# Services organisation in IoT : mixing Orchestration and Choreography

Sylvain Cherrier, Rami Langar

# Services organisation in IoT : mixing Orchestration and Choreography

Sylvain Cherrier, Rami Langar
Universit Paris-Est , Laboratoire d'Informatique Gaspard Monge (CNRS : UMR8049)

***Abstract*—The Internet of Things (IoT) is the extension of the Internet of Data to the physical world. It integrates a plethora of networks of sensors and actuators (WSAN). But it lacks a commonly adopted architecture, able to deal with its important heterogeneity and its constraints. The major trend in IoT Applications conception is based on a centralized architecture. This centralization, known as Orchestration, is a legacy of sensors and actuators networks. IoT Orchestrations are proposed by the industry to properly manage the Objects of a single protocol.**

**However, a distributed approach can be imagined. The distributed architecture, called Choreography, can offer some solutions for interoperability and security, which are important issues of the IoT. This paper explores the pros and cons of both architectures. It also presents a mix of the two, and an implementation in our own platform (BeC$^3$) of distributed applications for the IoT.**

***Index Terms*—Services architecture, IoT, Choreography, Orchestration.**

## I. Introduction

The *Internet of Things* (IoT) is still an evolving research theme. As no common solution seems to be adopted by the market, there are still challenging issues to be addressed.

IoT applications proposed on the market are mainly centralized ones. One of the main characteristics of Objects of the IoT is their heterogeneity. As a result, industrials propose proprietary silos applications, adapted to their hardware specificities. They provide up-to-date software controllers, efficiently driving their Objects, but unable to integrate or even inter-operate with other hardware.

To cope with hardware heterogeneity, a software Service approach can be proposed. The *Service Oriented Approach* (SOA) frees the programmer from constraints related to the specificities of the hardware and programming tools supported by the platform. *Services* are described by an *Interface*, and then implemented on different hardware, with any programming tool. Once the *Services* are implemented, an application can invoke them. Creating an application that uses and combines *Services* provides loosely coupled systems. *Services* can be implemented in any language, on any platform. They must rely on a well defined *Interface*, that describes the requests accepted and the responses given by the *Service*.

Then again, *Services* may interact depending on a centralized organization or a distributed one [4]. In an *Orchestration*, all the *Services* are under the control of a central node, called an *Orchestrator*. The *Orchestrator* has a global view of the overall logic of the application, and is in charge of calling and using the *Services*. The alternative to this centralized approach is the *Choreography*. In this distributed version, each *Service* reacts to its environment, and make decision on its own. Then, it can use other *Services*. In that case, there is no central point of control, every stakeholder follows it own logic. When the combination is correctly built, the resulting system is stable, and able to cope and react to multiple stimuli. Each solution has its pros and cons.

Issues in the IoT domain are often studied by researchers focusing on one aspect, e.g. the network protocol, the security, the application [2]. IoT is a result of the convergence of different visions. These "*oriented visions*" come from the researchers' "*perspective, depending on their specific interests, finalities and backgrounds*" [3]. L. Atzori et al. had classified these perspectives in 3 "visions": "*things*" (i.e. sensors, hardware, and protocols of WSAN), "*semantic*" (i.e. context awareness, analytic and pattern matching), and finally "*Internet*" (i.e. upper layers of IP, applications, Web...). Depending on their background, and on the point they want to address, these research may not pay attention to some layers that they consider as fully functional. In their survey, S. Verma et al. list several papers where some points (e.g. "*network attributes required*" or "*IoT Analytic structures*") (are) "*take*(n) *for granted*" [16].

*Services Approach* offers a solution to get rid of the issues of heterogeneous hardware and programming platforms. Still, the warning raised in [16] must be taken into account. Does the architectural organization of a *Service Oriented Approach* have an impact on some layers of the *Internet of Things* ?

In this paper, we will limit our study to the pros and cons of the two main architectures used in SOA : *Orchestration* (i.e. centralized) and *Choreography* (i.e. distributed). The remainder of this paper is organised as follow: Section II presents the *Orchestration* and discusses advantages and disadvantages. Section III is about the alternative, the *Choreography*. Section IV presents a hybrid approach, an implementation in our framework and a use case. The final section summarizes the study, and presents future works.

## II. Services Orchestration in the IoT

### A. The centralized approach

IoT takes its roots in the *Wireless Sensor and Actuators Network* (WSAN). WSAN refers to a group sensors and actuators that are connected through an often self-organized network, according to the rules defined in their specific protocols [1]. Usually, sensors gather data from the physical world and send them to the sink. Actuators receive the orders to execute from the sink.

