



CISTER

Research Centre in
Real-Time & Embedded
Computing Systems

Journal Paper

DroneTrack: Cloud-Based Real-Time Object Tracking using Unmanned Aerial Vehicles

Anis Koubâa*

Basit Qureshi

*CISTER Research Centre

CISTER-TR-180205

2018/03

DroneTrack: Cloud-Based Real-Time Object Tracking using Unmanned Aerial Vehicles

Anis Koubâa*, Basit Qureshi

*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

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

Abstract

Low-cost drones represent an emerging technology that opens the horizon for new smart Internet-of-Things (IoT) applications. Recent research efforts in cloud robotics are pushing for the integration of low-cost robots and drones with the cloud and the IoT. However, the performance of real-time cloud robotics systems remains a fundamental challenge that demands further investigation. In this paper, we present DroneTrack, a real-time object tracking system using a drone that follows a moving object over the Internet. The DroneTrack leverages the use of Dronemap planner (DP), a cloud-based system, for the control, communication, and management of drones over the Internet. The main contributions of this paper consist in: (1) the development and deployment of the DroneTrack, a real-time object tracking application through the DP cloud platform and (2) a comprehensive experimental study of the real-time performance of the tracking application. We note that the tracking does not imply computer vision techniques but it is rather based on the exchange of GPS locations through the cloud. Three scenarios are used for conducting various experiments with real and simulated drones. The experimental study demonstrates the effectiveness of the DroneTrack system, and a tracking accuracy of 3.5 meters in average is achieved with slow-speed moving targets.

DroneTrack: Cloud-Based Real-Time Object Tracking Using Unmanned Aerial Vehicles Over the Internet

ANIS KOUBAA^{1,2,3} AND BASIT QURESHI¹, (Member, IEEE)

¹Prince Sultan University, Riyadh 12435 3276, Saudi Arabia

²Gaitech Robotics, Shanghai 201101, China

³CISTER, INESC-TEC, ISEP, Polytechnic Institute of Porto, 4200-465 Porto, Portugal

Corresponding author: Anis Koubaa (akoubaa@psu.edu.sa)

This work was supported in part by Gaitech Robotics, China, in part by the Robotics and Internet-of-Things Lab, and in part by the Research and Initiative Center, Prince Sultan University, Saudi Arabia.

ABSTRACT Low-cost drones represent an emerging technology that opens the horizon for new smart Internet-of-Things (IoT) applications. Recent research efforts in cloud robotics are pushing for the integration of low-cost robots and drones with the cloud and the IoT. However, the performance of real-time cloud robotics systems remains a fundamental challenge that demands further investigation. In this paper, we present DroneTrack, a real-time object tracking system using a drone that follows a moving object over the Internet. The DroneTrack leverages the use of Dronemap planner (DP), a cloud-based system, for the control, communication, and management of drones over the Internet. The main contributions of this paper consist in: (1) the development and deployment of the DroneTrack, a real-time object tracking application through the DP cloud platform and (2) a comprehensive experimental study of the real-time performance of the tracking application. We note that the tracking does not imply computer vision techniques but it is rather based on the exchange of GPS locations through the cloud. Three scenarios are used for conducting various experiments with real and simulated drones. The experimental study demonstrates the effectiveness of the DroneTrack system, and a tracking accuracy of 3.5 meters in average is achieved with slow-speed moving targets.

INDEX TERMS Unmanned aerial vehicles, object tracking, cloud computing, Internet of Drones.

I. INTRODUCTION

In recent years, there has been an explosive growth in the usage of unmanned aerial vehicles (UAVs) commonly known as drones. Markets and Markets [1] reported that the UAV market is estimated to be USD 13.22 Billion in 2016 and is expected to surpass USD 28.27 Billion by 2022, reflecting expansive growth in a short period of time. Indeed, the majority of commercial UAV solutions such as 3DR Solo, DJI Phantom, Erle Copter, to name a few, rely on point-to-point communication between the drone and the ground station. These communication mechanisms rely on long-range telemetry devices, or WiFi channels using TCP/UDP protocols therefore restricting the operation of drones within a restricted geographic area limited by the communication range.

The control of drones through the Internet is a viable solution to overcome this limitation and has been recently proposed. Kuffner [2] and Chaari *et al.* [3] proposed the

usage of the Internet-of-Things (IoT) and cloud computing resources for developing new robotics applications. Chen *et al.* [4] specified the concept of Robot-As-A-Service (RAAS), which defines a service-oriented framework for robots to interact with the cloud. In [5], the DAVinci system was proposed to offload extensive computation from the robots, however, reliability and real-time were not addressed. In [6], the European project consortium developed the World Wide Web for Robots to share knowledge among robots for accomplishing complex tasks. Ng *et al.* [7] proposed a cloud-robotics platform to assist mobility-impaired people to navigate in a museum using Robot Operating System (ROS) enabled robots. However, no validation was made for this proposal. In [8], the ROSLink protocol was proposed to control ROS-enabled robots through the cloud. The authors simulated and evaluated the performance of an open-loop control system in terms of bandwidth and real-time constraints. However, the evaluation study neither considered real drones,

nor a closed loop control system unlike the tracking application addressed in this paper. While the aforementioned works provided significant milestones to cloud robotics, they lack in explicit evaluation of cloud robotics platforms application in for real-time scenarios. In fact, there is a wide gap in the literature that addresses performance evaluation of real-time UAV applications utilizing cloud robotics platforms.

To the best of our knowledge, there is no prior work that addressed the real-time tracking of moving objects using drones over cloud robotics platform. In this paper, we address this gap, and conduct a performance evaluation study of real-time object tracking through a cloud robotics platform. We first present the Dronemap Planner (DP) [9] cloud robotics platform that was designed for the monitoring and control of robots and drones through the Internet. Next, we present the design and architecture of DroneTracker, an object tracking application built on top of the Dronemap Planner Cloud. DroneTracker is exposed as a cloud service that provides interaction between a user/drone and DP through Websockets and MAVLink Proxy. We present a worst-case delay analysis model for the tracking application using Network Calculus and we demonstrate through extensive simulations and experimental studies the effectiveness of the proposed platform considering real-time constraints.

The contributions of the paper are three folds:

- We propose the design of DroneTrack, a cloud-based object tracking application over the Internet using a service-oriented architecture.
- We provide a deterministic model of worst-case delay and tracking distance using Network Calculus. The model presented is used to explain the impact of acceleration and speed in the experimental performance evaluation.
- We experimentally deploy and evaluate the performance of the cloud-based tracking application using real and simulated drones, and demonstrate the effectiveness of the system.

The remainder of this paper is organized as follows. Section II discusses related works. Section III presents the architecture of the Dronemap Planner cloud robotics platform. Section VI presents the DroneTracker, object follower application, and details its integration with the Dronemap Planner cloud. Section V describes performance evaluation criterion for delay analysis used in this study. Detailed simulation and experimental results from various scenarios are presented in section VI. Finally, Section VI provides a discussion and lessons learned, concluding the paper.

II. RELATED WORKS

There are several up to date research efforts in formulating robust futuristic cloud-based robotic applications. We classify these efforts in two sets, Cloud Robotics Systems and Drone based Systems.

A. CLOUD ROBOTICS SYSTEMS

Du *et al.* [10] presented algorithms and an implementation of cloud-based system namely, Robot Cloud, with the aim to leverage the cloud-based robot systems flexibility, re-usability and extensibility. They develop a prototype of Robot Cloud using the service-oriented architecture (SOA) which is deployed on Google App Engine. Bozcuoğlu and Beetz [11] address the Cloud based system for predicting consequences of a robotic systems actions just before execution. They build openEASE system that allows researchers and robots to execute complex mental simulation problems remotely on the cloud utilizing the massive storage and computation capacities of the cloud. The system uses learning algorithms and suggest solutions to the robot how to handle the situation. Huang *et al.* [12] present an early implementation of a Cloud Robotics Middleware that allows offloading of computation and storage from robots to the cloud. All of the above mentioned works deploy an early implementation of cloud robotics systems.

Hu *et al.* [13] address the lack of adequate onboard computation resources in a robot for execution of Simultaneous Localization and Mapping (SLAM) commonly used for drawing map of the surroundings. They propose Cloudroid, a QoS aware software framework that allows deployment of robotics packages to the cloud as cloud services. They further evaluate the performance in terms of request response time, in highly dynamic and resource-competitive environments. Wan *et al.* in [14] present Context Aware Cloud Robotics (CACR) System that provides decision making mechanisms to handling of industrial robots such as automated guided vehicles. The proposed architecture of the system utilizes cloud-enabled implementation for simultaneous localization and mapping. The researchers study the improved energy efficiency and cost saving as main benefit of using the cloud based system. In [15] Tian *et al.* describe Berkeley Robotics and Automation as a Service (Brass), a RAaaS prototype that allows robots to access a remote server that hosts a robust grasp-planning system. The cloud based system maintains data on hundreds of candidate grasps on thousands of 3D object meshes. The system uses perturbation sampling to estimate and update a stochastic robustness metric for each grasp. Their results suggests increase in grasp reliability in remote computation with acceptable network latencies for robots located thousands of miles away.

