CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

A Service-Oriented Cloud-Based Management System for the Internet-of-Drones

Anis Koubâa*
Basit Qureshi
Mohamed-Foued Sriti
Yasir Javed
Eduardo Tovar*

*CISTER Research Centre

CISTER-TR-170305

2017/04/26

A Service-Oriented Cloud-Based Management System for the Internet-of-Drones

Anis Koubâa*, Basit Qureshi, Mohamed-Foued Sriti, Yasir Javed, Eduardo Tovar*

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: aska@isep.ipp.pt, emt@isep.ipp.pt

http://www.cister.isep.ipp.pt

Abstract

Deploying drones over the Cloud is an emerging research area motivated by the emergence of Cloud Robotics and the Internet-of-Drones (IoD) paradigms. This paper contributes to IoD and to the deployment of drones over the cloud. It presents, Dronemap Planner, an innovative service-oriented cloud based drone management system that provides access to drones through web services (SOAP and REST), schedule missions and promotes collaboration between drones. A modular cloud proxy server was developed; it acts as a moderator between drones and users. Communication between drones, users and the Dronemap Planner cloud is provided through the MAVLink protocol, which is supported by commodity drones. To demonstrate the effective- ness of Dronemap Planner, we implemented and validated it using simulated and real MAVLink-enabled drones, and deployed it on a public cloud server. Experimental results show that Dronemap Planner is efficient in virtualizing the access to drones over the Internet, and provides developers with appropriate APIs to easily program drones 19 applications.

A Service-Oriented Cloud-Based Management System for the Internet-of-Drones

Anis Koubâa **‡, Basit Qureshi **, Mohamed-Foued Sriti ‖, Yasir Javed **¶, Eduardo Tovar ‡

**Prince Sultan University, Saudi Arabia.
‖ Al-Imam Mohammad Ibn Saud Islamic University, Saudi Arabia.
†† King Saud University, Riyadh, Saudi Arabia.
‡ CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal.

akoubaa@coins-lab.org, qureshi@psu.edu.sa, mfsriti@ccis.imamu.edu.sa,
yasir.javed@coins-lab.org, emt@isep.ipp.pt

***Abstract*—Deploying drones over the Cloud is an emerging research area motivated by the emergence of Cloud Robotics and the Internet-of-Drones (IoD) paradigms. This paper contributes to IoD and to the deployment of drones over the cloud. It presents, Dronemap Planner, an innovative service-oriented cloud based drone management system that provides access to drones through web services (SOAP and REST), schedule missions and promotes collaboration between drones. A modular cloud proxy server was developed; it acts as a moderator between drones and users. Communication between drones, users and the Dronemap Planner cloud is provided through the MAVLink protocol, which is supported by commodity drones. To demonstrate the effectiveness of Dronemap Planner, we implemented and validated it using simulated and real MAVLink-enabled drones, and deployed it on a public cloud server. Experimental results show that Dronemap Planner is efficient in virtualizing the access to drones over the Internet, and provides developers with appropriate APIs to easily program drones' applications.**

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are becoming increasingly popular. Although these have been around for already a few years, while more recently application developers and researchers are realizing the potential of these flying robots in applications such as remote sensing, smart cities, surveillance, disaster management and recovery, patrolling, aerial survey, border security etc. to name a few. Although these low cost UAVs can be utilized in many beneficial ways, the strictly limited processing capabilities as well as the low on-board storage raise several challenges. In fact, low-cost and battery-powered UAVs are unable to cope efficiently with the requirements of computation demanding applications (e.g., on-board image processing) encompassing real-time data and reliability constraints. Furthermore, with the limitation of the communication range of wireless-based drones, even when using telemetry systems (with range up to 5 Km), it is not possible to manage drones' mission in large environments such as at the scale of a city, or a country. In this paper, we propose a solution to this problem by integrating drones with the cloud and the Internet-of-Things (IoT).

In the context of the Cloud Robotics [1], [2], coined by James Kuffner [3], several works since 2010 attempted to integrate robots with the cloud through the Internet [4], [5], [6], [7]. In [4], the author proposed one of the first papers that specified the concept of Robot as a Service (RaaS). Yinong et al. proposed a cloud framework for interacting with robots in the area of service-oriented computing. In [5], the authors proposed the `DAvinCi` system for offloading computation from robots to a cloud-based distributed computing system based on the Apache Hadoop, but they did not address reliability and real-time control. In [6], the European project consortium proposed to build a World Wide Web for Robots for sharing knowledge about actions, objects, and environments between robots, and prototypes were implemented for service robots. In [7], the author proposed a SOAP-based service-oriented architecture that virtualizes robotic hardware and software resources, and exposes them as services through the Web.

