DPWSim: A Simulation Toolkit for IoT Applications using Devices Profile for Web Services

Son N. Han, Gyu Myoung Lee, Noel Crespi
Department of Wireless Networks and Multimedia Services
Institut Mines-Telecom, Telecom SudParis
91011 Evry, France
{son.han, gm.lee, noel.crespi}@it-sudparis.eu

Nguyen Van Luong
Department of Software-Networks
Institut Mines-Telecom, Telecom SudParis
91011 Evry, France
van.nguyen@it-sudparis.eu

Kyongwoo Heo
Electronics and Telecommunications Research Institute
University of Science and Technology
Daejeon 305-700, Korea
hkw06@etri.re.kr

Mihaela Brut, Patrick Gatellier
Smart Systems Laboratory
Theresis, Thales Services S.A.
91767 Palaiseau, France
{mihaela.brut, patrick.gatellier}@thalesgroup.com

***Abstract*—The OASIS standard Devices Profile for Web Services (DPWS) enables the use of Web services on smart and resource-constrained devices, which are the cornerstones of the Internet of Things (IoT). DPWS sees a perspective of being able to build service-oriented and event-driven IoT applications on top of these devices with secure Web service capabilities and a seamless integration into existing World Wide Web infrastructure. We introduce DPWSim, a simulation toolkit to support the development of such applications. DPWSim allows developers to prototype, develop, and test IoT applications using the DPWS technology without the presence of physical devices. It also can be used for the collaboration between manufacturers, developers, and designers during the new product development process.**

Keywords*—*Internet of Things, DPWS, Web Service, Simulation.

I. Introduction

The Internet of Things (IoT), after years of development, has fostered the advancement of many technologies especially in the fields of low-power wireless communication and computing paradigms for resource-constrained environments. One of the approaches is to bring Web service into smart devices to seamlessly integrate their functionalities into the World Wide Web (or Web), which is predominant on today's Internet. This envisages a booming chance of new IoT applications where devices can be interconnected with a plethora of existing Web resources and services. OASIS standard Devices Profile for Web Services (DPWS) [1] is such a technology arriving to realize this vision. DPWS enables secure Web service capabilities on resource-constrained devices. It has an architectural concept similar to Web Service Architecture [2] but different in several ways to better fit in resource-constrained environments and event-driven scenarios. DPWS is based on Web Service Description Language (WSDL) [3] and Simple Object Access Protocol (SOAP) [4] to describe and communicate device services, but it does not require any central service registry such as Universal Description, Discovery and Integration [5] for service discovery. Instead, it relies on SOAP-over-UDP [6] binding and UDP multicast to dynamically discover services. DPWS offers a publish/subscribe eventing mechanism for clients to subscribe for device events, *e.g.*, a device switch is on/off or sensing when temperature reaches a predefined threshold. When an event occurs, notifications are delivered to subscribers via separate TCP connections.

DPWS is the key technology of several European projects supported by Information Technology for European Advancement (ITEA) and Framework Programme (FP) initiatives such as SIRENA, SODA, SOCRATES, and on-going IMC-AESOP and WoO. These projects have solved many technical problems and successfully released several implementations of DPWS stacks for resource-constrained devices, which preliminarily enables the adoption of the DPWS technology. However, there are currently very few development tools for applications using DPWS. We therefore have developed DPWSim, a simulation toolkit for DPWS devices to help developers to prototype, develop, and test their applications during the development process. DPWSim mimics all the software and protocol features of DPWS under an intuitive graphical user interface to provide an efficient way to simulate and manage DPWS devices. Key features of DPWSim are: (1) Platform Independence - DPWSim is written in Java programming language and can run on any machine with Java Virtual Machine installed; (2) Flexibility - There are several ways to define a new DPWS device ranging from manually creating, importing from a file to automatically generating from a physical device; (3) Transparency - IoT applications working with virtual devices provided by DPWSim can immediately work properly on physical devices without any change in code. (4) Physical Device Generation - This toolkit can generate virtual devices from physical DPWS devices; (5) Graphical User Interface - An elegant and intuitive graphical interface is a key to accelerate the development process.

The remainder of the papers is organized as follows. Section II summarizes DPWS technology. Section III presents DPWSim core components and functionalities followed by some use cases. Section IV is about the experiments of DPWSim, and section V concludes the paper.