Reid *et al.* in [16] develop cloud computing infrastructure for networked heterogeneous robotic systems in open-source robot operating system (ROS). This work demonstrates the minimal impact on network performance by devices which use a significant amount of local processing for their operation. In their test-bed, various Turtlebots are connect to the cloud using wireless network. They carry extensive testing to evaluate the performance of the system using low and high bandwidth channels to study the latency, data integrity of the communications. Li *et al.* in [17] propose a novel hybrid architecture for cloud robotics, named RoboCloud.

The main objective of this research is integration of robots with cloud and providing task specific services without degrading the QoS. RoboCloud introduces a task-specified mission cloud with controllable resources determined by predictable behavior. They test the proposed architecture by analyzing the QoS parameters such as latency in a cloud service that provides cloud based object recognition.

B. DRONE BASED SYSTEMS

There have been a few attempts to integrate drones with the cloud and IoT. Gharibi et. al [18], presented a conceptual model for the Internet-of-Drones. The proposed architecture covers three major networks namely are air traffic control network, cellular network and Internet. The layered architecture provides generic services for different UAV applications, namely delivery, surveillance, search and rescue, etc. The paper did not present any implementation or realization of this architecture and only outlines 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.* [19] presented a model of a drone usage in natural disaster recovery where drones scan an environment first before starting the rescue operation using dense 3D scan and then continue operation using light 3D scan using monochrome images that can be helpful in rescue operations. Batim and Mellouk [20], present an intelligent traffic control architecture based on cloud that is based on conventional cloud for provision of services with static cellular network helping in identification of traffic, parking area and other services while the dynamic feature of proposed architecture will help in preparing a temporary cloud between vehicles, person and vehicles after provision to be part of network. It will allow temporal local data center storage and local service for quick and fast response. The proposed architecture is three layered with customer at one end and cloud at the other end and a middle layer joining them.

Bona [21], presents a cloud robotic platform called as FLY4SmartCity that is based on ROS. The proposed architecture contains basic features to create instances of drones as nodes where they are handled by platform manager in terms of planning and event management. The platform manager is supported by service manager for provision of services in case of events while rule manager to handle the actions. Ermacora *et al.* [22], presents a cloud robotics platform for emergency monitoring based on ROS. It allows leveraging the advantages of cloud to offload the data and computational capabilities. The layered architecture provides services built on API provided by applications built of drone capabilities and adaptation. Drones form the physical layer of the architecture.

Yanmaz *et al.* in [23] detail a high level architecture for the design of a collaborative aerial system consisting of drones with on-board sensors and embedded processing, sensing, coordination, and networking capabilities. They implement a multi-drone system consisting of quadcopters and demonstrate its potential in disaster assistance, search

and rescue, and aerial monitoring. The evaluate the performance of the system using parameters such as latency in communications and effectiveness of mission planning. The implementation lacks cloud based interaction with the drones.

It is important to study the QoS parameters for Cloud Drones applications in order to provide scalable, reliable and efficient systems. In the literature there is a wide gap in QoS evaluation of cloud robotics applications that we intend to fill. In this paper, we present DroneTrack, a Cloud based Real time object tracking using Unmanned Aerial Vehicles (UAV)s. We present design and architecture of Dronemap planner, a cloud based UAV tracking and monitoring system. Further, we focus on development and deployment of a real time object follower application using the Dronemap planner. Moreover, we provide a detailed study on the performance evaluation of the real-time tracking application and accuracy of the application in terms of network latency and tracking distance for various conditions and settings.

III. DRONEMAP PLANNER ARCHITECTURE

Dronemap planner is a cloud based system that realizes the concept of Internet-of-Drones(IoD) where multiple autonomous drones can be controlled and managed by users. A user behind the cloud defines a mission (e.g. visiting a set of waypoints) requesting its execution. The system defines virtual UAVs which are mapped to physical UAVs using a service-oriented approach, typically implementing SOAP or REST Web services. Once a mission request is received, the selected UAVs execute the mission and report in real-time the data of interest to the cloud service, which in turn will store, process and forward synthesized results to the user. The following presents an overview of the Dronemap planner architecture.

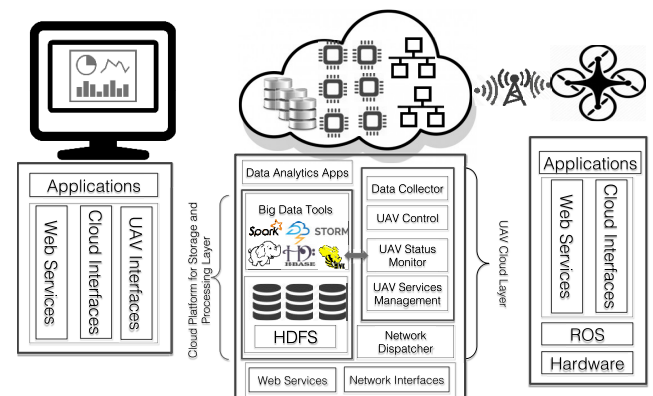


FIGURE 1. DroneMap system architecture.

A. DRONEMAP ARCHITECTURE

The system consists of three abstraction layers namely, UAV layer, Cloud Services layer and Client Layer. Figure 1 presents the architecture of Dronemap.

1) THE UAV LAYER

This layer exposes system resources to the end-user as services. The UAV layer provides interaction with the hardware using Robot Operating System (ROS) and MAVLink communications protocol. ROS is one of the widely used middleware to develop robotics applications. MAVLink is a communication protocol built over various transport protocols (i.e. UDP, TCP, Telemetry, USB) that allow exchange of pre-defined messages between the drones and between drones and ground stations. Together, ROS and MAVLink provide a high-level interface for applications developers to control and monitor drones without the need for direct programming and interaction with the hardware.

2) CLOUD SERVICES LAYER

This layer is responsible for realization of cloud services using three sets of components, i) cloud based storage, ii) remote computation and iii) communication interactions.

Streams of data originated from UAVs are stored in the Cloud. Information such as a UAVs environment variables, localization parameters, mission information, and transmitted data streams including sensor data and images with timestamps are stored in the cloud using 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. The system supports real-time and batch processing of data. In case of Real-time data input stream, 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. In case of batch processing, the Incoming data is stored in the HDFS distributed file system which can later be used for further analysis.

The system provides remote computation in the cloud. Various computation intensive algorithms using libraries for image processing and data analysis are provided. In addition, Map/Reduce jobs running in Hadoop allow applications to run in parallel improving the processing time, therefore increasing system efficiency. Additionally Data Analytics algorithms can be executed on the stored set of large scale data.

Communications interfaces is the third aspect of the Cloud services layer. Network interfaces and web services are two types of interactions supported by the system. The network interface implements network sockets on the server side that listens to JSON serialized messages sent from UAVs. 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 applications using websockets. The web services 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 various alternatives to control and monitor the drones through invocation of Web services. While

network interfaces are utilized mostly to handle continuous streams, Web services are used for sending control commands to the drones and getting information from the cloud.

3) CLIENT LAYER

This layer provides interfaces for both end-users and drones' applications developers. For end-users, the client layer executes `dronemap` client side Web applications, that provide interface to the cloud services layer as well as the UAV layer. End-users can register multiple UAVs, define and modify mission parameters based on results provided by the cloud. The application allows users to monitor and control the UAVs and their missions remotely. Front-end interface supports functions to connect/disconnect, use available physical UAVs and their services, configure, control a mission and monitor the parameters of the UAVs. For developers, the client layer provides several APIs for different programming languages to easily develop drones' applications.

B. SYSTEM COMPONENTS

Dronemap Planner system adopts a modular component-based software design, where components are loosely coupled and each component implements a specific aspect of the application. In the preceding text, we refer to *agent* as a drone, user or a cloud.

Figure 2 shows the component diagram of the software architecture.

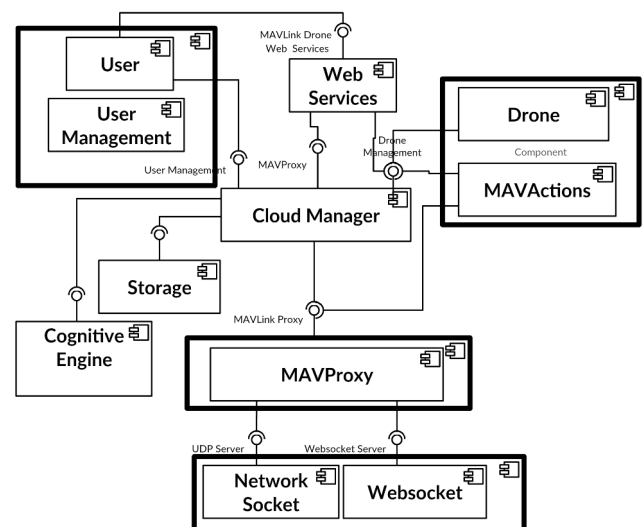


FIGURE 2. Dronemap planner software architecture: component diagram.

The software system is decomposed into five main sub-systems, each of which contains a set of components. These subsystems are:

- **Communication:** This subsystem implements the basic building block for network communications. The two main components, are (i.) Network sockets and (ii.) Websockets. Network sockets allow agents to exchange JSON serialized messages between each other

through the network interface using sockets. 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.

