An agent-based simulation of SOA-ready devices

Stamatis Karnouskos , Mian Mohammad Junaid Tariq

SAP Research
Vincenz-Priessnitz-Strasse, 1 Karlsruhe, Germany
E-mail: {stamatis.karnouskos, mian.mohammad.junaid.tariq} @sap.com

Abstract

In the future Internet of Things, intelligent embedded devices are expected to not only offer their functionality as a web service, but also to be able to discover and cooperate with other devices and services in a peer-to-peer way. The new capabilities of the emerging infrastructure will have a significant impact on industrial applications, services and business practices. In this paper we explore the simulation of such an infrastructure composed of heterogeneous web-service enabled (SOA-ready) devices with the help of a multi-agent system.

1. Introduction

We are witnessing the miniaturization of computing devices, and the expansion of their computing as well as communication capabilities. In the envisioned "Internet of Things" [1] large numbers of distributed networked embedded devices (NEDs) will be able to collaborate autonomously in order to achieve their goals. As embedded devices are becoming more sophisticated we see slowly a paradigm change characterized mostly from the efforts to migrate advanced functionality previously hosted in powerful static back-end systems, towards more lightweight mobile distributed embedded devices.

Interoperability will be a major challenge in this highly heterogeneous infrastructure. Although special flavors of the Internet Protocol (such as the 6lowpan [2]) are expected to efficiently connect large populations of these devices with each other, being able to seamlessly support interoperable Machine to Machine (M2M) interaction over the network is a must in order to provide wide access to its resources and functionality. A concept successfully tested in the business environment to tackle interoperability is that of Service-Oriented Architectures (SOA) where web services are used in order to focus on the functionality rather than the underlying implementation.

Web services nowadays can be implemented directly on devices, providing them with the necessary technology abstraction and making them easily integrateable in heterogeneous environments. As an example the SOCRADES project [3] is implementing web services on devices and integrates them with modern enterprise systems [4]. By including the NEDs in enterprise applications, new innovative services are expected to empower business solutions and provide new approaches to known problems that were not possible today due to the missing granularity and real-time delivery of information such the one swarms of NEDs can deliver. Therefore it is expected that services depending on these devices will be vital for future business scenarios in a number of industry domains [5].

2. Requirements analysis

Simulating an infrastructure populated by a high number of web service enabled devices is not trivial, but it could provide a very useful tool in the hands of enterprise application developers. In order to achieve simulation of even the most basic characteristics of it, some basic requirements need to be fulfilled.

- Transparent simulation devices and their functionality. This implies that the simulated devices should not be distinguished in their behavior from the devices they actually simulate. This implies web-service direct access to the simulated devices, dynamic discovery of their services, management of their lifecycle etc.
- Mobility support: trends in shop-floor show an increasing penetration of mobile devices. As such simulating the mobility/roaming of devices and their services in distributed should be feasible.
- Dynamic Environments: the new infrastructure is expected to be highly volatile since it will be composed of multiple heterogeneous devices and services that will appear or disappear/fail etc. As such any simulator should be able to provide a flexible

way of trying out such non-deterministic models.

- Micro and macro simulation support: Micro simulation techniques model behaviors of individuals and fit very well in the envisioned infrastructure. The web service enabled devices are expected not only to provide their functionality as a service, but compose sophisticated services running also in other devices and/or enterprise systems and cooperate with them. Therefore we want to be able to simulate swarm behaviors as a system (macro) but also as individuals (micro).
- Self-X behavior: Future infrastructures will depict self-X characteristics e.g. self-configuration (automatic configuration of components), self-healing (automatic discovery and correction of faults), self-optimization (automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements) and self-protection (proactive identification and protection from arbitrary attacks). Therefore our simulation approach should be able to capture this dimension.

Of course there are many other requirements if we were to fully simulate all features of such a dynamic complex infrastructure. However our aim is to start with some very initial ones and provide the first to our knowledge such simulator of its kind. Have also in mind that our interest is more on the enterprise integration side and the effects it might have on concepts and integration of business applications.

3. Technologies

Picking up the right technology is always a challenging issue as they have an impact on the overall architecture. We focus here mostly on the choice of technologies for the web-service enabled devices and their simulator.

