symmetry

MDPI

Article

A Security Generic Service Interface of Internet of Things (IoT) Platforms

Mi Kim [1,*], Nam Yong Lee [2,*] and Jin Ho Park [2]

[1] Department of Computer Science, Graduate School, Soongsil University, Seoul 06978, Korea
[2] Department of Computer Science, School of Software, Soongsil University, Seoul 06978, Korea; j.park@ssu.ac.kr
[*] Correspondence: pytwoori@gmail.com (M.K.); nylee@ssu.ac.kr (N.Y.L.); Tel.: +82-010-5518-8911 (M.K.); +82-010-5362-5656 (N.Y.L.)

Academic Editors: Young-Sik Jeong and Laurence T. Yang
Received: 24 February 2017; Accepted: 9 August 2017; Published: 26 August 2017

Abstract: Internet of Things (IoT) platforms are the key for the development of scalable IoT applications and services that connect real and virtual worlds between objects, systems, and people. However, as the IoT platform market represents a truly new market segment that was almost non-existent a few years ago, the platforms are complex and changing quickly. These IoT platforms perform simple functions such as providing useful information, and others can provide services through collaborations with IoT devices. This situation needs a generic service interface, and results in a range of IoT architectures through not only the configuration setting of IoT devices and resources but also the varied environments of collaboration of each device. Due to these heterogeneities, it is quite challenging to develop applications working with diverse IoT services, and it is even more difficult to maintain such applications. Therefore, this paper presents a security generic service interface with the effective common characteristics of an IoT platform by defining a set of generic interfaces and adopting well-known design patterns. The generic interface solves the heterogeneity-driven problems and makes it possible to effectively adopt a platform-independent Generic Interface that could be operated in diverse IoT platforms.

Keywords: IoT platform; internet of things; generic service interface; platform; framework

1. Introduction

With an Internet of Things (IoT) paradigm, applications can deliver useful services and convenience by interacting with services connected to IoT platforms. IoT platforms perform simple functions such as providing useful information, and others can provide services through collaborations with IoT devices. This situation needs the security generic service interface and results in a range of IoT architectures through not only the configuration settings of IoT devices and resources but also varied environments of collaboration for each device. Due to these heterogeneities, it is quite challenging to develop applications working with diverse IoT services, and it is even more difficult to maintain such applications. All these problems result from frequent portability and mobility among IoT devices. Therefore, this paper presents a security generic service interface with the effective common characteristics of IoT platforms by defining a set of generic interfaces and adopting well-known design patterns. The generic interface solves the heterogeneity-driven problems and makes it possible to effectively operate diverse IoT platforms.

The generic interface for an IoT platform consists of a four-layered architecture based on two generic interface schemes with its data flow at the connection to the device and the service layer. The best effort was also based on a seamless connection to the device and a set of interface headers.

The security generic service interface is proposed in order to build an IoT-based configurable resource–time and battery–consumption service platform for design. A number of things connecting it led to the proposal of a common service to create flexibility in the common characteristics where heterogeneous environments control the flow of data. The rest of this paper is organized as follows: Section 2 discusses the related work, Section 3 presents a Security Generic Service Interface, Section 4 presents a case study and evaluation, and Section 5 gives a conclusion.

2. Related Work

Most IoT management platforms, such as Philips Hue [1] and Internet of Things Environment for Service Creation and Testing (IoT.est) [2], focus on home automations or sensor networks.

By the end of 2011, more than 4.5 billion USD were spent thanks to the introduction of application programming interfaces (APIs) [3] over IPv6 over Low power Wireless Personal Area Networks (6LoWPAN). This solution was seen as highly flexible and powerful. Moreover, a reduced version of representational state transfer (RESTFul) for constrained devices was proposed as a constrained application protocol (COAP). The solutions for IoT implementation focus on uniform naming [4] and addressing, while common protocols for ubiquitous smart objects center on exposing virtual representations of physical objects to develop a common communication platform. In this way, the SENSEI project [5] introduces the concept of resource, which corresponds to a physical entity in the real world and modern power of the management of IoT [6–8].

Furthermore, the resource is related to a software process (called Resource EndPoint), which represents the physical resource in the resource layer and implements a set of resource access interfaces.

A dynamic architecture for services orchestration and self-adaptation has been proposed in the IoT.est [9]. To overcome technology and sector boundaries, dynamically design and integrate new types of services, and generate the new business opportunities mentioned, a dynamic service creates an environment that gathers and exploits data and information from sensors and actuators that use different communication technology formats. The service creation environment will enable the acquisition of sensors, objects, and actuators.

Networked Smart Objects (NOSs) collect data transmitted by different kinds of sources (i.e., nodes). The system has been designed to support both registered sources and anonymous ones, each characterized by different communication technologies and providing diverse quality levels for their computational data. They connect between the interface IoT device, a centralized platform [10]. Message Queue Telemetry Transport (MQTT) is a machine-to-machine (M2M)/IoT connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport [11]. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. NOS and MQTT retrieve them, through Hypertext Transfer Protocol (HTTP), to requests messages.

The security generic interface comparison with MQTT and NOSs are through a broker message, not only available for the heterogeneous device described but also proposed. The generic interface is defined as a driver interface and service interface without broker messages. Also, the proposed generic interface uses a common defined external agent to communicate with the heterogeneous device, among each device a connection through a common connecting message is made.

MQTT has been used in sensors communicating to a broker, via mobile link, over an occasional dial-up connection with healthcare providers as well as in a range of home automation and small device scenarios. Therefore, it is only ideal for mobile applications, however the proposed generic interface is adapted to all networks. It also has an effectively low power usage, minimized data packets, and efficient distribution of information to one.

