# Service Composition for IP Smart Object using Realtime Web Protocols: Concept and Research Challenges

Son N. Han[a,*], Imran Khan[a,c], Gyu Myoung Lee[b], Noel Crespi[a], Roch H. Glitho[c]

[a] *Telecom SudParis, 9 Charles Fourier, 91011 Evry, France*
[b] *Liverpool John Moores University, Byrom Street, Liverpool, L3 3AF, UK*
[c] *Concordia University, 1515 St Catherine Street West, Montreal, Quebec Canada H3G 2W1*

**Abstract**

The Internet of Things (IoT) refers to a world-wide network of interconnected physical things using standardized communication protocols. Recent development of Internet Protocol (IP) stacks for resource-constrained devices unveils a possibility for the future IoT based on the stable and scalable IP technology much like today's Internet of computers. One important question remains: how can data and events (denoted as services) introduced by a variety of *IP networked things* be exchanged and aggregated efficiently in various application domains. Because the true value of IoT lies in the interaction of several services from physical things, answers to this question are essential to support a rapid creation of new IoT smart and ubiquitous applications. The problem is known as service composition. This article explains the practicability of the future full-IP IoT with realtime Web protocols to formally state the problem of service composition for IP smart objects, provides literature review, and discusses its research challenges.

*Keywords:* Internet of Things, Smart Object, Service Composition

*Corresponding author
*Email address:* vnhanson@gmail.com (Son N. Han)

## 1. Introduction

The Internet of Things (IoT) is the next major evolution of the Internet where heterogeneous devices and machines are being connected to the Internet, to each other, and to people. With more than 10 billion microcontrollers being shipped each year [1], each of which can potentially be connected through the Internet, a huge variety of intelligent and networked devices are becoming available, from digitally-enhanced objects, to motion sensors, health-monitoring devices, electric meters, and even to street lights. These devices are referred to as smart objects characterized by sensing, processing, and networking capabilities.

The Internet Protocol (IP), meanwhile, has proven itself a long-lived, stable, and highly-scalable communication technology supporting a wide range of applications, devices, and underlying communication technologies. Recent advancement in IoT technologies such as low-power wireless communication links (*e.g.*, IEEE 802.15.4), light-weight IP networking stacks (*e.g.*, uIP), and new routing protocols (e.g., IPv6 Routing Protocol for Low-power and Lossy networks - RPL) allows smart objects, regardless of their limited computing and communication capabilities, to be part of the global network, the Internet. Such IP networks of smart objects are able to merge with the conventional Internet of computers and bare a great deal of opportunities. We call the future of this IP-based ecosystem as full-IP IoT.

In the full-IP IoT, making smart object services (data and events) available and accessible to different end-user applications, using open and standardized protocols is still a challenging task. *The question is not only how to make smart objects be able to communicate over the Internet, but also how their services can be composed to create new and creative applications.* The answer to the former part of the question is being approached by some realtime Web protocols specially designed for IP smart objects and compliant to open Web standards such as Devices Profile for Web Services (DPWS) [2] and Constrained Application Protocol (CoAP) [3]. Services from such smart objects can be directly accessed

on the Web and can interact with a plethora of existing conventional Web services to form a new generation of ubiquitous applications. For the latter part of the question, it can be realized as service composition problem, one of the core principles of Service-Oriented Computing (SOC) [4]. Advanced functionalities can then be created by combining a set of atomic services in the form of composite services [1]. These composite services can be used in different scenarios to meet various user requirements. The true value of the IoT and new opportunities to create a smarter world will become apparent when data and events from an increasing number of smart objects can be easily and dynamically composed to create novel applications.

Service composition has been extensively studied in the context of Web services and business processes [5]. A number of standards have been developed and are being used in real-world deployments to support the service composition. However, the characteristics of IoT systems, such as resource-constraints and data/event-driven devices render some of the techniques devised for traditional Web service composition inadequate. Therefore, new composition models with respect to new requirements of IoT systems are expected. In this article, we consider IP smart objects as the core components of the future full-IP IoT to introduce the problem of service composition and new research challenges.

This article is structured as follows. Section 2 provides the background of IP protocols for smart objects and how they shape the future of full-IP IoT. Service composition problem along with its requirements and some use cases are introduced in Section 3. Section 4 surveys early-stage service composition solutions on various types of smart objects. Section 5 discusses research challenges. Some concluding remarks are drawn in Section 6.

[1] The terms of atomic and composite are used in service composition to refer to participating services and the output service of a composition process

## 2. Full-IP IoT and Smart Object Services

Since its debut more than a decade ago [6], the IoT has gained attention not only from the research community but also from end-users, businesses, and even lawmakers. Numerous research efforts have been undertaken to identify efficient solutions in various problem domains such as enabling technologies (*e.g.*, identification, sensing, and communication) and middleware solutions [7]. Several IoT platforms have been released to tackle technical problems inherent in the development of new IoT products and systems. The business world, inspired by considerable results from research and development activities, has started introducing novel and appealing commercial products in a variety of application domains such as home automation, industrial management, logistics, smart cities, environment control/surveillance, and healthcare. IoT has become an importing computing paradigm in the next Internet, and is known with several names such as Machine to Machine/Object to Object Communication, the Internet of Everything, Industrial Internet, Programmable World, Web of Things, and Cloudy Things. We, in this section, streamline the interpretations of the IoT around smart objects, which can be found common in all IoT approaches.

### *2.1. Smart Objects: Cornerstones of IoT*

According to [8], *A smart object is an item equipped with a form of sensor or actuator, a tiny microprocessor, memory, a communication module, and a power source*. This definition covers different types of *things* in IoT (see Figure 2) including embedded systems (embedded in mechanical/electrical systems with communication modules), mobile phones, Radio Frequency Identification (RFID) tags/readers, and Wireless Sensor Networks (WSNs) sensor nodes. The sensor or actuator gives a smart object the ability of interacting with the physical world, and the microprocessor enables it to process the data captured from its environment. The memory is used to store the operating system, drivers, and software components. The communication module allows smart objects to communicate data to the outside world and receive input from other smart objects. The power source supplies electric power to smart objects.

Sensor nodes are typical smart objects while embedded systems exist in a huge number of devices/machines such as traffic lights, washing machine controller, and medical equipment. When the host mechanical/electrical systems include communication modules, embedded systems become a considerable source of smart objects. Mobile phones equipped with microphone, camera, and sensors make up another large proportion of smart objects. RFID tags are mainly used to identify objects or to track their location without providing any indication about the physical condition of the objects [9]. They only work in the presence of the readers which actually send the information stored in the tags to a software system. In that sense, RFID tag/reader pairs can be classified as a smart object since they have the ability to interact, process, and communicate data. The IoT, therefore, can be defined as *a loosely coupled, decentralized system of smart objects, which are autonomous physical objects characterized by sensing, processing, and networking capabilities* [8, 10, 11].

### *2.2. Internet Protocol for Smart Objects*

To support the large number of emerging applications for smart objects, the underlying networking technology must be inherently scalable, interoperable, and have a solid standardization base to support future innovation as the application space grows. IP has become widely used these days as a standard communication technology supporting a large range of applications, services, devices, and other networking technologies. Many standardization bodies such as Internet Engineering Task Force (IETF), European Telecommunications Standards Institute (ETSI), and Internet Protocol for Smart Objects Alliance (IPSO), as well as many researchers and practitioners have been working to advocate the future IoT with IP networked smart objects. IP has a long-term support to Internet applications for decades such as email, the Web, Internet telephony, video streaming, and many collaborative tools. IP runs over almost any underlying communication technology, ranging from high-speed wired Ethernet links to low-power IEEE 802.15.4 radios. For long-haul communication, IP data is readily transported through encrypted channels over the global Internet.

