future internet

MDPI

Article

Contribution of the Web of Things and of the Opportunistic Computing to the Smart Agriculture: A Practical Experiment

Lionel Touseau * and Nicolas Le Sommer *

IRISA-UBS, Université Bretagne Sud, BP 573, 56017 Vannes CEDEX, France

* Correspondence: lionel.touseau@univ-ubs.fr (L.T.); nicolas.le-sommer@univ-ubs.fr (N.L.S.)

Received: 31 December 2018; Accepted: 29 January 2019; Published: 1 February 2019

Abstract: With the emergence of the Internet of Things, environmental sensing has been gaining interest, promising to improve agricultural practices by facilitating decision-making based on gathered environmental data (i.e., weather forecasting, crop monitoring, and soil moisture sensing). Environmental sensing, and by extension what is referred to as precision or smart agriculture, pose new challenges, especially regarding the collection of environmental data in the presence of connectivity disruptions, their gathering, and their exploitation by end-users or by systems that must perform actions according to the values of those collected data. In this paper, we present a middleware platform for the Internet of Things that implements disruption tolerant opportunistic networking and computing techniques, and that makes it possible to expose and manage physical objects through Web-based protocols, standards and technologies, thus providing interoperability between objects and creating a Web of Things (WoT). This WoT-based opportunistic computing approach is backed up by a practical experiment whose outcomes are presented in this article.

Keywords: Web of Things; intermittent connectivity; opportunistic computing; opportunistic sensing; precision agriculture; smart agriculture

1. Introduction

With the recent progress achieved in the automation of farm machinery and in the monitoring of the physical environment, agriculture moves towards more environmentally-friendly, smart and precise agriculture, addressing challenges ranging from manpower shortage, adaptation to climate changes, optimization of environment resource usage, reduction of the usage of chemical crop protection products, etc. The Internet of Things (IoT)—which aims at connecting to the Internet physical objects that can sense, communicate, compute and sometimes actuate—can indubitably contribute to the development of the smart agriculture and farming. A few research works have recently experimented with the usage of IoT platforms in the agriculture and farming context [1–3]. These experiments have highlighted several issues that must be addressed to efficiently and automatically collect data in such challenging contexts, especially regarding the network connectivity, the network architecture and the interoperability between physical objects. Applications and services dedicated to smart agriculture and farming can tolerate, in some cases, delays and disruptions in data exchanges. Indeed, some physical objects do not need to be connected to the Internet permanently, because they measure environmental properties that do not evolve quickly (e.g., soil moisture and plant growth), and because they are in sleep mode most of the time for energy saving purposes. An alternative or a complement to long range, low power and low rate wireless standards such as LoRa/LoRaWAN and SigFox, could be to interconnect such devices to gateways opportunistically using data mules and short radio range wireless interfaces such as IEEE 802.15.4 and IEEE 802.11—that

allow transfering, thanks to a higher throughput, a significant amount of data in a short time from sensors (e.g., camera) or to agribots (field maps, tasks to achieve, etc.). Such an alternative, which relies on opportunistic communication and computing techniques, is known as the opportunistic sensing paradigm [4].

Besides these problems, the lack of semantic description and logical representation of physical objects does not allow easily consolidating collected data with data from external Web services (e.g., weather forecast and plant growth models), to assemble objects to build sophisticated systems, and to create high level services and applications that should help farmers to take decisions, or even to perform tasks automatically. The Web of Things (WoT) [5] is an attractive solution to address these issues. Indeed, the WoT extends the IoT, and aims at doing for physical objects what the Web did for information resources, namely an identification of resources using Uniform Resource Identifiers (URIs), a semantic description of resources, an access of resources through standard protocols, and the inclusion in resources of links to other resources to enable a scalable discovery of highly distributed resources. Following the trend of Web 2.0 participatory services, in particular Web Mashups, it thus is possible to create applications mixing physical devices with virtual services on the Web—this type of applications is often referred to as physical Mashup [6,7].

In this paper, we present a WoT oriented platform called ASAWoO (https://asawoo.gitlab.io/ [8]), that implements opportunistic and disruption-tolerant networking and computing techniques. This platform makes it possible to define and instantiate logical extensions, called Avatars, of physical objects. Physical objects can be configured and controlled through their avatar. Avatars provide a semantic description of physical objects and of their functionalities. Functionalities are implemented as REST services, and can be deployed dynamically by the Avatars themselves according to the description of the physical objects and of the execution context of these ones. Similar to the functionalities, execution contexts are described semantically, thus allowing Avatars to reason on these descriptions and to perform adaptations dynamically. REST services can be invoked using standard protocols such as HTTP or CoAP, either using Internet-legacy transport protocols or disruption-tolerant and opportunistic protocols. This platform can be deployed on physical objects themselves provided that they have enough processing and memory capacities, on gateways to which physical objects are connected or on cloud infrastructure. Objects can be fixed or mobile. For instance, data generated by the fixed sensors can be collected by data mules (e.g., tractors) and transmitted to their logical representations deployed in a cloud infrastructure. These features make this platform a relevant support for the development and the deployment of cyber-physical systems and applications dedicated to the environmental sensing and to the smart agriculture. In this paper, we also report experiments we achieved with this platform in an experimental agricultural farm.

The remainder of this paper is structured as follows. Section 2 presents research works dealing with opportunistic environmental sensing, and details their limitations. Section 3 presents the ASAWoO middleware platform and the concept of Avatars, and discusses the advantages such a platform could bring in an opportunistic sensing and actuating context such as smart agriculture and farming. Section 4 describes the experiments we performed with the ASAWoO platform, and presents some results we obtained. Section 5 concludes this paper by summarizing our contribution.

2. Related Work

Over the last years, a large majority of the research dealing with agricultural crop monitoring has focused on the development of small-scale testbeds and specialized applications built on wireless networks formed by fixed sensors (WSN) [3,9,10]. Kim et al. [11] presented an example of such works that rely on infrastructure, where all the devices—a meteorological station, a sprinkler system and humidity sensors deployed a the field; a base station installed near the field and connected to the network infrastructure, which plays the role of gateway between the network of sensors; and the infrastructure—are in radio range. Some works consider the issues of distances and of intermittent connections—mainly induced by the duty cycles of sensors for energy saving purposes—between the

devices, and solves them by adding redundant relays and gateways to build a connected network. A few works dealing with precision and sustainable agriculture propose to exploit the mobility of devices carried by people or embedded in agricultural vehicles (e.g., farm tractors and agribots) in order to collect data generated by sensors or to deliver data to actuators (e.g., sprinkler system and agribots). Such a collection process, which leverages the contact opportunities between devices, is referred as being opportunistic sensing.

Opportunistic sensing and communication techniques, which have an indubitable interest in the agricultural domain, have also been considered in a few systems that propose to exploit mobile devices (carried by people or embedded in vehicles) to collectively measure social or environmental data in large-scale pervasive environments. These systems rely on opportunistic and people-centric sensing techniques. They differ from first-generation sensor networks, not only in their goal to support concurrent people-centric and opportunistic sensing applications in urban environments, but also in their hardware and software heterogeneity, high volatility, and very large scale. CarTel [12], MetroSense [13], MobiScopes [14], SenseWeb [15], Urbanet [16] and Urban Atmospheres [17] are examples of such systems. To report information from a given region using these systems, applications run sensing tasks on the mobile devices that are located in this region and that have chosen to participate to the sensing process. Mobile devices report sensor data through opportunistic network connections, such as those they establish with third-party access points existing in their radio range. For instance in CarTel, a network node is a mobile device coupled to a set of sensors. Each node gathers and processes sensor data locally before delivering them to a central portal on the Internet, where the data are stored in a database for further analysis and visualization. Mobile devices opportunistically communicate, using their wireless interface (e.g., Wi-Fi and Bluetooth), with other CarTel mobile devices and with Internet access points found in urban areas. The portal and the mobile nodes use a delay-tolerant network stack, called CafNet, to communicate. CarTel applications run on the portal, and submit SQL queries to a delay-tolerant continuous query processor called ICEDB. This query processor maintains a list of queries submitted by the applications, and pushes them to remote nodes using CafNet. It receives results from remote nodes, and puts them into a relational database. MetroSense adopts a similar approach, and envisions a future Internet in which a large part of the data traffic is generated by sensors carried by people during their daily life. BikeNet [18] and CenceMe [19] are examples of applications based on the MetroSense system. In BikeNet, bicycles equipped with multiple sensors communicated with neighboring bikes as well as the network infrastructure, and deliver sensing data to a Web portal that promotes social networking among cyclists. Similarly, in CenceMe, mobile phones collect data about a user's activity and share it with buddies via a social network. Several other sensing applications have been proposed by project Urban Atmospheres [17] and the MIT Senseable City Lab [20]. Project SenseWeb proposes a Web-based platform and tools to publish and query sensor data across Internet. Sensors can measure various physical variables, can be static or mobile and carried by humans, or embedded in vehicles or on robots. To hide the heterogeneity and the intermittent connectivity of sensors, sensors are connected to a gateway that provides a uniform interface to communicate with them. Urbanet also focuses on sensing in spontaneous urban networks composed of heterogeneous and potentially mobile sensors. However, unlike the above-mentioned works, it does not rely on central collection points across the Internet that perform data and task management, and act as mediators between users and the network. Urbanet proposes three middleware solutions to develop distributed sensing applications that do not require servers or Internet connectivity; sensing applications running on mobile devices. These solutions, respectively, present the network: (1) as a distributed sensor database that can be queried via a SQL-like language; (2) as a single virtual name space that applications use to access individual resources offered by nodes; and (3) as a client-service model where services migrate to different network nodes to maintain a semantically correct and continuous interaction with clients.

