# The "Object-as-a-Service" paradigm

Sylvain Cherrier, Yacine Ghamri-Doudane

# The "*Object-as-a-Service*" Paradigm

Sylvain Cherrier*, Yacine M. Ghamri-Doudane†
* *Université Paris-Est , Laboratoire d'Informatique Gaspard Monge (CNRS : UMR8049)*
† *L3i Lab, University of La Rochelle, La Rochelle, France.*

***Abstract*—The increasing interest about the Internet of Things (IoT) is almost as remarkable than its practical absence in our everyday lives. Announced as the new breakthrough in IT industry, the domain is characterized by a large number of architecture propositions that are in charge of providing a structure for applications creation. These architectures are needed because of the heterogeneity of stakeholders involved in IoT Applications. Programming languages, operating systems, hardware specificities, processing power, memory, network organization, characteristics, constraints, the world of IoT is so diverse. Furthermore, these architectures should provide an easy access to users that are not aware of IT technologies involved. The Services Oriented Computing (SOC) has shown in the past its relevance to the decoupling constraints interoperability among stakeholders. The composition of loosely coupled services facilitates the integration of very varied elements and provides agility in the creation of new applications. But unlike the approach inherited from the SOC in pre-existing services are composed to obtain a specific application, we propose a more dynamic notion of service. Our "Object-as-a-Service" point of view is based on the notion of building dynamically the service needed on each Object and then integrate it in the whole composition. This paper focus on the gain of this approach for the IoT by promoting the "Object-as-a-Service" paradigm as a basis for the creation of dynamic and agile user-made applications.**

***Keywords*-Internet of Things; Services Oriented Computing**

## I. Introduction

With the spread of wireless communications and the rise of the number of devices capable of processing data, the idea of connecting everything to everything, through the universal medium offered by the Internet, opens the way to a new era in the computing domain, usually called the "Internet of Things". The "Major trends in computing", as defined by M. Weiser [31] has occurred. At the beginning, the "mainframe" era (*one computer for many users*), followed by the "personal" era (*one computer for one person*), we are now entering the "ubiquitous" computing era (*many computers for one user*). Desktop computers, laptops, but also smartphones, palm, internet Boxes, and now connected TVs, household appliances, phydgets, single-board micro-computers, we are surrounded by "*smart*" objects, able to process data, communicating through networks, and having new capabilities of sensing or acting on the real world.

The main idea of the Internet of Things is to expand the network, extending it into houses, buildings (walls, floor, windows, doors, etc.) and cities (car park, traffic light, lights, etc.). After its connection to offices, then homes, the Internet now connects farther, to the Objects. But for what purpose? Do we really need this Objects interconnection with the public network? Users will want applications to take advantage of their objects and their new capabilities: measuring the real world and acting on it, processing informations, and communicating with the network. Automatically. Without effort. Easily.

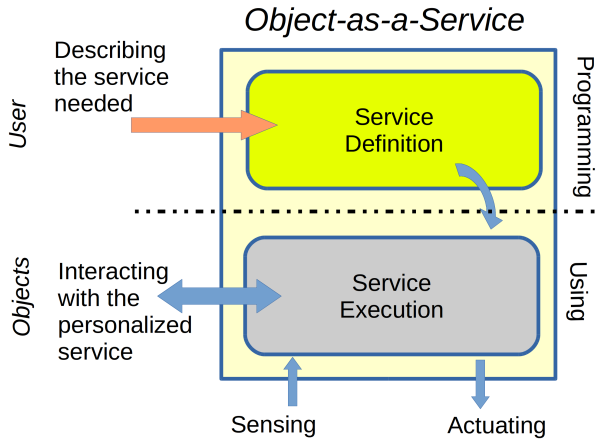


Figure 1. The "*Object-as-a-Service*" architecture. A service (at the bottom of this figure) runs on the Object, using its functionalities (sensing, actuating, computing). This specific service is dynamically created and tailored for user's need, on-the-fly, following the description given by the user on the programming layer (the top service of the figure) offered by this Object.