- **MAVProxy:** This component sits on top of the communication subsystem and incorporates all the protocol-related operations including message parsing, dispatching, and processing. It supports the MAVLink protocol which is based on binary serialization of messages and operates on various transport protocols, including, UDP and TCP. The MAVProxy is responsible for (i.) processing MAVLink messages received from the drones, (ii.) dispatching messages to users through the Websockets protocol, and (iii.) updating the received information on agents in the Cloud Manager. The cloud manager is a component in the Cloud sub-system presented later.
- **Cloud:** The cloud subsystem is responsible for managing all the computing, storage and networking resources of Dronemap Planner. It is composed of four components, (i.) Cloud Manager, (ii.) Storage, (iii.) Web Services components and (iv.) Cognitive Engine. Central to this subsystem is the Cloud Manager component, which orchestrates all the processes in Dronemap Planner and knits all components together. It interacts with 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 communicate with the MAVProxy, ROSLinkProxy and Storage components. The main role of the Storage component is to provide interfaces to store data in various storage media. SQL databases are used to store information about users credentials, information on drones and their missions. NoSQL databases (e.g. MongoDB) are used for unstructured data storage including the data collected from the drones's sensors. The Cognitive Engine (CE) component provides support to requests on intelligence related computations that rely on problem solving using artificial intelligence techniques. Furthermore, it aims to address scenarios where real-time analysis of data is required by an intelligent application such as object detection where object parameters needs to be compared and analyzed against set of rules pre-defined in the system. The Web services (WS) component provides interface between the Dronemap Planner cloud and the client applications. 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.

- **Drone:** The Drone subsystem addresses all aspects of information related to drones. The Drone component addresses *resource* in the Dronemap Planner cloud which is accessed by client applications through Web services. The MAVAction component addresses 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.
- **User:** The User subsystem maintains information about users accessing the dronemap Planner cloud. Users need to provide credentials in order to access the system with appropriate privileges. As a user registers in the system, the Cloud Manager provides appropriate mapping between this user and available drones. There are various strategies of mapping between users and drones, including: (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 provided to the users either through SOAP and REST Web services to execute command, or through Websockets to receive drones' MAVLink data streams.

IV. DroneTrack: A REAL-TIME OBJECT TRACKING APPLICATION

In this section, we present the architecture of DroneTrack, a real-time object tracking application. The system architecture is presented in Figure 3a and shows the main actors of the follower application. The drone is connected to the cloud using 3G/4G Internet connection and uses the MAVLink protocol to communicate with the cloud. The user communicates with the cloud using a mobile device through the Internet, using Web services and Websockets interfaces. A DP Web application can be used to control the mission remotely. The component diagram presented in Figure 3b shows more details about the internal structure of the cloud services, drones services and user application services and how they interact with each other.

A. APPLICATION SCENARIO

We consider the following scenario that we use to present the design and the implementation details for DroneTrack.

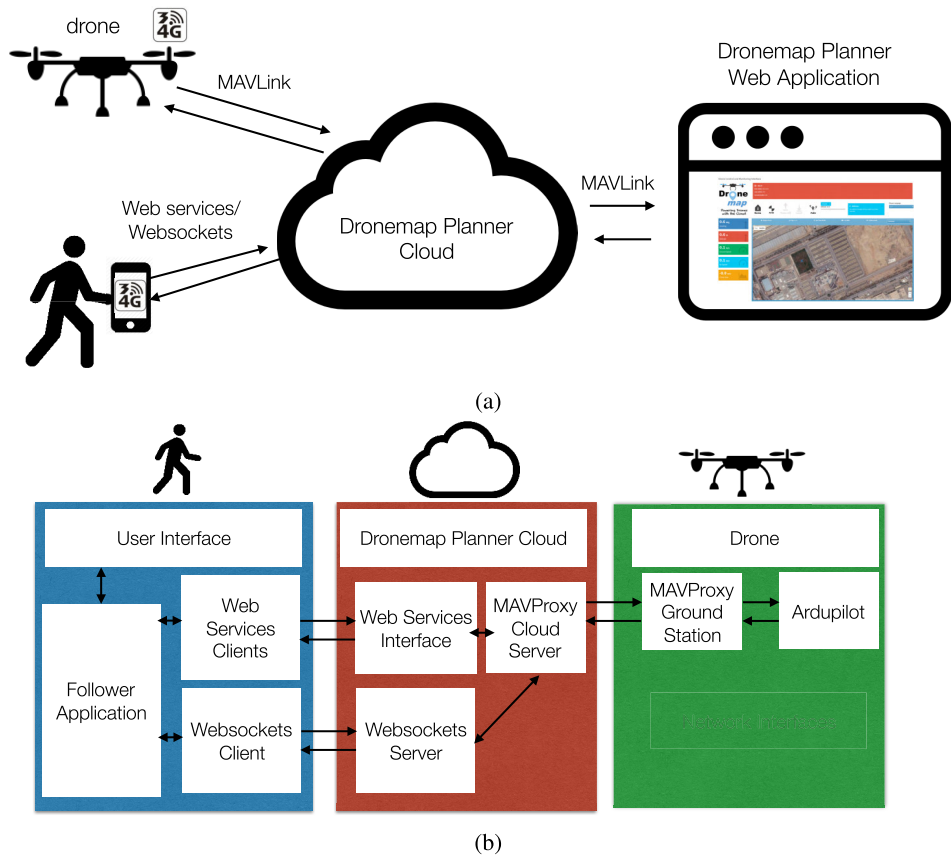


FIGURE 3. DroneTrack system architecture. (a) DroneTrack information flow. (b) DroneTrack components diagram.

Assume that an object needs to be followed by a drone, the object could be a person walking in a suburban environment, or a vehicle driving within the city. The drone needs to interact with the command center through the Internet and constantly updates the command center about its GPS location and related parameters. The scenario presented here could be used in search and rescue missions or by law enforcement agencies to track illegal activities. Based on this scenario, we build DroneTrack that provides a middleware between the moving object and the Dronemap Planner Cloud services. The only requirement for the tracked object is to carry on-board a mobile device with GPS localization capabilities that sends its GPS coordinates regularly. DroneTrack interacts with the GPS localization data from a mobile device on board the moving object, to provide connectivity to the DP Cloud service.

The sequence diagram of a successful mission of the cloud-based follower application is illustrated in Figure 4.

As can be seen in Figure 4, the mission is initiated by sending a follow request from the mobile device of the moving object to the cloud. The Dronemap Planner cloud will look for available drones that are already registered and are available to execute the mission. It will select the appropriate drone that will execute the mission with an optimal cost (e.g. the closest to the moving object). Once a drone is allocated

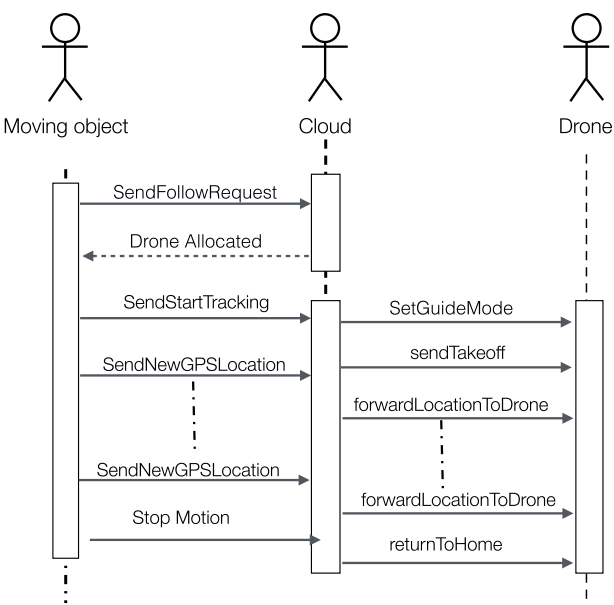


FIGURE 4. UML sequence diagram of follower application.

and the client application is notified, the user can start the tracking session by sending a request to Dronemap Planner, which will send the command to the drone to start following

the moving object. It will configure the drone to operate in GUIDED mode, which is the mode used in Ardupilot to autonomously navigate towards specific GPS locations. The client applications keep updating the Dronemap Planner with its GPS locations in real-time while moving, and the Dronemap Planner will forward these locations as soon as received to the drone to keep following the moving object. In what follows, we present the software architecture of the follower applications and the different Web services and how it was integrated into the Dronemap Planner cloud.

B. SOFTWARE DESIGN

The follower application was integrated into Dronemap Planner cloud as an independent software module that interacts with the other cloud modules, namely MAVProxy, and Drone modules, and exposes new Web Services methods to the end-users that use the follower application. Thus, from the end-user perspective, the follower application is exposed as a set of Web services methods that can be invoked by the client application. On the other hand, during the tracking mission, the exchange of GPS locations is performed through a Websockets connection, which is more appropriate for reliable bi-directional real-time streaming.