In this paper, we specifically address the use of drones over the Internet and we define the concept of Internet of Drones (IoD). The objective is to develop new IoT applications by leveraging cloud computing, web technologies, and service-oriented architecture (SOA). We outline two main benefits from integrating drones with IoT and cloud: (1) *Virtualization*: the cloud infrastructure helps virtualizing UAV resources through abstract interfaces. (2) *Computation offloading*: the cloud plays the role of a remote brain for the UAVs by providing storage and computation services. This approach overcomes the computing and storage resources' limitations of the UAVs, as intensive computation is not performed on-board, but rather offloaded to the cloud. As such, the UAV will typically act as a mobile sensor and actuator decoupled from heavy computations. In this paper, we particularly address the virtualization aspect of this integration.

The main contributions of this paper are as follows. Firstly, we specify our vision to the concept of Internet-of-Drones (IoD), its requirements, challenges, and importance (Section 2). Secondly, we propose an architecture for the IoD with cloud integration and present a software architecture for a cloud-based management of drones connected through the Internet (Section 3). Thirdly, we demonstrate the feasibility and effectiveness of Dronemap Planner in handling drones's mission using several use cases involving single and multiple drones missions' control over the Internet (Section 4).

II. RELATED WORKS

There have been a few attempts to integrate drones with the cloud and IoT. In [8], Gharibi et. al presented a conceptual model for the Internet-of-Drones. The proposed architecture covers three major networks: air traffic control network, cellular network and Internet. The layered architecture provides generic services for different UAV applications, namely delivery, surveillance, search and rescue. The paper did not present any implementation or realization of this architecture and only outlined general concepts of the IoD. In our paper, we present both an architecture for IoD and validate it through a real implementation and experimentation.

Apvrille et. al [9] presented a model of a drone usage in natural disaster recovery. The objective is to help rescuers in finding victims after a disaster. The proposed system mainly addressed the development of onboard autonomous drone missions based on image processing rather than controlling the drone over the Internet or offloading computation to the cloud.

In [10], the authors proposed a SOA model for collaborative UAVs. A mapping between cloud computing resources and UAVs' resources was presented. Furthermore, essential services and customized services were proposed. The paper only provides a high-level description of the system architecture, components and services without any specific details on their implementation.

In [11], the same authors extended their previous work [10] and designed a RESTful web services model by following a Resource-Oriented Architecture (ROA) approach to represent the resources and services of UAVs. A small prototype was implemented on an Adruino board that emulates a UAV and its resources. However, the experimental prototype is very limited as it does not demonstrate a sufficient proof of concept on real drones, but on a simple Arduino board. Thus, the feasibility of the approach was not effectively demonstrated.

In [12], an experimental testbed for an emulated Arduino UAV system was used, and several sensors were used (including temperature and humidity, ultrasonic for distance measurements). RESTful web services were defined and implemented for manipulating each type of sensor through a Web interface. The authors evaluated the performance of their system. However, the experimental system remains limited in terms of validating the scalability issues, and also the experimental setup only applies to a small local network. In our paper, we consider real drones communicating with the MAVLink protocol [13] to validate our architecture.

III. INTERNET-OF-DRONES

The Internet of Drones (IoD) can be defined an architecture for providing control and access between drones and users over the Internet. In fact, Drones are increasingly becoming commodity items widely available off the shelf, thus allowing its use by any user to fly various missions using multiple drones in a controlled airspace. Whereas technology is helping the miniaturization of a UAV's onboard components including processors, sensors, storage as well as improving the battery life, the limitations of these components hinder the performance and lower the expectations. IoD provides a vehicle for coupling of Internet of Things as well as cloud robotics technologies to allow remote access and control of drones as well as the seamlessly scalable computation off-loading and remote storage capabilities of the Cloud.

There are various challenges associated with the implementation of IoD. Reliable point-to-point communications, mission control, seamless wireless connectivity, effective utilization of onboard resources are just a few of the concerns. Furthermore, Quality of Service (QoS) stands-up as a crucial issue that must be considered in the design of the IoD. Security is also an important challenge as access to drones' resources must be authenticated and secured. In addition, the IoD system must be immune to attacks like drone impersonation, flooding, sniffing, etc. Another important aspect is to hide the underlying technical information from the user which is possible by using a service-oriented approach, typically implementing SOAP or REST Web services [14]. Users do not need to be technically savvy in order to program or develop missions, rather the web services based system would provide easy access to on board resources through various APIs. In this paper, Dronemap Planner provides a solution to seamlessly accessing drones resources over the Internet and control their mission.