The above-mentioned works have drawbacks regarding the opportunistic protocol they implement, as well as Web-based approach they propose. Indeed, similar to CafNet [21], opportunistic

protocols used in these works often rely on a simple epidemic (nodes exchange with the other nodes they encounter all the messages they have). However, such an epidemic approach is known to not be the more efficient one, especially when there are many data to exchange; more sophisticated and efficient opportunistic protocols have indeed been proposed to address the issues of the epidemic routing protocol [22]. Moreover, these works propose a Web portal that renders the data that have been collected from the sensors, and do not propose logical extensions of physical objects in the Web that are able to provide a semantic description of these objects and of their functionalities, and that offer means to configure and to control them through Web standards and protocols. However, as mentioned in the Introduction, such an approach offers significant advantages, the physical Mashup being one of them. WoT solutions have been investigated in the agricultural context [23,24] (and also in the smart home context [25,26]) to improve the processing of dataflows produced by sensors and to expose API of physical objects in the Web. Agri-IoT [23] is a data analytics platform composed of multiple layers, allowing to communicate with devices (lower layer), to analyze data (intermediate layers) and to present information to end-users either using dashboards or mobile applications (higher layer). In this platform, sensors, and the data they produce, are semantically described using agricultural ontologies. However, again, these solutions do not implement disruption-tolerant or opportunistic networking techniques to cope with the connectivity disruptions that can occur in smart agriculture and farming application domains, and more generally in scenarios involving mobile devices communicating with short range wireless interfaces.

In the next section, we present a WoT dedicated platform called ASAWoO. This platform implements opportunistic networking and computing mechanisms, and addresses the drawbacks of above-presented works.

3. Leveraging the Web of Things and the Opportunistic Computing to Build Systems for the Smart Agriculture

Smart agriculture mainly consists in growing seedlings and plants while preserving natural resources (e.g., water) and limiting crop protection products. To do so, both environmental properties (soil moisture, ambient temperature, sunshine, etc.) and crop growth are monitored using fixed sensors or sensors embedded on field machines, drones or agribots. Collected data are compared to agricultural data and models to make decisions about the treatments to be performed and the means to achieve them (traditional agricultural machines, agribots, and drones). We argue that models, protocols, techniques and approaches developed both in the WoT and in the opportunistic computing can help develop and deploy cyber-physical systems and applications dedicated to the smart agriculture. In the remainder of this section, we develop and justify these words by explaining successively how the opportunistic computing and the WoT can contribute to smart agriculture. Before detailing these contributions, we remind the key features of the opportunistic computing and of the WoT.

3.1. Contribution of the Opportunistic Computing

This subsection first explains what opportunistic computing is and to what extent it could benefit smart agriculture. Then, it thoroughly describes how a support for opportunistic computing was implemented in a Web of Things platform.

3.1.1. Definitions of Opportunistic Computing and Sensing, and Their Suitability for Smart Agriculture

Opportunistic networking is a research field that can be considered as being a derivative of the Delay-Tolerant Networking (DTN) research domain. While DTN [27] characterizes networks with long-delay communications and/or with an unreliable connectivity, opportunistic networks denote mobile networks that exploit transient—and sometimes non-predictable—radio contacts between mobile devices to exchange data [28]. Opportunistic networking therefore mostly targets disconnected mobile ad hoc networks (D-MANETs), as illustrated on Figure 1. However, the differences between these two approaches, which both rely on the "store-carry-and-forward" principle, are marginal,

and both terms are indistinctly used in this paper. Numerous opportunistic forwarding protocols have been proposed over the past years [22], investigating forwarding strategies based notably on epidemic approaches or on social criteria (computed from mobility patterns, history of contacts or location information). Opportunistic computing [29] is an extension of opportunistic networking, where resources offered by devices are, in general, abstracted and made available through services, thus allowing devices to access resources provided by the devices they have encountered directly or via intermediate devices. Opportunistic computing and networking do not make assumptions about the network topology and support frequent and unpredictable changes in this one, as well as its fragmentation.

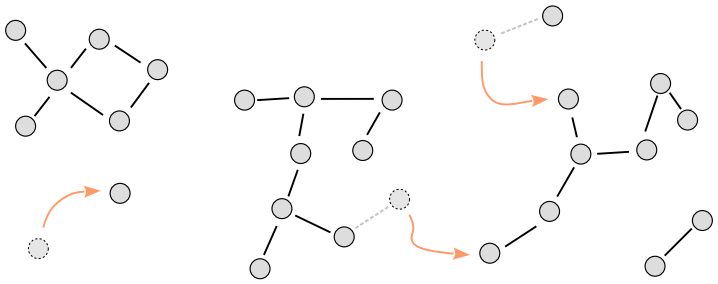

Figure 1. Opportunistic networking in a disconnected mobile network.

Such features make opportunistic computing and networking interesting for the development of systems dedicated to the smart agriculture. Unlike urban areas that may offer solid network infrastructure favorable to environmental sensing, farms are usually located in rural areas where the cellular network coverage is often lackluster and therefore does not provide a reliable network infrastructure. Furthermore, devices involved in smart agriculture (e.g., sensors, agribots, and drones) are by nature likely to fall in the D-MANET category, because devices may run out of battery or enter a sleep mode for energy-saving purposes, and because mobile devices enter and exit the communication radius of other nodes as they roam. Computing systems relying on opportunistic communications, however, make it possible to collect data generated by sparsely distributed sensors, even if these ones are not up at the same time and/or if it does not exist an end-to-end path between some sensors and the gathering point. For this purpose, mobile devices (e.g., tractors, agribots, drones, and smartphones carried by farmers) can in addition be used as "data mules" to collect data from sensors directly or to transfer data between the different parts of the network that are isolated from one another. Opportunistic computing furthermore allows farmers to deploy sensors or actuators in their fields and/or in their farms, even on their animals [30], without having to worry about the location, the radio range, the duty cycle and the sleep phases of these devices.

For the sake of illustration, let us consider a crop irrigation management application that can process soil moisture data produced by sensors deployed in agricultural fields, and that can trigger the watering system if the soil is too dry and if no rainfall is expected soon. As shown in Figure 2, two farming vehicles A and B, which act as data mules in this smart agriculture scenario, can participate to the collect of data and to their transmission to the gateway, where they will be analyzed by a WoT application in order to decide if the watering system must be triggered or not. In Figure 2A, gateway *G* emits a service invocation message to query sensor *S*. This message is stored and carried by tractor A. In Figure 2B, the tractor enters in the radio range of sensor *S*, and delivers the message to *S*. Finally, in Figure 2C, the service response sent by sensor *S* is forwarded back to the gateway *G* by tractor *B*, which returns to its shelter located in the gateway vicinity. The experiment further presented in this paper implemented a similar scenario.