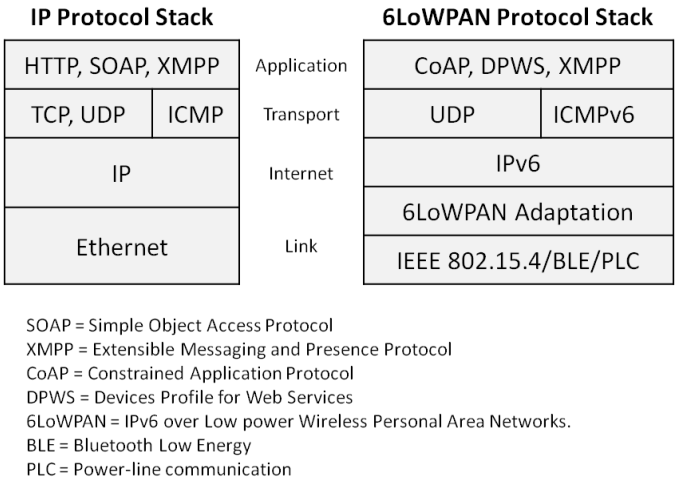


Figure 1: IP and 6LoWPAN protocol stack in reference to layers of the TCP/IP networking model.

Internet Protocol version 6 (IPv6), standardized by the IETF, is expected to accommodate the huge number of Internet-connected smart objects. IETF and other standardization bodies are making great efforts to reduce the footprint of IPv6 for resource-constrained devices. For example, by adding an adaptation layer to the IPv6 protocol, IPv6 over Low-power Wireless Personal Area Networks (6LoWPAN) [12] enables the use of IPv6 in Low-power and Lossy Networks (LLNs), such as those based on IEEE 802.15.4. In addition, RPL [13] routing protocol is used for smart object internetworking over LLNs. These are key areas of IP networking protocols to seamlessly integrate smart objects into the Internet.

Figure 1 shows a comparison between typical networking stacks of regular IP networks and 6LoWPANs following 4-layer TCP/IP model (RFC 1122): Link, Internet, Transport, and Applications. The key difference lies at 6LoWPAN adaptation layer, which adds a specific layer and IPv6 header compression before forwarding to regular IPv6 destination. This technology gives the efficient extension of IPv6 into the 6LoWPAN domain, thus enabling end-to-end IP networking features for a wide range of IoT applications. In order to connect 6LoWPAN networks to other IP networks, 6LoWPAN Edge Routers (6EdRs)

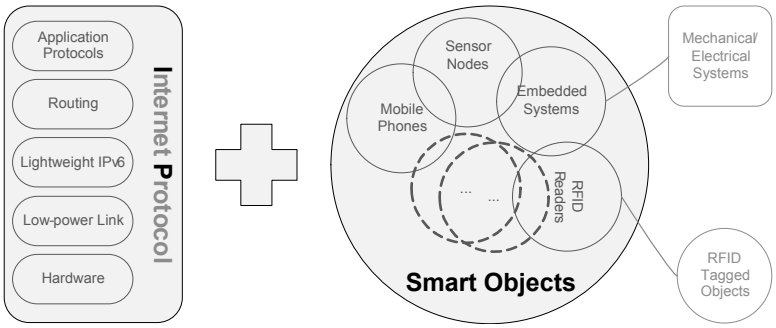


Figure 2: IP smart object services are the cornerstones of future IoT by using advances in IP technology for low-power and resource-constrained devices

are usually deployed at the edge of 6LoWPAN to perform two essential tasks: adaptation between 6LoWPAN and regular IPv6 networks and routing the IP traffic in and out of the 6LoWPAN. This transformation is transparent, efficient and stateless in both directions.

Figure 2 shows the formation of IP smart objects by the adoption of feasible IP stacks. These IP smart objects will be the cornerstones of the next IoT, *full-IP IoT*, and could eliminate the need for protocol translation gateway such as in [14]. Protocol gateways are complex to design, manage, and deploy; their network fragmentation leads to non-efficient networks because of the inconsistent routing, QoS, transport, and network recovery. End-to-end IP architecture is considered suitable and efficient for scalable networks of large numbers of communicating devices such as the IoT.

### *2.3. Realtime Web Protocol and Smart Object Services*

*How will the smart objects interact in the future IoT applications?* There are currently several candidate realtime Web protocols for IoT communication including Message Queue Telemetry Transport (MQTT) [15], Extensible Messaging and Presence Protocol (XMPP - RFC 3920), DPWS, and CoAP. Notably, DPWS and CoAP aim to bring functionalities of smart objects (data and events) to the Web in the form of services. By following Web design principles, these services can acquire open Web standards to enable them to understand the Web languages and protocols, denoted as smart object services.

DPWS is an application standard for bringing Web services to smart objects by adopting the concepts of the W3C Web Service [16] and SOAP-over-UDP binding [2]. DPWS is lightweight and can comply with many of the requirements of event-driven and pervasive IoT applications such as dynamic discovery and event notification. The Web Services for Devices initiative [3] provides and maintains the open source implementations of DPWS. To date, many experiments have been carried out to evaluate DPWS in (even) highly resource-constrained devices such as sensor nodes. The results show reasonable ROM footprints [17]. Other technical issues of DPWS have also been put under research such as encoding and compression [18], the integration with IPv6 infrastructure and 6LoWPAN [19, 20], the scalability of service deployment [21], and the first security implementation has been included in the latest release of DPWS stacks.

CoAP is developed by the IETF working group on Constrained RESTful Environments (CoRE) following REST architectural style [22]. It comprises a minimal subset of REST along with mechanisms of resource discovery, subscription/notification, and security measures for resource-constrained objects. It is similar to the Hypertext Transfer Protocol (HTTP) and can be easily translated to HTTP for a transparent integration with the Web, while also meeting the smart object requirements such as multicast support, very low overhead, and publish/subscribe model. The CoAP protocol provides a technique for discovering and advertising resource descriptions via CoAP endpoints using CoRE Link Format (RFC 6690) of discoverable resources. Its request/response interaction model between application endpoints is based on key concepts of the Web such as Uniform Resource Identifiers (URIs) and Internet media types.

In a nutshell, the aforementioned technologies with enhancements in complexity and overhead enable IP protocols feasible for smart objects. Furthermore, by using realtime Web protocols, smart object will become available on the Internet in the form of Web services (or Web APIs), just like today's millions

[2] http://schemas.xmlsoap.org/ws/2004/09/soap-over-udp
[3] http://www.ws4d.org/

of Web services. This is a vivid picture of the future full-IP ecosystem where conventional Web services can thrive along with smart object services to create a new generation of truly smart and ubiquitous applications. These new Web services of smart objects will be existed in three forms:

- Static Services: These are services to provide specific information from smart objects such as current temperature or status of a washing machine at home, and services to carry out specific tasks such as switching a TV on or setting the light level in living room.

- Event Services: When smart objects are carrying out tasks, there are many situations where events occur such as a coffee maker finishes its task or a printer runs out of ink. They are event services that happen unexpectedly and require applications to wait for their occurrences. The mechanism here is applications subscribe to an event; when it happens, notifications will be sent to its subscribers to handle.

- Periodic Services: These services appear in application such as monitoring environment by periodically pushing sensed data to the network. For example, a service to send the current temperature or CO2 level in a forest to data center every 5 minutes. It requires applications to process these services properly.

## 3. Service Composition for Smart Objects

With IP support and efficient realtime Web protocols for smart objects, it is one step away from the arrival of IoT applications: *how to aggregate smart object services to create novel applications*. A successful deployment of smart object services will create a unique opportunity for developers to build a new generation of IoT application as easily as to work with today's Web applications. Traditional software composition focused on defining languages, techniques, and models for building systems out of reusable software components [23]. As software components evolved into services [24], the notion of composition remained

as one of the core principles of SOC [4]. Service composition aims to reuse several existing component services (or atomic services) by joining them in creative ways. The idea, when applied to IoT, promises to bring in an acceleration for the creation of IoT applications.

### *3.1. Use Cases*

Service composition in the full-IP IoT, can bring new solutions to solve the integration issues that have existed in many classical IoT scenarios. A smart home is equipped with different types of IP appliances: battery-operated temperature sensor, battery-operated motion sensor (using Passive Infrared sensor), TV, lamps, air conditioner, security camera, alarm, electronic door lock, etc. They create a (ad hoc) home network in which these appliances can communicate to each other to carry out automated tasks.