II. Devices Profile for Web Services

DPWS was debuted in 2004 by a consortium led by Microsoft and became an integrated part of Microsoft's Windows Vista and Windows Rally (a set of technologies from Microsoft intended to simplify the setup and maintenance of wired and wireless networked devices). DPWS defines a set of implementation constraints to provide a secure and effective mechanism for describing, discovering, messaging and eventing of services for resource-constrained devices. Many global technology companies such as ABB, SAP, Schneider Electric, Siemens, and Thales have participated in Enropean research and development projects (ITEA and FP initiatives) and standardization activities (OASIS) related to the DPWS technology. Research result of the SIRENA project [7] is now available in Web Service for Devices (WS4D) website [8] to provide an open-source implementation of different DPWS stacks. Up to date, there are four DPWS stacks having been implemented and verified including WS4D-gSOAP (C), WS4D-uDPWS (C), WS4D-JMEDS (Java), and WS4D-Android (Java).

There are two types of services in DPWS: hosting service and hosted service. The former is a special service representing a device to participate in discovery, and to describe other services hosted in it. These services present the functionalities of each device and are called hosted services. DPWS uses SOAP, WS-Addressing [9], and MTOM/XOP [10] for messaging and supports SOAP-over-HTTP and SOAP-over-UDP bindings. It uses WS-Discovery [11] for discovering a hosting service (device), and WSDL to describe the hosted service (device service). It uses Web Services Metadata Exchange [12] to define metadata about the device, Web Services Policy [13] to define a policy assertion to indicate compliance of the device with DPWS, and WS-Transfer [14] to retrieve service description and metadata information about the device.

In addition to the fundamentals of DPWS, many researches on DPWS have been carried out with the consideration of several technical issues and different scenarios. It has been shown that DPWS is a promising technology to seamlessly integrate device functionalities and events into plenty of existing resources, services, and applications on the Web. DPWS thus far has been widely used in automation industry, home entertainment, and automotive systems [15] and also applicable for enterprise integration [16]. Experiments on WS4D-uDPWS stack show that DPWS is able to be implemented into (even) highly resource-constrained devices such as sensor nodes with reasonable ROM footprints [17]. Encoding and compression issues are discussed and preliminarily put under a careful consideration to improve the performance of SOAP messages in DPWS [18]. One of the important issues in IoT, the integration of DPWS into IPv6 infrastructure and 6LoWPAN, also well investigated in several works such as [19] and [20]. The scalability of service deployment was first exploited in [21] showing a prototype for a dynamic and scalable deployment of DPWS devices. The latest version of WS4D-JMEDS offers the security features by using private keys for encrypting SOAP messages. In addition, real applications adopting DPWS technology have started to gain attention such as the DPWS-based building automation system that was introduced in [22].

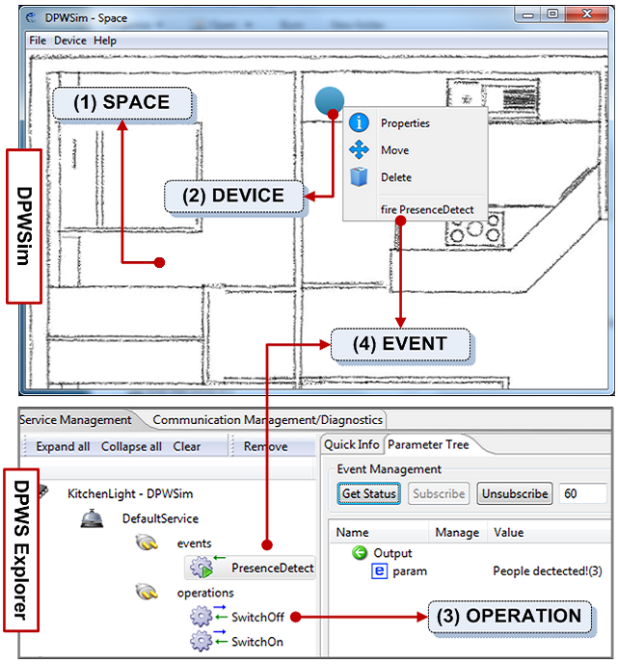

Fig. 1. Core components of DPWSim: Space, Device, Operation, and Event.