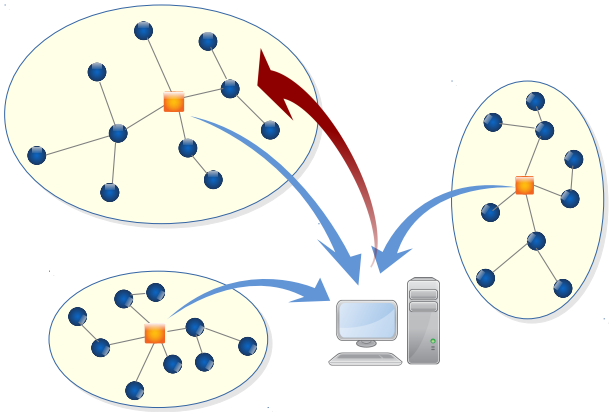

Fig. 1. IoT Services Orchestration: 3 WSAN gather data and send them to the cloud. This central point of control analyses data (blue arrows), takes decision, and sends back an action to be executed on the physical world (red arrow).

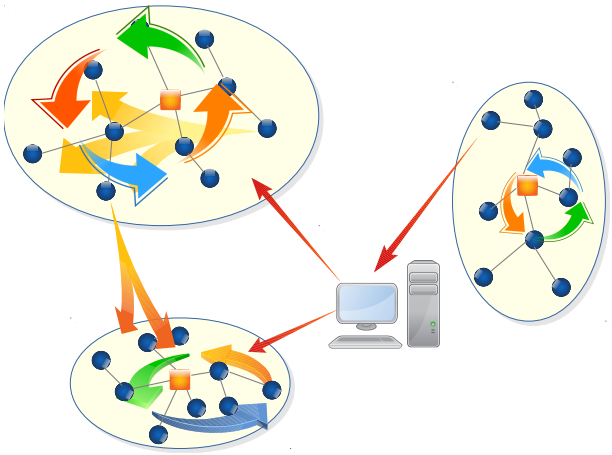

Fig. 2. IoT Services Choreography: 3 WSAN gather data and react directly, sending messages to other nodes (inside their network or even outside). They all collaborate, but no one has a complete view of the scenario. As an example in this figure, the WSAN on the right sends events to a service on the Internet, and this Service reacts by sending event to the other two WSAN, that react to them.

When IoT connects WSAN to the Internet, the sink, which is often the network organizer, is in charge of forwarding all data to the outside world using Internet protocols. The IoT extends the access to data initially offered by the Internet to the physical access given by WSAN. As the IoT evolves, it includes more powerful Objects that can access the Internet on their own.

In an *Orchestration*, the central point of control gets a complete view of the data (see Fig 1). Often, the application gives access to a supervision control panel, where all the data are displayed in graphics. The data gathered can be stored. An important part of IoT research studies are linked to Big Data approach. In that case, patterns are searched, or complex data analysis are done on the data corpus in order to obtain a precise view. Then, decisions are taken by the central *Orchestrator*. The actuators available are requested to make actions on the physical world as described by the central *Orchestrator*.

In this approach, using *Services* is a solution to ease the integration of new devices and new network protocols. As *Services Oriented Architecture* (SOA) provides a loosely coupled systems [4], it simplifies the port of the *Services* on new hardware and/or networks. By hiding the specificities of the new element, SOA makes it easier to cope with IoT heterogeneity. Then, the central point of control invokes the new port of the same *Service*. In the SOA approach, IoT platforms can include new stakeholders with less difficulties.

In the centralized approach, the collected data is used to build graphical reports (e.g. graphs, percent, and different data visualisations), for IoT data analytics [16] and IoT big data [6]. They are major and evolving fields in research.

### B. Do we need all these data ?

Collecting the data raises important issues, as the WSAN networks are constrained in terms of bandwidth, throughput and payload. Most of all, inside WSAN, the energy must be saved, and sending all these data represents an important consumption. Massive IoTs may encounter issues in term of data volumetric [16]. Security and confidentiality are also problematic [15] [5] [19].

Depending on use-cases, applications may not need to collect all the data. Some applications need information to take decision on-the-fly, data analytics and storage are less important [18]. In many cases, users need a summary or an analysis for decision making, and not the detailed data.

Finally, confidentiality and ownership of data can also raise issues. Using cloud services or proprietary solutions may be subject to contractual licenses that place restrictions on the use or ownership of collected data. These restrictions may be incompatible with the user's needs.