For example, when an owner arrives home, the door lock can identify him and open the doors, signal the light on, and communicate with other appliances to prepare a *home-arrival* composite service. Similarly, a *new-day* service can brew his coffee when he is waking up, update his social page, switch the TV on for his favorite news channel, and check the charge in his electrical vehicle and may suggest him to use a public transport in case there's a technical problem with his vehicle. New-day service can also notify the system in his office when he is about to leave his house to prepare his working session for the day.

This type of scenario orchestrate different types of smart objects to achieve common goals. This can only be fulfilled by an appropriate service composition model based on open standardized protocols. Furthermore, in the future IoT, novel applications will make use of composite services to fulfil complex requirements of inter-domain applications. In the above example, the automation system in a house is providing certain services; the next level of composite service would be the one offered by city administrators making use of the services offered by automation systems in multiple households and other public spaces.

### 3.2. Problem Statement

The problem we address in this article concerns composite services in full-IP IoT by aggregating existing smart object services (based on Web protocols) in a meaningful way that cannot be achieved otherwise.

**Problem.** *Given a set of smart object services as the union of three sets of three service types: static services, event services, and periodic services, a set of requirements R, and a set of cost functions C, find a composite service by aggregating selected services in an appropriate order to meet the requirements R and minimize the cost C.*

Service composition allows the aggregation of smart object services to meet complex requirements from various application domains. It can be used to create innovative applications in an efficient manner. A robust service composition mechanism also makes it possible to support applications in a dynamic network environment. This is the key to foster the development of IoT applications.

Figure 3 illustrates the future Internet application domain supported by service composition of conventional Web services from computer networks and smart object services. Two types of services involve a composition process: atomic service and composite service. The former are constituents of the target, more complex, and more powerful composite services. The latter can also be an atomic service participating in other composition processes.

### 3.3. Requirements

#### 3.3.1. Resource Constraint

Although advances in miniaturization technology enable chip manufacturers to produce powerful small devices, smart objects are still resource-constrained with limited wireless communication bandwidth and low processing capabilities. There is a large number of electronic devices with 8-bit/16-bit microcontrollers and hundreds or thousands byes of memory. It makes heavyweight W3C Web Service modeling languages, protocols, and frameworks such as Simple Object Access Protocol (SOAP) [25], Web Services Description Language (WSDL) [26],

Web Services Business Process Execution Language (WS-BPEL) [27], and Business Process Model and Notation (BPMN) [28] inapplicable. DPWS is the only technology thus far effectively lessens the SOAP messages and incorporates the discovery and the eventing mechanisms for smart objects. DPWS hints the use of WS-BPEL for describing composite services by workflows. However, there is still no WS-BPEL lightweight version to support DPWS messages and the discovery/eventing. A new design for a compact WS-BPEL language following the specification of DPWS (including discovery and eventing) is one possible solution to support service composition in DPWS applications.

When service composition becomes prevalent with plenty of composite services being used in multiple applications, multiple concurrent requests to a smart object become a new challenge. Smart objects with limited resources can only support a limited number of simultaneous incoming requests. This can cause *on-going* transactions freezing or block new coming requests. Therefore it is

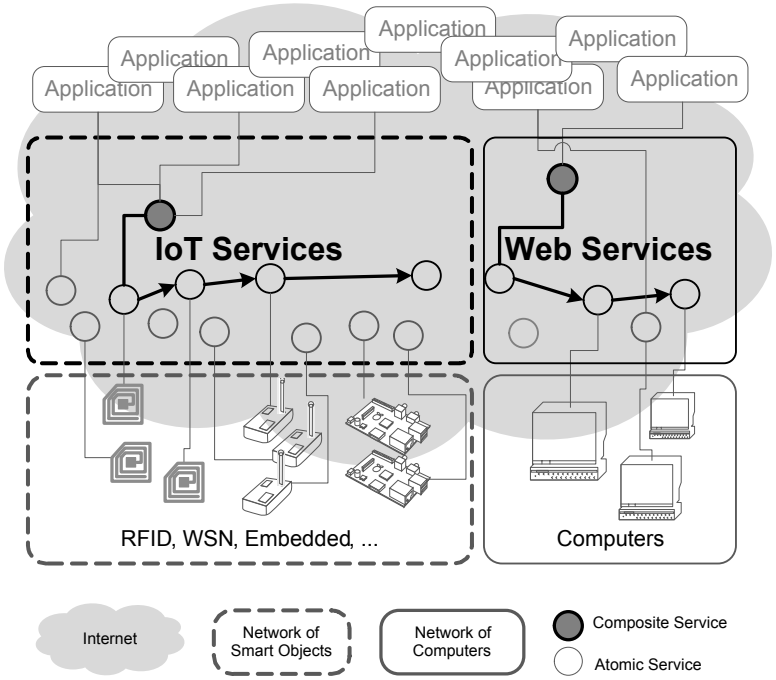


Figure 3: Future Internet application domain is an full-IP ecosystem of smart objects and conventional Web services.

a critical requirement of service composition in IoT to effectively consume the limited resources.

### *3.3.2. Low-power and Lossy Communication Link*

Many of smart objects are parts of LLNs consisting of constrained nodes (with limited processing power, memory, and sometimes energy when they are battery-operated or energy scavenging). The nodes are interconnected by lossy links, typically supporting only low data rates that are usually unstable with relatively low packet delivery rates. This characteristic requires service composition in IoT to be aware of the lossy and low data rate nature of the link.

### *3.3.3. Power Efficiency*

A smart object is driven by electronics, and electronics need power. Today, the most common power source is a battery, but there are several other possibilities for power, such as solar cells, piezoelectricity, radio-transmitted energy, and other forms of power scavenging. Power scavenging is a technique in which devices harvest power from the physical environment. Solar cells represent the most common form of power scavenging. They harvest their power from the ambient and direct light hitting the smart object. Piezoelectricity is another source for power scavenging. Many smart objects are battery operated such as sensor nodes and mobile phones. Mobile phone battery is a critical issue now as more and more background services are using the limited battery power. With sensor nodes, energy consumption is more critical to ensure the operation of nodes for a long time.

For battery-powered smart objects, the batteries typically cannot be recharged. For solar-powered smart objects, and those powered by power scavenging, energy is difficult to be stored for extended periods of time. For this reason, both the hardware and the software of the smart object must be designed to meet stringent power requirements. To achieve this, low-power radio hardware is not sufficient. Existing low-power radio transceivers use too much power to provide long node lifetimes on batteries. Radio duty cycling mechanisms (*e.g.*, Con-

tikiMAC) have been developed to deal with the problem in the object scope. It is based on the principles behind low-power listening but with better power efficiency to reduce power consumption and maintain good network conditions. Therefore, service composition should minimize the energy consumption for exchange messages during its life cycle.

### *3.3.4. Data/Event-driven Services*

In the Web services and business processes, the service model follows a process- (or operation-) oriented paradigm, whereas IoT applications implement a data- and event-driven model. DPWS has an event mechanism allowing clients to subscribe state changes (events) in smart objects; CoAP also provides subscription/notification model. One crucial requirement is in the way service composition in IoT deals with this new type of services. With DPWS, composition involves the execution of reactive and *always-on* atomic services. With CoAP, composition is concerned with the matching data of input from proactive events to another event's output data where the execution of services cannot be applied. These results in the ineffective use of the workflow-based languages such as WS-BPEL. Similar issue happens in mashups as static unions of RESTful services and use synchronous request/response transmissions of HTTP. To support this characteristic, either with DPWS or CoAP, it is required to meet the data- and event-driven characteristics for service composition.

### *3.3.5. Asynchrony*

In smart object architectures, data processing and response generation may not happen immediately. This would require long-lived connections. Especially in dynamic and mobile device scenarios, asynchronous short duration transmissions are required to overcome this problem. This is a common feature of real-time Web protocols for smart objects. CoAP, unlike other REST architecture HTTP deals with these interchanges asynchronously over a datagram-oriented transport such as User Datagram Protocol (UDP). This is done logically using a layer of messages that supports optional reliability. DPWS makes use

of WS-Addressing to overcome this problem. WS-Addressing includes message IDs in every request to assign responses to the correlated request. Therefore composition process is required to cope with this characteristic.