In summary, while there are several papers available, describing exactly what a heterogeneous environment should offer from a special structure for platform, hardly any domain is available on the common capabilities of existing IoT services. However, relevant research builds upon relevant IoT application problems. Therefore, this paper tries to contribute to the security generic service interface suggestion of IoT platforms by an IoT architecture model.

2.1. Analysis of Existing IoT Platforms

Three key criteria were applied to select IoT platforms for analysis: (1) market share of an IoT platform as of today; (2) strategic positioning of a platform provider in the market; and (3) degree of platform innovation. An IoT platform can, through connection with everything in different places, communicate and cooperate with smart products.

2.1.1. Axeda

Axeda [12] was founded in 2000 and is a classical M2M solutions provider.

An IoT platform simplifies connectivity, device management, machine/sensor data management, and the development of applications that monitor, manage, and control connected machines and devices.

The Axeda Platform includes six major areas of capabilities: connectivity, data management, system management, infrastructure, application enablement services, and connected machine management. For the purpose of this paper, a machine is the business asset being connected (i.e., asynchronous transfer mode (ATM), magnetic resonance imaging (MRI), vehicle, storage equipment, etc.) and a device is the hardware used for data communication (i.e., a module, communication device, or M2M gateway business model). The Existing Internet of Things (IoT) Platforms as shown in Figure 1.

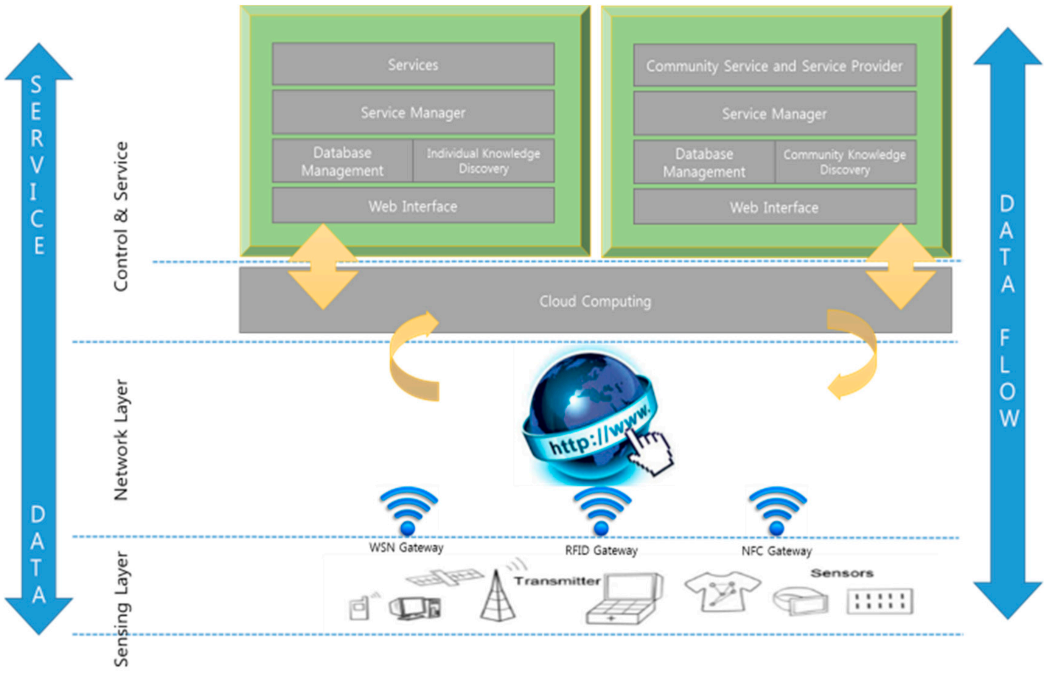

Figure 1. The existing Internet of Things (IoT) platforms scheme. WSN: Wireless Sensor Network; RFID: Radio-Frequency Identification; NFC: Near Field Communication.

2.1.2. EVRTYHNG—Enhanced Objects Intelligence

EVRYTHNG [13] is a U.S.-based IoT platform that gives physical objects an online identity. This facilitates the exchange of data in real-time throughout a product's lifecycle, thus, allowing consumers to control household appliances and personal possessions remotely, in addition to enabling insurers to lower loss ratios for home and property insurance lines.

This platform allows products to be configured in order to store information about themselves and its environment for being queried, displayed, and analyzed. In order to enable connectivity in a straightforward manner, it provides a RESTful API that connects physical objects with the virtual world of things, that is accessing, creating, storing, manipulating, sharing, or searching the Analog Devices (ADIs) for every real-world object as well as providing a set of services and tools that can be useful for such applications.

2.1.3. ThingWorx—Transversal Global Intelligence

The ThingWorx [14] platform bases its operation model in treating all things (considering people, the physical world, and systems) at the same level. This enables it to create processes to connect things in any possible combination. The platform stores information about people, environments, and systems, creating applications that evolve and grow together.

ThingWorx reduces the time, cost, and risk required to build innovative M2Mand IoT applications. ThingWorx accelerates IoT application development by compressing the design-develop-deploy cycle, reducing the time to market and spurring innovation. Cloud, On-Premise, Federated, or Embedded, ThingWorx lets you deploy exactly the way you want to fit the needs of any scenario.

2.1.4. Eclipse M2M

The Eclipse M2M [15,16] Working Group is a collaboration of companies and organizations focused on developing open-source protocols, frameworks, and tools for M2M software development. The goal of the Eclipse M2M Working Group is to make it easy for software developers to create M2M applications that are based on open standards and open-source technology. Eclipse M2M has three existing open-source projects: Koneki, Mihini and Paho, which provide open-source technology for M2M developers.

Existing IoT platforms cannot be a community solution, hence, the need to support common functions for IoT platforms.