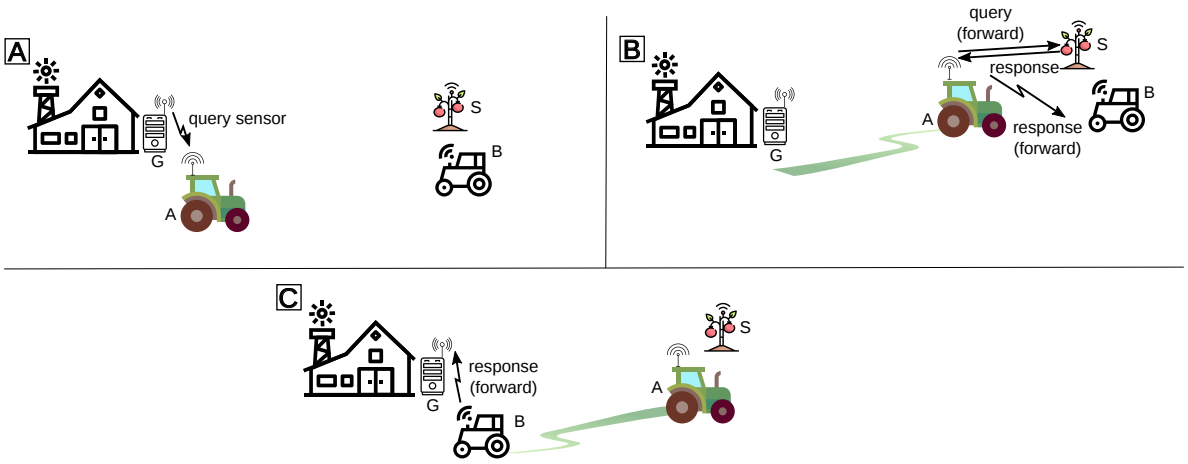

Figure 2. Opportunistic Sensing in a crop monitoring scenario. (**A**) A sensor query is emitted by gateway G. (**B**) Tractor A carries the query to sensor S. (**C**) Another tractor B forwards sensor data back to G.

3.1.2. Opportunistic Computing Implementation in ASAWoO Web of Things Platform

In project ASAWoO, we designed and developed a disruption-tolerant RESTful support for the WoT [31]. In this opportunistic computing implementation, resources offered by physical objects are identified by URIs and accessed through stateless services. Service requests and responses are forwarded using the "store-carry-and-forward" principle. This communication support provides a complete service invocation model, allowing to perform unicast, anycast, multicast and broadcast service invocations using either HTTP or CoAP, which makes it particularly suited for the WoT. It is compliant with the six constraints—the sixth one is optional—defined by the REST architecture style for performance, scalability and simplicity purposes. Indeed, it does not maintain any session state, and implements a loosely-coupled client/server architecture where resources offered by physical objects are accessible through well defined service interfaces. By implementing the "store-carry-and-forward" principle, it makes it possible to store service responses in the cache of clients and of intermediate nodes, but also the service requests that have been sent in the network. Intermediate hosts equipped with this communication system can also act as proxies, and can respond on behalf of a server if they have the response in their local cache, when this one is still valid. Such an approach improves the performance and the scalability of the system, because it naturally performs load balancing and data caching on intermediate hosts, and thus fulfills both the "system layering" requirement and the "response cacheablilty" requirement promoted by the RESTful architecture style. Finally, as WoT applications can be partially or fully developed in Javascript, a part of the application can be executed on the client side, thus allowing it to be compliant with REST optional "code-on-demand" constraint, and to reduce the computation load on the physical objects.

As with most disruption-tolerant communication systems, our system implements a multiple copy forwarding strategy. It can thus take advantage of these message transmission models to increase the message delivery probability and to reduce the response time in a WoT context. For instance, several sensors can simultaneously be invoked using a multicast transmission model without naming them explicitly. All the responses returned by these sensors will be transmitted to the client.

The system we have developed allows to invoke services using the HTTP and CoAP application-level protocols. The application-level messages (i.e., HTTP and CoAP messages) can be encapsulated in UDP datagrams, in TCP segments or in messages of a given disruption-tolerant communication system in order to be transmitted to their destination. Different wireless technologies (e.g., Bluetooth, Wi-Fi, and ZigBee) can be employed to communicate with physical objects. As shown in Figure 3, this system is implemented by two main elements, namely an HTTP/CoAP proxy and a DTN adapter. The HTTP/CoAP proxy makes it possible for standard HTTP and CoAP clients and servers to send and receive service requests and responses using disruption-tolerant communication

techniques. Thanks to this proxy, programmers can develop HTTP and CoAP WoT applications using regular HTTP and CoAP, libraries. Moreover, standard HTTP and CoAP servers do not need to be modified. This proxy can also invoke remote REST services using Internet-legacy routing protocols (i.e., TCP/IP). The HTTP/CoAP proxy is a common element shared between all the implementations. As for the DTN adapter, it binds the proxy and the disruption-tolerant communication system in charge of forwarding messages in the network. Hence, the DTN adapter depends on the underlying communication system and is specifically developed for each different system. Two adapters have been developed: one relying on the Bundle Protocol (BP) [32], which is the standard message-based protocol over the DTN architecture [33], and another one relying on the C3PO opportunistic communication platform C3PO [34]. This platform provides interesting features to implement the anycast, multicast and broadcast communication models efficiently. For instance, service responses are used to stop the propagation of service requests in the network, and messages can be restricted to a given geographical area. In the experiment presented in the next section, we used the C3PO adapter. The original and efficient features implemented in this RESTful opportunistic computing middleware support allows platform ASAWoO to not have the same drawbacks as those presented in Section 2, and make it an interesting platform adapted to the opportunistic sensing and to the smart agriculture and farming.

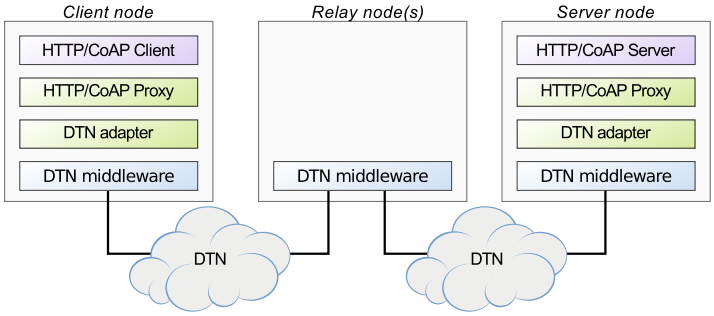

Figure 3. Overview of the implementation of the RESTful opportunistic computing middleware support.

3.2. Contribution of the Web of Things

The Web of Things allows controlling and communicating with physical objects through the logical software entities that represent them in the Web, and via Web standards and protocols. These logical software entities allow: (1) describing the objects; (2) syntactically and semantically describing functionalities (i.e., the Web services) providing access to hardware capabilities of the physical objects; (3) establishing communications with the physical objects, by eventually adapting the Web protocols and standards to the proprietary protocols and data formats used by the objects; and (4) providing environment to deliver and to run WoT applications. Thanks to these logical software entities, physical objects can be combined together using Web mashup techniques to create complex cyber-physical systems that can be, for instance, designed for the smart agriculture. For instance, a WoT application built on the logical extensions of a crop irrigation system, of sensors deployed in fields to measure soil moisture, and of a weather forecast Web service available on the Internet, could easily be developed using mashup techniques so as to process data produced by sensors, and to trigger the watering system if the soil is too dry and if no rainfall is expected soon. As mentioned before, the data generated by the sensors can be collected by data mules (e.g., tractors) and transmitted to their logical representations deployed in a cloud infrastructure. The application can query these logical representations to obtain the environmental measures, invoke a forecast Web service available on the Internet, and decide on the basis of these pieces of information to trigger the watering or not. If so, this application will act on the watering system through its logical representation.

In the remainder of this section, we present the concept of Avatar [35] and the middleware platform we have proposed in project ASAWoO, and how they can be used in a smart agriculture scenario similar to that described in the previous lines. We also give a comparison of the concept of Avatars with the concept of Servient proposed by the W3C [36].

3.2.1. Overview of the Avatar Concept

The purpose of an Avatar is to provide device vendors, software developers and end-users with a comprehensible abstraction that makes physical objects accessible on the Web and that extends their status and capabilities (processing, acting, sensing, etc.) into homogeneous and user-understandable functionalities. An Avatar can be viewed as a "Web-based cyber-physical object" that defines and embodies high-level behaviors for physical devices.