3.1 Web services for devices

In the past there have been efforts (e.g. Jini, UPnP) to integrate devices into the networking world and make their functionality available in an interoperable way. The latest one, coming from UPnP and attempting to fully integrate with the web-service world, is DPWS (Devices Profile for Web Services [6]), which defines a minimal set of implementation constraints to enable secure Web Service messaging, discovery, description, and eventing on resource-constrained devices. DPWS builds on several core Web Services standards such as WSDL 1.1, XML Schema, SOAP 1.2, WS-Addressing, WS-Metadata Exchange, WS-Transfer, WS-Policy, WS-Security, WS-Discovery and WS-Eventing.

The main issues associated with DPWS and relevant to our requirements are i) it runs on resource-constrained devices, ii) allows dynamic discovery of devices and services running on devices, iii) developer implementations of it are available as open source in Java and C. A DPWS implementation (WSDAPI) is also included by default in Microsoft Windows Vista and Windows Embedded CE.

Although primarily developed for the home and office environment, it is being piloted also in other domains [9] such as the automation one [7] by major industrial players. Initial efforts indicate positive results, and therefore it is expected that in the future many devices and their services will be able to be discoverable in a web-service enabled way [8]. DPWS provides a solid framework that satisfies our requirements.

3.2 Multi-agent system simulator

Agents are considered one of the most important paradigms for conceptualizing, designing, implementing and simulating software systems. Multi-agent systems (MAS) [10] support group behavior of agents in dynamic situations, and are capable of simulating systems with large number of heterogeneous entities behaving differently. As such MAS are more suitable for evaluating distributed systems that involve complex interaction between entities, e.g., humans, industrial robots, smart devices. As agents are autonomous and operate without human intervention MAS can model really complex non-deterministic systems governed by common and possibly even conflicting goals.

Taking into account our requirements, the availability of DPWS implementations and considering that our focus is not on the core protocol simulation but on the devices using it, we concluded that multi-agent systems are powerful enough to simulate the complex infrastructure envisioned.

Due to the fact that significant research has already been done especially on beliefs, desires, and intentions (BDI), cooperation, and coordination, organization, communication, negotiation, dependability and fault-tolerance, we could build on top of that to simulate a very dynamic, distributed and highly heterogeneous infrastructure composed of large numbers of autonomous devices that can offer their functionality and consume web services. To our knowledge there is no simulator available for the DPWS protocol nor for web-service enabled devices. Therefore, we need to develop a new one, preferably as an extension to one of the existing frameworks, and in our case using MAS.

For the work presented in this paper we have used the Java Agent Development framework (JADE [11]), which is a platform allowing the coordination of multiple FIPA-compliant agents. Although in principal any other multi-agent system could be used, we concluded using this one due to its Java

implementation, FIPA compatibility, open source availability, extensibility and the tools it comes with.

4. Architecture

The envisioned overall architecture making use of the simulator is depicted in Figure 1. We can clearly see three layers. At the bottom layer there are the devices (device layer), whose functionality is available via web services. These devices directly implement web-services (e.g. via the DPWS protocol) or if this is not possible (e.g. due to resource constraints) they are connected via a gateway that is capable of doing so. Typical examples of devices that implement web services (SOA-ready) are PLCs, robotic arms [7] etc, while legacy devices that connect via a gateway could be RFID tags etc.

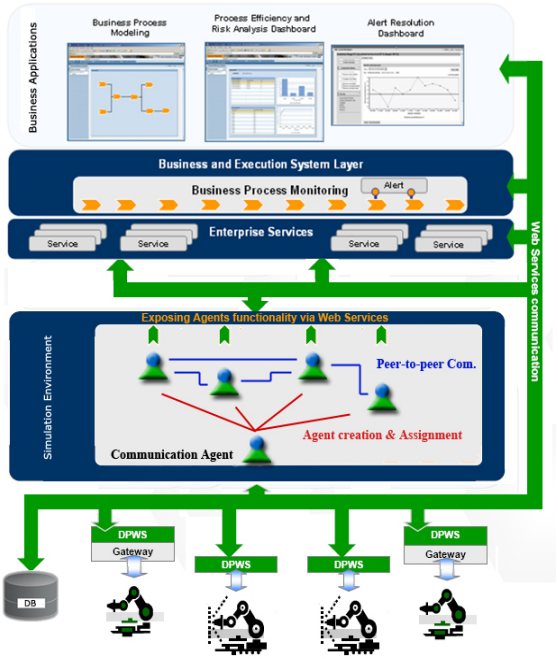

Figure 1: Architecture of an agent-based simulator of web-service enabled devices

On the top layer we can see the enterprise layer, where several enterprise services and business applications reside. Between the enterprise layer and the device layer is the core of our work, i.e. the device simulator. This is communicating via web services directly both to the different devices below and the services/applications on top, which makes it easy for our approach to fit in modern SOA systems. In the simulator itself each agent represents one SOA-ready device. The agents can either get initial or continuous data from the devices they simulate in real-time by connecting to them via WS, by getting predefined values stored in a database or even by generating their own based on internal algorithms.

