A RESTful Task Allocation Mechanism for the Web of Things

El-Mehdi Khalfi, Jean-Paul Jamont
Université Grenoble Alpes, LCIS
F-26000 Valence, FRANCE
firstname.surname@lcis.grenoble-inp.fr

Michael Mrissa, Lionel Médini
CNRS, Université de Lyon
LIRIS UMR 5205, Université Lyon 1
F-69622, Villeurbanne, France
firstname.surname@univ-lyon1.fr

***Abstract*—The Web of Things extends the Internet of Things with Web technologies to interconnect smart things and build scalable, adaptable, and interoperable applications. In this context, smart things expose their services as RESTful resources, to be combined in composite applications. Though, due to the proliferation of connected objects, multiple smart things often are available to provide a single service, showing the need for a task allocation mechanism for the Web of Things.**

Our work addresses the task allocation problem, which consists in selecting a set of objects to realize a complex application in a setup where several objects provide the same functionality. We rely on a multi-agent approach to describe this problem as a Distributed Constraint Satisfaction problem. We propose a solution that respects the fundamental principles of the Web by respecting the REST architectural style and design principles. We implemented our solution with a typical smart city scenario to show the applicability of our approach.

I. Introduction

The Internet of Things (IoT) refers to the current trend of connecting everyday objects to the Internet Procotol (IP) world. It aims at establishing identification, connectivity and communication to devices, ranging from the simplest sensors to the most complex robots. As more and more devices are getting connected to the Internet, a promising next step is to use the Web and its associated technologies as a platform for smart things : this vision is called the Web of Things (WoT) [1]. It refers to the use of Web technologies to address and consume services exposed by physical things. More concretely, the WoT promotes the use of the HTTP protocol and resource-oriented computing to allow exposing object functionnalities as RESTful Web Services (also called Web resources or simply resources). Hence, HTTP methods allow generic HTTP clients seamless access to any Web resource. As well, Uniform Resources Identifiers (URIs) provide a generic addressing scheme for physical things. For example, a HTTP GET on `http://lamp.office.home/status` would retrieve the state (on or off) of a lamp.

While Web technologies help communicating with connected objects, it remains difficult to manage or anticipate the complexity of a typical WoT setup, including interactions among things. In an open environment like the WoT, physical devices are heterogeneous (in terms of CPU, storage, capabilities, energy consumption, etc.). They reply to requests of different consumers, by providing and requesting services. Indeed, the WoT forms a complex system in which entities interact with each other and with their environment. *"The basic idea of this concept is the pervasive presence around us of a variety of things or objects which are able to interact with each other and cooperate with their neighbors to reach common goals"* [2]. Among manifold visions of the WoT paradigm, Multi-agent systems (MAS) provide an attractive solution in order to enable cooperation between objects in such context [3] as they allow autonomous behaviour and distributed decision-making. However, the implementation of typical distributed algorithms used in multi-agent systems (such as the asynchronous backtracking algorithm [13]) according to the principles of resource-oriented computing remains a challenge. To the best of our knowledge, no work has proposed a RESTful mapping of such a distributed algorithm.

In this paper, we setup our work in a typical WoT context where several objects can provide different functionalities to be combined in a single application. We propose a multi-agent approach for solving the task allocation problem that consists in deciding which functionalities of which objects should be selected to run a WoT application. From a technical aspect, we provide a resource-oriented implementation of a distributed artificial intelligence algorithm by providing a mapping between exchanged messages and the REST architectural style. We show the applicability of this mapping with the help of a typical "smart city" motivating scenario.

This paper is structured as follows. In Section II, we provide an illustrative scenario that presents a smart city environment and highlights the challenges we address in this work. Section III overviews related work and shows the originality of our approach. Section IV details our contribution, then we show how our implementation applies to the illustrative scenario in Section V. We draw conclusions from the results obtained and give guidelines for future work in Section VI.