1) CLOUD-SIDE WEB SERVICE MODULES

There are six Web services methods available for the end-user client applications, namely:

- **Follow Request Web service method:** this method enables a user to send a follow request to the cloud. In Figure 4, it is the first message sent to the cloud. The request has as parameter the location of the user to track. Once the request is received, the follower cloud application will search for an available drone among all drones registered in the cloud and select the one that will minimize the cost of the mission and with sufficient energy. In the current implementation, we consider the cost of the distance to the person to follow, the closest drone to the person will be selected, and allocated for the tracking mission.
- **Cancel Follow Request web service method:** this method allows the user to cancel the request, if not started, and to release the allocated drone.
- **Start Tracking web service method:** this method gives the user the ability to start the tracking mission. Once the user sends the start tracking message to the cloud, the latter will change flight mode of the drone to GUIDED mode through the MAVLink protocol (using the Dronemap Planner API to interact with the drone), then, will send an arm and takeoff messages to the drone, which will fly at the desired altitude as a consequence, then will go towards the longitude and latitude location of the moving object. When the tracking starts, a Websockets connection is open between the follower client application and the follower cloud application. On the follower application side, the Websockets receive

```
<definitions
targetNamespace="http://apps.cloud.dronemap.org/" name="ApplicationsWSImplService">
  <types>
    <xsd:schema
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://apps.cloud.dronemap.org/"
      schemaLocation="http://192.168.100.3:10500/ApplicationsService?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="followRequest" ... />
  <message name="followRequestResponse" ... />
  <message name="cancelFollowRequest" ... />
  <message name="cancelFollowRequestResponse" ... />
  <message name="startTrackingMission" ... />
  <message name="startTrackingMissionResponse" ... />
  <message name="stopTrackingMission" ... />
  <message name="stopTrackingMissionResponse" ... />
  <message name="disableTracking" ... />
  <message name="disableTrackingResponse" ... />
  <message name="enableTracking" ... />
  <message name="enableTrackingResponse" ... />
  <service name="ApplicationsWSImplService">
    <port name="ApplicationsWSImplPort" binding="tns:ApplicationsWSImplPortBinding">
      <soap:address location="http://192.168.100.3:10500/ApplicationsService"/>
    </port>
  </service>
</definitions>
```

FIGURE 5. Excerpt of the WSDL document of the follower cloud Web services.

messages with updated locations of the moving object. If the tracking is enabled, this new location is sent to the drone through the call of the specific method that sends new waypoints to the drone using MAVProxy instance of the cloud. As such, the drone will head towards this new location as soon as the new mission item is received. On the follower client application side, the Websockets will receive the location of the drone and will update it in the GUI using Google Maps so that the user can track the location of the drone.

- **Stop Tracking Mission web service method:** When this method is called (see last command in Figure 4), it will stop the tracking mission, and the drone will return to its home position and the tracking mission is completed.
- **Enable/Disable Tracking web service method:** this command allow the user to enable or disable the tracking without completing the mission. When the mission is disabled, the drone will still be flying and allocated to the user but will track its new locations until the tracking is enabled again.

The Web Service Description Language (WSDL) document of the follower application is presented in Figure 5.

2) WEB SERVICE CLIENT APPLICATION

We developed a follower client web application that allows a user to allocate a drone and execute a tracking mission through the dronemap planner cloud. The web application is then converted to Android and iOS application using GoNative (<https://gonative.io/>). A screenshot of Android interface is presented in Figure 6.

The Graphical User Interface provides all interfaces with the web services described above to execute the tracking mission. The application was tested to track walking persons and moving cars through Dronemap Planner cloud.

The accuracy of real-time object tracking depends on the delay that location messages take to reach the drone and be executed. In the next section, we will derive a formal model to estimate the maximum delay of the follower application

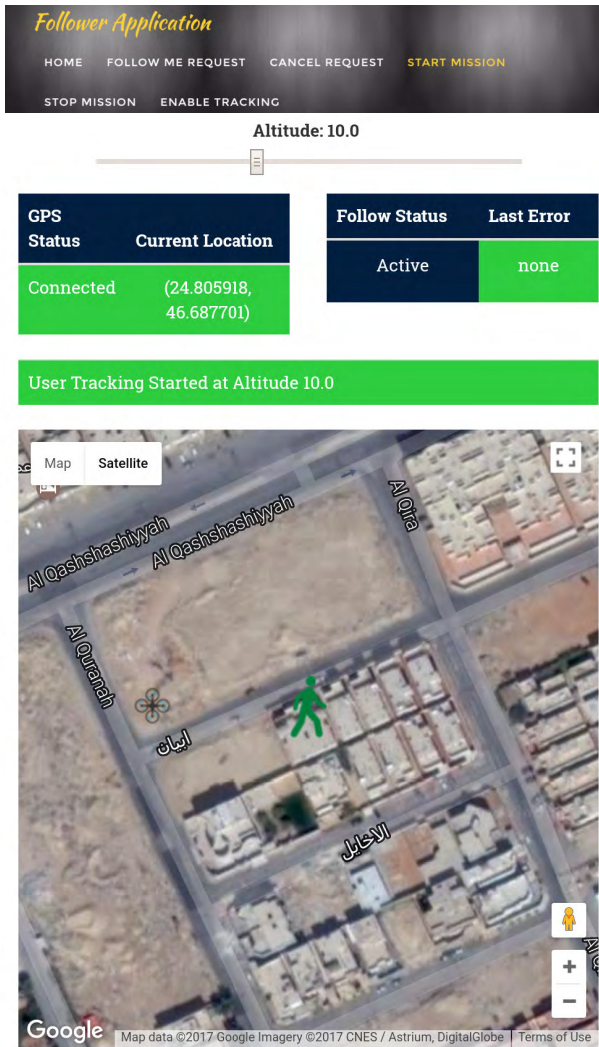


FIGURE 6. Mobile follower client application.

through the cloud and assess its impact on the accuracy of tracking.

V. WORST-CASE DELAY ANALYSIS

In this section, we present a mathematical model using Network Calculus Theory [24] to evaluate the worst-case delay of the DroneTrack object follower application through the cloud and investigate its impact on the tracking accuracy. Network Calculus is mathematical formalism used to evaluate the deterministic performance of queuing systems and derive upper bounds on quality-of-service performance metrics, including delay and buffering requirements. Our objective is to find the maximum delay between the drones and the moving objects during a tracking mission.

A. SYSTEM MODEL

We assume that the client tracking application generates GPS locations with a cumulative arrival function bounded by the linear arrival curve $\alpha(t) = b + r \cdot t$. The network service

pipeline model is composed of three stages, namely (i.) Transmission from the user to the cloud (ii.) processing inside the cloud, (iii.) Transmission from the cloud to the drone. Our objective is to estimate the maximum end-to-end delay of the follower application.

For this analysis model, we assume a reliable communication between the drone and the end-user through the Internet. The limitation of this worst-case delay analysis model is that it does not take into account probabilistic communication losses. However, they can be modeled as additional latencies that affect the system performance. In real time deployment, the tracking application will require reliable communication for its correct operation, and this could be achieved by using high-quality wireless communication with certain guarantees from network service providers [25].

The transmission from the user to the cloud is modeled as a *rate-latency* service curve as follows:

$$\beta_{R_{toCloud}, T_{toCloud}}(t) = R_{toCloud} \cdot (t - T_{toCloud})^+ \quad (1)$$

where $R_{toCloud} \geq r$ is the guaranteed network bandwidth from the user to the cloud, and $T_{toCloud}$ is the maximum latency of the service of the network from the user to the cloud, and $(x)^+ = \max(0, x)$.

Likewise, the transmission from the cloud to the drone is modeled as *rate-latency* service curve:

$$\beta_{R_{toDrone}, T_{toDrone}}(t) = R_{toDrone} \cdot (t - T_{toDrone})^+,$$

where $R_{toDrone} \geq r$ is the guaranteed network bandwidth from the cloud to the drone, and $T_{toDrone}$ is the maximum latency of the service of the network from the cloud to the drone.

The cloud processing component is a multi-threaded process that takes care of the incoming messages before these are forwarding to the drone. This service is modeled as a latency service, where the latency $T_{processing}$ corresponds to the maximum time needed to process the message by a thread in the cloud, before being forwarded.

Using the concatenation theorem of rate-latency service curves [24], the end-to-end service curve is derived as:

$$\begin{aligned} \beta_{R,T}(t) &= \beta_{R_{toCloud}, T_{toCloud}}(t) \\ &\otimes \beta_{R_{toDrone}, T_{toDrone}}(t) \otimes \beta_{T_{processing}}(t) \\ &= \beta_{\min(R_{toCloud}, R_{toDrone}), [T_{toCloud} + T_{toDrone} + T_{processing}]}(t) \end{aligned} \quad (2)$$

As a consequence, the end-to-end delay bound for a data flow with linear arrival curve $\alpha(t) = b + r \cdot t$ guaranteed by the service curve $\beta_{R,T}(t)$ of Equation 2 is:

$$D_{max} = \frac{b}{R} + T \quad (3)$$

Where $R = \min(R_{toCloud}, R_{toDrone})$ and

$$T = [T_{toCloud} + T_{toDrone} + T_{processing}]$$

The maximum delay in Equation 3 represents the maximum time gap between the drone and the moving object. This

TABLE 1. Experimental scenarios.