Creating applications for the IoT is not an easy task, because stakeholders are very varied, with multiple and very different constraints. The idea of using a Service approach (proposed by *Services Oriented Computing, SOC*) can facilitate this integration. Combining services is a way to easily create applications.

Our approach calls for this *Service* vision of an object. But we go further. Usually, the Service provides an access to the data, for example the sensed measure of a physical data (or to an action, in the case of an actuator). We advocate a generic system that allows us to understand the object as a dynamically built services provider. This means that the object must provide the user a way to describe the service to be rendered (see Figure 1). In the manner of "application server"[1], we would like to consider each object

[1]such as *Tomcat*, *JonAs*, *JBoss*, *WebSphere*...

as a computer that can be remotely programmed. Object-as-a-Service (OaaS) is an approach in which we can define and create, on-the-fly, new original services on each object, dynamically. OaaS gives the ability to remotely program an Object in order to include its actions in a global set of interactions.

This position paper is organized as follow: Section II presents the interest of Services for the IoT. Section III lists and defines the different elements usually found in IoT architecture. A classification model of IoT architectures is proposed in Section IV, in which some contributions are classified and compared (including OaaS). Finally, concluding remarks are given in Section V.

## II. Services in IoT

### A. The needs

Most papers describing the IoT offer impressive figures about the number of objects connected to the Internet within a near future [6] [12] [14]: already, the number of objects connected to the Internet exceeds the number of users, and there will be between 16 to 50 billions devices[2] connected in 2020.

The main modes of use of these objects are limited at the moment to simple remote controls. The user accesses his Objects through his smartphone, gathering data collected, or triggering actions on them. But this is a narrow conception of IoT, in which the Internet part is used for the universal connectivity it provides. From our point a view, the interest of IoT increases when Objects interact with each others, without human intervention. As introduced by M. Weiser in his reference article [31], the "*calm computing*" should hide the complexity of electronic devices settings. It should even hide the need for a human to be involved in the chain of reactions. IoT applications should seamlessly analyse sensed data and drive actions on actuators following the global instructions given by the user, automatically and without his mediation.

As the number of connected devices increases, and because Objects can collect huge variety of data (or trigger actions), the possibilities offered for applications are wide. Unlike computers and their finally restricted number of different kind of peripherals (hard disk, keyboard, mouse, webcam), the combination of interactions and the diversity of the effects that a user may want lead to less generic applications than in the usual domain of data computing. We believe that a user will not easily find applications "on-the-shelf" fitting his needs, compared to office or home computing domain in which the proposed generic and "already-made" software are more adapted.

In facts, it seems difficult to imagine real "*calm*"[3] IoT applications "on-the-shelf". Generic applications won't be able to provide sufficient flexibility and adaptability. In the "*Internet of data*"[4], common use-cases mimic the user's way of consuming services as it was before the Internet arises: storing and accessing data, read and writing more or less structured informations, etc. The transposition to the Internet of services organisation and composition was quite trivial.

On the contrary, the IoT opens a new way of understanding interactions between services, especially because these services relate to the real world. Despite the fact that the global trend of proposed demonstration regarding the IoT are often limited to "*remote control*" of distant devices, we believe that there is a new and interesting approach: comparable to Machine-To-Machine, the IoT may propose seamless interactions between Objects, providing an automatic and pervasive control of the real world based on the user's needs.

For us, the future of IoT usages, or a part of these use-cases, is original. The needs of IoT users are varied, specific to each of them (individual or organization, home automation, smart building, smart cities) [23]. The proposed solution of "on-the-shelf" applications, as in the "*Internet of Data*", is much more restricted. IoT Applications must be precisely configured according to user needs. In IoT use-cases, the focus is on user's objects combination. Combinations settings take precedence over application algorithms (which are not very complicated in IoT domain, mostly actions and reactions to gathered data, or data computation). And these settings, and the objects on which they apply, are extremely varied. This is the reason why the proposal of "on-the-shelf" applications is less relevant, and that we plead for more adaptable and dynamic applications.