The simulator is part of an ecosystem and completely transparent to the other actors of the infrastructure including the enterprise applications and other services and devices. As an example, SAP's xApp Manufacturing and Intelligence (xMII) product that typically links a shop floor system with an ERP and provides enterprise services with information from plant floor applications and systems, can discover the hundreds of devices created by the simulator and can not distinguish them from the real devices. This allows us to create large-scale infrastructures in agnostic-ways for the other layers.

We have to note, that agents representing devices can communicate with the outside world via the DPWS, while internally the facilities offered by the agent platform can be used e.g. the Agent Communication Language (ACL). This gives us extended capabilities as we have two different communication and control planes (that of agent system and of DPWS-devices) that can be used independently.

5. Coupling agents and DPWS devices

As mentioned earlier, the JADE multi-agent platform is used to create the agents representing DPWS devices. Each agent represents one DPWS device which needs to be created using the DPWS toolkit (www.soa4d.org). This integration has been achieved by creating two types of agents interacting with the DPWS toolkit i.e.

- a DPWS Client Agent (DC-Agent), and
- a DPWS Server Agent (DS-Agent).

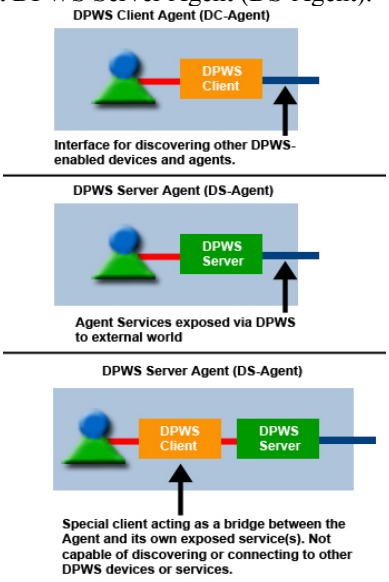

Figure 2: Implementation of DC- and DS- Agents (direct and bridge)

DC-Agent: A DC-Agent implements the client part of DPWS toolkit, acting as client for consuming services offered by devices as well as for services offered by DS-Agents. This agent has no public interface (Figure 2) meaning, it does not expose any service nor it will be visible to the external world. The DC-Agent also acts a bridge between a device and a DS-agent offering service(s) to applications. To keep implementation simple, it is assumed that no agent can be DC-Agent and DS-Agent at the same time. The main usage of DC-Agent is to:

- Discover a DPWS-enabled device,
- Get services and data offered by this device,
- Process data according to user requirements,
- Expose customized data to applications via the DPWS protocol

DS-Agent: This agent implements the server part of DPWS toolkit and is more complex as it consists of two distinct components i.e. a server and a service. The server part instantiates the services, registers them and listens at specified port for client requests. The service part is exposed to the external world and handles all the client requests.
As the DPWS toolkit is not part of the JADE environment, we have to develop bridging concepts that allow for functionality inclusion and interaction between the agents and the DPWS services. Therefore this interaction can be can be implemented in two different ways:

1. Direct implementation: Here the server part is directly implemented within the agent. At service instantiation, the reference of the service is stored, and the service part is modified to implement a protected interface in order to aid the internal communication between the agent and the service (as depicted in figure 2).
2. Bridge implementation: In this case, the implementation is similar to the "direct implementation" with the difference that we neither store the service reference, nor modify the service. Instead we implement a special DPWS client in the agent that acts as a bridge between the agent and its service (as depicted in figure 2).