In summary, existing IoT platforms have a usual, specific purpose but a need has developed for a common purpose for a generic platform for IoT.

Introducing proposed generic platforms that have automatic configurations of IoT Platforms as environments is not necessary for configurations of IoT devices and environments.

In the next chapter, we suggest generic architecture designs for IoT platforms in various environments, collaboration devices, and user runtime environments.

3. A Security Generic Service Interface Architecture

3.1. Framework for Collaborative Environments in IoT Platforms

There are design advantages to the security generic interface composed below:

Firstly, automatic service configuration: services can be used to compose other service environments.

Secondly, service discovery: services are supplemented with communicative meta data by which they can be effectively registered and discovered.

Finally, service reusability: logic is divided into various services to promote reuse of code.

In this section, a meta model [10] of the IoT platform is presented with an effective key to collaborative environments and relevant concepts. Through detailed surveys on IoT, the four key elements of entity objects, IoT device, IoT framework, and IoT service are derived [17,18].

3.2. Meta–Model of IoT Platform Environments

Through a detailed existing IoT platform, the effective key elements of objects, IoT device, IoT framework, and IoT service are demanded. A meta-model of the IoT platform is described by using these key elements and relations among them as shown in Figure 2.

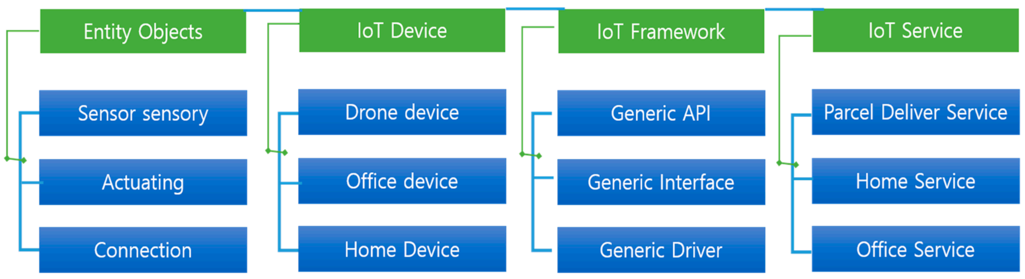

Figure 2. Meta-model of the IoT platform. API: Application Programming Interface.

Objects are a unit of things including a person, sensor, actuator, and even environments.

Each object has a set of configurations that represents the environment of objects. For instance, user acquisition data and user "light on" include user location.

IoT device is a physical entity as a flying drone device, Augmented Reality (AR)-free flight device and leap motion device. The leap motion controls the software then compares the 2D frame data in order to generate 3D positioning information. When an IoT device is set to monitor a specific entity, then one or more sensors capture the states of the target entity in the form of raw data.

IoT framework manages connections to IoT devices, acquires sensory data from IoT devices, stores them into a repository, and analyzes those services. The IoT framework should have a collaboration with IoT devices and monitoring services such as a remote heating system or parcel delivery application service similar to that of Amazon. The IoT framework determines the most efficient set of sensors and application framework for higher efficiency.

IoT service is to provide functions to its users based on active, personalized, and intelligent features.

Acquired from a set of sensors on IoT devices, each IoT service provides a convenient life for humans and a service meta model which specifies a set of target service configuration environment schemes to capture the services as a remote heating system and parcel delivery service.

3.3. Framework for Collaborative Environments

Framework development in IoT objects requires iterative processes: repository design, developing application, and configuring an IoT platform from the initial developmental stage to the final developments. Thus, IoT platforms with collaborative environments need to have structures that have accuracy configuration, manage with limited battery, and match the IoT service and IoT device. An IoT framework needs accuracy in its requirements to provide functionality.

3.4. Design of Generic Architecture

The design for the IoT platform consists of the IoT generic service interface and can be connected between the IoT service and IoT platform as shown in Figure 3.

A security generic driver interface also connected with the IoT platform, which is composed of the API components: the meta model, device manager, configuration manager, and security manager.

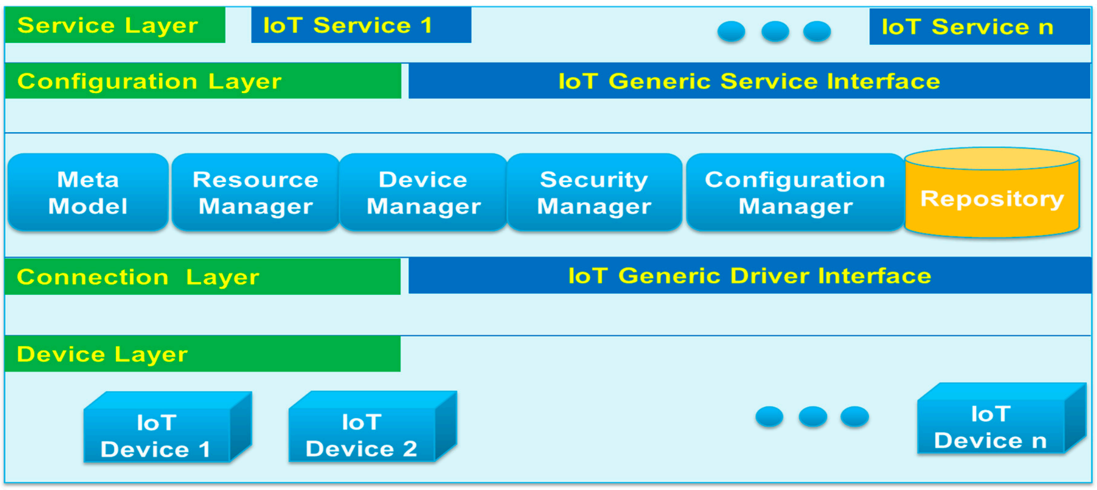

Figure 3. Generic architecture platform.

