1*, 58–65. [CrossRef]
46. Castro, M.; Jara, A.J.; Skarmeta, A.F. Enabling end-to-end CoAP-based communications for the Web of Things. *J. Netw. Comput. Appl.* **2016**, *59*, 230–236. [CrossRef]
47. Esquiagola, J.; Costa, L.; Calcina, P.; Zuffo, M. Enabling CoAP into the swarm: A transparent interception CoAP-HTTP proxy for the Internet of Things. In Proceedings of the 2017 Global Internet of Things Summit (GIoTS), Geneva, Switzerland, 6–9 June 2017; pp. 1–6.
48. Sheng, Z.; Zhu, C.; Leung, V. Surfing the Internet-of-Things: Lightweight access and control of wireless sensor networks using industrial low power protocols. *EAI Endorsed Trans. Ind. Netw. Intell. Syst.* **2014**, *14*, e2. [CrossRef]
49. Jelicic, V.; Magno, M.; Brunelli, D.; Bilas, V.; Benini, L. Analytic comparison of wake-up receivers for WSNs and benefits over the wake-on radio scheme. In Proceedings of the 7th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks, Paphos, Cyprus, 21–22 October 2012.
50. Jin, W.; Kim, D. A Sleep Scheme Based on MQ Broker Using Subscribe/Publish in IoT Network. *Int. J. Adv. Sci. Eng. Inf. Technol.* **2018**, *8*, 639–645. [CrossRef]
51. Jin, W.; Kim, D. A Sleep-Awake Scheme Based on CoAP for Energy-Efficiency in Internet of Things. *JOIV Int. J. Inform. Vis.* **2017**, *1*, 110–114. [CrossRef]
52. Amjad, M.; Ahmed, A.; Naeem, M.; Awais, M.; Ejaz, W.; Anpalagan, A. Resource Management in Energy Harvesting Cooperative IoT Network under QoS Constraints. *Sensors* **2018**, *18*, 3560. [CrossRef] [PubMed]
53. Liu, J.; Xiong, K.; Fan, P.; Zhong, Z. RF energy harvesting wireless powered sensor networks for smart cities. *IEEE Access* **2017**, *5*, 9348–9358. [CrossRef]
54. Zheng, H.; Xiong, K.; Fan, P.; Zhou, L.; Zhong, Z. SWIPT-aware fog information processing: Local computing vs. fog offloading. *Sensors* **2018**, *18*, 3291. [CrossRef] [PubMed]

55. Ahmad, S.; Malik, S.; Ullah, I.; Fayaz, M.; Park, Do.; 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]
56. Ahmad, S.; Malik, S.; Ullah, I.; Park, D.; 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]