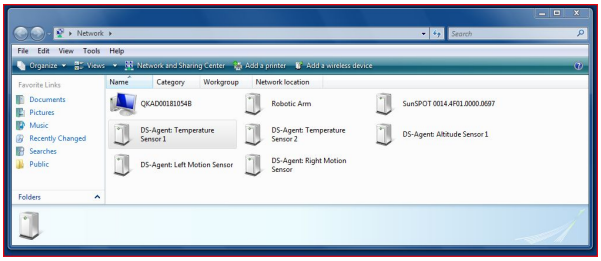

Figure 3: DC-/DS-Agents discoverable in Windows Vista network neighborhood

As it can be seen in Figure 3, all simulated devices can be discovered by third party DPWS clients; as an example in the Windows Vista network neighborhood they appear as normal devices (distinguishable only by their name), and coexist with other devices such as a robotic arm, a SunSPOT sensor (www.sunspotworld.com) and a windows Vista machine. This makes it obvious that the simulator created devices can at least be discovered/used by other infrastructure actors in an agnostic, non-intrusive way.

6. Scenario Implementation

Our initial aim was to create the basic tools for being able to simulate SOA-ready devices. In order to demonstrate this, we have created an infrastructure consisting of real devices, among others a temperature sensor. As an example of demonstrating the simulator capabilities, one can consider a simple device amplification scenario: In heterogeneous industrial environments, it is quite expensive and difficult to evaluate the impact of large number of devices on applications using physical hardware. Such large-scale evaluations can be done using simulated devices. Consider a requirement to simulate an infrastructure of 5000 temperature devices sending temperature readings every one second to specific application(s), while physically only 50 temperature sensors are available.
The simulator can achieve this task in the following fashion: A management agent (DPWS – enabled) can scan and discover all 50 physical temperature sensors and then create 4950 agents simulating the behavior of the 50 physical temperature sensors with a value-adjustment algorithm. Each of these agents exposes a temperature service via DPWS. To other devices and business applications these agents will appear as if they are real physical temperature sensors, in other words applications will not see any difference between simulated devices and real devices. Any discovery request will result in finding 5000 temperature sensors (the sum of real and simulated ones), each offering a temperature service via WS, that any application can subscribe to and get the temperature value every one second.

For our demo, the physical temperature sensor is a SunSPOT while its functionality i.e. the temperature is captured and represented as a web service via DPWS [7]. As it can be seen in Figure 4, we are able to discover the device in Windows Vista and by clicking on its properties we are able to get more info e.g. serial number, MAC address, IP address etc including a device web page that can offer us the WSDL file location and a human-readable form of the service (i.e. temperature) value. The DPWS enabled SunSPOT offers its temperature as a service that other DPWS clients can subscribe to and get notified if it changes.

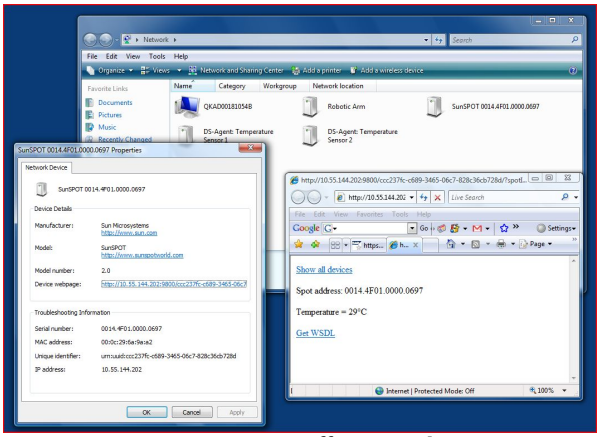

Figure 4: A SunSPOT offering with a DPWS Temperature Service

The SunSPOT is the real device, whose services we will use in the simulated device (the agent). The DC-Agents created in the simulator can discover the SunSPOT and the DS-Agents can be discovered by other DPWS-enabled devices as it can be seen by the Windows Vista screen shots. The DC-Agent is able to subscribe to the temperature service offered by the SunSPOT, process the value returned by the service according to its simulation parameters (e.g. normalize it within a specific range) and then communicate this via agent-specific ways (such as ACL, or internal call) to the DS-Agent. The results of this would be at least one simulated device (DS-Agent) to be discoverable, that offers the same functionality as the original SunSPOT device.