### *3.3.6. Discovery*

There is a need to discover available services to carry out service composition. Web services are usually discovered by querying registries using interfaces such as Universal Description Discovery and Integration (UDDI). While it can be a convenient way to discover services, its centralized nature can lead to many issues such as fault tolerance, performance, and scalability. One way to deal with such issues would be to envisage, similarly to network management, a broadcast discovery protocol.

In DPWS, WS-Discovery mechanism [4] using multicasting does not require any central service registry. When an application tries to locate a device in a network, it sends a UDP multicast message (using the SOAP-over-UDP binding) carrying a SOAP envelope containing a WS-Discovery Probe message with the search criteria, *e.g.*, the name of the device. All the devices in the network (local subnet) that match the search criteria will respond with a unicast WS-Discovery Probe Match message (also using the SOAP-over-UDP binding). To achieve resource discovery, CoAP servers provide a resource description available via a well-known URI such as /.well-known/core (RFC 5785). This description is then accessed with a GET request on the URI.

However, broadcasting at the global level could waste network bandwidth and increase the latency of the communication. Therefore, composition methods are required to minimize the message exchanges during discovery phase. Appropriate global discovery mechanisms are also required.

### *3.3.7. Management Requirements*

*Verification*: Composition correctness is required to check if certain properties of the produced composite service hold, such as the fact that it is guaranteed

[4] http://schemas.xmlsoap.org/ws/2005/04/discovery/

to produce a certain set of outputs given a certain set of inputs and under a certain set of conditions.

*Execution Monitoring*: Service composition regardless of its underlying technology should support the concept of a client making requests by *invoking methods*. Many smart object services are time-consuming therefore it is required that composite services be monitored and reported when they finish.

#### *3.3.8. QoS Awareness*

QoS-aware approaches take into account not only functional characteristics of services but also non-functional ones, dealing with quality aspects such as response time, price, availability and so on. Considering QoS aspects when deciding which services to include in a service composition schema is important when functional requirements are satisfied by more than one service. As a result, composite services produced by QoS-aware approaches not only offer the capabilities requested by the user but also guarantee the best possible quality.

## 4. A Review of IoT Service Composition

This section provides a review of early-stage studies on service composition in IoT classified by smart objects technologies (RFID, WSN, and others) and composition approaches (SOC composition and Mashup).

### *4.1. Smart Objects Technologies*

Since the original idea of the IoT stemmed from RFID and WSN technologies [7], service composition in RFID and WSN are considered early-stage studies in the field. When IoT extended to cover many other networked objects, service composition also evolved. In this sub-section, we review the service composition research in RFID systems, WSNs, and other types of smart objects.

#### *4.1.1. RFID Systems*

RFID was first introduced to overcome the limitations of the barcode technology and primarily focused on tagging objects by attaching identifiers to them

[29]. While the original idea was to tag items for retail and logistics, the application of RFID tags to any object allowed the development of numerous disruptive services. Since the application of RFID technology became prevalent, the need to aggregate RFID data/event and enterprise services has been considered to link logistics to other parts of the enterprise operations such as supply chain management, production, and customer relationship management. Systems such as Electronic Product Code Information System (EPCIS) [5] and SAP Auto-ID Infrastructure [6] were designed to meet this requirement.

The work in [30] focuses on designing a system to support the collaborative scenario of program committee meetings using presence-aware technology, RFID, and Web service. RFID tags are used to detect persons in the meeting room. When a person (carrying the RFID tag) enters the room, RFID reader detects and sends data to a Presence Manager Service, a service to determine the list of persons present in the room. This service is apparently implemented by W3C Web Service with WSDL description file. The system also supports core services to be leveraged and combined to form more powerful services and applications. The composition process is based on workflow approach with the use of WS-BPEL and service registry. Xingyi *et al.* [31] introduce a two-phase model for RFID data composition directly working on the RFID primitive data. The first phase is to process primitive events generated by RFID readers to produce predefined basic events (service abstraction). This phase aims at data filtering to reduce the data redundancy dramatically. The second phase is to union these basic events into several temporal complex events (semantic data composition).

In [32], authors provide a preliminary design for using semantics for service composition in the Border Control domain. The idea is applied to a shipment monitoring application, an open and multimodal end-to-end tracking and tracing system on SAP Auto-ID infrastructure. The work [33] presents a logistics

[5] http://www.gs1.org/epcis
[6] https://help.sap.com/aii

service platform being able to discover, combine services and track the goods status automatically. It is based on the RFID middleware to abstract RFID events to services and uses semantic Web standards to carry out service composition.

The work in [34] provides an early approach in bringing RFID data into the Web and then applying Web 2.0 mashup for composing these services. Guinard *et al.* in a number of studies [35, 36, 37, 38] show an effort to integrate RFID objects into the Web through RESTful APIs. Thereafter, it supports a low-cost RFID/EPC system by using cloud service and open Web standards for the communication between tagged objects and the EPCIS. Illustrated systems feature mashups by using RESTful APIs representing RFID collected data following Web 2.0 standards with the support of mashup editors such as Clickscript[7].

#### *4.1.2. Wireless Sensor Networks*

WSN is considered as another pillar of the IoT as the emergence of small scale sensor nodes has made profound impact on the way we can interact and manage our surroundings. In WSNs, the resources are constrained and communication among nodes is error-prone and unreliable. Such a dynamic environment requires a continuous adaptation of the composition of services.

Movahedi and Defude [39] introduce an abstract composition model for WSNs structured in three levels. The concrete service level relies on components of WSNs such as different types of sensors, dynamic nodes, and gateways. The gateways offer the interconnection between WSNs and upper layers by Web APIs using SensorML [8] repositories. Abstract block level abstracts the concrete services and stores them in a repository in this level. Application designed as an orchestrator of abstract blocks is called an abstract graph and stored in another repository. The prototype uses composition standards such as WS-BPEL and BPMN.

Toure *et al.* [40] present MASCO, a movement-assisted service composition

[7] http://clickscript.ch/
[8] http://www.opengeospatial.org/standards/sensorml

optimization method, to deal with the mobility and the availability of sensor nodes for optimizing service composition in dynamic networks such as WSNs. Authors assume that services are widely distributed in a dynamic network of self-organized multi-hop sensor nodes. The method is to select a short execution path by reducing the number of hop-counts of the services. This is accomplished using a node repositioning algorithm. The work excludes details on composition process in terms of communication protocols and Web service standards that should be used for real implementation.

In [41], the authors propose a service composition design in WSNs that allows composing efficient services when there are persistent service queries. The aim is to minimize the service composition cost when a service is required for a longer period of time with the high possibility of service interruptions, *e.g.*, due to sleep mode of sensor nodes. The work consists of a service-oriented query routing protocol, a greedy algorithm to optimize the service requests, and a dynamic programming algorithm to help minimizing the service composition cost. Several algorithmic proofs and solutions are provided along with some simulation results; however no detail of simulation or implementation are discussed.

In [42], the authors present a graph-based formulation for modeling sensor services for analysis and based on it, formulate the service composition problem in WSNs as NP-complete. Two heuristic methods are presented to solve the service composition problem. In top-down method the high-level specifications of a composite service are defined first and then after a series of steps, primitive services are composed and then can be used as the input to the desired composite service. In bottom-up method, a subset of the already composed primitive services is used by a composite service. Top-down method provides cost-efficiency whereas bottom-up method provides robustness. The proposed solutions are only implemented in NS-2 simulator using Ad hoc On-demand Distance Vector (AODV) routing (RFC 3561) and IEEE 802.11.