### B. Applications creation issues

The main characteristic of IoT applications stands in the wide variety of elements that composes it. Besides their number, and the diversity and the specificity of the applications in which they are involved, Objects greatest feature is their heterogeneity.

Objects can be categorized depending on:

- Object's Physical constraints: number and type of varied importance (from an almost total absence of constraints to enormous constraints in terms of memory, processing power, and high energy resource limitation, depending on the type of object)
- Networks Constraints: high speed, with large payload, reliable, or unreliable, with low throughput and limited payload. Wireless Sensors and Actuators Network(*WSAN*), often integrated to the IoT, have lot of limitations. Their combined constraints (low throughput, limited energy and limited reliability) have serious consequences.

[2]Ericsson white pages (2012) http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf

[3]as described by M. Weiser

[4]in opposition to the notion of "*Internet of Things*"

As a result, the user may experience difficulties in the creation of his specific application (lack of common tools, absence of generic platform). And if he succeeds, the application strong dependences on hardware, network and software infrastructures strongly limits its re-usability. Similarly, one can have serious doubts about its scalability, and its maintainability (that can even be impossible). For example, what happens in case of hardware failure and its replacement by another, equivalent, offering the same capabilities, but with different equipment?

These strong constraints (the disparity of hardware capabilities and heterogeneity of components) limit the creation and reuse of any creation. Development costs may be important, slowing the growth of the domain. The difficulty of creating custom applications must be solved.

### C. Service approach in the IoT

The "*Service*" approach limits dependency between stakeholders by introducing loosely coupled links. By providing an interface describing the exchange and hiding its implementation (and its possible change or modification), the dependence of the consumer to the producer is shrinking. The service is provided universally, and the specificities related to the real hardware are hidden. Calls to the service, or calls between services, are executed in a standardized manner. This facilitates interactions and limits dependences to the mere compliance with the interface description. This approach has been successful in the Internet of Data, where no user wonders which kind of hardware or operating system is running the web service he is currently using.

As part of the IoT, using the services approach would accelerate the creation of applications by limiting it to generic services calls. It would also increase the scalability as the infrastructure, composed of many devices, may change continuously, according to the breakdowns and replacement of elements that constitutes it [25].

The "*Service*" approach allows the introduction of genericity and flexibility, and thus facilitate applications creation. Reusing a service in a new composition, creating new service, or making it evolves while offering the same interface, open new prospects for IoT applications maintenance and their easy evolution. IoT applications using Services approach become more hardware independent, and can be adapted to new needs.

### D. Object-as-a-service

Introducing a Service approach in the IoT means to consider each Object as a Service Provider. This approach is presented for example by N. Priyantha et al. [24], D. Guinard et al. [15] [16] or E. Wilde [32]. Usually, the Service proposed by authors gives an access to the data gathered by Objects (sensors), or a way to trigger actions on Objects (actuators). This data-oriented approach gives a "*remote-control*" vision of the Internet of Things. Many demonstrations show a user gathering data from his home with his smartphone (temperature, energy consumption, alarm control centre), acting on it (open/close shutters, setting the central heating, etc), or a car driver finding an empty car park place. Then, we can imagine a central program that will use these services for managing a house, a smart building, some public services in a city, etc.

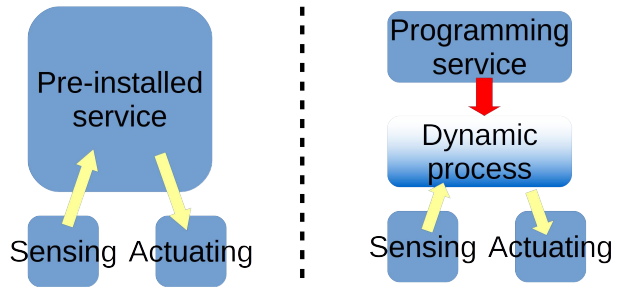


Figure 2. The service approach is often structured as the left part of this figure, in which the service is only able to give access to data gathered by the sensor(or trigger an action on the actuator). On the right, the *Object-as-a-Service* service gives the user a way to explain his algorithm. Then, a dynamic service is build, using Object processing capabilities to execute the algorithm.