The main objectives of an IoD management system, like the Dronemap Planner that we propose in this paper, include:

- to provide seamless access to and real-time control and monitoring of drones for end-users
- to offload extensive computations from drones to the cloud
- to schedule dynamically the missions of multiple drones
- to provide collaborative framework for multiple drones
- and to provide cloud-based programming APIs for developers to develop drones' applications through the cloud.

To motivate the need for IoD and in particular for the Dronemap Planner cloud-based system, let us consider the following illustrative scenario: a team of multiple autonomous UAVs deployed in an outdoor environment in their depot waiting for the execution of certain missions. A user behind the cloud defines a mission (e.g. visiting a set of waypoints) and requests its execution. The user may either select one or more virtual UAVs from the list of available UAVs registered in the cloud, or may send his request to the cloud to auto-select one or more UAVs to execute the mission. Each virtual UAV is mapped to a physical UAV by the cloud using a service-oriented approach, typically implementing SOAP or REST Web services [14]. Once the mission request is received, the selected UAVs execute the mission and report in real-time the data of interest to the cloud layer, which in turn will store, process and forward synthesized results to the user.

Another use case is when the user selects locations of interest to visit on the map and sends them to the drone. By default, the drone will execute the mission by visiting the locations according to their sequence number. However, this might be not the optimal way to visit the waypoint. The

problem becomes even more complex when there is a need to optimally assign multiple locations to multiple drones.

So, a cloud-based system will definitely help in offloading such extensive computation from the drone to optimize the mission execution, and thus extending the energy lifetime of the drone.

IV. Dronemap Planner Architecture

A. General System Architecture

Figure 1 presents the architecture of Dronemap addressing the above requirements.

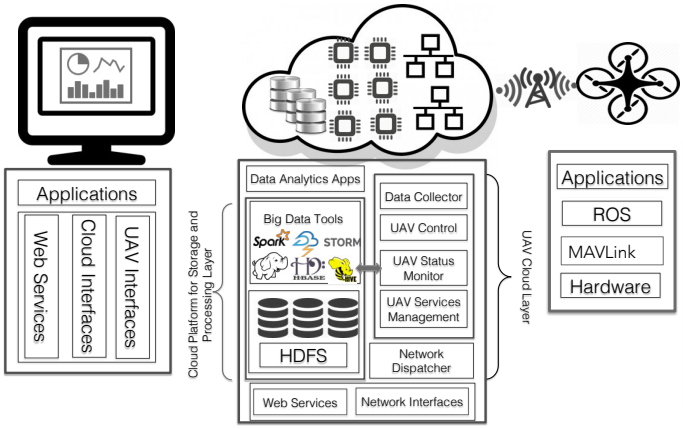

Fig. 1. DroneMap System Architecture: Abstraction Layers

The following layers are considered.

The UAV Layer: The UAV represents a set of resources exposed as services to end-users. The UAV has several layers of abstractions. On top of the hardware, ROS and MAVLink both provide two different ways for hardware resources abstraction. Robot Operating System (ROS) is one of the widely used middleware to develop robotics applications. On the other hand, MAVLink is a communication protocol built over different transport protocols (i.e. UDP, TCP, Telemetry, USB) that allow to exchange pre-defined messages between the drones and ground stations, which provide a high-level interface for applications developers to control and monitor drones without having to interact with hardware. These two alternatives allow software developers to focus more on the high-level development without having to deal with hardware issues.

Cloud Services Layer: Three components types are defined:

(1) *Storage components:* They provide storage services for streams of data originated from UAVs. Each UAVs environment variables, localization parameters, mission information, and transmitted data streams including sensor data and images with time-stamps are stored in the cloud either in regular SQL database or in a distributed file system (i.e. HDFS, NoSQL database such as HBase), depending on the applications requirements. Storage in distributed file systems helps to perform large-scale batch processing on stored data using tools like Hadoop Map/Reduce. There are two types of data processing on cloud computing infrastructures: (*i.*) *Real-time stream processing*: the cloud processes incoming streams of data for detecting possible critical events or threats that require immediate action or performs dynamic computation in a distributed environment. (*ii.*) *Batch processing*: Incoming data is stored in the HDFS distributed file system for increased reliability as well as post-processing using a distributed parallel computing approach. Batch processing can be used to look for particular events into the log file, for example, how many intruders detected in unauthorized area over a certain period of time. The cloud services layer implements a cluster of compute nodes running `Hadoop HDFS`.