An Avatar is composed of seven modules (see Figure 4). The interoperability module makes it possible to configure and to manage the physical object linked to the Avatar. It acts as an adaptation layer that hides objects' own protocols. The communication module provides Avatar-to-Avatar and Avatar-to-Web-browser communications. Different kinds of applicative protocols (e.g., HTTP and CoAP) and transport protocols (UDP, TCP, and the disruption-tolerant protocol provided by the opportunistic communication support presented previously in this section) are supported to communicate with the Avatars. Moreover, different wireless technologies are currently supported to achieve communications with physical objects, such as Bluetooth, Wi-Fi, Wi-Fi Direct and XBee. The functionality module manages the capabilities and the functionalities of an Avatar. Similar to functionalities, capabilities are pieces of code that make it possible to exploit the resources offered by the physical objects. However, unlike functionalities—which are device independent—capabilities depend on the physical objects. They can be viewed as implementations of elementary functionalities. Thanks to this approach, an agribot can for instance exposes a higher-level of abstraction for the navigation functionality that exploits lower-level functionalities, such as GPS positioning and steering capabilities. Both capabilities and functionalities are described semantically. Reasoning about semantic descriptions of the capabilities and of the functionalities help inferring complex functionalities involving sub-functionalities, thus allowing to define high-level representations that better match the users' needs than the low-level capabilities. Based on the inferences made by the semantic reasoner, the functionality manager can incrementally deploy new functionalities. These functionalities are filtered by the context manager according to the information it processes continuously, thus allowing to select the implementations of the functionalities that are the most suited to the running conditions of the physical object, as well as to perform dynamic adaptations. Various contextual properties can be taken into account, such as the location of the object, its processing and memory capacities and its power budget. Functionalities are exposed as REST services by the WoT module, and therefore can be invoked remotely by WoT applications (WoTApps) or by other Avatars for collaboration purposes. The WoT application container deploys and manages the life-cycle of WoTApps. WoTApps can fully run server-side (i.e., on the Avatar), be distributed and executed both on the Avatar and in a Web browser, or be exclusively run in a Web browser. Finally, Avatars implement a collaboration module relying on an agent-based approach. This module aims at endowing an autonomous behavior to the Avatar and physical object, so that they can discover the functionalities they can partially achieve and that require the help of other Avatars to be performed. After discovering the missing functionality in another avatar, they can enrol this Avatar to provide the functionality.

The main differences between the proposition of the W3C for the WoT, called Servient [36], and the Avatar-based models reside in the ability of Avatars to tolerate the connectivity disruptions, to dynamically adapt their behavior and the functionalities they provide to their running context, to automatically deploy new functionalities inferred on the basis of the semantic description of more elementary ones, and to collaborate with other Avatars. Indeed, similar to Avatars, Servients implement a legacy communication module that makes it possible to communicate with physical objects using proprietary protocols, a protocol binding module that allows using several types of applicative protocols (MQTT, HTTP, and CoAP) to communicate with the Servients, a model of resources and of description of objects, and a runtime environment to run WoTApps. These differences between avatars and Servients make Avatars more suited to develop cyber-physical systems that integrate autonomous mobile objects, and therefore more adapted to systems dedicated to the smart agriculture and farming. Furthermore, the reference implementation of the W3C standard is not as mature as that proposed by

the ASAWoO project. Unlike the solutions proposed in [23–26], both Servients and Avatars provide users and developers with a comprehensive virtual representation of physical objects to access and control them. This unified resource representation helps coping with device heterogeneity, which is not well supported in Agri-IoT according to its authors.

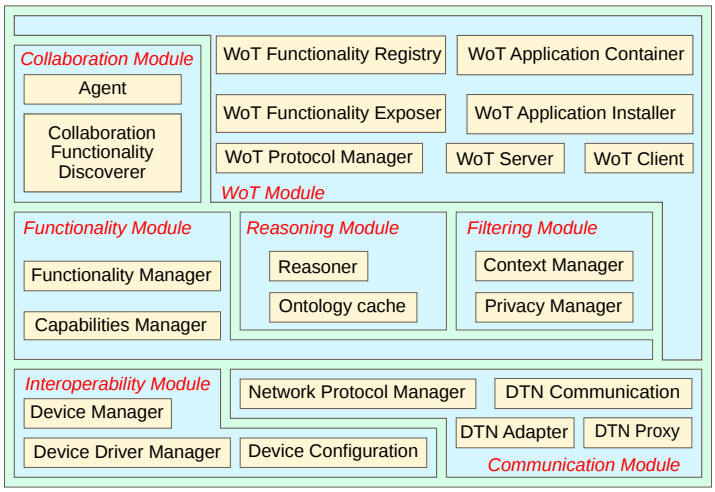

Figure 4. Architecture of an Avatar.

3.2.2. Avatar Runtime Environment

To configure themselves automatically, Avatars rely on a set of semantic and code repositories, as represented in Figure 5. These repositories contain device setups, semantic descriptions of capabilities and functionalities of Avatars and semantic descriptions of execution contexts. The code repositories provide independent syntactic descriptions of REST interfaces of functionalities, and specific implementations of capabilities, functionalities and WoTApps. Most capabilities are device dependent, which is not the case of the functionalities. The implementations of functionalities can still differ for non-functional purposes such as quality of service requirements. Avatars download pieces of code from the code repositories and deploy them locally, doing so they instantiate new capabilities and functionalities, and infer new high-level ones progressively. Avatars are executed in a runtime environment that manages their life-cycle and that exposes them in the Web. An Avatar is instantiated in the runtime environment either automatically when a new physical object is discovered in the local area network or when a user registers a new device manually. The Avatar runtime environment and the code repositories can be installed either in a cloud platform, on a gateway, or even on a physical object if this one has enough processing power and memory capacity. Needless to say, the repositories and the Avatar runtime environments can be deployed on separated devices.

3.2.3. Technical Implementation

The ASAWoO runtime environment is developed in Java and is based on the OSGi Apache Felix platform (http://felix.apache.org/). It is designed as a set of services that are deployed using bundles (i.e., a JAR augmented with metadata), and that manage the life-cycle of Avatars. An Avatar (i.e., the different modules and managers forming the Avatar's architecture) is also implemented as a set of OSGi services. Configuration files and the pieces of code of capabilities, functionalities and WoTApps are also deployed using bundles. Code repositories are implemented by OSGi Bundle Repositories (OBR), which can be either local on remote. An OBR offers a customizable dependency management that has been tuned to take into account functionalities and capabilities requirements. The semantic reasoner module depends on Apache Jena (https://jena.apache.org/) inference engine and queries Fuseki SPARQL endpoints. In the current implementation, the Vert.x (https://vertx.io/) stack is used to serve WoTApps and to expose functionalities as REST services over HTTP. Vert.x handler mechanism also helps in managing asynchronous calls to functionalities. The opportunistic

communication module implementation based on C3PO has already been thoroughly covered in Section 3.1.2 and in [31].

ASAWoO source code which is distributed under the CeCILL license (compatible with GNU GPLv2), is available in GIT repository (https://gitlab.com/asawoo) and users and developers documentation can be accessed on ASAWoO website (https://asawoo.gitlab.io/).

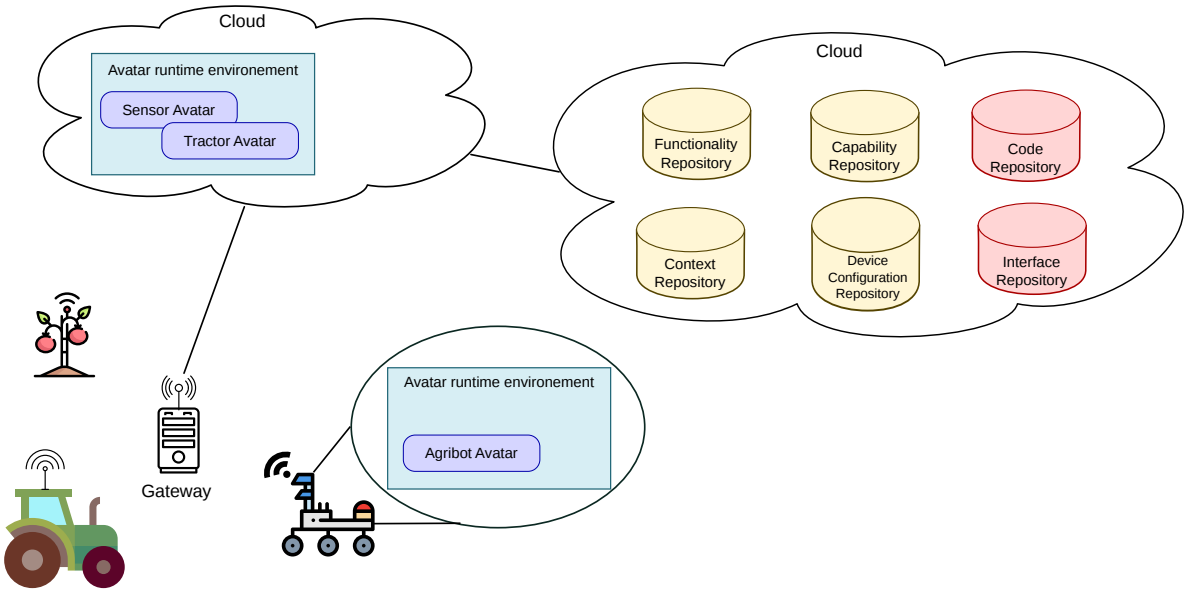

Figure 5. Avatar runtime environment.