	Drone Type	Target Type	Runs	Environment	Internet Connectivity
Scenario 1	Real drone	Walking/running person	run1: walking and running	Football field	Drone-Cloud: 3G User-Cloud: 3G
Scenario 2	Simulated drone	walking person	run1: walking run2: walking run3: walking	Residential area	Drone-Cloud: Optical Fiber (40 mbps) User-Cloud: 3G
Scenario 3	Simulated drone	Moving vehicle	run1: low speed, low acceleration run2: high speed, high acceleration	Residential area	Drone-Cloud: Optical Fiber (40 mbps) User-Cloud: 3G

gap will be smaller as long as the maximum speed of the drone is greater or equal to the maximum speed of the moving object. Consequently, the gap will increase indefinitely if the moving object keeps moving faster than the drone, this scenario should be avoided in a real time case. It has to be noted that the difference in acceleration between the drone and the moving object will also affect the distance gap among them. A moving object with a higher acceleration compared to the drone will not be reachable by the drone. The relation between speed, acceleration, tracking distance and latency will be further explained in the experimental evaluation section.

We analyze the relation between the speed and the tracking delay, considering an equal null acceleration between the drone and the moving object (i.e. constant speed), if the speed of the drone is given by V_{drone} whereas the speed of the moving object is given by V_{object} , then the maximum distance between the drone and object, which refers to the accuracy of the tracking, is expressed as:

$$Distance_{max} = D_{max} * V_{object} \quad (4)$$

The tracking is possible if and only if $V_{object} \leq V_{drone}$.

Hence, the accuracy of tracking can be measured with reference to the distance between the drone and the moving object as expressed in Equation 4.

The next section presents a experimental performance evaluation study of the DroneTrack follower application and demonstrates how the cloud based application is able to meet the real-time requirements of a tracking application.

VI. PERFORMANCE EVALUATION

In this section, we detail results from various experiments to evaluate the performance of real-time tracking application using DroneTrack. The primary focus of this work is the tracking accuracy in terms of network delay and tracking distance for various parameters.

A. EXPERIMENTAL SETUP

The experimental testbed consists of three different scenarios which are used for the performance evaluation of the real-time tracking application. The first scenario was evaluation of a real drone test flight at the Soccer field in Prince Sultan University. The other two scenarios correspond to the use of simulated drones in public residential areas in Riyadh city, due to the restrictions imposed by local laws govern-

**FIGURE 7.** Custom drone used in experiments.

ing aviation regulations prohibiting flying drones in public areas. Alternatively the real time evaluation of real drone's flight experimentation was conducted at the University Campus.

In experiments using real drones, we utilized our custom built drone shown in Figure 7. Our custom built drone is a 450 mm quadcopter with DJI F450 Frame equipped with a Navio2 autopilot on top of a Raspberry PI 3 single board computer running an embedded Raspbian linux image. Raspberry PI 3 has Quad Core 1.2GHz Broadcom 64-bit CPU, with 1GB or RAM and MicroSD card for storage. It has an embedded WiFi and Bluetooth interfaces. Navio2 autopilot board is a drone controller hardware that is equipped with the UBlox NEO-M8N embedded GNSS receiver to track GPS signals with an external antenna, while allowing the connection of external GPS devices in its UART port. It has a dual IMU with two 9 degree-of-freedom IMUs, namely the MPU9250 and LSM9DS1. Each IMU contains an accelerometer, a gyroscope, and a magnetometer which are fused together to estimate drone acceleration and speed. The Dronemap Planner cloud server is hosted on the DreamCompute cloud `gpl.wrapspeed` instance, which has 4 virtual CPU cores, 80GB hard disk, and 8GB or RAM, running the Ubuntu 14.04 operating system.

B. EXPERIMENTAL SCENARIOS

In what follows, we present the experimental scenarios. The summary of the scenarios is presented in Table 1.

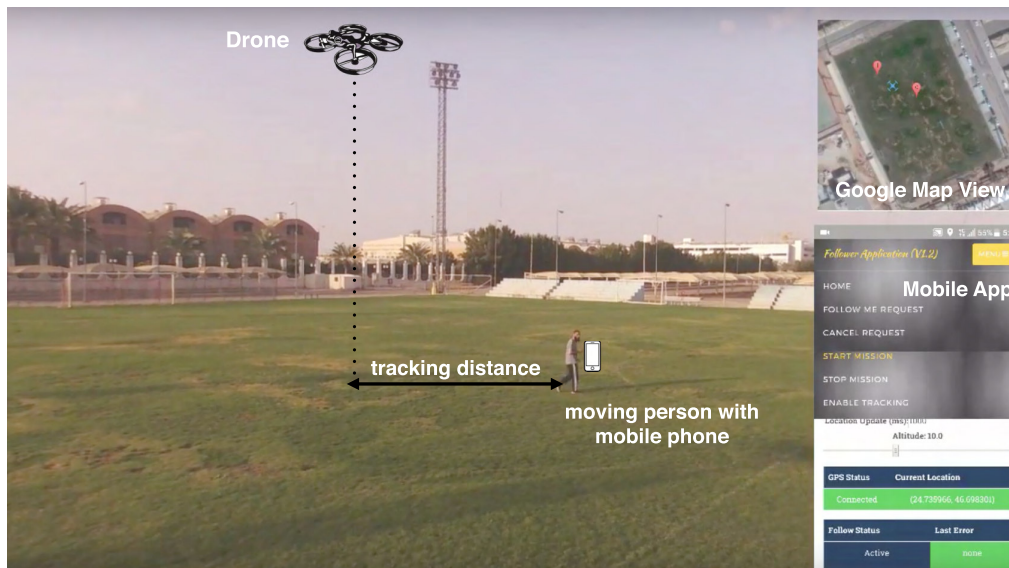


FIGURE 8. Experimental environment of Scenario 1: Football Field of Prince Sultan University with a real drone.

1) SCENARIO1: WALKING PERSON IN A FOOTBALL FIELD USING A REAL DRONE

The football field is located at Prince Sultan University and covers an area of 120 meters by 90 meters. The scenario consists of a person walking and running randomly in the football field while the drone tracks him during his movement. The real drone connects to the Dronemap Planner cloud using a HUAWEI E5775 LTE Portable Router WiFi router operating with 3G/4G connection. The smart phone device used in the experiments with the follower client application also used a 4G connection to communicate with the Dronemap Planner cloud. The user follower client application is installed onto an Android smart phone and is configured to read the location of the user from GPS device in a pre-defined time interval. Meanwhile, in the background, we monitor the experimental process of the tracking application through Dronemap Planner web client application using a MAC Book PRO laptop connected to Dronemap Planner cloud service through a WiFi router with 3G/4G connection. This assists in having a visual validation of the tracking process in real-time. A video demonstration of the scenario 1 is available at [26]. In order to analyze the effect of delay for intermittent communication, we induce a controlled and temporary service disconnection between the mobile phone and the drone. The experimental environment of Scenario 1 is depicted in Figure 8.

2) SCENARIO 2: WALKING PERSON IN CITY QUARTER USING A SIMULATED DRONE

This scenario utilizes a simulated drone to track the motion of the walking person using DroneTrack. In this scenario, a user walks into a residential district area in the city and initiates contact with a drone using the Dronemap Planner Cloud service. The user starts a tracking mission as can be seen

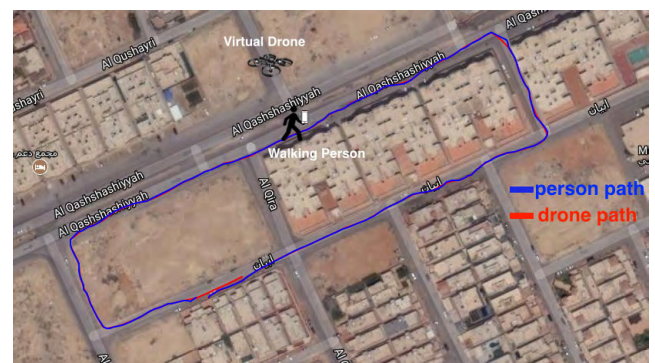


FIGURE 9. Experimental environment of Scenario 2: paths followed by the walking person and the drone over 880 m distance.

in Figure 4. Figure 9 shows the experimental environment of Scenario 2 as well as depiction of the various paths traversed by the walking person. The trajectory followed by the walking person is $350\text{m} \times 90\text{m}$ resulting in a total traveled distance of 880 meters.

The simulated drone is executed with the Ardupilot Simulation-In-The-Loop (SITL) simulator [27]. The simulated drone is connected to the Dronemap Planner cloud hosted on our DreamCompute cloud instance. The rest of the environment settings are the similar to Scenario 1. The only notable difference is the connectivity, the simulated drone is connected to the Internet through a high-speed Internet connection (40 Mbits/sec) through optical fiber network, instead of the 4G connection in Scenario 1. It guarantees a more reliable communication channel between the drone and the cloud with reliable connectivity. Meanwhile, the walking person connects to the Cloud service using a 3G connection on a mobile WiFi router.

TABLE 2. Statistical results of the tracking distance (meters) and speed (Km/h).