(2) *Computation components:* Various computation intensive algorithms are deployed in the cloud. Image processing libraries process stored data available in HBase to detect possible event. In addition, Map/Reduce jobs running on the Yarn cluster allow applications to run in parallel reducing the processing time, therefore improving performance. Additionally Data Analytics algorithms can be executed on the stored set of large scale data.

(3) *Interface components:* We used two types of interfaces: (*i.*) *Network interfaces* that implement network sockets and Websockets interfaces on the server side. They listen to JSON serialized messages sent from UAVs. In particular, Websockets is the most appropriate protocol to reliably handle streaming applications. In the context of Dronemap Planner, MAVLink messages are received from the drones through network sockets (UDP or TCP), and then forwarded to the client application through Websockets. The reason is that Websockets are supported by all programming languages (e.g. Java, Python, C++) including Web technologies. The use of UDP or TCP sockets for streaming to the client will induce more restrictions to the development of Web clients applications. (*ii.*) *The web services interfaces* allow clients to control the missions of the drones and their parameters. Both SOAP and REST web services are used to provide the end-users and clients applications different alternatives to control and monitor the drones through invocation of Web services. While network interfaces are used to mostly handle continuous streams, Web services are used for sending control commands to the drones and getting information from the cloud.

Client Layer: It provides interfaces for both end-users and drones' applications developers. For end-users, the client layer runs `dronemap` client side Web applications, which provide interfaces to the cloud services layer as well as the UAV layer. Users have access to registering multiple UAVs, defining and modifying mission parameters and decision making based on data analysis provided by the cloud. The application allows users to monitor and control the UAVs and their missions remotely. Front-end interfaces provide the functionalities to the user to connect/disconnect, use available physical UAVs and their services, configure and control a mission and monitor the parameters of UAVs. For developers, the client layer provides several APIs for different programming languages to easily develop drones' applications and interact with their drones.

B. Software Architecture

In this section, we present the Dronemap Planner software architecture. We adopted a modular component-based software promoting, where components are loosely coupled and each component implements a specific behavior of the application. In our architecture, we refer to *agent* as a drone, user or a cloud.

1) Architecture Components: Figure 2 shows the component diagram of the software architecture.

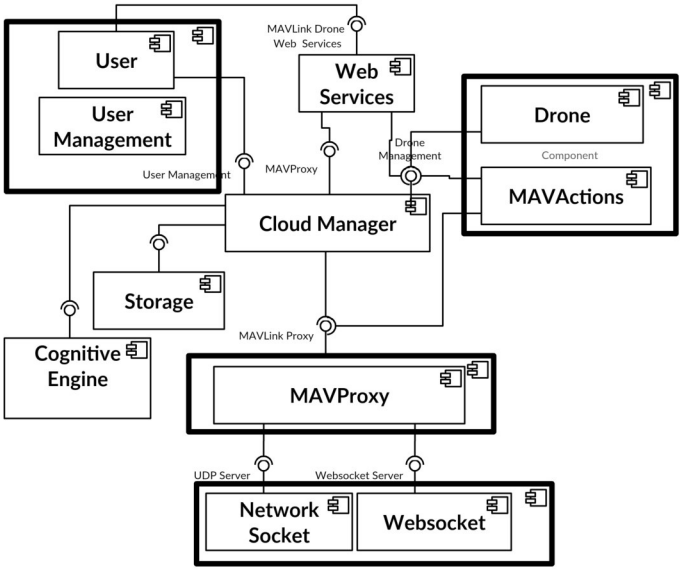

Fig. 2. Dronemap Planner Software Architecture: Component Diagram

The software system is decomposed into five main subsystems (or layers), each of which contains a set of components. These subsystems are:

- **Communication:** This subsystem represents the basic building block for network communications between the drones, users and the cloud. There are two main components, namely (*i.*) Network sockets and (*ii.*) Websockets. On the one hand, Network sockets allow agents (drones, users, and cloud) to exchange JSON serialized messages between each other through the network interface using sockets. The use of JSON message format is beneficial for interaction between heterogeneous systems as it is platform-independent and less verbose than XML. On the other hand, Websockets interfaces are used to handle data streaming between the cloud and the user applications. As explained above, we opted for the use of Websockets technology because it is supported by different programming languages including Web technologies.
- **Proxy:** This layer acts on top of the communication layer and incorporates all the protocol-related operations including message parsing, dispatching, and processing. This layer supports the MAVLink protocol. This communication protocol is the de-facto standard for the communication between ground stations and drones. The MAVLink protocol is based on binary serialization of messages and operates on different transport protocols, namely, UDP, TCP and serial. The MAVProxy is responsible for (*i.*) processing MAVLink related messages received from the drones, (*ii.*) dispatching messages to users through the Websockets protocol, (*iii.*) updating the information of drones objects of the `Cloud Manager`. It is multi-threaded server that was designed to effectively handle MAVLink data streams and messages. At the reception of a MAVLink message, a new thread will be created to process that individual message and extract related information, depending on its message type.
- **Cloud:** The cloud layer is responsible for managing all the computing, storage and networking resources of Dronemap Planner. It is composed of four components, namely (*i.*) Cloud Manager, (*ii.*) Storage, (*iii.*) Web Services components and (*iv.*) Cognitive intelligence. Central the cloud layer is the `Cloud Manager` component, which orchestrates all the processes in Dronemap Planner and links all other components together. It uses the interfaces provided by `MAVProxy` and `ROSLinkProxy` components, in addition to the storage component. On the other hand, it provides interfaces to the Drone and Users components, so that they do have access to `MAVProxy`, `ROSLinkProxy` and `Storage` components. The main role of the `Storage` component is to provide interfaces to store data in different storage media including SQL/NoSQL databases and distributed file storage, i.e. HDFS. Different type of data needs to be stored, retrieved and accessed. For example, SQL databases may be used to store information about users, and their credentials, or also information about drones and their missions. NoSQL databases (e.g. MongoDB) are used for more unstructured data such as data collected from the drones's sensors for further analysis. HDFS storage can be used to store data that requires further batch processing using distributed computing techniques, like Map/Reduce. For example, data related to drones' missions can be dumped from SQL or NoSQL databases to HDFS to process it either with batch processing system like Map/Reduce or real-time processing systems like Storm, and extract useful information for dumped data.
  The `Cognitive Engine` (CE) component aims at performing computations on cloud data to reason, plan and solve problems using artificial intelligence techniques. For example, the CE component may include algorithms to assign multiple targets locations to visit to multiple drones or to a single drone to optimize their missions. This is known as an instance or the typical traveling salesman problem (TSP). Another example would be to process received images or sensor data from drones using real-time processing systems (e.g. Apache Storm) to detect possible events or threats. In general, the CE component will contain intelligent applications to provide smart functionalities and reasoning.
  The `Web services` (WS) component is the main interface between the Dronemap Planner cloud and the client

applications (i.e. users). It provides platform-independent interfaces to end-users and leverages the use the service-oriented architecture (SOA) paradigm. Both SOAP and REST Web services are defined. The REST API was developed to allow developed accessing cloud public resources through simple http requests. The SOAP API was designed for a more formal and structured service-orientation to for remote procedure invocation, which is basically used to send commands to the drone from the client application. There has been long discussions about the pros and cons of REST and SOAP web services and the reader may refer to [14] for more details. In our architecture, we opted for providing both types of Web services as interfaces with end-users for increased flexibility.

- **Drone:** The `Drone` subsystem contains all information related to drones and actions that could be performed on them. The `Drone` component represent *resource* in the Dronemap Planner cloud. This resource is basically accessed by client applications through Web services. In addition, the `MAVAction` component represents all the MAVLink protocol actions that could be executed on the drone including taking-off, landing, waypoint navigation, getting waypoints list, changing operation mode, etc. The `Drone` component maintains the status of the drone, which is updated whenever a new MAVLink message is received. In addition it provides an interface to access and modify the parameters of a drone. Note that the cloud manager maintains a list of drones into a map data structure, as mentioned above.
- **User:** The `User` subsystem contains information about users that access the dronemap Planner cloud. Each user should be registered to the Dronemap Planner cloud to have access to drones based on his profiles and privileges. A user might be able to control a single drone, or multiple drones or all drones based on his privileges. The mapping between drones and users is made through the Cloud Manager during the registration of the user to the system, and based on approval of the cloud administrator. There are different possible strategies of mapping between users and drones, namely: (*i.*) Single User / Single Drone, where one user is allowed to access and control a single physical drone, (*ii.*) Single User/Multiple drones, where one user is allowed to access and control multiple physical drones, (*iii.*) Single User / Virtual Drone(s), where one user is not allowed to control a physical drone, but sends its request to the cloud, which will decide on which drone(s) to execute the mission of the user. Each user should have an access key that allows him to access a certain drone resource over the cloud or to develop applications for a particular drone resource. The access to drone resources on the cloud is given to the users either through SOAP and REST Web services to execute command, or through Websockets to receive drones' MAVLink data streams.

V. EXPERIMENTAL EVALUATION

In this section, we present the experimental evaluation study to demonstrate the effectiveness and performance of Dronemap Planner in meeting the objectives for Internet-of-Drones.