4. Experiment

This section describes the experiment conducted during four months in Montoldre (France) experimental farm in cooperation with IRSTEA (http://www.irstea.fr/en/research/research-units/regions-jru). This experiment aimed firstly at evaluating the effective functioning of the ASAWoO midldleware platform and its performances in real conditions, and secondly at studying the relevance and the viability of a Web of Thing approach in an agricultural context.

4.1. Presentation of the Experiment Environment

Montoldre experimental farm is a large plot of agricultural land dedicated to the experimentation of agricultural techniques, especially fertilizer spreading techniques. The area shown in the satellite view in Figure 6 is composed of a dozen fields (in red) maintained by a handful of agricultural vehicles.

The application considered in this first experiment consisted in collecting environmental data (humidity and temperature) measured by sensors. This experiment was run in the early stages of ASAWoO development. Even though the whole middleware was not complete at that time, the opportunistic communication module was operational. Hardened sensors that can be exposed to bad weather conditions were also not ready to use for this experiment. Consequently, we used tiny single-board computers (RaspberryPis) equipped with Wi-Fi antennas that produce data exactly as the sensors would have done. As these devices are not hardened, they were placed in farm buildings, as shown in Figure 7. These experimental conditions may appear as degraded, but are not in reality because the communication behavior of sensors was reproduced as accurately as possible. This experiment allowed us to validate the RESTful opportunistic communication support presented in Section 3.

Three instances of ASAWoO runtime were deployed on Raspberry Pis equipped with 5 dB Wi-Fi antennas. The first one, identified as RASP00, acted as a base station located in IRSTEA office at the limit of Wi-Fi communication range from the tractors' warehouse. The two other RaspberryPis (RASP01 and RASP02) were equipped with an external battery pack and a GPS module. The kits were

embedded on two tractors. GPS antennas were placed on the tractors' roofs and the battery power cords were plugged into the tractors' cigarette lighters to power up/on the Raspberry and recharge. Two other self-powered RaspberryPis with smaller range acting as sensors were deployed in farm buildings located on tractors paths. Locations of the five nodes are shown on the map in Figure 7. The embedded system deployed on sensors was restricted to ASAWoO communication module since the whole middleware stack was not required. During this experiment, the sensors produced data every 30 min.

Figure 6. Montoldre experiment farm.

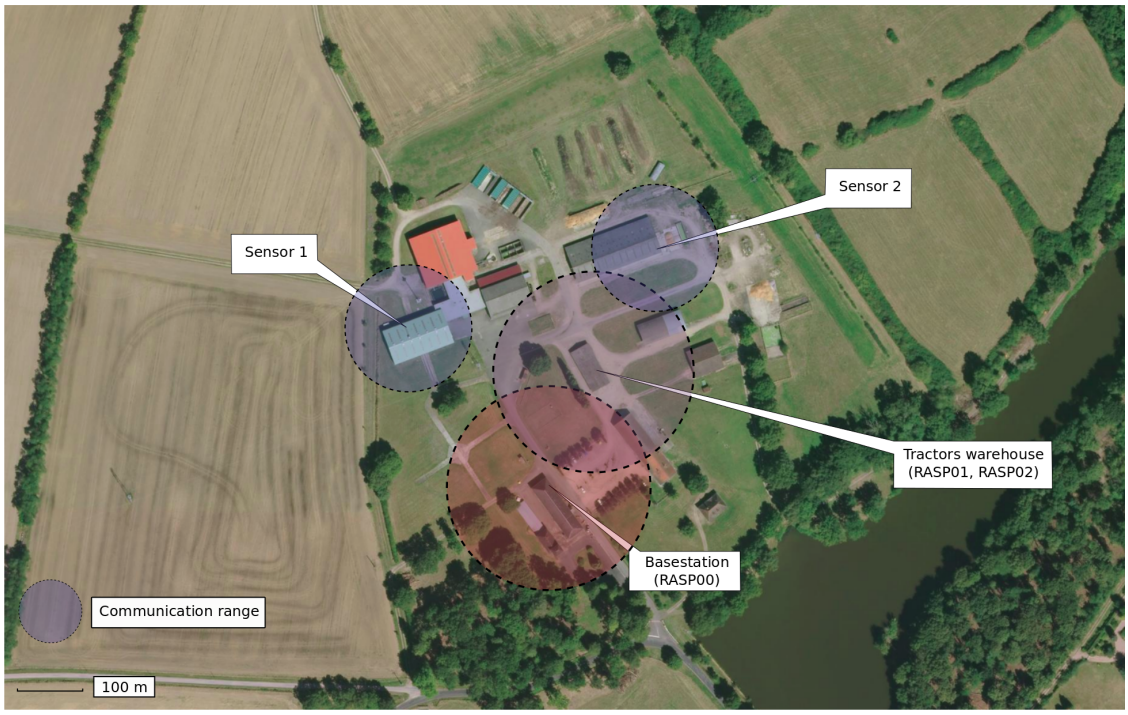

Figure 7. Node locations and their approximate communication ranges.

During the four months of experimentation, over 30 trips were monitored. Mobility traces of RASP01 and RASP02 (i.e., tractors) were stored in log files, as well as the contacts and exchanges

between tractors and the base station). Logs were compiled and sent to the base station when entering its radio range. A ssh tunnel between the base station at IRSTEA and our lab was established to access the logs, update the code if necessary, and remotely monitor the experiment throughout its progress. The following subsection presents the experiment outcomes, focusing on a few trips involving both tractors or longer activity periods.

4.2. Experiment Results

After four months of sparse activity, logs gathered on the base station were compiled, sanitized and analyzed. Since the two equipped tractors were rarely operating simultaneously, we focused on the logs of days providing the most extensive data. A summary of the activity during these days is presented in Table 1. Day 1 was the experiment setup; we checked that communication was well established between the three nodes, and moved RASP01 and RASP02 to make them enter and exit communication range with each other and with sensors. On Day 3, the tractor carrying RASP01 was spreading fertilizer, and on Day 4 it sowed the fields. Day 5 featured the longer trip from RASP02.

Table 1. Data of active days.

	Trip Duration	**Number of Messages Exchanged**			**Number of GPS Positions**	
		RASP00/RASP01	**RASP00/RASP02**	**RASP01/RASP02**	**RASP**01	**RASP02**
Day 1	5h09	46	110	48	5	147
Day 2	4h50	28	47	3	0	28
Day 3	6h21	56	0	0	266	0
Day 4	7h31	334	0	0	1005	0
Day 5	7h44	219	0	51	0	2063

Table 2 compiles global observations made throughout the experiment. When tractors were active, we reported an average speed of 16 km/h, which was not an hindrance to message delivery. Opportunistic sensing and the data mule approach is therefore a viable solution in the agricultural context where vehicles speed is moderate. Although cellular network coverage was quite poor on Montoldre site, opportunistic networking via ad hoc Wi-Fi proved to be effective to convey small messages such as environmental data. Under the right circumstances, the communication range exceeded our expectations with a recorded maximum range of 468 m. This result is further supported by Figure 8, which shows the number of actual contacts at a given time between two nodes versus an estimate derived from the actual node positions and a fixed range of 200 m. More often than not, actual contacts were established, whereas the estimate did not consider the nodes to be within radio range.

Table 2. Tractors average speed and radio-contact range.

Average Speed	**Communication Range**		
	Maximum Range	**Average Range**	**Median Range**
15.9 km/h	468 m	237 m	185 m

Contact durations (i.e., the time periods during which the nodes are in contact and can communicate) are presented in Figure 9. On Day 5, half of the contacts lasted less than 3 min while only a tenth of the contacts exceeded 10 min. The few contacts exceeding 1 h were due to RASP02 instance remaining active in the warehouse, within range of the base station. The relatively moderate speed of the tractors combined with the long range Wi-Fi interfaces resulted in long enough contacts allowing for wide communication time periods. Since the size of exchanged messages, whether they were gossiping relating to the opportunistic communication protocol or sensor data, was quite small (a few bytes), such a time frame is sufficient to perform opportunistic sensing (i.e., collect sensor data and opportunistically forward these data).