	Average Distance (m)	Distance Std Dev	Coeff of Var of Distance	Average Speed (Km/h)
Scenario1 (Tracking Window run)	4.5652	3.9589	0.8672	6.275
Scenario 2 (all runs)	3.5622	0.9935	0.2789	4.5324
Scenario 3 (run1)	14.4080	7.7187	0.5357	17.6055

**FIGURE 10.** Experimental environment of Scenario 3: paths followed by the moving car and the drone over 3 Km distance.

3) SCENARIO 3: MOVING VEHICLE WITH IN CITY QUARTER USING A SIMULATED DRONE

In order to evaluate the impact of speed and acceleration on the tracking quality, we conducted a set of experiments involving a moving vehicle using a simulated drone. All the environment settings are the similar to the settings presented in Scenarios 1 and 2. The moving vehicle is allowed to traverse a total length of about 3 Km. Scenario 3 is illustrated in Figure 10. Identical to the Scenario 2, the simulated drone is connected to the Internet through a high-speed Internet connection (40 Mbits/sec) through optical fiber network. The smart phone device with the follower application was placed inside the car with reliable GPS signal and Internet connection. The connectivity to the Internet is provided using the 3G cellular network. The vehicle was driven at different speeds and variations of acceleration to investigate its impact on the tracking quality of DroneTrack in terms of distance and delay. Figure 15 shows the four different runs of Scenario 3: two runs with low speed and low acceleration, and two runs with high speed and high acceleration. We focus on run1 and run2 considering their similarity to run3 and run4.

C. DATA COLLECTION PROCESS

We collect the experimental data during the execution of the tracking mission into log files for offline analysis. The data collected using the Dronemap Planner Cloud service which is used by the DroneTrack follower application. For every new GPS location received at the DroneTrack from the client application, the DroneTrack records the following

data into a specific log file: (1) the latitude and longitude GPS locations for user and the drone, (2) the altitude of the drone, (3) the long-term link quality, the short-term link quality, and the intercommunication between the drone and the cloud, (4) the ground and air speed of the drone. A log file is created for every single mission created by the user. It has to be noted that the Dronemap Planner cloud has a global view and knowledge of all data and statuses related to the walking person and the drone. The link quality is defined as packet reception ratio (PRR), which is the number of packets correctly received divided by the total number of sent packets. The long-term PRR is based on collection of all packets sent and received. The short-term PRR is based on packets sent and received over a certain period of time window. experimentation, we used 10 seconds as time window size to account for the link quality in short-term. Data collected was processed and analyzed using MATLAB. Results and observations are presented in the next section.

D. EXPERIMENTAL ANALYSIS AND RESULTS

In this section, we analyze and compare the performance of DroneTrack for the three scenarios in terms of delay and tracking accuracy. The diversity of the scenarios enables to provide a better understanding of the experimental performance for different configurations, including *i*) real drone versus simulated drone and *ii*) moving person (low speed and acceleration) versus moving car (high speed and acceleration).

1) IMPACT OF THE SPEED ON THE TRACKING DISTANCE

As mentioned in the worst-case delay analysis Network Calculus model presented in the previous section, the tracking distance is affected by the speed of the moving object. Figures 11 to 16 illustrate this dependency for all three scenarios. Table 2 also provides the average, standard deviation and coefficient of variation of the tracking distance for the three scenarios.

It can be observed from Run1 in Scenario 3 involving a moving vehicle, the average distance is more than four times larger than the average distances of the two Scenarios 1 and 2 with the walking person. We also observe that the tracking distances are also proportional to the average speeds of the moving targets. These real-time results and observations prove the correctness of the deterministic delay analysis presented in previous section as determined by the Equation 4. These results demonstrate the existence of the correlation between the speed and the distance of the user with the drone. In addition, the coefficient of correlation between the average distances and the average speeds of

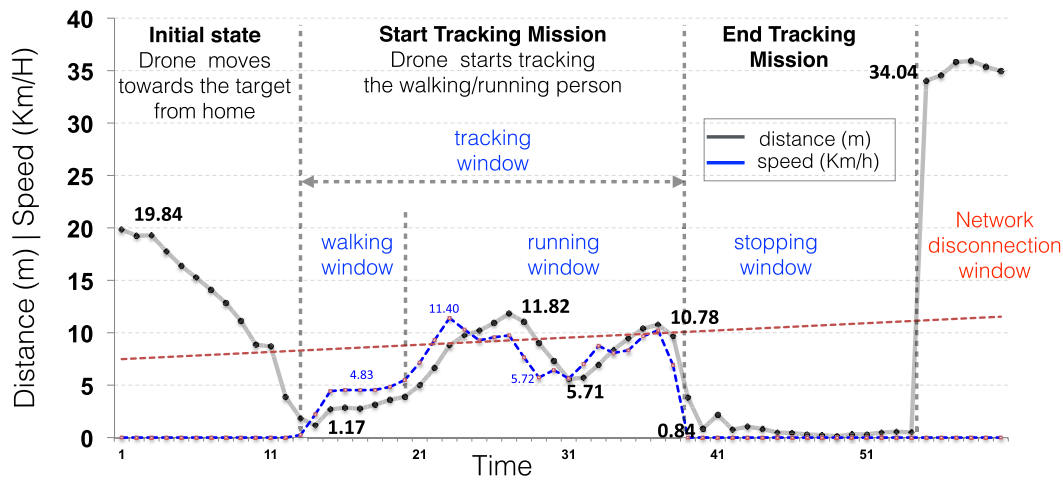


FIGURE 11. Scenario 1: tracking distance versus time.

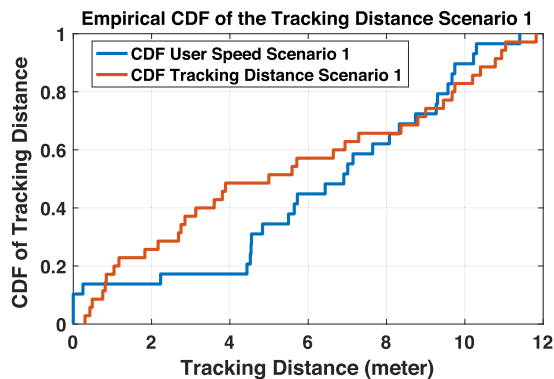


FIGURE 12. Scenario 1: CDF of the tracking distance.

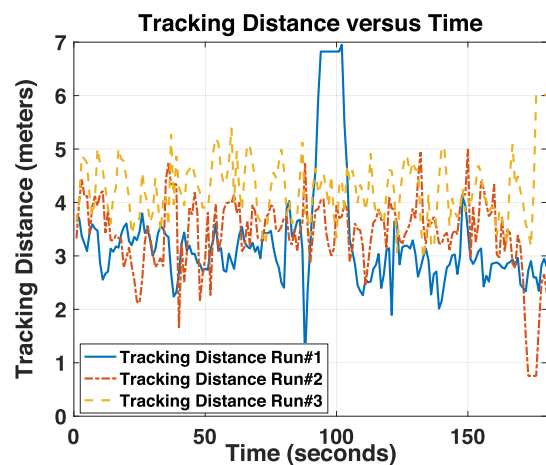


FIGURE 14. Scenario 2: tracking distance versus time.

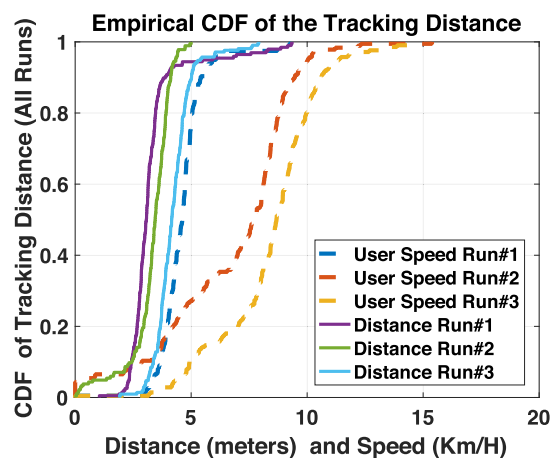


FIGURE 13. Scenario 2: CDF of speed vs distance.