II. Motivation and Problem Statement

A. Illustrative Scenario

On a rainy Saturday morning in Lyon, a student's usual plan is to do homework in the public local library. At 09:00, the "homework plan" prescribed in the student's smartphone shall be automatically set. One of the actions to be performed in the "homework plan" is "take a means of transportation". This

action needs a service composition to recommend a means of transportation for the student (Fig. 1). We consider that the student prefers cycling because it is much faster as it allows avoiding traffic jams, while in cold or rainy days, it is more convenient to take a public transport (bus, tram, metro) in urban areas. The Web application provides our user with a recommendation of the nearest means of transportation (bike or public transport), taking into account the (i) user's GPS location, the (ii) ambient temperature, and (iii) information about public transport stations and bike stands.

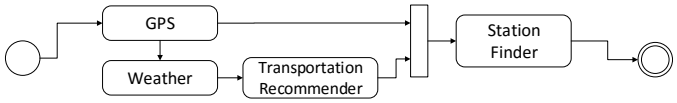

Figure 1: Flow of Services in the Smart City Scenario

The application relies on a composite service, as illustrated in Fig. 1, which combines the following atomic services from different providers :

- GPS : provides the position of the user (Long., Lat.).
- Weather : provides the weather condition and temperature according the user's current position.
- TransportationRecommender : recommends a means of transportation for the user. For the sake of simplicity, this service simply recommends a public transport in bad weather or biking if appropriate.
- StationFinder : Finds the name of the nearest station based on the recommended means of transportation and the user's GPS position.

We consider that atomic services are provided as follows :

- GPS is provided by the phone's embedded sensor,
- Weather service (temperature only) is provided by (i) the user's smartphone, (ii) a weather API, and (iii) an ambient temperature sensor located somewhere in the city.
- TransportationRecommender and StationFinder are RESTful Web services.

This illustrative scenario is simple enough to be particularly suited to dealing with several aspects of our problem. We use it to motivate our multi-agent vision and to explore the various barriers to realizing this vision. Real world scenarios involve more users, devices, and constraints.

B. Problem Statement

Our contribution aims at tackling the following problems :

- Task allocation : As illustrated in our scenario, in a Web of Things context, several objects can provide similar services. However, available resources (memory, storage, energy, processing, bandwidth) are often limited. Considering the size of a heterogeneous system such as the WoT, there is a need for applications to select the services they will use during runtime. To do so, and considering the distributed nature of the WoT, we shall rely on distributed algorithms.
- REST adaptation for distributed algorithms : While REST promotes adaptability, visibility, interoperability and loosely coupled interactions between objects, there is a need to adapt the execution of distributed algorithms used to solve the task allocation problem. Such adaptation should respect the REST constraints (uniform interface, protocol statelesness, etc), and at the same time allow for dynamic service composition to offer complex applications [4].
- Resource-oriented mapping : In daily Web practice, developers often misuse the resource-oriented paradigm as proposed by the REST model[1]. In this model, the fundamental architectural concept is the resource, and the interaction mode follows the semantics of the HTTP verbs. Another challenge is to be able to describe objects and their interactions from a resource-oriented (i.e. RESTful) perspective.

III. Related Work

Many works of different communities deal with the aformentionned scenario. We study different approaches that can guarantee the cooperative behavior in a Web of Things context.

1) WoT Platforms: In [5], "*Physical Mashups*" are proposed as a lightweight approach for combining software services with physical objects. For example, a physical mashup can be a Web dashboard for controlling and monitoring household appliances. Such "*Mashup Editors*"[2] are used to connect physical and virtual services to develop hybrid workflows. Since all services are exposed in a RESTful way in our illustrative scenario, the user can define rules and processes according to some situations. For example, one can list all the available services provided by devices in the vicinity, and define a rule like : *if it is rainy today, then find the nearest bus station*. More WoT platforms that support service composition have been surveyed in [6, 7]. This class of approaches is very successful in our context, however, decision-making is not delegated to physical devices which can cause problems related to scalability and adaptation to situations not predefined by the user via the mashup editor.