3.5. *IoT Generic Service Interface*

In this section, the IoT generic service interface for heterogeneous interfaces is suggested. It can invoke a method through one interface from heterogeneous services. To satisfy this method, a generic interface parameter is applied on the platform. This algorithm is to represent a generic interface for heterogeneous IoT service. This service has various applications that can be performed with the framework, such as drones, a wearable watch, and a smart car service. Next, Figure 4 shows a security generic service interface algorithm on the IoT platform.

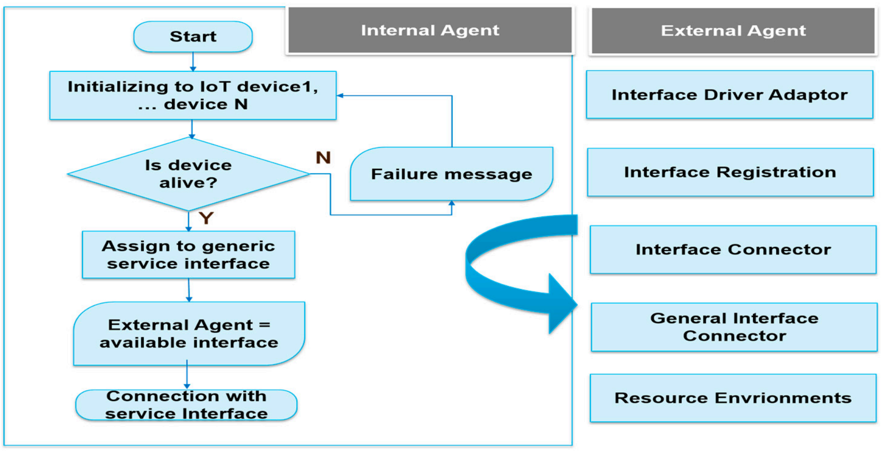
Start
Internal Agent
External Agent
Initializing to IoT device1, … device N
Interface Driver Adaptor
Is device alive?
N
Failure message
Interface Registration
Y
Assign to generic service interface
Interface Connector
External Agent = available interface
General Interface Connector
Connection with service Interface
Resource Envrionments

Figure 4. A security generic service interface.

3.6. *IoT Generic Driver Interface*

This security generic interface enables high flexibility and platform-independent communication of IoT platforms. On an IoT platform, the generic driver interface APIs can issue meta data to find out the list of drivers supported by the underlying platform and register their discovery in the driver meta data accordingly.

In the case of both the service interface and driver interface, queries are performed in the same fashion for information and notification requests, using a lightweight subset of the driver discovery and response registration query.

The device has various sensors and actuators. Algorithm 1 is a generic driver interface algorithm on IoT platforms.

Algorithm 1. Generic Driver Interface Algorithm

```
public class Generic_driver Interface
-------------------------------------------------------
public interconnect Device () {
Try {
Initializing Generic Driver Interface
Search Generic_driver_connection
Set Generic_driver_connection equal to discovery driver1
Mapping parameter driver1 interface parameter
Print driver1
For each driver assign to driver interface
    If driver connection to device
      then success device connection
          Return;
Search next device driver
    For each next driver
      If next device is discovery
          Then next driver assigns to device
     Connection close device
```

3.7. *Configuration Manager*

The collaboration manager flow assumes that IoT device information such as device name, id, or description are already stored in the repository of the IoT platforms. First, during initialization, the IoT device connects with an IoT collaboration device. After that, check the available devices and liveness and execute to check if the connected device is available in current.

If it is not an available device, it will send a failure message and if it is alive it should assign to a collaboration device, such as finding another available device as shown in Figure 5.

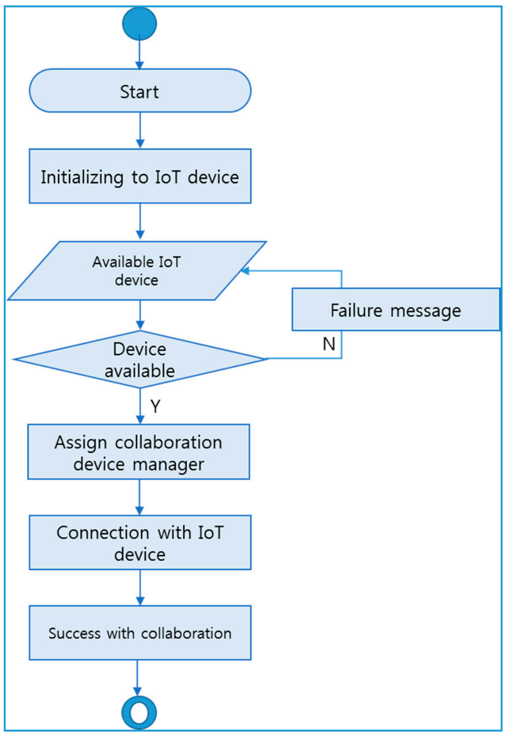
Start
Initializing to IoT device
Available IoT device
Failure message
N
Device available
Y
Assign collaboration device manager
Connection with IoT device
Success with collaboration

Figure 5. A configuration manager.

3.8. Security Manager

Secure execution environments refer to managed code and runtime environments designed to protect IoT platforms. Among a collaboration of devices, security from intermediate extortion can be needed to authenticate the procedure of a precise device. The aim of preventing intermediate extortion can be used as a design consideration of the authentication algorithm. Secure networks provide a network connection or service access only if the device is authorized. Security manager includes authenticating communicating peers, ensuring confidentiality and integrity of communicated data by preventing repudiation of the communication of transaction, and protecting the identity of the communicated entity as sensory data resources. Security for the repository involves the confidentiality and integrity of sensitive information stored in the system. The following figure is an authentication algorithm as shown in Figure 6.

