Computer Communications 35 (2012) 405–417

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Combining heterogeneous service technologies for building an Internet of Things middleware

Kiev Gama *, Lionel Touseau, Didier Donsez

University of Grenoble, Laboratoire d'Informatique de Grenoble, France

ARTICLE INFO

Article history:
Received 30 September 2010
Received in revised form 8 September 2011
Accepted 3 November 2011
Available online 12 November 2011

Keywords:
Service-oriented computing
Internet of Things
RFID
Middleware

ABSTRACT

Radio-frequency identification (RFID) is a technology that allows ordinary objects to be uniquely identified by "smart tags" which are also capable of storing small quantities of data. The term Internet of Things was originated from a vision strongly coupled with supply-chain concerns and RFID tagged objects. However the idea of such Internet of Things has evolved in a wider sense, referring now to a ubiquitous object society combining RFID, sensor networks and pervasive computing technologies. This scenario involves different requirements such as heterogeneity and dynamicity of objects, sensors, applications and protocols as well as the need for allowing the dynamic evolution of such applications. These issues seemed to be easily addressed if the principles of service-oriented computing (SOC), like loose coupling and heterogeneity, are used for constructing such architectures and applications. In this paper we underline what benefits SOC can offer to constructing a middleware for the Internet of Things. These concepts have been applied in a service-oriented middleware that tries to leverage the existing Internet of Things architectural concepts by using SOC principles in order to bring more flexibility and dynamicity. We describe the approaches used in that middleware and the lessons learned from that experience. This middleware was initially tested on an application for tracking and monitoring supply-chain objects, and later extended to target wider application domains that are also described in this paper. The project described here has become part of the OW2 AspireRFID open-source project.

1. Introduction

For the last decade, technological progress has led to miniaturization of computer resources. Processors, memory and data storage are now embedded on small devices spread in the environment. As these devices are more and more equipped with communication capabilities, they tend to be integrated to larger systems. Some objects with communication and sensing capabilities can produce measurements and provide information systems with data on the physical world. Radio-frequency identification (RFID) technology allows ordinary objects equipped with radio-frequency tags to be uniquely and remotely identified. RFID tags are able to store identifiers and small quantities of data associated to a given object (e.g., the credit balance of a skipass, a URL in an NFC smart poster).

The MIT Auto-ID Center, one of the major actors in RFID systems standardization, coined the term "Internet of Things" [2] for referring to a network where objects would be individually and instantly identified with RFID tags to form an Internet of Things. However, the standards produced by the MIT Auto-ID center, and EPCglobal (non-profit consortium that controls such standards) were strongly coupled with RFID as the means of product identification. They target mainly supply chains and goods traceability, allowing easily querying to find information about a specific product. However, RFID tags themselves are not able to provide the aforementioned sensing capabilities. Different wireless and mobile technologies play an important role for bringing an enhanced vision of the Internet of Things (IoT) a step further: in a wider sense, the idea of an IoT evolved as to refer to a ubiquitous object society where different objects are connected, combining RFID, sensor networks and ubiquitous technologies to achieve this concept [30]. RFID laid the initial bricks for such IoT, by allowing to uniquely identify and to store information about individual objects. However, the IoT is not only limited to RFID but could have similar functionality by using different types of object identification [31] such as standards like linear barcode and datamatrix. Not only objects that participate in a supply chain should be traceable in the IoT. As many others, we believe that ordinary objects either through their own means (e.g., smartphones) or by other means of connectivity should also take part in this Internet of Things.

As a consequence of this evolution, software engineering should propose innovative infrastructures fulfilling the requirements related to the nature of the Internet of Things, which are among

* Corresponding author.
E-mail addresses: kiev.gama@imag.fr (K. Gama), lionel.touseau@imag.fr (L. Touseau), didier.donsez@imag.fr (D. Donsez).

0140-3664/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2011.11.003

others: heterogeneity (e.g., different objects, sensors, protocols and applications), dynamicity (e.g., arrival and departure of objects and sensors) and evolution (e.g., support for new protocols, sensors).

At the same time, service-oriented computing (SOC) [22] has also become a major research topic drawing the attention of academics and industrials. A service-oriented architecture (SOA) brings loose coupling and flexibility to applications, and allows the seamless integration of heterogeneous platforms and applications. Dynamic platforms such as the OSGi service platform, which also applies service-oriented principles, provide the ability of dynamic evolution of code (i.e., component updates at application runtime). For these reasons, SOC seemed as a good way to tackle the above issues. Hence the purpose of this paper, which is the experimentation of leveraging a middleware for the Internet of Things by using service-oriented computing (SOC) for building it. The role of the middleware is to track not only RFID-tagged objects but also other objects that can provide relevant information. The data mediation to be performed should handle the data produced by such objects forwarding them to information systems using existing standards.