What we call "*Object-as-a-service*" goes further than mimicking the Data vision offered by "Service Oriented Computing" in the Internet of Data. The IoT domain could be an opportunity to evolve the concept of SOC to something more valuable. Rather than being limited to Objects sensing and actuating functionalities, the access to their processing capabilities is certainly a great evolution that IoT should offer. These processing capabilities are a great added value offered by connected Objects. Nowadays, these processing capabilities are mainly used for organizing the network, and implementing the different protocols at work to offer access to data or actions.

Giving remote access to the Objects processing capabilities opens the way to the *dynamic definition of new services* (Figure 1). This offers a wide latitude of creation. We propose to see each Object as an application server (such as Tomcat in the Internet), which allows deploying and managing an application on a remote server (Figure 2). Having this ability seems a promising path for the IoT. "*Object-as-a-service*" is the possibility to access and use its programming functions. *OaaS* is a service for **creating** new services.

*OaaS* can be done with a generic programming language, when they are available for Objects. For example, Maté [20], a virtual machine embedded in small Objects, can be programmed to accomplish new services. Darjeeling [4] is a version of an embedded Java virtual machine for that purpose. Our *OaaS* running solution is called D-LITe [8], a very small virtual machine that gives access to the programming, sensing and actuating abilities of Objects. The Contiki-OS [11] version of D-LITe fits in the 48KB

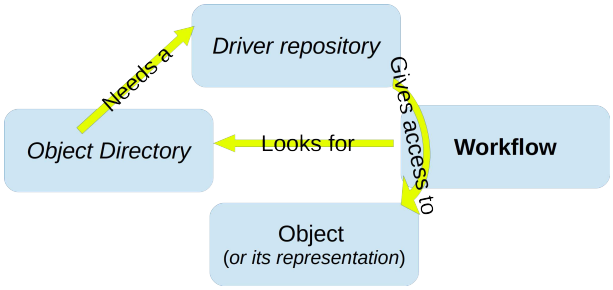


Figure 3. In order to build IoT applications, the proposed architectures often use some or all these common software components, under various organization.

of a TelosB[5] (with 6LowPAN [19] and CoAP [26]). This *OaaS* is accessible through CoAP, and the logic to be run is described with a language called SALT [9]. D-LITe is provided for Android too, and there is a Java version for more powerful Objects. So a user can program dynamically, through the network, a new specific service to be executed on his Objects. He can then make them communicate through CoAP, starting a new genuine services interaction, with less dependence on the characteristics of each real Object.

Consequently this distributed intelligence, focusing on local information processing, reduces network load (and avoid to stress networks, some of which are unreliable). This **programming service** is the main feature of the *"Object-as-a-service"*, and could ensure the success of IoT.

## III. Software components

IoT applications, by using sensed data or acting on the real world, imply a software architecture designed in order to request, find, and access Objects that provide them. The *OaaS* paradigm alters this organisation by pushing the "intelligence" into end-devices. Generic IoT application components (the different elements that compose them, and their roles) have already been described [2] [28] [16]. In that section, we propose to categorize different IoT architecture designs regarding their usages of these common software elements (Figure 3), including *OaaS*.

### A. Composition Engine

The composition engine is the visible part of the whole architecture. Its role is to help the user to express his combination of actions, and to describe how the different elements must interact all together. When the architecture in based on a centralized organization, the Workflow remains under the control of that central point of the infrastructure. In such organizations, the central control point follows the Workflow defined by the user, collects data from Objects (or their representation) or triggers actions on them, computes results and organizes the whole application. In a distributed organization, the logic is spread over different stakeholders, and there is no central control. The composition engine is still present at the conception time, to help the user to describe his logic and the Objects interconnections. But here, each object is in charge of its own activity and its behaviour.