For now, the service device is initialized. After that, an authentication key is assigned to the IP, session, and time. Then, the server is checked to validate the exact device. The application is able to use the stored functions by finding a real device function through a key. This method of finding, with a real device, has precision that is with a suggested algorithm.

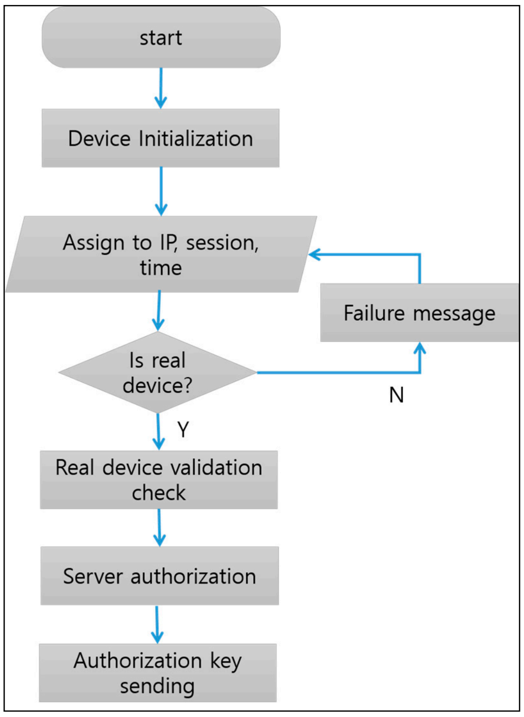
start
Device Initialization
Assign to IP, session, time
Failure message
Is real device?
N
Y
Real device validation check
Server authorization
Authorization key sending

Figure 6. Security manager.

3.9. Resource Manager

The resource manager manages resources such as memory, bandwidth, network, and limited power as energy efficiency. The resource manager process finds resources such as memory, limited battery, Wi-Fi, Near field communication (NFC), etc. After that, it finds available resources and allocates them. Check validation resources as alive resources (i.e., Wi-Fi, NFC, etc.) and leave the current network zone and physical collision while moving are not alive resources.

When alive resources finally register, the resources are in the resource manager, as shown in Figure 7.

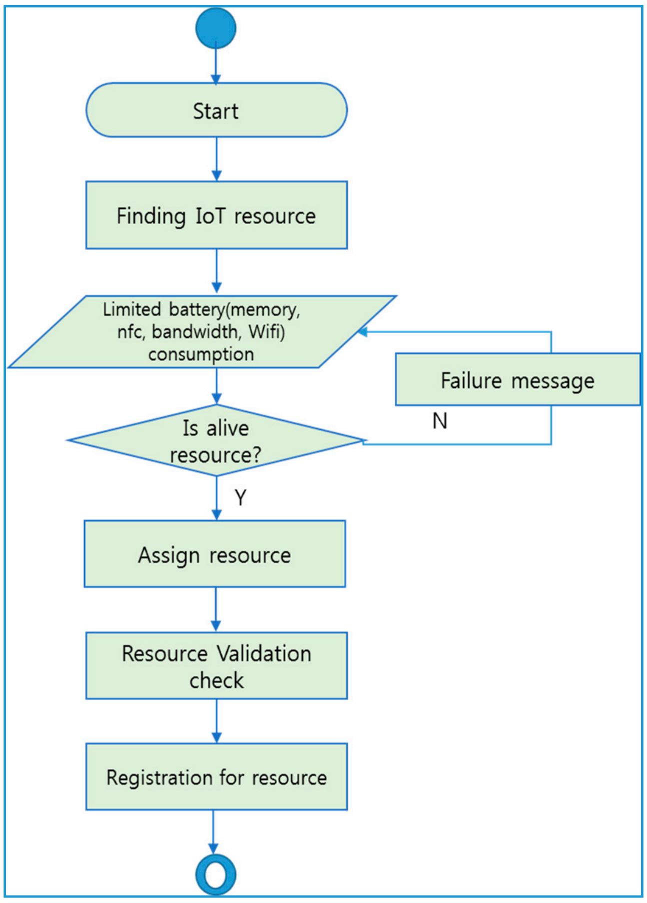
Start
Finding IoT resource
Limited battery(memory, nfc, bandwidth, Wifi) consumption
Failure message
N
Is alive resource?
Y
Assign resource
Resource Validation check
Registration for resource

Figure 7. A resource manager.

3.10. Collaboration Manager

The collaboration manager flow assumes that IoT device information such as device name, id, or description are already stored in the repository of IoT platforms. First, the initialized IoT device connects with the IoT collaboration device. After that, it checks the available devices and liveness to execute the check if the connected device is currently available.

If it is not available, the device will send a failure message and if it is alive it should assign a collaboration device such as finding another available device as shown in Figure 8.

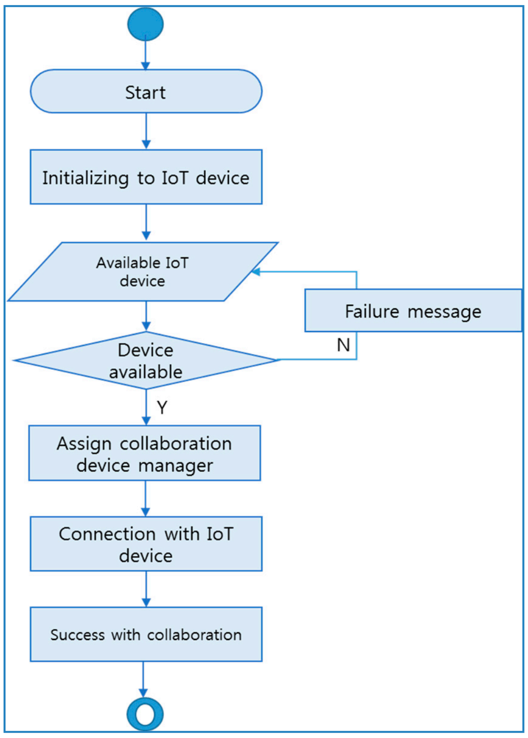
Start
Initializing to IoT device
Available IoT device
Failure message
Device available
N
Y
Assign collaboration device manager
Connection with IoT device
Success with collaboration

Figure 8. Collaboration device process.