## III. Services Choreography in the IoT

Y. Akyildiz et al. [1] describe the differences between a centralized and a distributed approach in WSAN. In the centralized organisation, called *Semi-Automated*, sensors send data through the sink and actuators receive orders from the central point of control. The alternative, named *Automated*, allows sensors to directly interact with actuators. In IoT SOA approach, the distributed architecture is a *Choreography*.

### A. Distributing the control

The *Automated* WSAN is based on a distributed architecture. Creating a distributed application in WSAN raises issues, due to the poor computation capabilities of the nodes, and their heterogeneity. Processing power of Sensors is limited, as the hardware is more oriented on sensing than computing. But to be able to participate to a self-organized network means some data processing capabilities. And this restricted computational power can be used to process data.

IoT extends the WSAN to the world of Internet services on one side, and also imports the Internet approaches in the

sensors. Objects at stake offer more computational power as the market grows, and they may improve network accesses to their capabilities. Some of them even give a full *IPv6* access through *6LowPAN* [12] [13] and simulate *REST* with *CoAP* [14]. The more powerfull objects are (even if it is still very limited), the more work they can do. In our platform called BeC$^3$ [10], we use this processing capability to process data everywhere we need, and to build direct interactions between Objects (see Fig 2).

### B. Energy and security constraints

Sending all the physical measure gathered by all the sensors can generate an important energy consumption. Often users need to get all these data for decision making. The issue here is that sending data to the sink uses important energy resources. The wireless network used at this level is often not reliable, slow, and limited in throughput. A solution to leverage the impact of such a use of the network and the objects can be based on network coding [17].

Processing sensed data directly on the node is often less energy consuming that sending them. On a *Texas Instrument TI CC2538* System-On-Chip, receiving data consumes[1] from 20 to 27 *mA*, transmitting consumes[1] from 24 to 34 *mA*, while computing costs[1] from 7 to 13 *mA*. Furthermore, building the packet adds computational energy cost.

Security is a big issue in IoT. Due to the lack of energy and the limited computational capabilities of the Objects, the level of encryption of transmitted data is not at the level of that usually encountered on the Internet. By impact, one of the strengths of *Choreographies* is that it limits access to data from the outside. Local processing increases confidentiality by limiting the data dissemination, even if it does not protect against an attack. The less data is transmitted, the more confidentiality is assured. An example of the utility of data local processing, linked to the security issue, is presented in the port of blockchain for the IoT [11].

### C. Coherence and expressivity issues

*Services Choreography* raises its own challenges. On one hand, distributed architectures leverage the impact on the network, save energy and protect from security issues. On the other hand, the lack of central point of control may lead to loss of consistency in the application logic. If messages get lost, some Objects of the *Choreography* may stand in an invalid state, loosing the coherence of the whole application. Some mechanisms can be used to avoid it, or to take into account this issue, as we have described in our previous work [7].

Another issue is the expressiveness of the *Choreography*. As it is composed of a combination of *Services*, the application is limited to the capabilities of each *Service* to cope with the logic to implement. Expressive power is always relative to the algorithms that the *Services* can describe. Each IoT *Service* is implemented on the limited processor of Objects, so restrictions are important. However, the IoT applications needs in terms of expressiveness are not very challenging, because the data processing is often quite simple, linked with the limited semantic content of the data measured (i.e. physical value such as temperature, pressure, light, etc, or simple on-off, as in contact, presence, switch, etc). The limited expressivity is offset by the high distribution of processing and the simplicity of the data to which it applies.

[1]http://www.ti.com/lit/ds/symlink/cc2538.pdf

## IV. An hybrid approach

BeC$^3$ [10] is our framework for building IoT distributed applications. With this framework, it is possible to define a combination of *Services* that will be created on-the-fly on the Objects, by programming them remotely [9]. This "*Object-as-a-Service*" approach gives the ability to dynamically create new logical interactions between user's Objects.

BeC$^3$ was designed to process the data locally. It focuses on checking the exchanges validity, in order to allow the interactions between logical blocks implemented on the participating Objects. No data are transmitted, but orders from Objects to Objects. The entire application corresponds to a combination of events, each of which is deduced from the data analysis carried out by each *Service*, locally, closest to their source.

### A. Why choosing ?

This radical choice has not stood the test of real users needs. Even if the local processing of data gives a good protection against hackers, users still often need to access their data, to process pattern matching or simply to log a part of them. An a-posteriori analysis enhances the knowledge of the environment, and helps improve applications.