In [43], authors present a distributed composition service to automate the re-organization process that can occur due to the inherit limitation of a WSN. Their main goals are to design and implement an API and to provide an ac-

ceptable level of network functions under abnormal conditions. The proposed composition method uses three servers for i) lookup service, ii) adaptation service and iii) composition service. These services allow service discovery, its adaptability to tackle the network changes and the service execution. After detecting an event, a group of sensors sends a composition request to the composition server. In reply, the sensors receive the group information, such as the group leader, its members as well as their own IDs to communicate with each other. The adaptation server helps in group reconfiguration when a sensor fails. The main drawback of their solution is the total dependency on composition server.

Authors in [44] provide a composition method based on logical programming through backward chaining for chaining services. They model services as statements, whose truth depend on their predicates, and set certain statements true when these predicates are satisfied. These statements are further used by other services as predicates. The method is used for automated inference in WSNs.

#### *4.1.3. Miscellaneous Smart Objects*

Most of IoT systems consider the heterogeneity of smart object for service composition. These smart objects vary in hardware, software, and communication protocols but share some common characteristics such as they are all able to run the same operating system. Beside smart objects, there are also many approaches taking into account *always-on* services hosted in conventional computer networks such as enterprise services and Web services for their service composition models.

Authors in [45] propose two types of mashup for service composition on embedded systems: physical-virtual and physical-physical mashups. EnergyVisualizer is a physical-virtual mashup that offers a Graphical User Interface (GUI) on the Web to monitor the power consumption and to control different home appliances. EnergyVisualizer is built by using the self-defined RESTful Plogg APIs and Google Web Toolkit APIs. Ambient Meter on a SunSPOT is a physical-physical mashup that polls a predetermined URL using GET method to get the

energy consumption of all the devices in a room from a smart gateway.

The work in [46] also provides service composition on embedded systems but in service-oriented approach by integrating real-world services (running on smart embedded systems) with business services. The proposed architecture consists of Discovery, Query, Selection, and On-Demand Provisioning. The composition is based on selection of appropriate services.

With similar purpose, [47] discussed the problem of integrating sensor services with traditional information systems and tried to solve it using the concepts of service orchestration and choreography undertaking the scalability and dynamicity issues of IoT in order to extend the existing (adaptable) service composition mechanisms. The orchestration process works at device level to integrate the obtained services from several sensors or smart devices while the choreography module is used to invoke several other (Internet) services to extend the cooperation between virtual and physical worlds.

The DPWS technology was used as a part of a service composition platform developed within ITEA2 DiYSE project [48]. It focused on the presentation layer of IoT to enable the creation of new user-generated composite applications. After that, Han *et al.* [49, 50] are among the first to explore the service composition explicitly over the DPWS. Authors present a semantic context-aware service composition model for DPWS devices. The model is the main component of a building automation system. A context-based description language and ontology are proposed to support the dynamic composition of smart device services offered by SOAP based DPWS services.

The authors in [51], [52] extend DPWS standard to specify the service interfaces of the smart objects for service provisioning in future service-oriented Internet by using W3C Web Service technology. Additionally, a verification technique is proposed to check the viability of the composition of things. It is argued that the behavior of the heterogeneous things is critical during their composition in future Internet and the services should be invoked in the correct order to avoid violation of the behavior of things. To detect invalid mashup invocation at runtime, a mediation platform is used to allow things to receive

mashup request compatible to their current behaviors. A motivating example of a mashup of things with two services is presented. Finite state machines are used to represent the relationships between the messages in these services. WSDL is used for service description while SOAP for transporting messages. The work lacks any real-world implementation of the proposed solutions.

In [53] a context-based service composition method is proposed along with a test application. Two types of context metrics are considered during service composition, one is computational context and the second is service quality context. In the beginning the service selection is divided into two sub-parts, first using computational context appropriate services are selected, then service quality context is used to provide the best available service for the service composition process. Web Ontology Language (OWL) [54] is used to construct context ontologies consisting of three categories, user, computing environment, and physical environment. A hierarchical ontology design model is followed to reduce the scope of the context information. A device monitoring service is discussed as a test without any implementation.

In [55] the authors present a distributed, scalable trust management protocol for IoT to compose services. The protocol takes social relationships into the account. The system architecture lacks a centralized trusted authority. The trust of the node is built when it interacts with other nodes. Each node uses the trust evaluation protocol independently to assess the other nodes it communicates with. This trust assessment is also shared with the other neighboring nodes as recommendations. The trust model has three trust properties, honesty, cooperativeness and community interest. The numerical results are presented as well as a simple description of the application composition with and without the use of the trust management protocol.

### *4.2. Service Composition Approaches in IoT*

There are currently two common approaches for service composition that are based on two different service models: service-oriented middleware and RESTful APIs.

#### *4.2.1. Service-Oriented Middleware and Composition*

Service-oriented middleware that supports the service-oriented interaction pattern through the provision of proper functionalities for deploying, publishing/discovering and accessing services at runtime, is the main service abstraction model to support service composition for smart objects [56]. In this abstraction model, to create a composite service, a description language is normally used to describe its component services and the interaction between these services in the form of a workflow. Workflows can be nested, so it is possible to call a workflow from inside another. The creation of complex processes can be represented as a sequence of coordinated actions performed by single components. This paradigm is supported by W3C Web Service technology with many open standards such as SOAP, WSDL, WS-BPEL, and BPMN.

Since service composition has been intensively investigated in the Web services and business processes, it is promising to use their results in the IoT environment. However, the problem cannot be tackled by simply extending existing approaches; it requires a paradigm shift from reliable and reactive Web services to dynamic and event-driven smart object services that demand for a more resource-aware event-driven composition process. Dar *et al.* [47] undertakes the scalability and dynamicity issues of very-large scale IoT systems to propose an orchestration/choreography for composing services of smart objects and business services. Graph-based modeling framework [42] formulates the process of sensor service composition to deal with the dynamicity of WSNs. Han *et al.* [50] propose a context-aware service composition to dynamically adapt the composite process to the changes in several types of context in a building environment.

#### *4.2.2. RESTful APIs and Mashup*

Web resources identified by Universal Resource Identifiers (URIs) are considered as the core of modern Web architecture. They are accessed by clients in a synchronous request/response fashion using Hypertext Transfer Protocol (HTTP) methods such as GET, PUT, POST, and DELETE. Resource state

is kept only by the server, which allows caching, proxying, and redirection of requests and responses. Web resources may contain links to other resources creating a distributed Web between Internet endpoints, resulting in a highly scalable and flexible architecture. These are the fundamental concepts of the Web, *i.e.*, Representational State Transfer (REST) [22].

REST has emerged as a predominant Web design model with more than ten thousand RESTful APIs (services) at the time of this article [9] to facilitate a similar number of mashups that are weaving today's Web. Mashups can be seen as composition applied at the User Interface (UI)/Presentation layer [57], where a fully integrated UI is built out of reusable widgets applied to different data sources. Mashups do not emphasize on the reusability of the composition, but only the ease with which it can be built [58]. Composite services in this context (mashups) therefore are meant to be primarily used as stand-alone services.

Similarly, the RESTful service abstraction advocated by many researchers and professionals is an essential step to provision mashups in IoT systems. Guinard *et al.* in several studies [59, 45, 35, 36, 37, 38] present a continuous effort to integrate smart objects of different forms ranging from RFID, to WSNs, to embedded systems, to the Web by representing their data and events using RESTful APIs. Based on that, authors develop two approaches for mashup: Physical-Virtual and Physical-Physical in a number of applications. Many other studies [60], [61], [62] also find their ways to explore this trend over sensor nodes and embedded devices.

RESTful abstraction of smart device data and events has many advantages: lightweight, uniform identification, Web integration, and open standards. Mashup is a preliminary form of RESTful service composition, in the future IoT, more comprehensive models for composition are expected to cover many existing as well as new issues from smart object services to efficiently compose their services.

[9] http://www.programmableweb.com

### *4.3. Comparison*

We summarize service composition models in regard to composition requirements presented in Section 3 (see Table 1). It appears that early studies on composition in RFID systems and WSNs do not take into account defining requirements related to IoT such as resource constraint, power efficiency, and data/event-driven. Recent efforts with DPWS such as [50, 52] start considering these requirements. In terms of application protocols, composition with CoAP is still a missing piece of the service composition puzzle.