III. DPWSim: A Simulation Toolkit

DPWSim is a cross-platform simulation toolkit to support the development of IoT applications using DPWS. The core function of DPWSim is to create virtual DPWS devices, which can be discovered on the network and can communicate with other devices or clients via DPWS protocols. Besides, it can simulate environments where DPWS devices reside in. It also has a management tool to create, manage, store, and load simulations with which, it offers a high flexibility for users to render their simulations. A graphical user interface designed in Java Swing provides an intuitive way for user to interact with the their virtual devices and environments. DPWSim plays a role as a supporting toolkit alongside the main development environment of each IoT application to help developers prototyping, developing, and testing DPWS functionalities.

The following sections describe DWPSim core components and functionalities, and use cases of DPWSim uses.

A. DPWSim Components

DPWSim has four basic components namely Spaces, Devices, Operations, and Events as shown in Fig.1.

Spaces: A space is a virtual environment representing a real-life setting in which DPWS devices reside in. It can be a home, an office, a train station, a public space, or simply a stand-alone device.

Devices: A device refers to both DPWS hosting service and hosted service. Since these two kinds of services, in reality, share similar characteristics, they are used interchangeably in DPWSim for simulation purpose. It contains two different endpoint addresses used for each type of services. For example, when taking part in the discovery, it uses the device endpoint address; when invoking an operation, it uses the service endpoint address.

Operations: A device (in this case, hosted service) includes a set of operations which reflect device functionalities. These operations are described in WSDL file and can be reached via a service endpoint address.

Events: An event, similar to operations, is also the implementation of a specific device functionality designated for tasks happening proactively in devices.

B. DPWSim Core Functionalities

DPWSim provides intuitive simulation tools to help researchers and developers to build IoT applications consuming DPWS services. DPWSim can support users to create virtual environments from a simple to a complex one, even a graphically-rich interface like in the Fig.3 with the aid of external computer graphics software and design skills. DPWSim acts as a dynamic mediator to generate different types of simulation meanwhile maintaining the DPWS functionalities.

New Space/Stand-alone Device: There are two options for creating a virtual environment: stand-alone device and space. These functions can be accessed through File menu or keyboard shortcuts. A space is a composite environment to host several devices. It is created by using a plan image (office plan, home plan, etc.). A stand-alone device is simply a virtual DPWS device with a hosted service containing operations and events. This kind of virtual environment can be stored in file and re-used in other virtual environments.

New Device: Devices can be created by several ways, each is associated with a submenu of the Device menu in DPWSim: Add New (new user-customized devices), Add Predefined (pre-configured devices by DPWSim), Add From File (importing device from saved device description), and Genereate from Physical Device (creating new device by mapping functionalities of a real device to a virtual one). Fig.2 illustrates a New Device dialog window, which is automatically filled with information captured from a physical DPWS device (on Raspberry Pi). User also can further customize these values to create a new device. This capability is especially useful when developers want to focus on designing the business logic of an IoT application rather than the physical performance of devices.

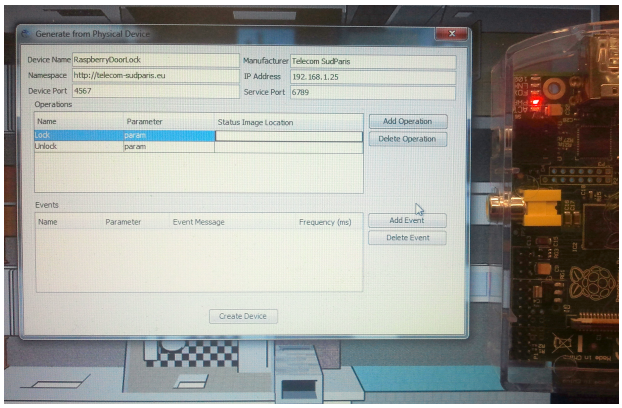

Fig. 2. A virtual device is being generated from a physical DPWS device implemented on a Raspberry Pi board.