In that purpose, our *Object-Service* has been enriched by a remotely configurable MQTT client. Thus, it is now possible to configure each object to transmit the sensed values to a given server. Each Object can be set (or not) as client of a given MQTT server, which can be different or unique. Depending on the accuracy of the data he wants to collect, each user can choose which object emits. For example, it can be limited to actuators, and recover only the received orders resulting from the *Service* that has made the data processing. It can also recover the original data by configuring the MQTT client of each sensors, or only some of them. Finally, it can use one or more MQTT servers, scattered in the network, near or far from the capture network.

Mixing *Choreography* and data gathering in the same application offer multiple interests to the user. The distributed approach reduces network traffic and limits power consumption in the WSAN at work. The user controls which amount of data is sent on the network, their destinations and the impact on the network. These destinations can be set at different places in the network or to a central point, on the Edge or in the Cloud. After an analysis, a new Choreography can be designed, and then deployed as a new IoT application.

### B. Implementation

There are several implementations of our tool providing the *"Object-as-a-Service paradigm*" [9], a virtual machine called

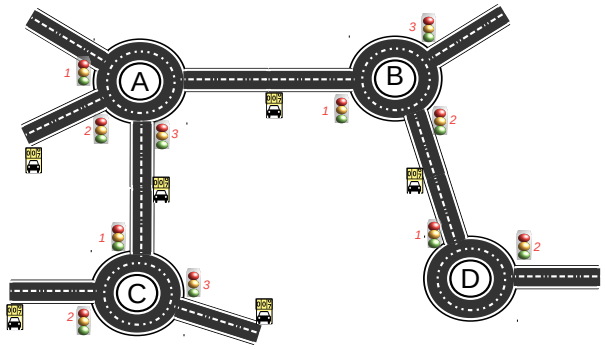


Fig. 3. Roundabouts Choreography. In each roundabout, the traffic light number 1 is the leader of the others. They obey to its rhythm. Car sensors are sending their sensed values to a MQTT Server.

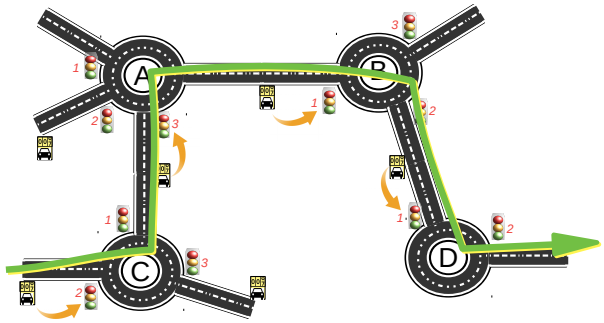


Fig. 4. The IoT data analysis has detected a car traffic jam, a new Choreography is deployed to provide a solution (Green way). Now, the car sensors are controlling the corresponding traffic light (Orange arrow), which are now controlling their roundabout. Traffic lights A3 and C2 are now leaders of their roundabout, while B1 and D1 remain leaders of their.

*D-LITe* [8]. An implementation of *D-LITe* is integrated in BeC$^3$ in order to offer the users a solution to integrate common Internet Services in their IoT applications (e.g. Twitter, web forms, etc). The integration of a MQTT client was not really an issue because there are no constraints of memory or processing power: BeC$^3$ is running on Internet servers, with their usual amount of resources.

We have also integrated the MQTT client in our Python port of the *D-LITe* virtual machine. This version is mainly used on Raspberry Pi. Raspberry Pi are used to control some common WSAN such as ZWave, EnOcean, or Bluetooth LowEnergy. It interconnects them to the Internet, acting as a gateway. Adding the MQTT support was done with the paho-mqqt package[1]. The size of this package is 272 KB. We add less than 10KB on our Python implementation to use it. There is also a Contiki-os[2] port of the virtual machine to integrate 6LowPAN objects in our platform. As there is no MQTT client library for Contiki, we could not integrate it in our Contiki version of *D-LITe*. Writing a MQTT client for this Operating System is a future work to be done.

### C. Smart-city use-case

To illustrate our proposal, we propose a smart-city use-case. In order to run this simulation, we have created in our platform a test of a road management system. This test uses our Python version of the *D-LITe* virtual machine. This road management system (see Fig 3 and 4) has some car detection sensors able to count the number of cars on a road, and their speed. Intelligent Traffic light are also available, and give the ability to control the traffic. We created these "*car sensors*" and "*traffic lights*" with Python scripts that simulate their behaviours.