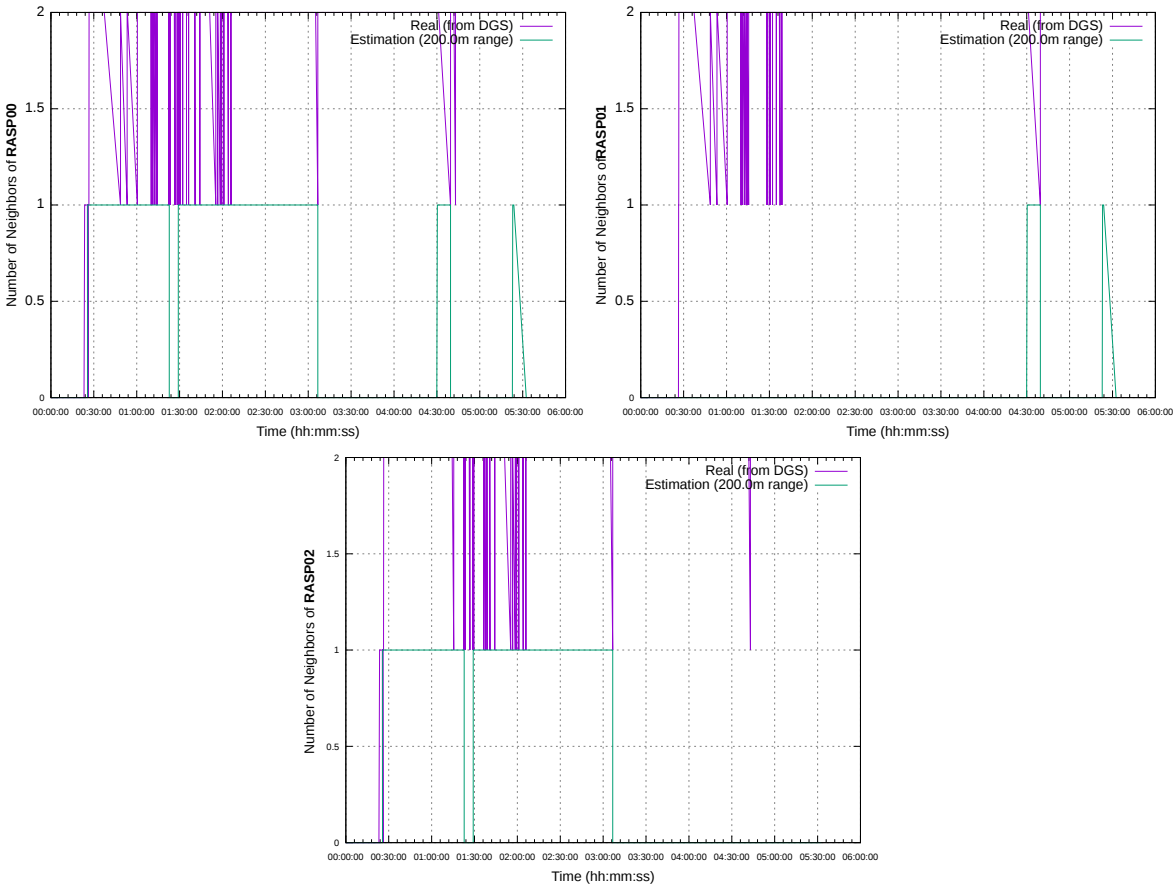

Figure 8. Real contacts vs. estimated contacts of RASP00, RASP01 and RASP02 on Day 1.

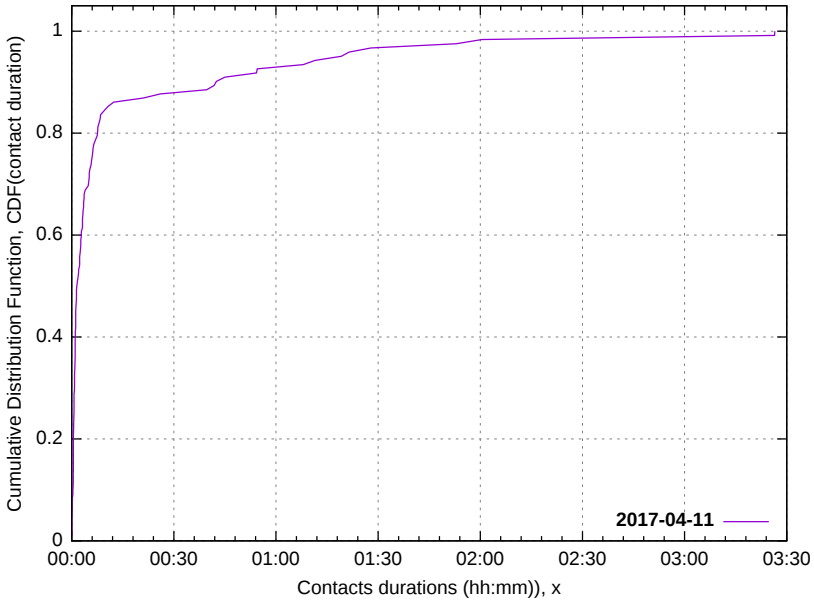

Figure 9. Cumulative distribution of contacts durations on Day 5.

Nevertheless, it is worth noting that the experiment did not go entirely smoothly. Sometimes the RaspberryPi was slow to receive the GPS signal, which not only led to missing positions, but also to clock desynchronization and erroneous timestamps since the system clock was set to GPS time. However, logs could be resynchronized using contacts and exchanged messages with the base station. Due to farming vehicle vibrations, sometimes the device got its GPS antenna or its power cord

unplugged, and the shutdown and powering up that should have been controlled by the battery was not always done properly, which led to corrupted log files that had to be discarded or corrected. In a concrete deployment, kits should be designed and built to better fit the specificities of agricultural vehicles (e.g., with antivibration material, properly integrated power management and a CMOS clock battery).

Lastly, the final encountered issue was the considerable size of the covered area combined with the fact that the two equipped vehicles were hardly running together, which resulted in too few contacts between the mobile nodes outside the vicinity of their rallying point (the warehouse). The lack of contact data does not affect the effectiveness of opportunistic sensing nonetheless. Even though contacts are more likely to occur in the rallying point area, tractors are still able to retrieve data from faraway sensors and forward it to the base station. The two mobile ASAWoO instances were indeed able to successfully gather data from sensors up to the base station without a permanent connectivity between each node.

4.3. Evaluation of the Web of Things Approach

The experiment described in the previous section focused on the validation of ASAWoO opportunistic communication capabilities, whereas the Web of Things aspect could not be tested at that time. WoT modules were nonetheless still validated in the later stages of ASAWoO development in another experiment, which is briefly presented in this subsection. During the Montoldre experiment, ASAWoO middleware mainly consisted of the Avatar core, the functionalities management and registry features (i.e., functionalities were advertised as RESTful services) as well as the opportunistic communication module. The second experiment was run in Bordeaux, France. It introduced the device manager and the semantic reasoning features to instantiate capabilities, simple and composite functionalities, as well as the WoT Application module. Unlike the farm experiment, which spanned over four months, this one lasted only one day. Since the communication module had already been validated in the long-run experiment, the goal was to validate ASAWoO final stage, particularly the WoT related modules (e.g., WoTApp manager, automatic instantiation of functionalities from device configurations and semantic repositories, etc.).

This experiment was based on the same data collection WoTApp and the same hardware kits. Although the Raspberry Pi-based kits (with GPS module and Wi-Fi interface) were reused, this time the data mule was a robot, the Jackal all-terrain mobile platform (https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/), on which was deployed an instance of ASAWoO hosting its own Avatar. Figure 10 shows a picture of the robot and the packaged base station kit on the experimentation field. In addition to its navigation capabilities, a camera was mounted on the platform. A ROS (Robot Operating System: http://www.ros.org/) bridge interoperability module was developed for ASAWoO to expose these capabilities, which in turn enabled higher level functionalities such as person detection and video surveillance. The video surveillance and the navigation functionalities were then combined to provide a patrol functionality.

The scenario written for this experiment consisted in making the robot patrol from sensor to sensor to collect data as long as it did not encounter a person. If the video surveillance functionality triggered an alarm, the reasoning module would give priority to the patrol functionality and use the navigation functionality to go back to a rallying point. This scenario is illustrated on ASAWoO code repository homepage (https://asawoo.gitlab.io/).

During the experiment, all devices hosting an ASAWoO platform were accessible through their second Wi-Fi interface set on access point mode. When connecting to a device's main WoTApp, we could navigate through its neighbors and query their functionalities. Figure 11 presents a screenshot of the main WoTApp hosted on the robot. It features system functionalities (e.g., functionalities registry, neighborhood, device configuration and WoTApp deployment), as well as other Avatars whose remote functionalities can be reached through multi-hop asynchronous DTN RESTful requests. The screenshot

shows a JSON-formatted GPS position in response to a GET request sent to a sensor's LocationService functionality. The retrieved sensor data were also displayed on the data collection WoTApp.

Figure 10. All-terrain mobile platform running ASAWoO next to the base station kit.