2) Service Composition: Motivated by the same concern, i.e. WoT platforms focus mainly on aspects related to basic device interoperation and spontaneous networking without prodiving for dynamic coordination and the realisation of more complex and intelligent functionalities that can be built at a higher application level, a SOA-based architecture adopting an artificial intelligence domain-independent planning approach to automatic service composition, and using OSGi as a platform for exposing RESTful web services is shown in [4]. Nonetheless, we believe that using distributed hypermedia system like REST, distributed models and algorithms for service composition is more relevant than applying CSP-based AI planner, as it is more adapted to the distributed Web setup.

3) Multi-Agent Approaches: Applying a different approach to our scenario, [8] proposes to use RESTful Web services as an abstraction layer of the proposed architecture to be accessed

[1] http://ruben.verborgh.org/blog/2012/09/27/the-object-resource-impedance-mismatch/

[2] http://clickscript.ch/

by users to monitor the Social WoT platform. However, the other layers are not Web compliant and use multi-agent techniques without respecting Web constraints.

Another approach [9] is suggested in a IoT context, taking advantage of using embedded RESTful Web services [10] for constrained embedded networked devices to expose smart objects resources, capabilities and services to the Web for human-machine interactions. The novelty of this approach is its straighforward adaptation to our scenario. However, the approach suggests to use mobile agents. This requires the support of distributed programming models : macroprogramming languages, code migration, task offloading, virtual machines and cyber foraging. The caveat of such approach is that decision making is not reached in a multi-agent way (multiple agents cooperating together to fulfill a complex task) but with agent migration between devices. This technique may open the doors to many security problems due to the agent migration.

Starting from the same area of interest, i.e. multi-agent coordination in ambient intelligence, [11] tackles the challenge of integrating diversified intelligent capabilities to create proactive assistant for everyday life in a domestic environment. This integration, within a coordination scheme, allows to leverage potentially sophisticated domestic services to obtain a smart home with more added value than the sum of is parts. Their contribution consists in reducing the service coordination to a multi-agent coordination, thus casting the smart home coordination to a DCOP (Distributed Constraint Optimization Problem). The fundamental difference related to our work is the use of a constraint optimization approach, while we opt for a constraint satisfcation one [12]. Another distinctive difference lies in the compliance of our work with the RESTful paradigm, which is much more convenient in a WoT setup.

IV. Approach

Providing users with understandable functionalities usually requires composing services offered in the user's vicinity. So, devices need to deal with the incoming requested tasks, called Functional Requirement Descriptions (FRD).

A. Functional Requirement Description

A FRD is a composed service that can be divided into multiple atomic services. For example "bring warm coffee" as a FRD includes the invocation of "go to the kitchen", "make coffee" and "carry plate" atomic services that are provided by a robot and a coffee machine. Fig. 2 shows the FRD of our scenario, composed of the GPS, Weather, Transportation-Recommender, and StationFinder atomic services. For a FRD, many agents are likely to be proposing a similar atomic service (e.g., in Fig. 2, temperature can be provided by the sensor (A4), weather API (A3), and device (A1) agents. The first aspect of our contribution consists in dealing with this task allocation problem in a Web of Things context. Ideally, we aim to assign each requested atomic service to a participating agent in the service composition.