[5]A wireless sensor for research/experimentation http://www.memsic.com

### B. Discovery and directory

The identification of different objects is crucial to obtain the most relevant informations to answer the user's needs. This identification assumes to be both able to retrieve the access to the object that provides data (or actions), but also to manage the data type that the object sensed (or the type of actions it can trigger) and how to invoke that access. The directory needs relate to both the container and contents. It can be considered as furnishing the same service that a DNS can offer, added with more semantic results, such as Web search engines.

However, we can observe differences in indexing the Internet of Data and the IoT. On one side, the number of accessible objects is already much higher than the number of Internet machines managed by DNS servers. On the other side, Objects contents are much less complex than those hosted by web servers for example (in terms of size, variety and complexity). Indeed, each object provides a rather small amount of data compared to the quantity of pages stored on a web server. However, the limited numbers of different data (or actions) provided by objects can take varied presentations, and therefore their format must be detailed.

### C. Drivers repository

IoT deals with a wide diversity of hardware, and the sensing capabilities offered by a sensor (and actions triggered by actuators) are also very varied. Some proposed architectures contain a repository of drivers that give access to all functionalities once the Object is discovered and its type identified. If the driver is available in the repository, the central system can build the application using this object functionalities. The drivers repository gives an high dynamicity to the architecture, because it is a central answer to the stakeholders heterogeneity. As soon as a driver is available, all Objects of this type can be involved in any application.

### D. Object virtualisation/Middleware

In most of IoT proposed architectures, the Workflow is centralized (See composition engine above). IoT applications created with these architectures are often under the control of a central point. This central control point is in charge of the execution of the overall application logic. Following this logic, the central control point recovers data gathered by sensors or triggers actions offered by actuators. Access to Objects are often seen through a representation of their instance. This allows to limit interactions with the real Object to a simple data flow, while the central application

uses its representation. This offers also a way to bind Objects dynamically, and to change that link if needed. This can also be used to represent a unified data, collected from different sources. For example, the notion of temperature or of movement, in a given area, can be gathered from several Objects. An application may consider that if any movement detector wakes up, the whole zone is to be checked. In the same manner, the temperature can be the average of all measures provided by the sensors over a given perimeter. These data are provided by a representation of the Object(s).

Accessing the representation of an Object can be done through two different ways: a virtualization of the Object, or through a middleware. The main difference between the two solution is their impacts inside the application code. While a virtualization hides totally the fact that it is not the real object that runs the code, the middleware is more invasive. Calls to the middleware specifically appear in the program. By providing a complete replica of the real object, the virtualization hides its presence and is agnostic to the program that uses it. Having a virtual Object, or a representation, provides a good (even if not transparent) access to it and may improve the energy efficiency of the solution [10].

## IV. Architectures

Because the *Oaas* paradigm alters the programming point-of-view by providing dynamicity directly on Objects, we propose a classification of the different IoT architectures in order to see the impact of that organization. This classification allows us to compare the different ways Objects are represented in IoT architectures, how applications interact with them, and those that offer a real programming service in Objects (corresponding to our "*Object-as-a-Service*" definition).

### A. Architecture layers

Our classification (Figure 4) references 3 different software layers and 3 different hardware that interact in IoT architectures:

- Software layers are *WorkFlow*, *Virtualization/Middleware* and *Sensing/Actuating* (the latter is the specificity of the IoT).
- Hardware encountered in IoT architectures are *Central Control Point*, *Gateways or Proxy* and *Objects* themselves.