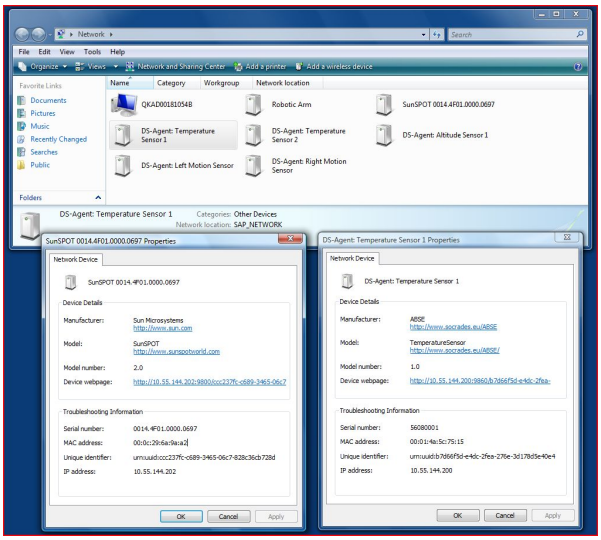

Figure 5: Real and simulated device info

In Figure 5, we can see on the top of it the different discovered devices, on the left side the info from the real device and on the right side the info from the simulated device.

7. Conclusion and Future Work

We have presented initial work towards simulating SOA-ready devices in future web-service dominated landscapes. As future enterprise services will heavily depend on the data acquired from millions of devices, simulating such infrastructures in order to test aspects of it such as communication overheads, performance, model services capable of dealing with dynamic changes etc will become critical. From our viewpoint the simulated framework depicted here can offer some initial insights on how this could be done.

The work presented although still at an early stage, is promising and shows how the real world devices can be coupled with virtual devices simulated by a multi-agent system and how both of them can create an ecosystem that can be used transparently by applications and other services relying at enterprise, network or device layer. As the basic components are implemented, our goal is to trial large-scale landscapes and go more in detail towards better satisfying the requirement expectations set upon such landscapes.

8. Acknowledgments

The authors would like to thank the European Commission and the partners of the European IST FP6 project "Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices" (SOCRADES - www.socrades.eu), for their support.

9. References

[1] E. Fleisch and F. Mattern, editors. "Das Internet der Dinge: Ubiquitous Computing und RFID in der Praxis: Visionen, Technologien, Anwendungen, Handlungsanleitungen", Springer, 2005.

[2] IPv6 over Low power WPAN (6lowpan), The Internet Engineering Task Force (IETF), http://www.ietf.org/html.charters/6lowpan-charter.html

[3] Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices (SOCRADES) project, www.socrades.eu

[4] Stamatis Karnouskos, Oliver Baecker, Luciana Moreira Sa de Souza, Patrik Spiess, "Integration of SOA-ready Networked Embedded Devices in Enterprise Systems via a Cross-Layered Web Service Infrastructure", 12th IEEE Conference on Emerging Technologies and Factory Automation, September 25-28,2007, Patras, Greece

[5] Patrik Spiess , Stamatis Karnouskos, "Maximizing the Business Value of Networked Embedded Systems through Process-Level Integration into Enterprise Software", The Second International Conference on Pervasive Computing and Applications (ICPCA 2007), July 26-27, 2007, Birmingham, United Kingdom.

[6] Shannon Chan et al., "Devices profile for web services", February 2006. http://schemas.xmlsoap.org/ws/2006/02/devprof/,

[7] Luciana Moreira Sa de Souza, Patrik Spiess, Moritz Koehler, Dominique Guinard, Stamatis Karnouskos, and Domnic Savio, "SOCRADES: A Web Service based Shop Floor Integration Infrastructure" , Internet of Things 2008 Conference, March 26-28, 2008, Zurich, Switzerland.

[8] E. Zeeb, A. Bobek, H. Bohn, and F. Golatowski, "Service-Oriented Architectures for Embedded Systems Using Devices Profile for Web Services", 21st International Conference on Advanced Information Networking and Applications Workshops., 2007.

[9] H. Bohn, A. Bobek, and F. Golatowski, "Sirena - service infrastructure for real-time embedded networked devices: A service oriented framework for different domains", International Conference on Mobile Communications and Learning Technologies, April 2006.

[10] Michael Wooldridge, "An Introduction to Multiagent Systems", February 2002, John Wiley & Sons (England). ISBN 0 47149691X.

[11] Fabio Luigi Bellifemine, Giovanni Caire, Dominic Greenwood, "Developing Multi-Agent Systems with JADE", Wiley Series in Agent Technology, 2007, ISBN-10: 0470057475