The proposed middleware, simply called RFID suite, was designed on a multi-layer architecture with SOC being present at each tier, and as a way for integrating them. The novelties and contributions that can be identified in our approach are: the service-oriented and multi-layer architecture in an RFID context; the dynamicity and flexibility of the SOC approach in the data collection layer; the protocol flexibility introduced in the intermediate layer; and the web service-oriented object naming service. This work has been conducted at the University of Grenoble, France, and it has been open sourced in the OW2 AspireRFID[1] project, part of the ASPIRE FP7 project.[2]

The remainder of this paper is structure as follows: the next section provides some background for understanding this article and explains the motivations behind the creation of a service-oriented middleware for the Internet of Things. Section 3 details the architecture of the RFID suite, and how SOC is used to implement the different middleware layers. Afterwards, Section 4 presents several validation scenarios where our middleware was used, and Section 5 discusses the effectiveness of the approach relying on SOC as well as the lessons learned concerning the encountered limitations of our approach. Section 6 overviews other approaches that are related to the RFID suite, and finally, Section 7 draws the conclusions and final considerations.

2. Motivations and background

Besides giving some background on some of the principles and standards used in the realization of our RFID suite, this section also gives some motivations on how service technologies could help overcoming challenges emerging from the Internet of Things and pervasive computing, particularly heterogeneity, dynamicity and evolution at runtime.

In the Internet of Things (IoT), devices/objects are made by different manufacturers that do not always comply with the same standards. This inevitably results in heterogeneous devices that cannot directly communicate because of different protocols involved, raising integration issues. Service-oriented computing (SOC) promotes encapsulation so that component specificities are hidden behind the concept of services, which provide a known contract without needing to know who provides the realization of such contract (i.e., an interface implementation). By using that approach, strong decoupling between components in the system can be enforced. It is therefore possible to represent things (i.e., communicant objects) as services, without needing to know the details (e.g., protocol adaptation code) behind such services. By doing so, devices heterogeneity is hidden from the service consumers, allowing applications to use those devices as compatible services.

Another point is the dynamic nature of such types of pervasive applications. Each thing follows its own life-cycle. A device such as a Bluetooth smartphone might become unavailable to a system as soon as it moves out of range. Regarding autonomy concerns, a simple sensor cannot perform its task anymore if its battery is depleted. As a consequence a system hosting IoT-based pervasive applications must be highly dynamic to manage the devices which continuously leave or enter the system. Since these kinds of applications are not static they cannot be described beforehand, which is the typical case when using architectural description languages (ADL). SOC can achieve a coupling loose enough to manage this dynamicity through service specifications. Moreover, dynamic service-oriented architectures [5] allow applications to react to service arrivals and departures, which is an issue to deal with in IoT-based applications.

Concerning evolution issues, pieces of the IoT are continuously evolving. This evolution may consist in replacing a physical object or upgrading the software driving or using a device. That is why a system targeting such infrastructures must be flexible enough to support this kind of runtime evolution. Modularization is a first step towards such support. A monolithic system can only evolve as a whole whereas a modular system can evolve piece by piece. Due to the loose coupling between its constitutive parts, SOC supports modularity: every service composing an application can evolve independently. Moreover, in many cases the application has to be stopped and restarted to perform the upgrading. This is not the case with service platforms where it is possible for applications to react to services upgrades without being stopped.

The high decoupling and ability to deal with dynamic scenarios lead us to choose the usage of SOC as a solution to answer the software infrastructure needs of an IoT. We decided to experiment this approach by designing and building our service-oriented middleware. This generic purpose middleware is extensible, dynamically adaptable and enables the mediation of data between the physical world and IT systems. Existing vendor solutions are closed solutions where extensibility is not the main issue, since they directly target supply chain customers in solutions that keep customers locked to providers. Thus, we kept flexibility in mind when designing the RFID suite. It should indeed take into account available resources and adapt to different physical infrastructures. In addition to the need for a flexible architecture, the RFID suite should also enable hot deployment (i.e., without rebooting the system upon an update), and a plug'n play approach (i.e., once a device is connected, its features become available represented by services). Other important features concern the ability to perform infrastructure management during execution. As a general characteristic, it should remain domain-independent and be usable in a wide range of IoT-related contexts.

2.1. Dynamic service composition in the OSGi platform

The OSGi platform [21] provides a lightweight dynamic service-oriented architecture enabling the deployment of modular and highly dynamic applications. Recent trends in software industry shows that OSGi has become the *de facto* module system for Java, being employed in diverse contexts (e.g., mobile, application servers, plugin-based development environments).

OSGi applications are built in a modular way using deployment units called bundles. OSGi indeed considers modularity and dynamicity as major concerns. It enhances extensibility and offers

[1] http://wiki.aspire.ow2.org.
[2] http://www.fp7-aspire.eu.