The *Worflow* is in charge of the application logic, at the higher software layer. It drives the whole behaviour and combines services calls and accesses to the Objects. The *Workflow* describes the user's needs and can make decisions depending on Objects responses, reactivity or failures. The *Workflow* triggers actions (through actuators) as the result of its analysis of sensed data gathered from other Objects. This *Workflow* invokes functionalities offered by Objects through the *Virtualization/Middleware* layer. The

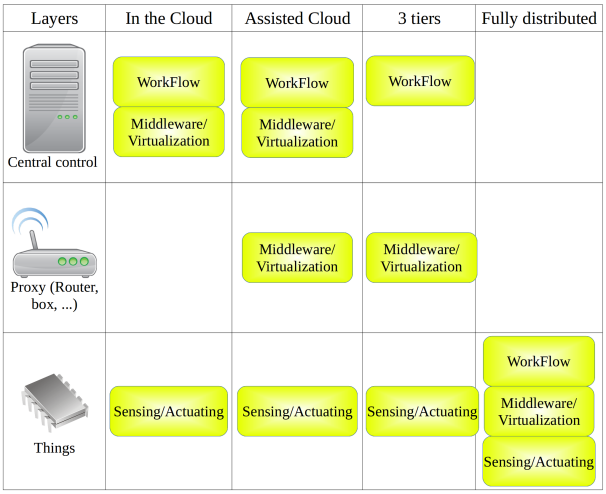


Figure 4. Our architectures classification for Internet of Things applications, according to the software components organization.

*Virtualization/Middleware* layer represents the real Objects (mostly a representation of this Object) and has the ability to conduct actions or retrieve data. As seen before, it can be accessible via a Middleware (through the addition of some specific instructions in the software or through a virtualization. Finally, *Sensing/Actuating* is really executed on Objects, which is the measure (in the case of sensors) or the actions (in the case of actuators). This is done at the lower level of our software layers.

Regarding the hardware, we consider at first the *Central Control Point*. The *Central Control Point* runs some (or all, depending on the organization) parts described in the previous Section: *Service discovery*, *Driver repository*, *Composition engine*, etc. It can be a real computer (in home automation, or Smart Cities), but is more and more often virtualized in the Cloud. *Gateways or Proxy* interconnect the local network to the Internet, for example 802.15.4 networks. Their processing capabilities can be used to do some computation on data. Finally, *Objects* are in charge of sensing or actuating the real world.

### B. Architectures classification

When comparing the different architectures that are described in the literature, a classification of their design can be proposed, based on the distribution of software components in relation to the unit responsible for its implementation. The different functionalities (*Workflow*, *Virtualization/middleware*) described above are executed on different hardware following the architecture organization proposed by the authors [25].

In *Cloud oriented* architectures, most of the functionalities are executed on a central server hosted by a provider. Even the access to Objects, by the meaning of a hardware virtualization, is done in the Cloud. This design is described

in the first column ("*In the Cloud*") of our classification Figure 4. These *Cloud-oriented* architectures offer an integrated installation and deployment that fit the user need of simplicity. But the distance between the virtualization and the real Objects may have an impact in term of networks load, especially because of the number of connected objects involved. This network loads could be an issue for the constrained and unreliable networks IoT applications require (see for example IoT-6 european project [29] [33], Future Internet FI-Ware [1] [30], i-Core [13] or Actinium [18]).

Taking into account the specificities of common Objects Networks used in the IoT (802.15.4 for example), the *Middleware/Virtualization* part of the solution can be executed closer to the Objects, for example in a router, or in the Internet Box. This *Proxy* approach offers the possibility to use different kind of protocols, different messages sizes, with a more or less complex auto-description, adapted to the throughput and bandwidth of the networks used. For example, rich describing SOAP messages (on the Internet side) are translated on the other side (the WSAN) into small CoAP exchanges, tailored for 6LowPAN networks. Depending on the computing capabilities of this Proxy, the data processing can be more or less important, lightening the workload of the various stakeholders. This design is called "*Assisted-Cloud*" or "*3 Tiers*" in our classification Figure 4. This approach can be found in S.K.Datta et al. [10] or WuKong project [21].