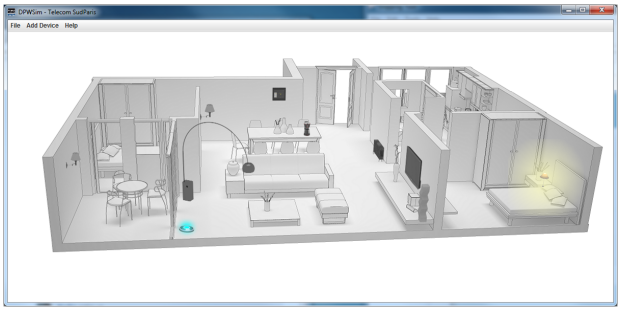

Fig. 3. A virtual home hosting several DPWS devices is designed by DPWSim with the help of a 3D artist (Sa Hoang from École Nationale Supérieure d'Architecture de Paris La Villette - ENSAPLV).

Device Management: Once a device has been created within a virtual environment or as a stand-alone device, it can be queried for DPWS information, re-located, deleted, or saved for future uses. Similarly, a virtual environment including its devices can be saved in the file system for being shared among co-workers.

To put everything all together, a home space with several devices can be created by DPWSim as shown in Fig.3. It provides an elegant simulation of devices, which are able to communicate through DPWS protocols.

C. DPWSim Use Cases

This section provides some typical scenarios in which DPWSim can be used.

Scenario one - Product Integrating: Device manufacturers can pre-provide the DPWSim-compatible **.dpws* file that describes functionalities of upcoming devices to developers. It enables them to test these devices in their real IoT applications before the official release of these products.

Scenario two - Product Protyping: Developers can also prototype new devices and new functionalities based on their application requirements without going through the complex manufacturing process. The final design then can be transferred to the manufacturer to work on it.

Scenario three - Resources Sharing: This scenario describes the situation when several teams, at the same time, develop different modules over the same devices. To solve the problem and speed up the development process, a new set of virtual devices is generated by DPWSim to share among developers. The simulation can also be used for demonstration purpose without the loss of the accuracy.

IV. DPWSim Experiments

DPWSim has been used and tested in different environments such as DPWS Explorer [8] (the standard tool of DPWS community), a Web application, and a research and development project. The following parts explain each of these experiments on DPWSim.

DPWS Explorer: DPWS Explorer is an analyzing tool for DPWS compliant services. It visualizes various aspects of both hosting and hosted service like metadata or message exchange and provides capabilities to call or subscribe to service operations and events. It is a standard tool for previewing DPWS

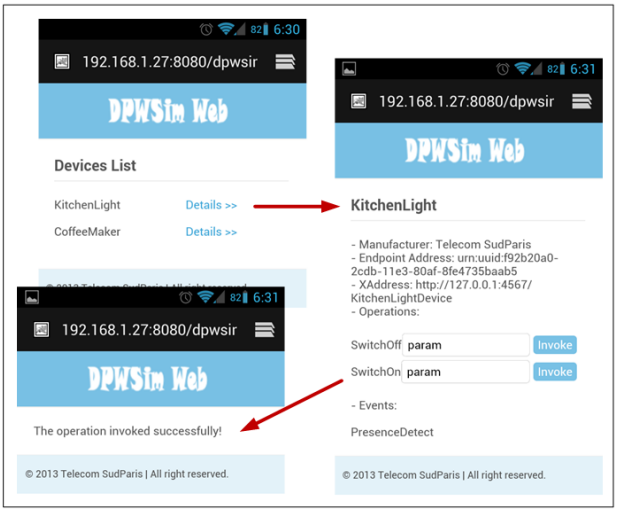

Fig. 4. A user can turn on the light bulb KitchenLight by invoking its SwitchOn operation via smartphone interface of DPWSim Web.

services during the development process. All DPWSim virtual devices can be discovered, their operations can be invoked, and their events can be subscribed from DPWS Explorer (see Fig.1).

DPWSim Web: DPWSim Web is a small Web application included in the release of DPWSim to illustrate how an application interact with DPWSim virtual environments. It is a Java Web application running on Apache Tomcat in form of a WAR file (dpwsimweb.war). Fig.4 shows a demonstration of DPWSim Web by using its smartphone interface to invoke an operation of a light bulb device.

WoO Project Scenario: DPWSim has been used within ITEA2 Web of Objects project to support the development of an incident management scenario for testing the contextual object collaboration. An intruder penetrates a restricted area and damages electrical equipment. There are three workflows from three different stakeholders. They share the common resources and will be triggered in response to the alarm: Video Analysis workflow is for intrusion detection; Fire Alert workflow is for indoor guidance such as notifying the fire agent's smartphone while providing the rights for discovering specific objects such as the closest water plug; Electrical Equipment Recovery workflow is raised by the equipment embedded sensors that will notify the smartphone of an authorized repairman, facilitating him indoor-guidance and technical record consulting. DPWSim has been used throughout the development to describe the common interface for the cooperation between devices upon the assigned rights and specific rules imposed in the whole system.