### *4.4. IoT Application Platforms*

Many IoT platforms have been developed to support the development of IoT application. As shown in the Table 2, these platforms mainly aim at integrating smart objects of different types into the Web through RESTful APIs or cloud services. These platforms provide mid-point services to encapsulate underlying heterogeneous smart objects into Web interfaces that can further integrate into modern Web infrastructures such as cloud and platform-as-a-service (PaaS). These approaches expose some difficulties to scale IoT systems since each platform has to handle routing discrepancy and protocol translation.

As we can observe from these platforms, RESTful APIs make service composition possible in the form of mashups; even though, there are only few of them being provided such as [63] and [64]. Beside providing the connectivity of smart objects to the Web, how multiple smart objects can interact is still missing in these platforms. Service composition models, therefore, are still missing not only in literature but also in practical.

Table 1: Comparison of state-of-the-art service composition models in IoT

| Study | Target Objects | Resource Constraint | Lossy Link | Power Efficiency | Data/Event -driven | Asynchrony | Discovery | Management | QoS Awareness |
|---|---|---|---|---|---|---|---|---|---|
| Kerer 2004 [30] | RFID | | | | | | ✓ | ✓ | |
| Xingyi 2008 [31] | RFID | | | | ✓ | | | ✓ | |
| Paliwal 2004 [32] | RFID | | | | | | ✓ | ✓ | |
| Fagui 2008 [33] | RFID | | | | | | ✓ | ✓ | |
| Vermeulen 2007 [34] | RFID | | | | | | ✓ | ✓ | ✓ |
| Guinard 2010, 2011 [35, 36, 37, 38] | RFID | | | | | | ✓ | ✓ | ✓ |
| VITRO [39] | WSN | | | | | | ✓ | ✓ | |
| MASCO [40] | WSN | | | | | | ✓ | ✓ | |
| Persistent Queries [41] | WSN | | | | | | ✓ | ✓ | |
| Graph-based [42] | WSN | | | | | | ✓ | ✓ | |
| Distributed [43] | WSN | | | | | | ✓ | ✓ | |
| Semantic Streams [44] | WSN | | | | | | ✓ | ✓ | |
| Guinard 2009 [45] | Embedded | | | ✓ | | | ✓ | ✓ | ✓ |
| Guinard2010 [46] | Embedded | | | ✓ | | | ✓ | ✓ | |
| Dar 2011 [47] | IP enabled | | | | | | ✓ | ✓ | |
| Han 2012, 2014 [49], [50] | DPWS | ✓ | | ✓ | ✓ | | ✓ | ✓ | ✓ |
| Cubo 2012, 2013 [51], [52] | DPWS | ✓ | | ✓ | | | ✓ | ✓ | |
| Li 2012 [53] | Not specified | | | | | | ✓ | ✓ | |
| Bao 2012 [55] | Not specified | | | | | | ✓ | ✓ | |

Table 2: IoT application platforms

| Platform | Target Objects | Service Modeling | Service Composition | Applications |
|---|---|---|---|---|
| Axeda [65] | IP networked | Cloud | N/A | Cloud-based platform |
| BUGswarm [66] | IP networked | RESTful APIs, Cloud | N/A | Cloud-based platform |
| Carriots [67] | Web-enabled | RESTful APIs | N/A | IoT platform |
| Etherios [68] | Embedded | Android M2M device | N/A | Platform-as-a-service (PaaS) |
| EVRYTHNG [63] | Web-enabled | RESTful APIs | Web 2.0 mashup | Personalize/Track/Socialize |
| GroveStreams [69] | Web-enabled | RESTful APIs | N/A | In-cloud real-time big data analytics for IoT |
| Nimbits [70] | WSN | RESTful APIs | N/A | In-cloud data processing |
| Open.Sen.se [71] | (Note specified) | RESTful APIs | Web 2.0 mashup (perspective) | Data storage Visualization |
| Paraimpu [72] | Web-eanbled | RESTful APIs | Web 2.0 mashup | Social Web of Things |
| NanoService [73] | Mobile phone Embedded | Nano Service Platform RESTful APIs | N/A | Embedded Web applications |
| SensorCloud [74] | MicroStrain WSN Android, iOS NI CompactRIO Web-enabled | SensorCloud OpenData APIs | N/A | Cloud Sensor Data Storage |
| ThingSpeak [75] | WSN | RESTful APIs | N/A | Sensor logging Location tracking Social network of things |
| ThingWorx [64] | (Not specified) | RESTful APIs Sockets, MQTT, AlwaysOn | Web 2.0 mashup | Cloud services Social services |
| Xively (Pachube) [76] | (Note specified) | RESTful APIs Sockets, MQTT | N/A | IoT Public Cloud Platform as a Service (PaaS) |
| Yaler [77] | Embedded (Arduino, BeagleBone Netduino, Raspberry Pi) | RESTful APIs SSH Service | N/A | Relay infrastructure for Web access of devices |

N/A = Not Applicable

## 5. Research Challenges

Since smart object services have many different characteristics compared to conventional Web services, service composition problem in the next full-IP IoT has many challenges that invoke the need of redesigning composition models to fully realize the its potential. We hereby discuss these challenges.

### *5.1. RESTful Service Composition*

Apart from DPWS, which is compliant to W3C Web Service, a majority of the current IoT application platforms (see Figure 2) use RESTful APIs to abstract smart object services in order to seamlessly integrate them into current Web infrastructure. Besides, Web 2.0 applications have moved away from SOAP services toward more cohesive collections of RESTful services (RESTful APIs) [78]. CoAP was designed according to the REST architecture, and thus exhibits functionality similar to that of the HTTP protocol.

We foresee that the RESTful service composition will play an important role, particularly in the context of smart objects in which identifying an atomic service modeling takes the center stage for the composition. RESTful composite services should be reusable and interactive rather than non-recursive and read-only mashups. It is required to devise novel languages and techniques helping to more effectively build composite RESTful services.

Service composition with RESTful services, however, cannot directly inherit the concepts of SOC since REST is based on resources (with HTTP methods) and the hyperlink between them rather than operations that can be logically described by description languages such as WS-BPEL and BPMN.

### *5.2. Composite Service Interface Description*

It is necessary for composite services to have interfaces that describe what they can do. Web service composition models use description languages such as WS-BPEL and BPMN to describe service operations defined by parameter inputs and value outputs. In the context of IoT, interface description is a challenge

due to the lack of the support to IoT characteristics such as data/event-driven, resource constraint, and asynchrony.

Another issue is with REST architecture. RESTful services comply with the uniform interface principle, where resources are manipulated using the GET, PUT, DELETE, and POST methods. A composite service should not only be able to invoke its component services, but it also should be able to handle these requests. Also, the underlying discovery mechanism will define in which way a composite service should be described. To summarize, the following issues should be considered when designing a description language:

- REST support
- Data/event-driven support
- Asynchrony support
- Discovery compatibility

### *5.3. Recursiveness*

Composite services can play the role of atomic services and can be recursively composed with other services to create other composite services. Such recursive composition of services is one of the most important features of SOC, allowing to rapidly build new application based on the existing services. As the amount services (and their compositions) grows, the easier it becomes to implement new applications.

To achieve the recursiveness of composition problem in IoT, the life cycle of atomic services needs to be reproduced in composite services such as: description, execution, and verification.

### *5.4. Semantic Composition*

Semantics technology has been used widely to automate and improve many aspects of the service composition [79]. Meanwhile, semantics are penetrating into the Web world as an increasing number of Web resources are marked up

with semantic annotations[10]. When smart objects find their way into the Web, the potential of the semantics technology cannot be ignored. It opens up an opportunity to support more intelligent composition processes but also brings in challenges such as semantic modeling of smart objects. Currently, approaches in fundamental issues of semantics in IoT are at an early stage. SPITFIRE [80] is a system trying to give semantic annotations for smart objects through a centralized service (server). This exposes many restrictions as the third-party services would be able to filter the information from the original smart object services, and again play a role of protocol translation gateways.