And finally, the *Fully distributed* solution (last column of Figure 4) is based on our *Object-as-a-Service* approach. If a specific service can be defined on each Object, dynamically, on-the-fly, it is possible to compose an IoT application in the manner as a program is build in a procedural programming language (such as the C language). This *Fully distributed* organization relieves the network of transmitting data as some computation can be directly done on the Object itself. The Object computes data, takes decisions on acting or invokes directly a service on another Object. Then, in cascade, Objects interact depending on their sensing capabilities or orders received from others Objects (see D-LITe [8] for example). Even if the lack of central point of control may lead to issues [3], the SOC [22] [5] [17] and the IoT [27] [7] community literature propose solutions to solve these problems.

## V. CONCLUSION

While the IoT becomes a major subject of interest in both the academy and the industry, a solid foundation still lacks for its development in our everyday life. The *Service* approach, borrowed from the Service Oriented Computing, lightens the difficulty of developing applications, and offers a solution to compose loosely coupled pieces of software, hardware independent, and interoperable.

In this paper, we advocate for an IoT specific view of the *Service* approach. We believe that a "*services creation*" service gives a real advantage for the IoT. It offers the ability to remotely program Objects, dynamically, in order to use their computing capabilities to distribute the logic directly on each of them. Then, Objects implied in the application directly invoke other Objects services, or only send the result of their logical computation, giving a richer information and reducing the network load.

Our "*Object-as-a-Service*" approach pleads for a rich and universal expressibility of the logic to be executed on each Object. Composing services (their results and their actions) helps to build fully distributed applications, in which each element is autonomous and responsible for its actions and measures, and also interacting with other stakeholders.

In this paper, we also propose a classification of the different IoT software architectures. This classification is based on the organization of the different components that are commonly found in such solutions. This classification describes the various IoT applications designs, how the different parts of the architecture are spread over the stakeholders, and includes our "*Object-as-a-Service*" approach.

## REFERENCES


[1] Fi-ware internet of things(iot) service enablement, 2011.

[2] J. Arkko, D. McPherson, H. Tschofenig, and D. Thaler. Architectural considerations in smart object networking. 2013.

[3] A. Barros, M. Dumas, and A. Ter Hofstede. Service interaction patterns. *Business Process Management*, pages 302–318, 2005.

[4] N. Brouwers, K. Langendoen, and P. Corke. Darjeeling, a feature-rich vm for the resource poor. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 169–182. ACM, 2009.

[5] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, and G. Zavattaro. Choreography and orchestration: A synergic approach for system design. *Service-Oriented Computing-ICSOC 2005*, pages 228–240, 2005.

[6] Y.-K. Chen. Challenges and opportunities of internet of things. In *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pages 383–388. IEEE, 2012.

[7] S. Cherrier, Y. Ghamri-Doudane, S. Lohier, and G. Roussel. Fault-recovery and coherence in web of things choreographies. WF-IoT 2014 (Soumis).

[8] S. Cherrier, Y. Ghamri-Doudane, S. Lohier, and G. Roussel. D-lite : Distributed logic for internet of things services. In *IEEE International Conferences Internet of Things (iThings 2011)*, pages 16–24. IEEE, 2011.

[9] S. Cherrier, Y. Ghamri-Doudane, S. Lohier, and G. Roussel. SALT: a simple application logic description using transducers for internet of things. In *IEEE International Conference on Communications - Communication Software and Services Symposium (ICC'13 CSS)*, Budapest, Hungary, June 2013.

[10] S. K. Datta, C. Bonnet, and N. Nikaein. An iot gateway centric architecture to provide novel m2m services. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 514–519. IEEE, 2014.

[11] A. Dunkels, B. Gronvall, and T. Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. local computer networks. In *Annual IEEE Conference on, 0*, pages 455–462, 2004.