V. Conclusions

DPWSim is a cross-platform software enabling the simulation of DPWS devices and protocols. The simulation is a transparent, dynamic, and efficient channel between manufacturers and developers for speeding up the development of IoT applications using DPWS technology. To boost the adoption of DPWSim in industrial context, more experiments with complex IoT applications requiring the dynamic generation of virtual devices will be further accomplished.

Acknowledgement

This work is supported by ITEA2 Web of Objects project.

References

[1] "Devices Profile for Web Services Version 1.1," OASIS, OASIS Standard, Jul. 2009. [Online]. Available: http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01/

[2] "Web Services Architecture," W3C, W3C Working Group Note, Feb. 2004. [Online]. Available: http://www.w3.org/TR/ws-arch/

[3] "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," W3C, W3C Recommendation, Jun. 2007. [Online]. Available: http://www.w3.org/TR/wsdl20/

[4] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, "Simple Object Access Protocol (SOAP) 1.1," W3C, W3C Note, May 2000. [Online]. Available: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[5] Universal Description, Discovery and Integration. [Online]. Available: http://uddi.org/pubs/uddi-v3.0.2-20041019.htm

[6] SOAP-over-UDP. [Online]. Available: http://schemas.xmlsoap.org/ws/2004/09/soap-over-udp

[7] ITEA SIRENA Project. [Online]. Available: http://www.sirena-itea.org/

[8] Web Service for Devices Initiative. [Online]. Available: http://www.ws4d.org/

[9] WS-Addressing. [Online]. Available: http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/

[10] Message Transmission Optimization Mechanism (MTOM). [Online]. Available: http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/

[11] Web Services Dynamic Discovery. [Online]. Available: http://schemas.xmlsoap.org/ws/2005/04/discovery/

[12] Web Services Metadata Exchange (WS-MetadataExchange). [Online]. Available: http://schemas.xmlsoap.org/ws/2004/09/mex/

[13] Web Services Policy Framework. [Online]. Available: http://schemas.xmlsoap.org/ws/2004/09/policy/

[14] Web Services Transfer (WS-Transfer). [Online]. Available: http://schemas.xmlsoap.org/ws/2004/09/transfer/

[15] T. Cucinotta, A. Mancina, G. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, and F. Rusina, "A Real-Time Service-Oriented Architecture for Industrial Automation," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 3, pp. 267–277, 2009.

[16] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. Souza, and V. Trifa, "SOA-based Integration of the Internet of Things in Enterprise Services," in *IEEE International Conference on Web Services (ICWS 2009)*. IEEE, 2009, pp. 968–975.

[17] C. Lerche, N. Laum, G. Moritz, E. Zeeb, F. Golatowski, and D. Timmermann, "Implementing powerful Web Services for highly resource-constrained devices," in *2011 IEEE International Conference on Pervasive Computing and Communications Workshops*, 2011, pp. 332–335.

[18] G. Moritz, D. Timmermann, R. Stoll, and F. Golatowski, "Encoding and Compression for the Devices Profile for Web Services," in *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2010, pp. 514–519.

[19] G. Moritz, F. Golatowski, D. Timmermann, and C. Lerche, "Beyond 6LoWPAN: Web Services in Wireless Sensor Networks," *IEEE Transactions on Industrial Informatics*, 2013, Early Access Article.

[20] I. Samaras, G. Hassapis, and J. Gialelis, "A Modified DPWS Protocol Stack for 6LoWPAN-Based Wireless Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 209–217, Feb. 2013.

[21] X. Yang and X. Zhi, "Dynamic Deployment of Embedded Services for DPWS-enabled Devices," in *2012 Int. Conf. on Computing, Measurement, Control and Sensor Network (CMCSN)*, 2012, pp. 302–306.

[22] S. N. Han, G. M. Lee, and N. Crespi, "Semantic Context-aware Service Composition for Building Automation System," *IEEE Transactions on Industrial Informatics*, 2013, forthcoming.