A. Drone Mission over the Web

We used Dronemap Planner cloud to control and monitor drones over the Internet, in particular through the Web. This is possible thanks to the Web services and Websockets interfaces provided by Dronemap planner.

1) Dronemap Planner Cloud Deployment: Dronemap Planner was deployed into a DreamCompute cloud instance provided by Dreamhost service provider. It is also possible to deploy it in any other cloud or public IP server like Amazon AWS or Azure. We used a minimal instance of the DreamCompute with 80 GB of storage, 512 MB or RAM, and 1 Virtual CPU. It is possible to extend the specification of the instance but was sufficient for our experimental study. Once Dronemap Planner is launched all of its services are available and accessible through its public IP address and corresponding port numbers.

2) Drone Configuration: We used low-cost quadcopters with Ardupilot autopilot, which natively supports the MAVLink protocol. Without loss of generality, any drone/aircraft with MAVLink support and WiFi Internet connection is compatible with Dronemap Planner. For a drone to be able to be controlled over Dronemap Planner, it is required that it streams its MAVLink messages to the cloud, which will be intercepted and processed by MAVProxy. We changed the network configuration of the drone so that it connects to a WiFi router and streams its MAVLink messages to the Dronemap Planner.

3) Web-based Ground Station: Figure 3 presents the list of active drones for an administrator user. It can be observed that five actives drones are shown in the interface giving the user with the choice to select to control and monitor any of the available drones. Each drone is characterized by its IP address, port number, and MAVLink System ID. In addition, the physical address of each drone is shown based on its GPS coordinates. Google Geolocation Web Services API were used to determine the location addressed based on the GPS coordinates.

Figure 4 depicts the Dronemap Web graphical user interface.

The web interface contains all information about the drone, including altitude, air/ground speeds, heading, battery level, location address based on GPS coordinates, and GPS fix status. These information are received through the JavaScript Websockets client that connects to the Websockets server of the MAVProxy. As mentioned above, this ensures a reliable bi-directional communication between the Web ground station of MAVLink streams and the MAVProxy through the Websockets protocol. A comprehensive JavaScript/Ajax library was developed to parse and process incoming MAVLink messages and update the Web interface in real-time. In what concerns control