4. A Case Study and Evaluation

4.1. A Case Study of IoT Platforms

To show the applicability of the proposed design of IoT platforms we conducted a case study by applying experiments and scenarios to IoT Platforms. We demonstrated usefulness and suggested security generic architecture for IoT Platforms and the above architecture by using a case study. We have suggested a scenario where a mobile application can be controlled according to the IoT monitoring service. It is assumed that the discovered special device driver to the end user contains discovering a light sensor and a temperature sensor. These sensors are connected to an IoT Platform. The IoT device first queries the platform to discover a driver set of sensors. In this case, the discovery reveals that the IoT monitoring service has a light sensor and a temperature sensor and both of these sensors are linked to IoT platforms. If the user selects a temperature sensor and weather domain, then the IoT monitoring service in the IoT repository needs to obtain a list of the available device drivers. The list consists of four such scenarios: (i) weather, luminosity and emotion; (ii) weather and wearable watch; (iii) weather and activities and (iv) weather, delivery service, and safety device. If the user chooses the last option, based on the temperature sensor measurement, the IoT monitoring service can deduce whether the authentic device is real.

According to the experiments, the user receives the service of exact weather temperature.

The generic interface can provide a common set for functionality.

The supposed scenario of the security generic IoT platform can easily discover the real device driver specially.

4.2. Experiments Settings

For the experiment, a generic interface is implemented with a heterogeneous interface and it deploys four different layers: service layer, configuration layer, connection layer, and device layer.

First, the service layer serves to identify home service and office service on a remote-control panel that turns on/off home light, etc. The service layer serves to analyze the user's activity and display its analytics results. It manages user's heating condition by measuring power switching by scenario. With those three IoT service layers, the experimental environment is set as shown in Figure 9.

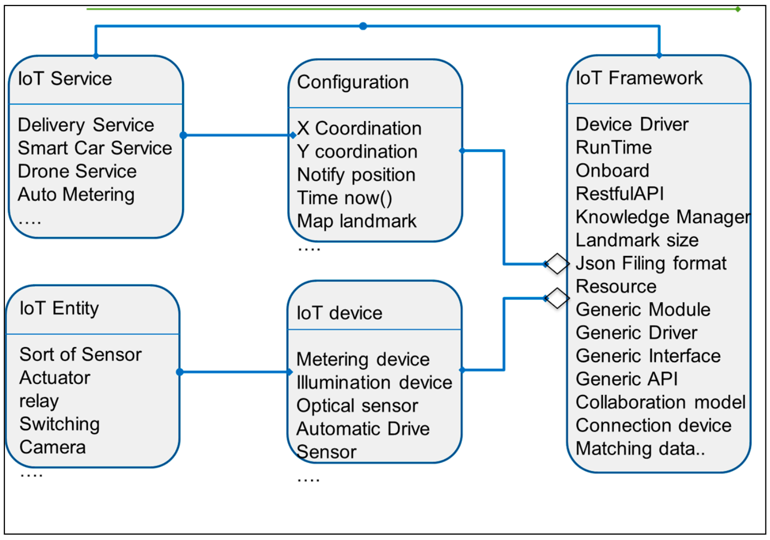

Figure 9. Framework for collaborative environments.

The IoT platform data flow on the framework is connected from various sensory data in the device layer and provides them to IoT Services based on their parameter spec. Each IoT service has a different layered architecture requirement a:

- EXPSET_IoT Service Layer = {Service, Global Positioning System(GPS)}
- EXPSET_IoT Configuration Layer = {Resource, Wi-Fi, GPS}
- EXPSET_IoT Connection Layer= {Connection, Parameter}
- EXPSET_IoT Device Layer = {Discovery, Registration}

Each type of data provides one or more device sensors as a requirement, as shown in Table 1. The cost price weight follows the result weight per response time.

Table 1. IoT requirement of IoT experiment spec.

Layer	Settings	IoT Parameter	Cost Price Weight
Service Layer	DevSET_IoT Service Layer	Cognitive WIFI	0.5
Configuration Layer	DevSET_IoT Configuration Layer	Resource, GPS	0.4
Connection Layer	DevSET_IoT Connection Layer	Connection Value1	0.3
Device Layer	DevSET_IoT Device Layer	Discovery Dev1	0.5

Each layer is set up to run the IoT parameter of cost price as its specified value by different weight. The cost price weight for the IoT parameter, such as high (0.5), medium (0.4), and low (0.2), would be weighed per response time represented.

The experiment has run each IoT layer element for measured response time efficiency and the accuracy of query for the IoT. Each layer is shown Table 2 and Figure 10.

Table 2. Results of efficiency and accuracy spec.

Settings Layer	IoT Query	Efficiency	Accuracy
DevSET_IoT Service Layer	Service Wi-Fi	95%	90%
DevSET_IoT Configuration Layer	Resource, GPS	86%	80%
DevSET_IoT Connection Layer	IoT Parameter	88%	70%
Device Layer	DevSET_IoT Device Layer	92%	90%

The next experiment is to show the data transmission of each platform on the framework for the device discovered. To do this, four types of experiments are conducted: response time rate applying the proposed security the generic IoT platform.