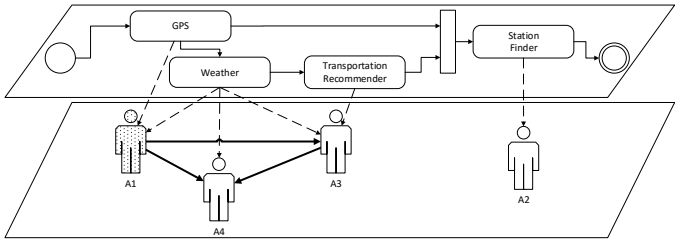

Figure 2: The task allocation problem for the Smart City scenario

B. Task Allocation as a "Distributed Constraint Satisfaction Problem"

In a Web of Things context, objects are confronted to multiple constraints. (i) Each device has its own limitations with respect to hardware capabilities, organizational policies, and cognitive abilities. (ii) Consumers' needs vary in terms of data quality, nature, format, etc. (iii) New constraints are imposed among services provided by physically distributed agents controlling objects.

We introduced in one of our previous works [12] a self-adaptive ecosystem of services based on a multi-agent perspective for dealing with the dynamic nature of physical environments. This work proposes an approach to face the challenge of service adaptation by modeling this problem as a Distributed Constraint Satisfaction Problem (DisCSP). Since we view this problem from a multi-agent perspective, we use this work as a task allocation mechanism for agents.

1) DisCSP: The formal definition of a CSP involves a set of variables $X_1,X_2,\ldots,X_n$ whose values are taken from finite, discrete domains $D_1,D_2,\ldots,D_n$ respectively, without violating a number of constraints $C_1,C_2,\ldots,C_n$. The set of constraints specifies allowable combinations of values for subsets of variables. A solution to a CSP is an assignment of every variable to a value in its domain such that every constraint is satisfied.

A Distributed CSP [13] is a CSP in which variables and constraints are distributed among multiple agents. Solving a DisCSP consists in finding a set of values for variables to satisfy inter-agent constraints to reach consensus among agents. Unlike the centralized CSP, no agent has a complete view of the states of involved agents. Many AI problems can be modeled as DisCSPs. In the same way, we propose to model the dynamic task allocation as a Distributed Constraint Satisfaction Problem.

2) DisCSP Model: To fulfill each FRD in the system, we represent it as a tuple $\langle A_x, SD_x, SC_x \rangle$, where :

- A_x is the set of agents $\langle a_1, a_2, \ldots, a_n \rangle$ which accepted to collaborate to fulfill the FRD;
- SD_x is a set of service domains $\langle sd_{a1}, sd_{a2}, \ldots, sd_{an} \rangle$, each service domain SD_x comprises services provided by the corresponding a_x in the service composition;
- SC_x is the set of constraints on A_x.

A solution to this DisCSP model consists of assigning services to agents without violating (intra-)constraints.

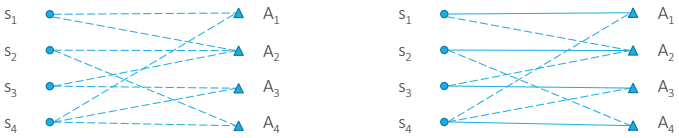

Figure 3: Task Allocation Problem represented as a (bipartite) graph

Task allocation can be represented with Fig. 3. On the left side, we have the required services for the FRD, and the agents that accepted to provide them, dashed links mean a potential participation of the agents. On the right side, normal links mean self-acquired obligation of agents to provide the linked services (We assume, there is a constraint preventing an agent to take a service provided by another).

To create this graph, an agent receives a requested FRD, it decomposes this FRD into atomic services, then sends atomic service requests to its neighbors. Each neighbor decides whether or not to provide the requested services. After receiving replies from neighbors, the agent forms a candidate table with the neighbors and the corresponding services (the services that they agreed to provide).

To provide a composite service, as illustrated in Fig. 1, we must solve the following task allocation problem modeled as a DisCSP (Fig. 4).