[12] D. Evans. The internet of things. *How the Next Evolution of the Internet is Changing Everything, Whitepaper, Cisco Internet Business Solutions Group (IBSG)*, 2011.

[13] V. Foteinos, D. Kelaidonis, G. Poulios, V. Stavroulaki, P. Vlacheas, P. Demestichas, R. Giaffreda, A. R. Biswas, S. Menoret, G. Nguengang, et al. A cognitive management framework for empowering the internet of things. In *The Future Internet*, pages 187–199. Springer, 2013.

[14] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services Cloud Computing and Scientific Applications-Big Data, Scalable Analytics, and Beyond.

[15] D. Guinard and V. Trifa. Towards the web of things: Web mashups for embedded devices. In *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain*. Citeseer, 2009.

[16] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio. Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. *Services Computing, IEEE Transactions on*, 3(3):223–235, 2010.

[17] L. Jing, Z. Huibiao, and P. Geguang. Conformance validation between choreography and orchestration. In *Theoretical Aspects of Software Engineering, 2007. TASE'07. First Joint IEEE/IFIP Symposium on*, pages 473–482. IEEE, 2007.

[18] M. Kovatsch, S. Mayer, and B. Ostermaier. Moving application logic from the firmware to the cloud: Towards the thin server architecture for the internet of things. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 751–756. IEEE, 2012.

[19] N. Kushalnagar, G. Montenegro, and C. Schumacher. Rfc 4919: Ipv6 over low-power wireless personal area networks (6lowpans): overview. *Assumptions, Problem Statement, and Goals*, 2007.

[20] P. Levis and D. Culler. Maté: A tiny virtual machine for sensor networks. In *ACM Sigplan Notices*, volume 37, pages 85–95. ACM, 2002.

[21] K.-J. Lin, N. Reijers, Y.-C. Wang, C.-S. Shih, and J. Y. Hsu. Building smart m2m applications using the wukong profile framework. In *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pages 1175–1180. IEEE, 2013.

[22] C. Peltz. Web services orchestration and choreography. *Computer*, pages 46–52, 2003.

[23] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies*, 25(1):81–93, 2014.

[24] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao. Tiny web services: design and implementation of interoperable and evolvable sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 253–266. ACM, 2008.

[25] C. Sarkar, S. Nambi, R. V. Prasad, and A. Rahim. A scalable distributed architecture towards unifying iot applications. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 508–513. IEEE, 2014.

[26] Z. Shelby, B. Frank, and D. Sturek. Constrained application protocol (coap). *An online version is available at http://www.ietf.org/id/draft-ietf-core-coap-18.txt*, 2010.

[27] P. H. Su, C.-S. Shih, J. Y.-J. Hsu, K.-J. Lin, and Y.-C. Wang. Decentralized fault tolerance mechanism for intelligent iot/m2m middleware. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 45–50. IEEE, 2014.

[28] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé. Vision and challenges for realising the internet of things. *Cluster of European Research Projects on the Internet of Things (CERP-IoT)*, 2010.

[29] I. Thomas, S. Ziegler, C. Crettaz, L. Fedon, and S. Gaide. Making it all work together enabling new business models in the web of everything. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 526–531. IEEE, 2014.

[30] T. Usländer, A. J. Berre, C. Granell, D. Havlik, J. Lorenzo, Z. Sabeur, and S. Modafferi. The future internet enablement of the environment information space. In *Environmental Software Systems. Fostering Information Sharing*, pages 109–120. Springer, 2013.

[31] M. Weiser and J. Brown. The coming age of calm technology. In *Beyond Calculation*, pages 75–85. Springer New York, 1997.

[32] E. Wilde. Putting things to rest. *School of Information*, 2007.

[33] S. Ziegler, C. Crettaz, L. Ladid, S. Krco, B. Pokric, A. F. Skarmeta, A. Jara, W. Kastner, and M. Jung. Iot6–moving to an ipv6-based future iot. In *The Future Internet*, pages 161–172. Springer, 2013.