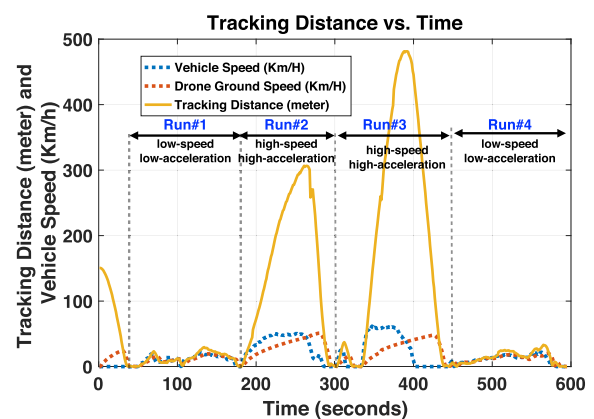


FIGURE 15. Scenario 3: tracking distance vs. time.

the three scenarios is equal to 0.99, highlighting the strong correlation between both. Figure 13 and Figure 14, show the empirical CDF of the tracking distance for the 3 runs in scenario 2 as well as the tracking distance versus time relationship. This concurs with the observation depicting the empirical cumulative distribution function (CDF) and the temporal curve, respectively, for the person walking speed and the

tracking distance of Scenario 2. Figure 9 shows the three different segments of the path shown in scenario 2. In Figure 13, the solid lines represent the CDF of the tracking distance and the dashed lines represent the CDF of the person speed. The speed of the user was estimated through numerical analysis for the moving object's GPS locations that were recorded in

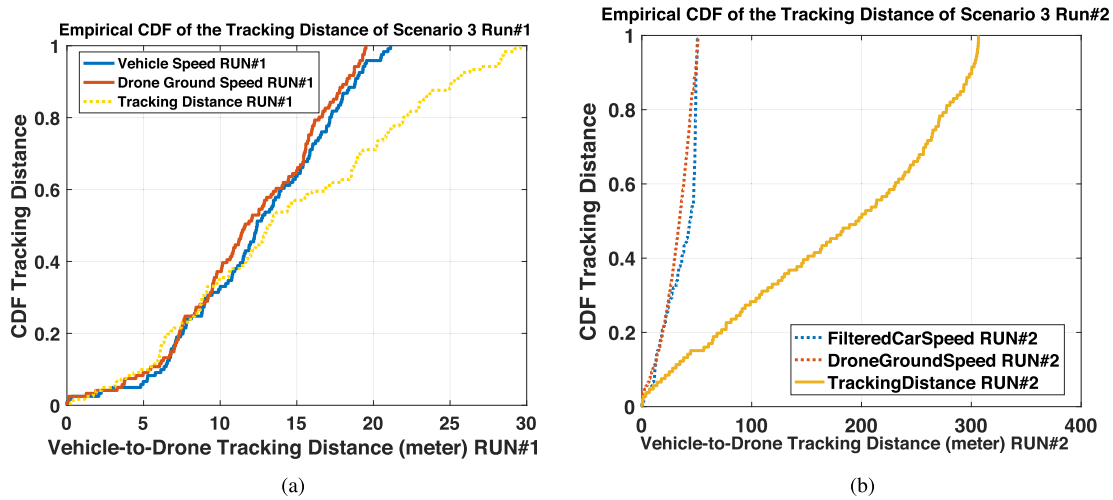


FIGURE 16. Scenario 3: tracking distance vs. time. (a) Scenario 3 Run1: tracking distance vs. time run1. (b) Scenario 3 Run2: tracking distance vs. time run2.

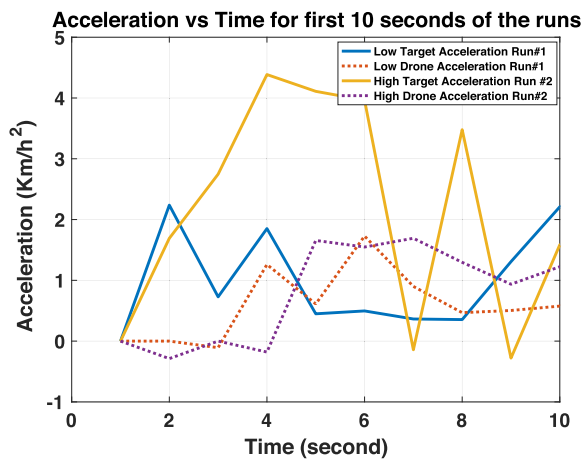


FIGURE 17. Scenario 3: acceleration vs. time for 10 first seconds.

the log file. In the worst case, with speeds lower than 5 Km/h, 80% of the tracking distance remains under 5 meters. Based on Equation 4, the maximum network delay is 3.5 s in 80% of the cases. This network delay includes the network latency T , which depends on the speed and acceleration of the target object. We further note that the tracking distances in run2 and run3 are higher than that of run1, because the latter run has slower average speed.

Figure 11 shows the tracking window of Scenario 1 involving a walking person and a real drone. The relation between speed and distance can be clearly observed when comparing the blue curve of the speed and the black curve of the tracking distance. At first, the person starts walking in the football field, he abruptly starts running for a short period and then resumes walking. The running window in Figure 11 shows an increase of the distance up to 11.82 meter, then abruptly, the distance decreases to 5.71 meters as soon as the moving person starts walking back, finally the distance goes back to 10.78 meters when running. The gap between the two curves represents the network delay as the drone will respond to

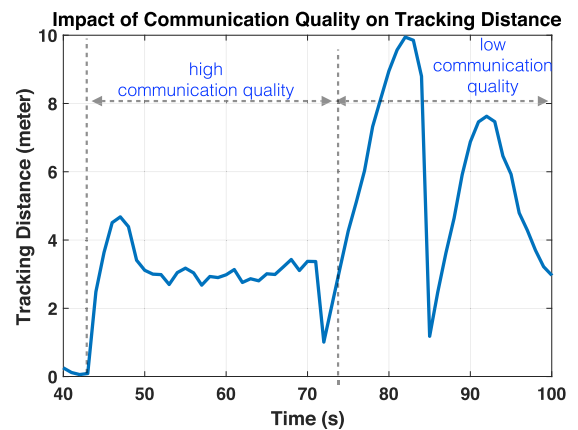


FIGURE 18. Impact of communication quality.

new location and move towards it as soon as a new GPS location is received by the drone. The coefficient of variation of the tracking distance is as high as 0.86 which is due to the variation of the speed in walking and running during the experiments. The video demonstration of this scenario is available at [26].

The results of the CDF of Scenario 2 as shown in Figures 13 and 12, in addition to Table 2, show that the tracking distance with a real drone is comparable to the one observed with a simulated drone.

2) IMPACT OF ACCELERATION ON THE TRACKING DISTANCE Figures 16a and 16b illustrate the CDF of the tracking distances for a vehicle in scenario 3. (run1) depicts the movement of vehicle with low speed and low acceleration, where as (run2) shows this movement for higher speed and acceleration.

In this scenario, the maximum speed of the drone did not exceed the maximum speed of the vehicle, which is 40 Km/h; however, the drone lost tracking accuracy and the tracking distance reached up to 300 meters. This increase of the track-

ing distance is mostly related to the difference between the drone acceleration and the vehicle acceleration rather than the impact of the network delay. In fact, Equation 3 of the maximum delay is composed of two factors (1) the network delay expressed as b/R , and (2) the latency T , which is the maximum period of time needed for the system to start reacting. In particular, the latency between the target and the cloud T_{cloud} is affected by the acceleration of the target (and thus the speed), and also the frequency of the GPS location updates. In case of high acceleration, the vehicle will move faster to new locations, which will be transferred to the drone only after a certain delay that includes the network delay b/R and other latencies $T_{processing}$ and T_{drone} . The faster the acceleration is, the larger the tracking distance will be due to slower response of the drone to new locations updates.

Another reason of this tracking distance increase is the difference in the acceleration of the vehicle and the drone. This is demonstrated in Figure 17, which shows the acceleration of the vehicle and the drone for the two runs during the first starting 10 seconds. In *run1* with low acceleration and low speed, the *difference of acceleration* between the vehicle and the drone was high and positive for the whole 10 seconds reaching a maximum of 3.0951 Km/h^2 and a average of 0.1472 Km/h^2 . In *run1* with low acceleration and low speed, the *difference of acceleration* between the vehicle and the drone was high and positive for the whole 10 seconds reaching a maximum of 4.5657 Km/h^2 and a average of 0.7223 Km/h^2 . This illustrates a limitation to use low-cost drones over the Internet to track target objects with high acceleration and speed. The use of powerful drones with high acceleration capabilities will mitigate the responsiveness problems and thus reduce the tracking accuracy.

3) IMPACT OF COMMUNICATION QUALITY

Communication quality plays an important role in the performance of the follower application over the cloud, this can be observed in Figure 18. In the time window 40s to 70s seconds with reliable communication quality, the tracking distance of the walking person in Scenario 2 is stable for a maximum distance of 3 meters confirming previous results. In the time window 70s to 100s, the 3G communication was temporarily unstable while going between buildings reflecting in increase of the tracking distance to a maximum of 10 meters.

Furthermore, as can be seen in Figure 14, we observe the same effect where the peak in the tracking distance of *run1* (solid blue) increases up to 7 meters. This is due to temporary loss of communication between the user and the DroneTrack. It can be seen that the system resumes normal and smooth tracking after a maximum delay of 5 seconds. Results in Table 1 confirm the smooth tracking behavior for a walking person scenario as the coefficient of variation remains as low as 0.27 meter.

Figure 11 also illustrates the impact of communication loss on the tracking accuracy. During the time window 41-51 seconds, the communication between the user and the cloud

stopped and thus no GPS updates were sent, consequently, the distance reaches 34 meters.