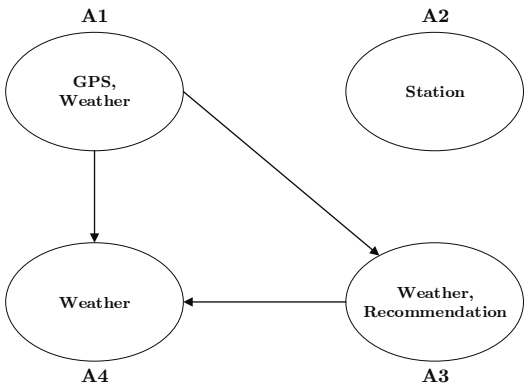

Figure 4: Task Allocation DisCSP Model of our Smart City Scenario

In this model, nodes represent agents, services provided by each agent are listed in the corresponding node, arrows between agents represent the following constraint : Two agents can not provide the same service. Each edge is directed from the agent with the higher priority to the agent with the lower priority. In this type of problems, it is highly recommended to establish a priority between the participating agents. We simply set the highest priority to the agent which provides the highest number of services. For each constraint, the lowest priority agent involved in the constraint is the agent that evaluates the constraint. The other agents in the constraint send their assignments to the first agent, which will check the consistency of the constraint.

V. Mapping DisCSP to the Web of Resources

For solving a DisCSP, one trivial approach is to collect all the information about the problem (variables, domains and constraints) into a single agent. Then, this agent solves the problem in a centralized way. However, centralizing the entire problem into a single agent could be very costly in terms of communication and scalability and is not relevant with respect to the distributed nature of the Web. Therefore, we do not consider a centralized algorithm for solving our DisCSP model. A common method for solving a DisCSP is by backtracking, that is, by repeatedly assigning values to the variables in a predetermined order and then backtracking whenever reaching a dead end. The solution involves assignments of all agents to all their variables. Agents exchange messages containing information about their assignments which allow them to check the consistency of assignments with respect to the problem constraints.

A. The Asynchronous BackTracking Algorithm

Following the pioneer work on the ABT (Asynchronous BackTracking) algorithm [13], manifold ABT-based extensions have been proposed for solving both distributed constraint satisfaction and optimization problems : ABT$kernel$, Asynchronous Distributed Constraints Optimization (ADOPT), ADOPT-ng, AgileABT, Dynamic Backtracking for distributed constraint optimization (DyBop), *etc.* . All these extensions share commun principles and assumptions with ABT. Therefore, in this paper, we found it logical to base our implementation on the original ABT algorithm as a cornerstone for adapting the implementation of more extensions to the Web. ABT is an asynchronous algorithm executed autonomously by each agent in the task allocation process. Each agent manages to find an assignment (provided service) satisfying the constraints with what is known from other agents in a distributed way. To give an idea about the progress of the algorithm (detailed in [13]), agents exchange many messages :

- *ok?* : sent after choosing a service.
- *nogood* : sent after finding that the current local view (assignment of neighbors) violates one of the constraints.
- *newlink* : sent to a non-neighbor to know its current assignment.

B. Mapping ABT messages to HTTP Requests

An important aspect of this paper lies in the alignment of the implementation of an AI distributed algorithm with REST architectural principles [14, 15]. Many of the successful RESTful Web applications are still limited to data exposure and manipulation, it is still not clear how to apply them to process-intensive systems [16]. So, we consider mapping the ABT algorithm to the REST design principles as a step forward towards further application to wider problem domains. We respect the following REST constraints as follows :

C1 Everything being resource identified through URI
The central principle of REST is to model as a resource any entity that needs to be used or addressed, which the