A more intuitive way of making objects semantically expressive should strictly follow the semantic Web guidelines and standards whereby smart objects themselves represent their services semantically. This can be done, in case of RESTful APIs, by adding media type of *application/rdf+xml* that notify clients to process semantic data from smart objects and content is represented in Resource Description Language linking to an appropriate ontology.

### *5.5. Context-awareness*

Context-awareness is important and generally defined in ubiquitous computing, where it is considered the key to the effort of bringing computation into daily lives [81]. The major task in context-aware is to acquire and use the knowledge about the state of users and devices including surroundings, situation, and (to a less extent) location information in order to provide the most appropriate services. Service composition in IoT are related to a particular person, place, time, or event, therefore, it is a challenge to any service composition solution to be aware of the changes in context of the device, the user, and the surrounding environment. As suggested in [50], a context-aware description language can be used to describe composite services, and semantics technology can provide the *intelligence* for the composite services to adapt themselves according to the changes in context of different entities.

[10] http://lod-cloud.net/

### *5.6. Hybrid Composition*

Future Internet has many different stakeholders using IP as the communication media. With IP support, smart objects become an integral part of the Internet where their services can be treated in the same way as other services offered by today's Internet. Service composition then can be done with the participation of not only smart object services but also from many other sources to create an Internet of services. This is a new form of the heterogeneous services, combining things and services all together under service-oriented paradigm. The fulfillment of this vision is a challenge where new standards, languages, frameworks for service composition are required to consider the various types of service sources.

### *5.7. Privacy and Security*

Composition languages such as WS-BPEL do not provide any security concepts that user could leverage. All security aspects are left to the WS-BPEL engine or, in other words, to the WS-BPEL engine wrapper. Similar issue would happen in IoT environment when developing composition technique and language. Security and privacy in IoT render even more complications than in classic Internet environment. Smart objects in the IoT often have a relationship to real persons, who could be owner(s), manufacturer(s), user(s), administrator(s), or many other functions. A product might be owned by a manufacturer first and subsequently by a user who bought the product. The owner, user or administrator of an object might change over time. Ownership and identity relationships in the IoT have an impact on other identity related processes like, *e.g.*, authentication, authorization. The owner of a thing might be challenged for authentication or be asked for authorization policies. Similarly with composite services, if such relationships are taken into account, it would be very challenging to identify the relationship between services and providers.

## 6. Conclusion

Service composition in IoT promotes the idea of assembling smart object services in novel and creative ways into composite services that can be used in multiple IoT application domains to augment the power of the IoT. As inspired from the composition concept of reusable components in software engineering, IP technology for smart objects, and the legacy of W3C Web Service composition methods, we foresee service composition will be a key to stimulate the development of future IoT applications to build a smarter world. Especially with the inevitable arrival of the full-IP IoT, we are facing new composition issues to support the creation of a new generation of IoT applications. By the analysis of the service composition problem and its challenges along with a review of its early-stage studies, we hope to serve as the start for upcoming research in the field and identify some possible approaches to these challenges.

## Acknowledgement

This work is supported by two European projects: ITEA2 Social Internet of Things: Apps by and for the Crowd (SiTAC) and ITEA2 Future Unified System for Energy and Information Technology (FUSE-IT).

## References

[1] M. Roberti, The internet of things revisited, RFID Journal.

[2] Devices Profile for Web Services Version 1.1, Tech. rep., OASIS (Jul. 2009).

[3] Z. Shelby, K. Hartke, C. Bormann, Constrained Application Protocol (CoAP), IETF Internet Draft – work in progress 18, IETF (Jun. 2013).

[4] M. N. Huhns, M. P. Singh, Service-oriented computing: Key concepts and principles, IEEE Internet Computing 9 (1) (2005) 75–81.

[5] S. Dustdar, W. Schreiner, A survey on web services composition, Intl. Journal of Web and Grid Services 1 (1) (2005) 1–30.

[6] Analyst Geoff Johnson interviewed by Sue Bushell, M-commerce key to ubiquitous internet (Jul. 2000).

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

[8] J.-P. Vasseur, A. Dunkels, Interconnecting Smart Objects with IP: The Next Internet, Morgan Kaufmann, 2010.

[9] A. Mitrokotsa, C. Douligeris, Integrated rfid and sensor networks: architectures and applications, RFID and Sensor Networks (2009) 511–536.

[10] G. Kortuem, F. Kawsar, D. Fitton, V. Sundramoorthy, Smart objects as building blocks for the internet of things, IEEE Internet Computing 14 (1) (2010) 44–51.

[11] G. Fortino, P. Trunfio (Eds.), Internet of Things Based on Smart Objects, Spr, 2014.

[12] Z. Shelby, C. Bormann, 6LoWPAN: The Wireless Embedded Internet, Wiley Publishing, 2010.

[13] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, R. Alexander, RPL: Ipv6 routing protocol for low-power and lossy networks, RFC 6550 (Proposed Standard) (Mar. 2012).

[14] S. N. Han, S. Park, G. M. Lee, N. Crespi, Extending the device profile for web services (dpws) standard using a rest proxy, IEEE Internet Computing 19 (1) (2015) 10–17.

[15] A. Stanford-Clark, H. L. Truong, Mqtt for sensor networks (mqtt-sn) protocol specification, Tech. rep., IBM (Nov. 2013).

[16] Web Services Architecture, W3C working group note, W3C (Feb. 2004).

[17] C. Lerche, N. Laum, G. Moritz, E. Zeeb, F. Golatowski, D. Timmermann, Implementing powerful web services for highly resource-constrained de-

vices, in: 2011 IEEE International Conference on Pervasive Computing and Communications Workshops, 2011, pp. 332–335.

[18] G. Moritz, D. Timmermann, R. Stoll, F. Golatowski, Encoding and compression for the devices profile for web services, in: 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2010, pp. 514–519.

[19] G. Moritz, F. Golatowski, C. Lerche, D. Timmermann, Beyond 6lowpan: Web services in wireless sensor networks, IEEE Transactions on Industrial Informatics 9 (4) (2013) 1795–1805.

[20] I. Samaras, G. Hassapis, J. Gialelis, A modified DPWS protocol stack for 6lowpan-based wireless sensor networks, IEEE Transactions on Industrial Informatics 9 (1) (2013) 209–217.

[21] X. Yang, X. Zhi, Dynamic deployment of embedded services for DPWS-enabled devices, in: 2012 Int. Conf. on Computing, Measurement, Control and Sensor Network (CMCSN), 2012, pp. 302–306.

[22] R. T. Fielding, R. N. Taylor, Principled design of the modern web architecture, ACM Trans. Internet Technol. 2 (2) (2002) 115–150.

[23] U. Aßmann, Invasive software composition, Springer, 2003.

[24] R. Sessions, Fuzzy boundaries: objects, components, and web services., ACM Queue 2 (9) (2004) 40–47.

[25] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer, Simple Object Access Protocol (SOAP) 1.1, W3C note, W3C (May 2000).

[26] Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C recommendation, W3C (Jun. 2007).

[27] Web Services Business Process Execution Language (WS-BPEL) Version 2.0, Tech. rep. (Apr. 2007).

[28] O. M. G. (OMG), Business process model and notation (bpmn) version 2.0, Tech. rep. (Jan. 2011).

[29] K. Finkenzeller, RFID Handbook, 3rd Edition, Wiley, 2010.

[30] C. Kerer, S. Dustdar, M. Jazayeri, D. Gomes, A. Szego, J. B. Caja, Presence-aware infrastructure using web services and rfid technologies, in: Proceedings of the 2nd European workshop on object orientation and web services, Oslo, Norway, 2004.

[31] J. Xingyi, L. Xiaodong, K. Ning, Y. Baoping, Efficient complex event processing over rfid data stream, in: 2008 Seventh IEEE/ACIS Intl. Conf. on Computer and Information Science (ICIS 08), IEEE, 2008, pp. 75–81.