At first, the *Choreography* deployed uses traffic lights (see Fig 3). For each roundabout, we chose a traffic light master. This traffic light changes from red to green every 90 seconds. By doing so, it sends an order to the other traffic lights of the same roundabout and asks them to change their colours. In this *Choreography*, each roundabout is autonomous and manages its own logic. In this scenario, the car sensors measure the traffic of the road and send the data to an MQTT server.

In this scenario, we assume that, at some point, we detect an abnormally high number of cars on a road. The analysis of the data reveals a car traffic jam in the part of the town. The user then decides to set up a priority path to facilitate the reduction of this car traffic jam (see the green arrow in Fig 4). For each roundabout, the car sensor will command the corresponding traffic light. As long as the car traffic jam remains on this portion, the flow will have priority, and the traffic light will stay green longer than red. Our test uses mandatory value (6 times longer in our test), but the idea is to make this value configurable instead of static. Each portion of the road receives this same algorithm, the car sensor controlling the main traffic light, the main traffic light controlling the secondary ones. Once deployed, this new *Choreography* has modified the role of each Object, thus in reaction of the data analysis done by the IoT data analysis software.

## V. Conclusion

In this paper, we have presented the pros and cons of the two different architectures available for *Services* in the IoT. On one hand, the centralized *Orchestration* that uses *Services* implemented on Objects, hiding their specificities. On the other hand, the distributed *Choreography*, which defines autonomous groups of Objects sharing a logical task. A mix of the two architectures is described. An implementation of a remotely configurable MQTT client is handled in our distributed platform BeC$^3$. In future works, we will concentrate on an IoT data analytics software able to automatically define new *Choreographies*. In that case, the parameters used in the on-the-fly created *Choreographies* can be set by the IoT analytics software. A reflexive loop can be built. In our car traffic jam use-case, the traffic light's "red and green" time difference can be continuously redefined as the environment evolves. Then, each new application can be deployed through our platform according to new requirements.

[1]https://pypi.org/project/paho-mqtt/
[2]http://contiki-os.org/

## REFERENCES


[1] I. Akyildiz and I. Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad hoc networks*, 2(4):351–367, 2004.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[3] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

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

[5] E. Bertino and N. Islam. Botnets and internet of things security. *Computer*, (2):76–79, 2017.

[6] H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos. Iot-based big data storage systems in cloud computing: Perspectives and challenges. *IEEE Internet of Things Journal*, 4(1):75–87, 2017.

[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 and Y. M. Ghamri-Doudane. The "object-as-a-service" paradigm. In *Global Information Infrastructure and Networking Symposium (GIIS), 2014*, pages 1–7. IEEE, 2014.

[10] S. Cherrier, I. Salhi, Y. Ghamri-Doudane, S. Lohier, and P. Valembois. Bec3: Behaviour crowd centric composition for iot applications. *Mobile Networks and Applications*, pages 1–15, 2013.

[11] Z. Huang, X. Su, Y. Zhang, C. Shi, H. Zhang, and L. Xie. A decentralized solution for iot data trusted exchange based-on blockchain. In *Computer and Communications (ICCC), 2017 3rd IEEE International Conference on*, pages 1180–1184. IEEE, 2017.

[12] 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.

[13] Z. Shelby and C. Bormann. *6LoWPAN: The Wireless Embedded Internet*. Wiley, 2010.

[14] 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.

[15] N. Sklavos and I. D. Zaharakis. Cryptography and security in internet of things (iots): Models, schemes, and implementations. In *New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on*, pages 1–2. IEEE, 2016.

[16] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato. A survey on network methodologies for real-time analytics of massive iot data and open research issues. *IEEE Communications Surveys & Tutorials*, 19(3):1457–1477, 2017.

[17] S. Wunderlich, J. A. Cabrera, F. H. Fitzek, and M. Reisslein. Network coding in heterogeneous multicore iot nodes with dag scheduling of parallel matrix block operations. *IEEE Internet of Things Journal*, 4(4):917–933, 2017.

[18] I. Yaqoob, E. Ahmed, I. A. T. Hashem, A. I. A. Ahmed, A. Gani, M. Imran, and M. Guizani. Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges. *IEEE wireless communications*, 24(3):10–16, 2017.

[19] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos. Security and privacy for cloud-based iot: challenges. *IEEE Communications Magazine*, 55(1):26–33, 2017.