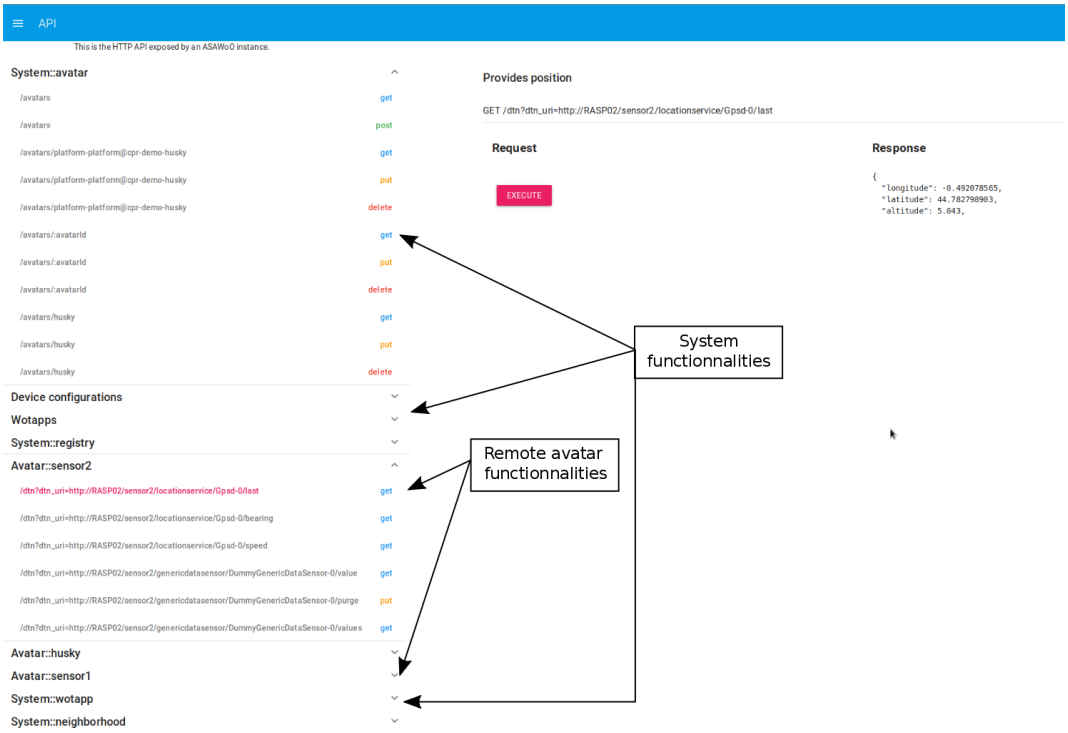

Figure 11. Screenshot of ASAWoO administration WoTApp.

It is worth noting that, although this paper mainly focuses on sensing functionalities, ASAWoO also supports actuators and the conclusions that we drew also apply to this class of devices. The general practice that has been followed in ASAWoO was to bind actuator functionalities to PUT and POST HTTP methods, while GET methods were used for sensor functionalities. In the patrol surveillance scenario, the navigation functionality is an example of an actuator functionality. Results regarding context adaptation and semantic reasoning performances are presented in [37].

4.4. Results Discussion

During the second experiment, the WoT approach allowed us to successfully interact with different devices through their logical representations. Devices heterogeneity was indeed hidden behind Avatars using web standards (JSON over HTTP requests). The web browsers from a laptop, a smartphone or a tablet were used interchangeably.

Both experiments were conclusive regarding the opportunistic computing aspect. The monitored contact durations and distances between mobile and/or static nodes, resulting from the data mules speed and their communication range, both support the idea that opportunistic sensing is a possible option for smart agriculture. The communication range measured in the second experiment confirmed the first results with a maximum range of 283 m between the base station and the robot. We however observed that farming vehicles were not operating on a daily basis whereas data gathering depends on their operation frequency for an opportunistic sensing approach entirely relies on nodes mobility.

Even though the following perspective is beyond the scope of this article, we argue that the node density issue can be addressed by emulation techniques. At the end of the first experiment, we obtained valuable data on the mobility of farming vehicles in actual conditions. These mobility data could be made available on a service such as CRAWDAD (https://crawdad.org/) and leveraged with simulation tools such as LEPTON [38] which is an emulation platform for opportunistic networking development. LEPTON allows the execution of functional code to run in an emulated environment under controlled and repeatable conditions, where only the mobility of nodes is simulated. The possibility to replay scenarios based on actual tractors paths gives the opportunity to evaluate our approach under different conditions and therefore to finely tune scenario parameters such as the number of data mules, their paths and speed, or the positions of sensors, without starting over an actual experimental campaign. Thus, testing different smart agriculture scenarios could indeed help in deducing an optimal setup to improve opportunistic sensing solutions for precision agriculture.

5. Conclusions

The Internet of Things will become a reality when connected object makers propose off-the-shelf cyber-physical objects that can communicate and interoperate through open standard protocols and data formats, or that can be connected to Web of Things (WoT) platforms that bridge the gap between proprietary protocols and Web standard formats and protocols. Such objects and platforms appear as attractive solutions to build complex systems, notably for environmental sensing and for smart agriculture and farming. Nevertheless, in challenging environments, such as in agricultural and farming environments, it is often difficult to deploy systems relying exclusively on infrastructure-based networks because rural areas often have poor network coverage, and because there are frequent and unpredictable connectivity disruptions resulting from both the mobility of agricultural machines (e.g., tractors, agribots, and drones) communicating with short range radio interfaces and from the sleep phases kept by some devices for energy saving purposes.

In this paper, we present how opportunistic computing can be combined with the WoT to contribute to the smart agriculture, and how such an approach was implemented in a middleware platform called ASAWoO. Opportunistic computing and networking techniques leverage the mobility of agricultural machines to provide intermittent connectivity between physical objects that would not have been connected by any route otherwise. This platform provides a virtual representation of the physical objects in the Web, called Avatars, which give access to the objects functionalities in order to allow WoT users to interact with devices at a higher level of abstraction.

ASAWoO platform was evaluated in a practical smart agriculture scenario through a four-month experiment. Two tractors carried ASAWoO instances responsible for collecting data of sensors disseminated in an experimental farm. Experiment results show that ASAWoO platform was able to perform environmental sensing using opportunistic computing in real conditions, and that the low pace of farming vehicles makes opportunistic computing possible in the agricultural context. Based on these outcomes, we conclude that both the integration of sensors and actuators such as agribots

in WoT applications, and the implementation of opportunistic computing mechanisms are likely to improve agricultural practices and compensate for the potential lack of network infrastructure. Such an approach is not limited to smart agriculture and farming, and could also be transposed to other related critical contexts such as disaster relief.

Author Contributions: Data curation, N.L.S.; Investigation, L.T.; Resources, N.L.S.; Software, L.T. and N.L.S.; Supervision, N.L.S.; Validation, L.T.; Visualization, L.T.; Writing—original draft, L.T. and N.L.S.; and Writing—review and editing, L.T. and N.L.S.

Funding: This work was supported by the French ANR (Agence Nationale de la Recherche) grant number ANR-13-INFR-012.