Each one of the four types of elements gains data transmission higher than other platforms.

The above efficient algorithms have been proposed as security generic IoT platforms and use the formal model. The proposed platform and its algorithms significantly reduce the resource consumption for CPU Usage and increase the data transmission rate for IoT platforms. The extensive experiments also support the benefits of the proposed platform.

The simulation of our proposed generic interface of platform is conducted using a discrete event java based simulator, which is designed based on a multi-server queueing system. In our simulation, it is assumed that the four operation incoming requests follow data transmission with mean light-emitting diode (LED) display brightness level consumption requests per energy. The service duration of each request follows a random distribution and the resource demand of the requests is assigned periodically. The memory requirement of the request ranges from 2-1024 MB and the CPU requirement of each request ranges from one to two cores. Similarly, the storage requirement of each request is taken to be 10 MB-10 GB.

Table 3 lists the CPU load results as measured from the Dalvik Debug Monitor Server (DDMS) Analysis of energy consumption for a specific time tool of Eclipse Android Development Tools (ADT) [19]. The loads are measured during the main four operational phases of the application.

Table 3. Power Consumption of Platform (mW).

Division	**Power Consumption(mW)**		
	Generic IoT	**EVRTYHNG**	**THINGWORX**
Sensor data	2970	3710	3920
Wi-Fi	3050	5210	4550

The IoT service operations and the CPU computations take minimal power. The dissipation on the display can be reduced by lowering the brightness value of the LED displays of the device. Figure 11 shows the energy consumption comparison of generic IoT and ThingWorx.

We have used analytic tools of power consumption in both figures to better visualize the results. Figure 12 shows the results of the response time of a generic IoT platform per layered architecture compared to ThingWorx. We knew that the measured response times are better and quicker than ThingWorx.

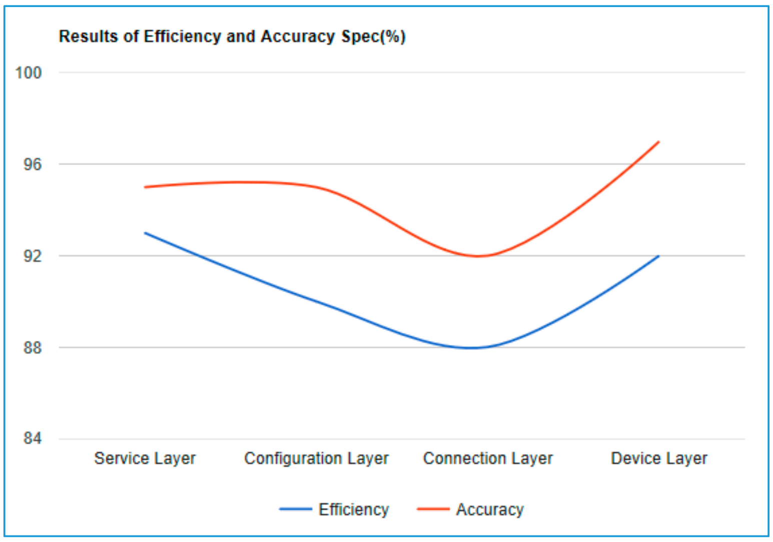

Figure 10. Results of efficiency and accuracy spec.

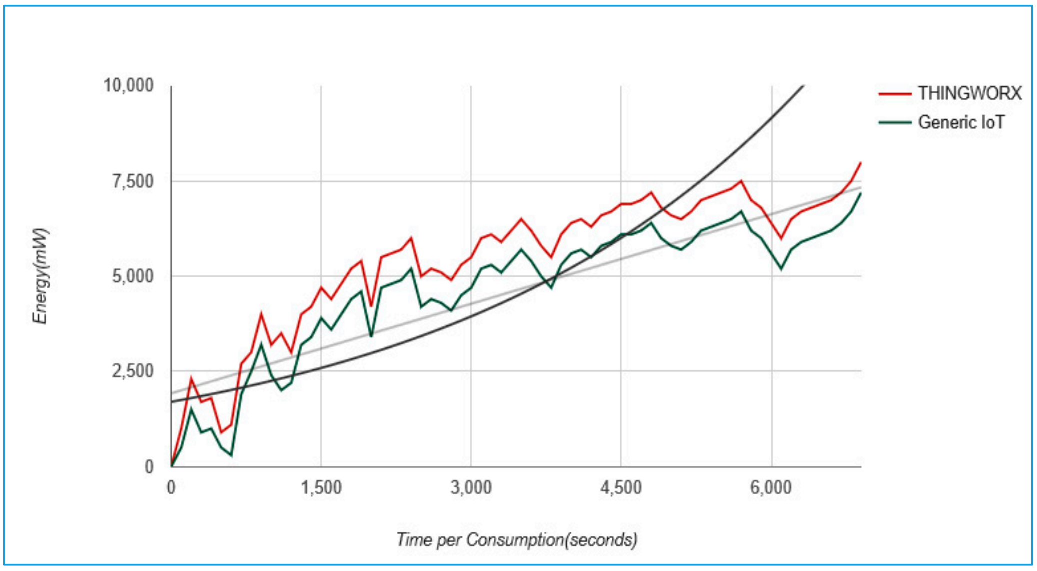

Figure 11. Power consumption of generic IoT (mW).