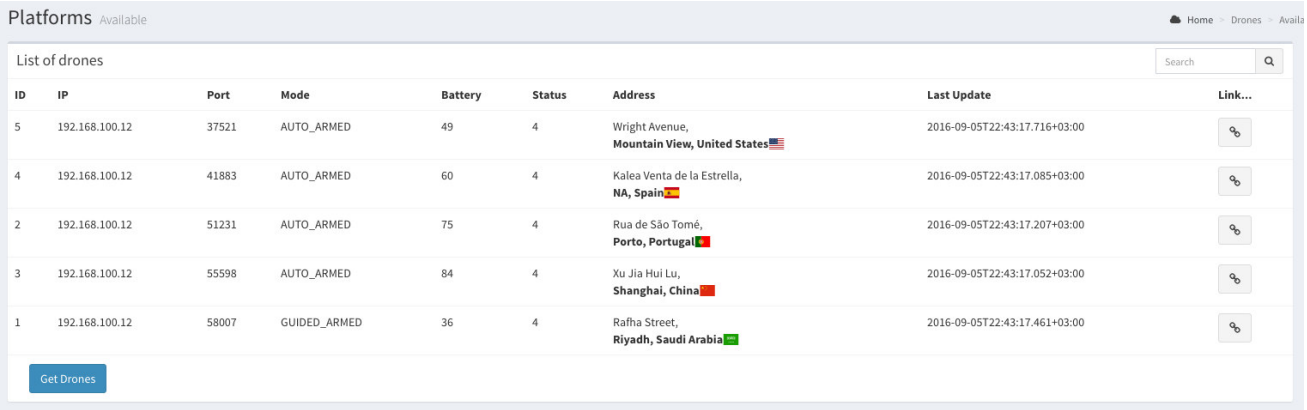

Fig. 3. List of Drones on Web Ground Station

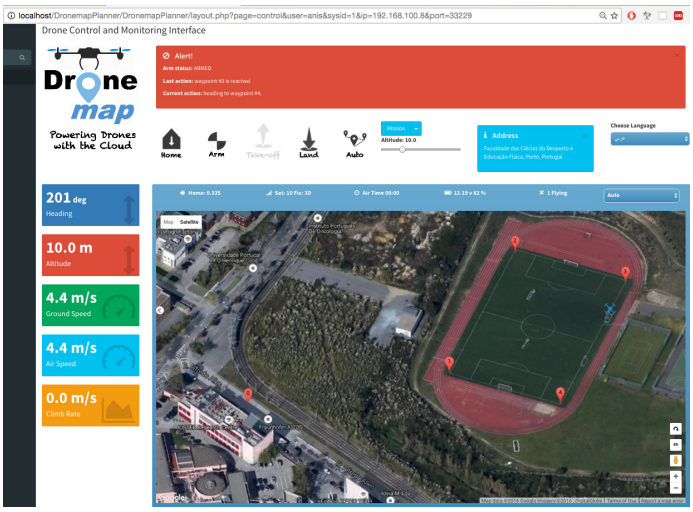

Fig. 4. Dronemap Web-based Ground Station

commands, the Web ground station allows the user to change the flight mode, arm/disarm the drone, take-off and landing, navigate to a waypoint in a guided mode, load and save a mission, execute a mission in autonomous mode, and return to launch. A mission refers to visiting a set of waypoints. The Web ground station allows to add and remove waypoints to a mission and save it to the drone, through the Dronemap Planner. All these control actions are performed through the SOAP Web services interface through remote method invocation. For example, to take-off, a Web service client invokes the take-off method of the Dronemap Cloud Manager Web services called `MAVLinkControllerService`.

This allows any developer to access Dronemap cloud resources through the Web service using any programming language as it will be illustrated in the next section.

It has to be noted that current browsers do not allow to define Web service clients using JavaScript for security reasons. This is a known issues that prevents from directly invoking Web services methods through client-side scripting. To overcome this problem, we developed an intermediate PHP SOAP client that invokes the SOAP Web services of the Dronemap Cloud Manager. As such, the parameters of the remote methods are sent from the browser in JavaScript using a typical Ajax GET request to the PHP SOAP client page, which prepares the corresponding SOAP message using the received parameters, invokes the Web service and return the result to the browser through Ajax response.

The Web ground station allows to define missions for the drones in real-time, change the mission dynamically by adding and/or removing waypoints as required, navigate to a particular waypoint in `Guided` mode.

B. Drone Client Programming API over the Cloud

The advantage of the cloud-based drones' missions control is to allow developers to develop applications and interact with drones leveraging the use of cloud web services. In fact, most programming languages fully support SOAP and REST web services and provide appropriate API to interact with.

In that sense, we demonstrate in this section how simple would be the programming of drones through cloud web services API. This definitely helps in providing pragmatic educational tools for teaching and learning drones' programming for students at undergraduate or even high-school levels, with no prior background on drones, robots or MAVLink is needed to develop programs. The program below defines a minimal mission for a drone written in Java using the drone client API developed to interact with the drones through the Dronemap Planner cloud. The `DroneClient` UML class diagram is presented in Figure 5. A similar API could be developed for other programming languages like Python.

In Line 3, a new `DroneClient` object is created specifying the IP address of the Dronemap Planner server. In Line 4, the drone client object attempts to connect to a drone with a system ID equal to 1. The connection is successful if the client is able to connect to all web services and Websockets servers, and a drone with the specified system ID exists. If the drone exists, the mission will be executed. Lines from 5-15 define a new mission that includes changing the flight mode to guided, arming the drone, taking-off for an altitude of 20.3 meters, going to a pre-defined location, and finally returning

to launch. The program above is rather illustrative and more sophisticated event-driven programs cloud be written based on the API.

```
public static void main(String []args){
    Gson gson = new Gson();
    DroneClient drone =
            new DroneClient("192.168.1.102");
    if (drone.connect(1)){
        System.out.println(drone.getDroneHostID(1))↩
            ;
        drone.flightMode(Config.↩
            MAVLINK_SET_MODE_GUIDED);
        Thread.sleep(5000);
        drone.arm();
        Thread.sleep(5000);
        drone.takeoff(20.3);
        Thread.sleep(10000);
        Location3D location = new Location3D↩
            (24.734840, 46.698421, 25.0);
        drone.goToGoal(gson.toJson(location));
        Thread.sleep(40000);
        drone.flightMode(Config.↩
            MAVLINK_SET_MODE_RTL);
    }
}
```