Acknowledgments: The experiment presented in this article would not have been possible without the helpful support provided by IRSTEA from Montoldre and Clermont-Ferrand. We would also like to thank other partners for their valuable contributions to the ASAWoO project: TWEAK team from Lyon 1 university LIRIS lab (http://liris.cnrs.fr/), COSY team from Valence-Grenoble LCIS lab (http://lcis.grenoble-inp.fr/), and Generation Robots (https://generationrobots.com/).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, Z.; Wang, J.; Higgs, R.; Zhou, L.; Yuan, W. Design of an Intelligent Management System for Agricultural Greenhouses Based on the Internet of Things. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017; pp. 154–160.
2. Vasisht, D.; Kapetanovic, Z.; Won, J.; Jin, X.; Chandra, R.; Sinha, S.; Kapoor, A.; Sudarshan, M.; Stratman, S. FarmBeats: An IoT Platform for Data-Driven Agriculture. In Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), Boston, MA, USA, 27–29 March 2017; USENIX Association: Boston, MA, USA, 2017; pp. 515–529.
3. Sreekantha, D.K.; Kavya, A.M. Agricultural crop monitoring using IOT—A study. In Proceedings of the 2017 11th International Conference on Intelligent Systems and Control (ISCO 2017), Coimbatore, India, 5–6 January 2017; pp. 134–139.
4. Zhao, D.; Ma, H.; Liu, L.; Zhao, J. On Opportunistic Coverage for Urban Sensing. In Proceedings of the 10th International Conference on Mobile Adhoc and Sensor Systems (MASS 2013), Hangzhou, China, 14–16 October 2013; pp. 218–226.
5. Guinard, D.; Trifa, V.; Mattern, F.; Wilde, E. From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices. In *Architecting the Internet of Things*; Uckelmann, D., Harrison, M., Michahelles, F., Eds.; Springer: Berlin, Germany, 2011; pp. 97–129. . [CrossRef]
6. Guinard, D.; Trifa, V.; Wilde, E. A Resource Oriented Architecture for the Web of Things. In Proceedings of the IEEE International Conference on the Internet of Things (IOT 2010), Tokyo, Japan, 29 November–1 December 2010; pp. 1–8. [CrossRef]
7. Wilde, E. *Putting Things to REST*; Technical Report; School of Information and UC Berkeley: Berkeley, CA, USA, 2007.
8. Mrissa, M.; Médini, L.; Jamont, J.P.; Le Sommer, N.; Laplace, J. An Avatar Architecture for the Web of Things. *IEEE Internet Comput.* **2015**, *19*, 30–38. [CrossRef]
9. Rehman, A.U.; Abbasi, A.; Islam, N.; Shaikh, Z. A Review of Wireless Sensors and Networks' Applications in Agriculture. *Comput. Stand. Interfaces* **2014**. *36*, 263–270. [CrossRef]
10. Ojha, T.; Misra, S.; Raghuwanshi, N.S. Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges. *Comput. Electron. Agric.* **2015**, *118*, 66–84. [CrossRef]
11. Kim, Y.; Evans, R.G.; Iversen, W.M. Remote Sensing and Control of an Irrigation System Using a Distributed Wireless Sensor Network. *IEEE Trans. Instrum. Meas.* **2008**, *57*, 1379–1387.
12. Hull, B.; Bychkovsky, V.; Zhang, Y.; Chen, K.; Goraczko, M.; Miu, A.; Shih, E.; Balakrishnan, H.; Madden, S. CarTel: A Distributed Mobile Sensor Computing System. In Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06), Boulder, CO, USA, 11–14 November 2006; pp. 125–138.

13. Campbell, A.T.; Eisenman, S.B.; Lane, N.D.; Miluzzo, E.; Peterson, R.A. People-centric Urban Sensing. In Proceedings of the Second Annual International Wireless Internet Conference (WICON '06), Boston, MA, USA, 2–5 August 2006; pp. 18–31.
14. Abdelzaher, T.; Anokwa, Y.; Boda, P.; Estrin, D.; Burke, J.; Leonidas, G.; Madden, S.; Reich, J. Mobiscopes for Human Spaces. *IEEE Pervasive Comput.* **2007**, *6*, 20–29. [CrossRef]
15. Kansal, A.; Nath, S.; Liu, J.; Zhao, F. SenseWeb: An Infrastructure for Shared Sensing. *IEEE MultiMedia* **2007**, *14*, 8–13. [CrossRef]
16. Riva, O.; Borcea, C. The Urbanet Revolution: Sensor Power to the People! *IEEE Pervasive Comput.* **2007**, *6*, 41–49. [CrossRef]
17. Urban Atmospheres. Available online: http://www.urban-atmospheres.net/ (accessed on 29 January 2019).
18. Eisenman, S.B.; Miluzzo, E.; Lane, N.D.; Peterson, R.A.; Ahn, G.S.; Campbell, A.T. The BikeNet Mobile Sensing System for Cyclist Experience Mapping. In Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys '07), Sydney, Australia, 6–9 November 2007; pp. 87–101.
19. Miluzzo, E.; Lane, N.D.; Eisenman, S.B.; Campbell, A.T. CenceMe—Injecting Sensing Presence into Social Networking Applications. In Proceedings of the Smart Sensing and Context, Second European Conference (EuroSSC 2007), Kendal, UK, 23–25 October 2007; pp. 1–28.
20. MIT Senseable City Lab. Available online: http://senseable.mit.edu/ (accessed on 31 January 2019).
21. Chen, K.W. CafNet: A Carry-and-Forward Delay-Tolerant Network. Master's Thesis, Massachusetts Institute of Technology, Cambridge, UK, 2007.
22. Mota, V.F.S.; Cunha, F.D.; Macedo, D.F.; Nogueira, J.M.S.; Loureiro, A.A.F. Protocols, Mobility Models and Tools in Opportunistic Networks: A Survey. *Comput. Commun.* **2014**, *48*, 5–19. [CrossRef]
23. Kamilaris, A.; Gao, F.; Prenafeta-Boldu, F.X.; Ali, M.I. Agri-IoT: A semantic framework for Internet of Things-enabled smart farming applications. In Proceedings of the 3rd IEEE World Forum on Internet of Things (WF-IoT 2016), Reston, VA, USA, 12–14 December 2016; pp. 442–447. [CrossRef]
24. Taylor, K.; Griffith, C.; Lefort, L.; Gaire, R.; Compton, M.; Wark, T.; Lamb, D.; Falzon, G.; Trotter, M. Farming the Web of Things. *IEEE Intell. Syst.* **2013**, *28*, 12–19. [CrossRef]
25. Kamilaris, A.; Trifa, V.; Pitsillides, A. HomeWeb: An application framework for Web-based smart homes. In Proceedings of the 18th International Conference on Telecommunications (ICT 2011), Ayia Napa, Cyprus, 8–11 May 2011; pp. 134–139. [CrossRef]
26. Kamilaris, A.; Pitsillides, A.; Trifa, V. The Smart Home meets the Web of Things. *Int. J. Ad Hoc Ubiquitous Comput.* **2011**, *7*, 145–154. [CrossRef]
27. Fall, K. A Delay-Tolerant Network Architecture for Challenged Internets. In Proceedings of the ACM SIGCOMM03, Karlsruhe, Germany, 25–29 August 2003; pp. 27–34.
28. Pelusi, L.; Passarella, A.; Conti, M. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks. *IEEE Commun. Mag.* **2006**, *44*, 134–141. [CrossRef]
29. Conti, M.; Giordano, S.; May, M.; Passarella, A. From Opportunistic Networks to Opportunistic Computing. *IEEE Commun. Mag.* **2010**, *48*, 126–139. [CrossRef]
30. Juang, P.; Oki, H.; Wang, Y.; Martonosi, M.; Peh, L.S.; Rubenstein, D. Energy-efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (APLOS X), San Jose, CA, USA, 5–9 October 2002; ACM: San Jose, CA, USA, 2002; pp. 96–107.
31. Le Sommer, N.; Touseau, L.; Mahéo, Y.; Auzias, M.; Raimbault, F. A Disruption-Tolerant RESTful Support for the Web of Things. In Proceedings of the 4th International Conference on Future Internet of Things and Cloud (FiCloud 2016), Vienna, Austria, 22–24 August 2016 ; pp. 17–24.
32. Scott, K.; Burleigh, S. Bundle Protocol Specification. IETF RFC 5050; 2007. Available online: http://www.hjp.at/doc/rfc/rfc5050.html (accessed on 31 January 2019).
33. Cerf, V.G.; Burleigh, S.C.; Durst, R.C.; Fall, K.; Hooke, A.J.; Scott, K.L.; Torgerson, L.; Weiss, H.S. Delay-Tolerant Networking Architecture. IETF RFC 4838; 2007. Available online: http://www.hjp.at/doc/rfc/rfc4838.html (accessed on 31 January 2019).
34. Le Sommer, N.; Launay, P.; Mahéo, Y. A Framework for Opportunistic Networking in Spontaneous and Ephemeral Social Networks. In Proceedings of the 10th Workshop on Challenged Networks (CHANTS'2015), Paris, France, 11 September 2015.

35. Médini, L.; Mrissa, M.; Terdjimi, M.; Khalfi, E.M.; Le Sommer, N.; Capdepuy, P.; Jamont, J.P.; Occello, M.; Touseau, L. Building a Web of Things with Avatars. In *Managing the Web of Things: Linking the Real World to the Web*; Shengn, M., Qin, Y., Yao, L., Benatallah, B., Eds.; Elsevier: New York, NY, USA, 2016.
36. W3C. Web of Things (WoT) Architecture. 2017. Available online: https://www.w3.org/TR/wot-architecture/ (accessed on 31 January 2019).
37. Terdjimi, M.; Médini, L.; Mrissa, M.; Le Sommer, N. An Avatar-Based Adaptation Workflow for the Web of Things. In Proceedings of the 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Paris, France, 13–15 June 2016. [CrossRef]
38. Sánchez-Carmona, A.; Guidec, F.; Launay, P.; Mahéo, Y.; Robles, S. Filling in the missing link between simulation and application in opportunistic networking. *J. Syst. Softw.* **2018**, *142*, 57–72, [CrossRef]