VII. DISCUSSIONS AND LESSONS LEARNED

In this paper, we proposed, DroneTrack, a cloud-based system for the real-time tracking of GPS location aware moving objects using Unmanned Aerial Vehicles (UAVs). The contributions of this work consist in the design, development, experimentation, and performance evaluation of the DroneTrack, a follower application, over the Internet, using the cloud-based Dronemap Planner drone management system. Extensive experimental study was conducted to validate the functioning of the system and evaluation of its performance and limitations. We considered three different scenarios with real and simulated drones following moving object and vehicle for various parameters under different situations. The experimental results provide a clear understanding on the advantages and limitations of the DroneTrack system, and provide a proof-of-concept of its effectiveness and feasibility under pre-defined operation conditions. We observe that the performance of DroneTrack is Dependant on seamless connectivity, variations in low speed and low acceleration of moving objects. These factors are rather dependent on the limitations and capabilities of the individual drones rather than the DroneTrack system's performance.

A salient design feature of the DroneTrack system lies in the adoption of a service-oriented architecture, that enables real-time tracking of target objects based on GPS locations using Web services and Websockets technologies. It demonstrates the potential of applicability of the design principles to similar technologies involving cloud robotics in addition to providing new robotics services through the Internet. We conclude that with the seamless connectivity between the drones, cloud, and users, the DroneTrack system can be reliably used to track moving targets anywhere and anytime regardless of communication range limitations between the target and the drone. We intend to focus on improving the tracking accuracy to less than 1 meter, this will require optimization of the process involving, reduction in processing delay in the cloud, network delays and the responsiveness of the drone to higher speed and accelerations.

Several lessons have been learned from this evaluation study, which are summarized as follow. First, the effectiveness of the cloud-based tracking system is demonstrated. However, the tracking accuracy remains with in the range of 3 to 4 meters scenarios involving a person moving at a walking pace with good communication quality using a low-cost drone. It is desirable to increase the tracking accuracy to a range of approximately 1 meters. This could however be achieved by reducing the network delay through end-to-end seamless and reliable communication.

Second, as demonstrated from our experiments, the effectiveness of the tracking system appears to be less relevant in scenarios with high speeds such as was observed in scenarios involving moving vehicles. The cumulative effect of network delay and the wide gap in acceleration introduced by sudden

change in speed affects the tracking accuracy of the DroneTrack system.

Third, reliable functioning of the DroneTrack is dependent on three important factors; (i) The accuracy of the GPS localization parameters: erroneous GPS location coordinates results in increased gap between the user and the drone. This sudden change in location can hinder the movement of the drone on the required trajectory until the GPS signal is stabilized, (ii) The frequency of update of new locations of the moving object. In this experiments, we used 1Hz frequency, and this can be further improved by sending more locations updates. However, we are investigating how fast the Google Geolocation API is able to provide new location updates through GPS. (iii) The communication quality between the user and the cloud: in this experiment we had a fluid communication between the user and the cloud. In further work we will investigate in more depth the impact of the latency in communication on the tracking quality.

REFERENCES

- [1] Markets and Markets. (Nov. 2016). *Unmanned Aerial Vehicle (UAV) Market, by Application*. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/unmanned-aerial-vehicles-uav-market-662.html?gclid=EALalQobChMIouHLwZXs2QIV7grTCh1AwDkEAAAYASAAEgKVx_D_BwE
- [2] J. Kuffner, "Cloud-enabled robots," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Jun. 2010, pp. 518–523.
- [3] R. Chaâri et al., "Cyber-physical systems clouds: A survey," *Comput. Netw.*, vol. 108, pp. 260–278, Oct. 2016.
- [4] Y. Chen, Z. Du, and M. García-Acosta, "Robot as a service in cloud computing," in *Proc. 15th IEEE Int. Symp. Service Oriented Syst. Eng. (SOSE)*, Jun. 2010, pp. 151–158.
- [5] R. Arumugam et al., "DAvinCi: A cloud computing framework for service robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2010, pp. 3084–3089.
- [6] M. Waibel et al., "RoboEarth," *IEEE Robot. Autom. Mag.*, vol. 18, no. 2, pp. 69–82, Jun. 2011.
- [7] M. K. Ng et al., "A cloud robotics system for telepresence enabling mobility impaired people to enjoy the whole museum experience," in *Proc. 10th Int. Conf. Design Technol. Integr. Syst. Nanoscale Era (DTIS)*, Apr. 2015, pp. 1–6.
- [8] A. Koubaa, M. Alajlan, and B. Qureshi, "ROSLink: Bridging ROS with the Internet-of-Things for cloud robotics," in *Springer Book of Robot Operating System (ROS)*, vol. 2. Springer, 2017.
- [9] A. Koubaa, B. Qureshi, M. F. Sriti, Y. Javed, and E. Tovar, "A service-oriented cloud-based management system for the Internet-of-drones," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, Apr. 2017, pp. 329–335.
- [10] Z. Du, L. He, Y. Chen, Y. Xiao, P. Gao, and T. Wang, "Robot cloud: Bridging the power of robotics and cloud computing," *Future Generat. Comput. Syst.*, vol. 74, pp. 337–348, Sep. 2017.
- [11] A. K. Bozcuoglu and M. Beetz, "A cloud service for robotic mental simulations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 2653–2658.
- [12] C. Huang, L. Zhang, T. Liu, and H. Y. Zhang, "A control middleware for cloud robotics," in *Proc. IEEE Int. Conf. Inf. Autom. (ICIA)*, Aug. 2016, pp. 1907–1912.
- [13] B. Hu, H. Wang, P. Zhang, B. Ding, and H. Che, "Cloudroid: A cloud framework for transparent and QoS-aware robotic computation outsourcing," in *Proc. IEEE 10th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2017, pp. 114–121.
- [14] J. Wan, S. Tang, Q. Hua, D. Li, C. Liu, and J. Lloret, "Context-aware cloud robotics for material handling in cognitive industrial Internet of Things," *IEEE Internet Things J.*, to be published.
- [15] N. Tian et al., "A cloud robot system using the dexterity network and Berkeley robotics and automation as a service (brass)," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 1615–1622.
- [16] C. Reid, B. Samanta, and C. Kadlec, "Network performance of wireless cloud-based robots with local processing," in *Proc. SoutheastCon*, Mar. 2017, pp. 1–6.
- [17] Y. Li, H. Wang, B. Ding, P. Shi, and X. Liu, "Toward qos-aware cloud robotic applications: A hybrid architecture and its implementation," in *Proc. Int. IEEE Conf. Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart World Congr. (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Jul. 2016, pp. 33–40.
- [18] M. Gharibi, R. Boutaba, and S. L. Waslander, "Internet of drones," *IEEE Access*, vol. 4, pp. 1148–1162, 2016.
- [19] L. Apvrille, T. Tanzi, and J.-L. Dugelay, "Autonomous drones for assisting rescue services within the context of natural disasters," in *Proc. Gen. Assembly Sci. Symp. (URSI GASS)*, 2014, pp. 1–4.
- [20] S. Bitam and A. Mellouk, "ITS-cloud: Cloud computing for Intelligent transportation system," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2012, pp. 2054–2059.
- [21] B. Bona, "Advances in human robot interaction for cloud robotics applications," Polytech. Univ. Turin, Turin, Italy, Tech. Rep., 2016.
- [22] G. Ermacora et al., "A cloud robotics architecture for an emergency management and monitoring service in a smart city environment," Polytech. Univ. Turin, Turin, Italy, Tech. Rep., 2013.
- [23] E. Yanmaz, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter, "Drone networks: Communications, coordination, and sensing," *Ad Hoc Netw.*, vol. 68, pp. 1–5, Jan. 2018.
- [24] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Berlin, Germany: Springer-Verlag, 2001.
- [25] A. Koubaa and B. Andersson, "A vision of cyber-physical Internet," in *Proc. Workshop Real-Time Netw.*, Jul. 2009, p. 2.
- [26] *Video Demonstration of the Follower Application the Football Field of Prince Sultan University With a Real Drone*. Accessed: Mar. 14, 2018. [Online]. Available: <https://www.youtube.com/watch?v=9a8Gn0BDe8U>
- [27] *ArduPilot Simulation in the Loop (SITL)*. Accessed: Mar. 14, 2018. [Online]. Available: <http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html>



ANIS KOOBAA is currently a Professor of computer science and the Director of the Robotics and Internet of Things Research Lab, Prince Sultan University. He is also a Senior Researcher with CISTER/INESC and ISEP-IPP, Porto, Portugal, and a Research and Development Consultant with Gaitech Robotics, China. His current research interests include providing solutions towards the integration of robots and drones into the Internet of Things (IoT) and clouds, in the context of cloud robotics, robot operating system, robotic software engineering, wireless communication for the IoT, real-time communication, safety and security for cloud robotics, intelligent algorithms design for mobile robots, and multi-robot task allocation. He is also a Senior Fellow of the Higher Education Academy, U.K. He has been the Chair of the ACM Chapter, Saudi Arabia, since 2014.



BASIT QURESHI (M'02) received the B.Sc. degree in computer science from Ohio University, Athens, OH, USA, in 2000, the M.Sc. degree in computer science from Florida Atlantic University in 2002, and the Ph.D. degree in computer science from the University of Bradford in 2011. He is currently with the College of Computer and Information Science, Prince Sultan University. His research interests include trust, security, and privacy issues in wireless networks, robotics, and smart cities applications. He is a member of the IEEE Computer Society, the IEEE Communications Society, and the ACM.

...