Listing 1. Sample Drone Client Mission Control Java Program

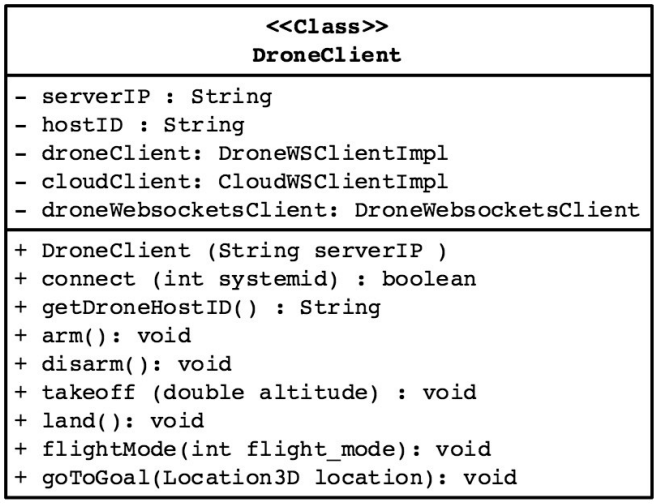

Fig. 5. DroneClient UML Class Diagram

VI. CONCLUSIONS

In this paper, we defined the concept of Internet-of-Drones and illustrated it through Dronemap Planner, a cloud-based management system for drones over the Internet. We discussed the challenges and requirements to build the Internet-of-Drones, and we proposed a modular system and software architecture for the management of drones' missions. The experimental study demonstrated through a proof-of-concept prototype a Web-based control of drones over the Internet and presented a simple API to develop drones applications through the cloud.

We believe that this work represents a major step towards enabling Internet-of-Drones. However, many challenges still need to be addressed as future work. First, security is a very important aspect to investigate. In fact, the MAVLink protocol is not secured and can be hacked quite easily. It is crucial to design secure mechanisms for authentication and encryption of MAVLink data streams to avoid harmful attacks through the cloud. Another challenge is monitoring the QoS of drone control over the Internet. It is needed to investigate the impact of wireless communication perturbation on the quality of drones' management over the network.

ACKNOWLEDGMENTS

This work is supported by the Dronemap project entitled "DroneMap: A Cloud Robotics System for Unmanned Aerial Vehicles in Surveillance Applications" under the grant number 35-157 from King AbdulAziz City for Science and Technology (KACST).

In addition, the authors would like to thank the Robotics and Internet of Things (RIoT) Unit at Center of Excellence of Prince Sultan University for their support to this work. Furthermore, the authors thank Gaitech Robotics in China for their support to this work.

REFERENCES

[1] R. Chaari, F. Ellouze, A. Koubaa, B. Qureshi, N. Pereira, H. Youssef, and E. Tovar, "Cyber-physical systems clouds: A survey," *Computer Networks*, vol. 108, pp. 260 – 278, 2016.

[2] B. Qureshi and A. Koubaa, "Five Traits of Performance Enhancement Using Cloud Robotics: A Survey," *Procedia Computer Science*, vol. 37, pp. 220 – 227, 2014.

[3] J. Kuffner, "Cloud-enabled robots," 2010.

[4] Y. Chen, Z. Du, and M. Garcia-Acosta, "Robot as a service in cloud computing," in *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*, pp. 151–158, June 2010.

[5] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, and G. W. Kit, "Davinci: A cloud computing framework for service robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3084–3089, May 2010.

[6] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. M. M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "RoboEarth," *Robotics Automation Magazine, IEEE*, vol. 18, pp. 69–82, June 2011.

[7] A. Koubaa, *Architecture of Computing Systems – ARCS 2014: 27th International Conference, Lubeck, Germany, February 25-28, 2014. Proceedings*, ch. A Service-Oriented Architecture for Virtualizing Robots in Robot-as-a-Service Clouds, pp. 196–208. Springer International Publishing, 2014.

[8] M. Gharibi, R. Boutaba, and S. L. Waslander, "Internet of drones," *IEEE Access*, vol. 4, pp. 1148–1162, 2016.

[9] L. Apvrille, T. Tanzi, and J.-L. Dugelay, "Autonomous drones for assisting rescue services within the context of natural disasters," in *General Assembly and Scientific Symposium (URSI GASS), 2014 XXXIth URSI*, pp. 1–4, IEEE, 2014.

[10] S. Mahmoud and N. Mohamed, "Collaborative uavs cloud," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pp. 365–373, May 2014.

[11] S. Mahmoud and N. Mohamed, "Broker architecture for collaborative uavs cloud computing," in *Collaboration Technologies and Systems (CTS), 2015 International Conference on*, pp. 212–219, June 2015.

[12] S. Mahmoud, N. Mohamed, and J. Al-Jaroodi, "Integrating uavs into the cloud using the concept of the web of things," *Journal of Robotics*, vol. 2015, September 2015.

[13] "The MAVLINK Protocol, website: http://qgroundcontrol.org/mavlink/start." http://qgroundcontrol.org/mavlink/start.

[14] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. "big"' web services: Making the right architectural decision," in *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, (New York, NY, USA), pp. 805–814, ACM, 2008.