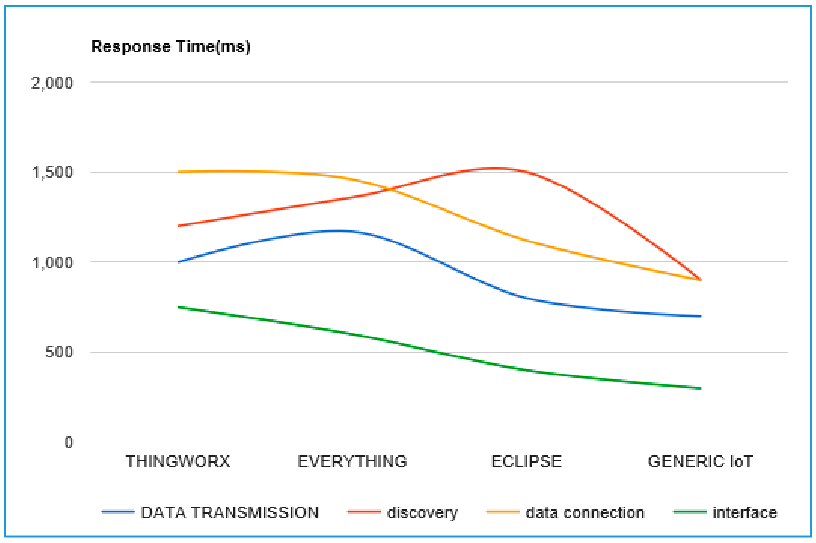

Figure 12. Data transmission time.

5. Conclusions

Due to software and hardware heterogeneities, it is quite challenging to develop applications working with diverse IoT services, and even more difficult to maintain such applications. All these problems result from frequent portability and mobility among IoT devices. Therefore, this paper presents a security generic service interface to effectively adopt the common characteristics of IoT platforms by defining a set of generic interfaces and adopting well-known design patterns. For each type, elements gain data transmissions higher than other platforms.

The above efficient algorithms have been proposed as security generic IoT platforms by using the formal model, the proposed platform, and its algorithms to significantly reduce the resource consumption for energy usage and to increase the data transmission rate for IoT platforms. The extensive experiments also support the benefits of the proposed platform. The security generic interface solves the heterogeneity-driven problems and makes it possible to effectively adopt a platform-independent generic interface that could be operated in diverse IoT platforms.

Also, the proposed generic interface, through CPU load results with data transmission, are efficient with an objective to minimize the communication cost price.

Acknowledgments: This work was supported by The Korea Educational Development Institutes and the Generic IoT project for their contributions in developing the Security Generic Interface Framework.

Author Contributions: M.K. conceived and designed the experiments; N.Y.L. and J.H.P. performed the experiments, power consumption and analyzed data and wrote the paper.

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

References

1. Philips Hue. Meet Hue. 2014. Available online: http://www.developers.meethue.com/ (accessed on 16 May 2016).
2. De, S.; Carrez, F.; Reetz, E.; Tonjes, R.; Wang, W. Test-enabled Architecture for IoT service creation and provisioning. In *The Future Internet: Future Internet Assembly 2013: Validated Results and New Horizons*; Springer: Berlin, Germany, 2013; Volume 7858, pp. 233–245.
3. Voxeo Labs Tropo Whitepaper. *Make the Shift from Telco Power to Telco Powered with the Tropo API*; Voxeo Labs: Menlo Park, CA, USA, 2013.
4. Batalla, J.M.; Krawiec, P.; Gajewski, M.; Sienkiewicz, K. ID layer for internet of things based on name-oriented networking. *J. Telecommun. Inf. Technol.* **2013**, *2*, 40–48.
5. EU FP7 SENSEI Project Consortium. *Final SENSEI Architecture Framework*, SENSEI Project Deliverable Report D3.6. 2011.
6. Maity, S.; Park, J.-H. Powering IoT Devices: A Novel Design and Analysis Technique. *J. Converg.* **2016**, *7*, 16071001.
7. Derdour, Y.; Kechar, B.; Fayçal-Khelfi, M. Using Mobile Data Collectors to Enhance Energy Efficiency and Reliability in Delay Tolerant Wireless Sensor Networks. *J. Inf. Process. Syst.* **2016**, *12*, 275–294.
8. Iam-On, N.; Boongoen, T. Generating descriptive model for student dropout: A review of clustering approach. *Hum.-Centric Comput. Inf. Sci.* **2017**, *7*, 1. [CrossRef]
9. IOT-EST Project. Available online: http://ict-iotest.eu/iotest/ (accessed on 16 May 2016).
10. Sicari, S.; Rizzardi, A.; Miorandi, D.; Cappiello, C.; Coen-Porisini, A. A secure and quality-aware prototypical architecture for Internet of Things. *Inf. Syst.* **2016**, *58*, 43–55. [CrossRef]
11. IBM, Eurotech. MQTTV3.1 Protocol Specification. Available online: http://public.Dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html (accessed on 15 April 2016).
12. Axeda Machine Cloud & M2M Platform. Available online: http://www.axeda.com/ (accessed on 11 September 2013).
13. EVRYTHNG. Available online: http://www.evrythng.com (accessed on 16 May 2016).
14. Enterprise IoT Solutions and Platform Technology. Available online: http://www.thingworx.com (accessed on 16 May 2016).
15. Eclipse M2M Charta. Available online: http://www.eclipse.org/org/industry-workgroups/m2miwg_charter.php (accessed on 16 September 2013).
16. Shamszaman, Z.U.; Lee, S.; Chong, I. WoO based user centric Energy Management System in the internet of things. In Proceedings of the 2014 International Conference on Information Networking (ICOIN), Phuket, Thailand, 10–12 February 2014.
17. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A Vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
18. Miorandi, D.; Sicari, S.; Pellegrini, F.; Chamtac, I. Internet of Things: Vision, Applications and Research Challenges. *Ad HocNetw.* **2012**, *10*, 1497–1516. [CrossRef]
19. ADT. Available online: https://developer.android.com/studio/tools/sdk/eclipse-adt.html (accessed on 1 April 2015).