client can consume or act upon. Each agent participating in the fulfillment of the FRD (Fig. 2) has a unique URI. As shown in Fig. 5, the agent's state is modeled as a Web resource. Its most relevant state attributes (localview, constraints) are modeled as sub-resources. Following this segmentation, the size of requests and responses is reduced, giving more scalability to distributed algorithms [17].


```
{
    "name": "a3",
    "address": "http://localhost:8080/RESTfulMAS/agents/a3/",
    "value": "Recommendation",
    "valuesDomain": [
        "Weather",
        "Recommendation"
    ],
    "mode": "VALUE_CHANGED",
    "parentsChildren": {
        "children": [
            "http://localhost:8080/RESTfulMAS/agents/a4/"
        ],
        "parents": [
            "http://localhost:8080/RESTfulMAS/agents/a1/"
        ]
    },
    "links": [
    {
        "link": "http://localhost:8080/RESTfulMAS/agents/a3/",
        "rel": "self"
    },
    {
        "link": "http://localhost:8080/RESTfulMAS/agents/a3/localView",
        "rel": "localView"
    },
    {
        "link": "http://localhost:8080/RESTfulMAS/agents/a3/constraints",
        "rel": "constraints"
    }
    ]
}
```


Figure 5: JSON Representation of an Agent resource

C2 Manipulation of resources through representations

We provide agents with a uniform interface through HTTP operations (GET,POST,PUT,DELETE) such that :

- GET is used to retrieve the agent's attributes,
- POST sets the agent's initial constraints (arcs in Fig. 4) and neighbors (parents and children),
- PUT is invoked to allow agents to exchange messages in a resource-oriented manner as illustrated in Fig. 6,
- DELETE is used to reinitialize agents states after the execution of the ABT algorithm : agents constraints and local view sub-resources are deleted after the adhoc task allocation phase.

C3 Stateless interactions

In the ABT algorithm, agents exchange their selected services and local views (a table containing services provided by neighbors). These asynchrnonous interactions are separate and do not depend on any additional context information for interpretation, thus we keep no state between interactions as illustrated in Fig. 6 which reduces the workload on the agents.

C4 Self-descriptive messages

This constraint tightly liked to C3, as interactions are self-contained. Standard HTTP operations are used to indicate semantics and exchange information (C2).

C5 Hypermedia as the engine of application state (HATEOAS)

Each agent resource contains links to other resources or sub-resources. This allows other agents or users to navigate through the accessed agent. Even if we have implemented this constraint, we do not use it in the distributed task allocation mechanism because the algorithm's steps are predefined, and there is no dynamic interactions in the process. Yet, we can fully exploit the potential of this constraint to provide users with links to others resources (agents, functionalities, services, etc.)

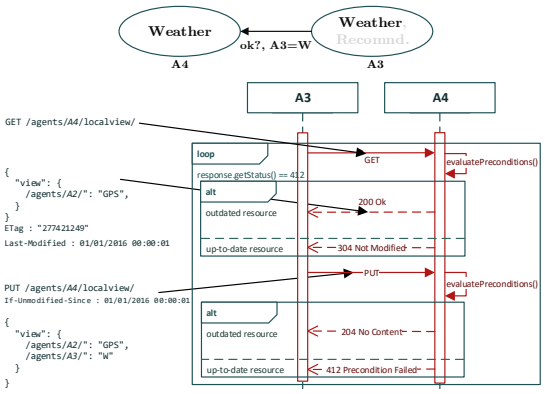

Figure 6: Sending an *ok?* message

C6 Cache

Caching is one of the many performance and scalability aspects of the Web. Agents use this information technology to store temporary data by caching responses. As a result, we prevent agents from sending unnecessary data in response to further requests, and reduce the time generating the results. By adding ETag and Last-Modified headers to responses, agents can get other agents state only when necessary. Requests containing these headers are called *Conditional Requests* and give the sender an opportunity to skip the response body if the representation has not changed since the last time it served the representation.

Therefore, our solution supports by design the constraints of the REST architectural style, which allows getting most benefits of the Web as a support platform for realizing our view of avatar [18] communities to build the Web of Things.