[32] A. V. Paliwal, N. Adam, C. Bornhövd, J. Schaper, Semantic discovery and composition of web services for rfid applications in border control, in: 3rd Intl. Semantic Web Conference (ISWC), Vol. 4, 2004.

[33] L. Fagui, L. Kun, Z. Yang, Semantic web services and its application in third-party logistics, in: International Workshop on Education Technology and Training and International Workshop on Geoscience and Remote Sensing (ETT/GRS 2008), IEEE, 2008, pp. 626–630.

[34] J. Vermeulen, K. Luyten, K. Coninx, Tangible mashups: exploiting links between the physical and virtual world, in: 1st Intl. Workshop on System Support for the Internet of Things (WoSSIoT'07), 2007.

[35] D. Guinard, Mashing up your web-enabled home, in: Current Trends in Web Engineering, Springer, 2010, pp. 442–446.

[36] D. Guinard, M. Mueller, J. Pasquier-Rocha, Giving rfid a rest: building a web-enabled epcis, in: Internet of Things (IOT), 2010, IEEE, 2010, pp. 1–8.

[37] D. Guinard, C. Floerkemeier, S. Sarma, Cloud computing, rest and mashups to simplify rfid application development and deployment, in: Pro-

ceedings of the Second International Workshop on Web of Things, ACM, 2011, p. 9.

[38] D. Guinard, M. Mueller, V. Trifa, Restifying real-world systems: A practical case study in rfid, in: REST: From Research to Practice, Springer, 2011, pp. 359–379.

[39] Z. Movahedi, B. Defude, A high-level service composition model for building applications on sensor networks, in: 2013 IEEE 22nd International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013, pp. 202–207.

[40] I. Toure, Y. Yang, Z. Q. Mi, L. Wang, Low redundant hop-counts for service composition optimization in dynamic network, in: 2011 Intl. Conf. on Cloud and Service Computing (CSC), 2011, pp. 26–31.

[41] X. Wang, J. Wang, Z. Zheng, Y. Xu, M. Yang, Service composition in service-oriented wireless sensor networks with persistent queries, in: 2009 6th IEEE Consumer Communications and Networking Conference(CCNC 2009), IEEE, 2009, pp. 1–5.

[42] S. Geyik, B. Szymanski, P. Zerfos, Robust dynamic service composition in sensor networks, IEEE Transactions on Services ComputingForthcoming.

[43] L. Zhang, M. Ma, G. Zhang, A. Lim, Distributed composition services for self-adaptation wireless sensor networks, in: 2013 Computing, Communications and IT Applications Conference (ComComAp), 2013, pp. 47–52.

[44] K. Whitehouse, F. Zhao, J. Liu, Semantic streams: A framework for composable semantic interpretation of sensor data, in: Wireless Sensor Networks, Springer, 2006, pp. 5–20.

[45] D. Guinard, 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 (Intl. World Wide Web Conferences), Madrid, Spain, 2009.

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

[47] K. Dar, A. Taherkordi, R. Rouvoy, F. Eliassen, Adaptable service composition for very-large-scale internet of things systems, in: Proceedings of the 8th Middleware Doctoral Symposium, ACM, 2011, p. 2.

[48] M. Roelands, J. Plomp, D. Mansilla, J. V. Salhi, G. Lee, N. Crespi, C. Valderrama, N. Menezes, M. Lopez-Ramos, C. van Nimwegen, D. D. Roeck, L. L. Bastida, Q. Reul, The diy smart experiences project: A european endeavour removing barriers for user-generated internet of things applications, in: D. Uckelmann, M. Harrison, F. Michahelles (Eds.), Architecting the Internet of Things, Springer, New York Dordrecht Heidelberg London, 2011, Ch. 11, pp. 279–316.

[49] S. N. Han, G. M. Lee, N. Crespi, Context-aware service composition framework in web-enabled building automation system, in: 2012 16th Intl. Conf. on Intelligence in Next Generation Networks (ICIN), 2012, pp. 128–133.

[50] S. N. Han, G. Lee, N. Crespi, Semantic context-aware service composition for building automation system, IEEE Transactions on Industrial Informatics 10 (1) (2014) 752–761.

[51] J. Cubo, A. Brogi, E. Pimentel, Behaviour-aware compositions of things, in: 2012 IEEE International Conference on Green Computing and Communications (GreenCom), IEEE, 2012, pp. 1–8.

[52] J. Cubo, L. González, A. Brogi, E. Pimentel, R. Ruggia, Towards runtime verification of compositions in the web of things using complex event processing, in: IX Jornadas de Ciencia e Ingeniera de Servicios (JCIS), 2013.

[53] K. Li, L. Jiang, The research of web services composition based on context in internet of things, in: 2012 IEEE Intl. Conf. on Computer Science and Automation Engineering (CSAE), Vol. 1, 2012, pp. 160–163.

[54] OWL 2 web ontology language document overview, W3C recommendation, W3C (Oct. 2009).

[55] F. Bao, R. Chen, Trust management for the internet of things and its application to service composition, in: 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE, 2012, pp. 1–6.

[56] V. Issarny, N. Georgantas, S. Hachem, A. Zarras, P. Vassiliadist, M. Autili, M. A. Gerosa, A. B. Hamida, Service-oriented middleware for the future internet: state of the art and research directions, Journal of Internet Services and Applications 2 (1) (2011) 23–45.

[57] F. Daniel, M. Matera, J. Yu, B. Benatallah, R. Saint-Paul, F. Casati, Understanding ui integration: A survey of problems, technologies, and opportunities, IEEE Internet Computing 11 (3) (2007) 59–66.

[58] S. Vinoski, Serendipitous reuse, Internet Computing, IEEE 12 (1) (2008) 84–87.

[59] D. Guinard, V. Trifa, T. Pham, O. Liechti, Towards physical mashups in the web of things, in: Proceedings of INSS 2009 (IEEE Sixth International Conference on Networked Sensing Systems), Pittsburgh, USA, 2009.

[60] D. Zhiquan, Y. Nan, C. Bo, C. Junliang, Data mashup in the internet of things, in: 2011 International Conference on Computer Science and Network Technology (ICCSNT), Vol. 2, IEEE, 2011, pp. 948–952.

[61] E. Avilés-López, J. A. García-Macías, Mashing up the internet of things: a framework for smart environments, EURASIP Journal on Wireless Communications and Networking 2012 (1) (2012) 1–11.

[62] K. Kenda, C. Fortuna, A. Moraru, D. Mladenić, B. Fortuna, M. Grobelnik, Mashups for the web of things, in: Semantic Mashups, Springer, 2013, pp. 145–169.

[63] Evrythng.
URL http://www.evrythng.com/

[64] Thingworx.
URL http://www.thingworx.com/

[65] Alexa.
URL http://www.axeda.com/

[66] BUGswarm.
URL http://developer.bugswarm.net/

[67] Carriots.
URL https://www.carriots.com/

[68] Etherios.
URL http://www.etherios.com/products/devicecloud/

[69] Grovestreams.
URL https://grovestreams.com/

[70] Nimbits.
URL http://www.nimbits.com/

[71] Open.Sen.se.
URL http://open.sen.se/

[72] Paraimpu.
URL http://paraimpu.crs4.it/

[73] Sensinode.
URL http://www.sensinode.com/

[74] SensorCloud.
URL http://www.sensorcloud.com/

[75] Thinkspeak.
URL https://www.thingspeak.com/

[76] Xively.
URL https://xively.com/

[77] Yaler.
URL https://yaler.net/

[78] Programmableweb.
URL http://www.programmableweb.com/

[79] N. Milanovic, M. Malek, Current solutions for web service composition, IEEE Internet Computing 8 (6) (2004) 51–59.

[80] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kroller, M. Pagel, M. Hauswirth, M. Karnstedt, M. Leggieri, A. Passant, R. Richardson, Spitfire: toward a semantic web of things, IEEE Communications Magazine 49 (11) (2011) 40–48.

[81] A. Schmidt, Ubiquitous computing-computing in context, Ph.D. thesis, Lancaster University (2003).