VI. Conclusion

In this paper, we have presented a solution to map a task allocation algorithm based on a cooperative distributed problem solving approach to the Web. We setup our contribution in a Web of Things context where agents are required to compose services offered in their vicinity to provide users with understandable functionalities. The novelty of our contribution lies in envisioning a WoT problem as a distributed constraint satisfaction problem (DisCSP), and in providing a translation of the Asynchronous BackTracking (ABT) distributed algorithm into the REST architectural style to solve this problem. We show that a Web-compliant mapping of the ABT algorithm is possible and offers interesting features. Future work includes studying to what extent other problems from the Web of Things can be modeled as distributed constraint satisfaction problems and solved with well-known distributed algorithms.

ACKNOWLEDGMENT

The authors would like to thank Mohamed Ayache for his help on previous versions of this paper. This work is supported by the French ANR (Agence Nationale de la Recherche) under the grant number <ANR-13-INFR-012>.

REFERENCES

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

[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] E. M. Khalfi, J.-P. Jamont, F. Cervantes, and M. Barhamgi, "Designing the web of things as a society of autonomous real/virtual hybrid entities," in *Proceedings of the 2014 International Workshop on Web Intelligence and Smart Sensing*. ACM, 2014, pp. 1–5.

[4] E. Kaldeli, E. U. Warriach, A. Lazovik, and M. Aiello, "Coordinating the web of services for a smart home," *ACM Transactions on the Web (TWEB)*, vol. 7, no. 2, p. 10, 2013.

[5] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *Internet of Things (IOT), 2010*. IEEE, 2010, pp. 1–8.

[6] D. Zeng, S. Guo, and Z. Cheng, "The web of things: A survey," *Journal of Communications*, vol. 6, no. 6, pp. 424–438, 2011.

[7] S. N. Han, I. Khan, G. M. Lee, N. Crespi, and R. H. Glitho, "Service composition for ip smart object using realtime web protocols: Concept and research challenges," *Computer Standards & Interfaces*, vol. 43, pp. 79–90, 2016.

[8] A. Ciortea, O. Boissier, A. Zimmermann, and A. M. Florea, "Reconsidering the social web of things: position paper," in *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. ACM, 2013, pp. 1535–1544.

[9] T. Leppänen, J. Riekki, M. Liu, E. Harjula, and T. Ojala, "Mobile agents-based smart objects for the internet of things," in *Internet of Things Based on Smart Objects*. Springer, 2014, pp. 29–48.

[10] Z. Shelby, "Embedded web services," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 52–57, 2010.

[11] F. Pecora and A. Cesta, "Dcop for smart homes: A case study," *Computational Intelligence*, vol. 23, no. 4, pp. 395–419, 2007.

[12] F. Cervantes, M. Occello, F. Ramos, J.-P. Jamont *et al.*, "Toward self-adaptive ecosystems of services in dynamic environments," *Advances in Systems Science*, vol. 240, pp. 671–680, 2013.

[13] M. Yokoo, T. Ishida, E. H. Durfee, and K. Kuwabara, "Distributed constraint satisfaction for formalizing distributed problem solving," in *Distributed Computing Systems, 1992., Proceedings of the 12th International Conference on*. IEEE, 1992, pp. 614–621.

[14] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000.

[15] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the internet of things to the web of things: Resource-oriented architecture and best practices," in *Architecting the Internet of Things*. Springer, 2011, pp. 97–129.

[16] X. Xu, L. Zhu, Y. Liu, and M. Staples, "Resource-oriented architecture for business processes," in *Software Engineering Conference, 2008. APSEC'08. 15th Asia-Pacific*. IEEE, 2008, pp. 395–402.

[17] A. Doniec, N. Bouraqadi, M. Defoort, V. T. Le, and S. Stinckwich, "Distributed constraint reasoning applied to multi-robot exploration," in *Tools with Artificial Intelligence, 2009. ICTAI'09. 21st International Conference on*. IEEE, 2009, pp. 159–166.

[18] M. Mrissa, L. Médini, J.-P. Jamont, N. Le Sommer, and J. Laplace, "An avatar architecture for the web of things," *Internet Computing, IEEE*, vol. 19, no. 2, pp. 30–38, 2015.