Software Architecture for Autonomous UAVs for Power line Inspection

Liping Shi,
Néstor J. Hernández Marcano,
Rune Hylsberg Jacobsen
Department of Engineering, Aarhus University
{liping,nh,rhj}@eng.au.dk

Golizheh Mehrooz,
Emad Ebeid,
Peter Schneider-Kamp
University of Southern Denmark
{mehrooz,esme,petersk}@sdu.dk

I. Introduction

Solutions based on the Unmanned Aerial Vehicle (UAV) technologies are increasingly deployed in many industrial use cases [1], e.g. infrastructure inspection, search and rescue operation, and transportation. To fully use the advantages of UAVs and system optimizations, e.g. efficiency and cost, software architecture needs to be investigated in terms of flexibility, reliability and scalability. Besides, the target use case will influence the architecture with specific application requirements. In this paper, we address the software architecture for power line inspection using UAVs. The power line inspection task is usually operated by a pilot with a remote-controlled UAV. Collected inspection images are manually evaluated. The purpose of the proposed software architecture is to enable power grid infrastructure operators on inspection tasks using autonomous technologies for human risk minimization and cost reduction. The software will provide two main automated services 1) UAV inspection and 2) inspection data analysis. The software architecture aims to support the whole service lifecycle from developing phase to operation phase with the ability of continuous improvements. This paper contains a presentation of the work-in-progress on our software architecture design inspired by microservices architectures [2].

II. Software architecture

The concepts of Microservices and Containers as a Service (CaaS) are selected for the software architecture. Microservices are used to structure UAV-based applications as a collection of loosely coupled services. Decomposing an application into smaller services improves modularity, makes the application easier to develop as well as easing independent test in parallel by multiple developing teams. Besides, the proposed architectural supports continuous delivery and deployment. Containers allow a logical packaging mechanism in which services will be abstracted from the deployed environment. This provides a consistent environment including software dependencies, specific programming language runtimes, libraries, etc. In addition, containers virtualize hardware and network resources. This provides isolated environments for target services and enables assignment of resources, e.g. CPU and GPU, with considerations of system and service. Fig. 1 presents the proposed software architecture for UAV-based

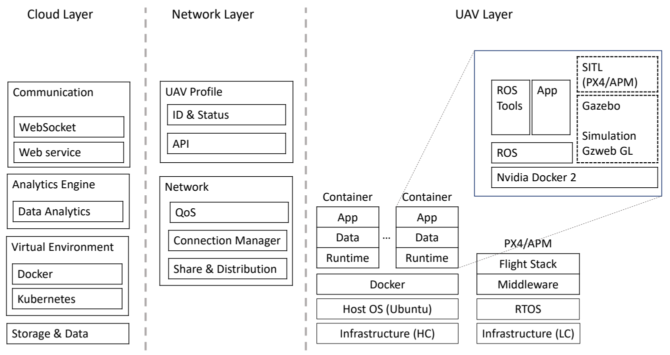

Fig. 1. Software Architecture: Containers as a Service is used to form the software application while assigning resources.

power line inspections. The layers used in our proposed architecture are described in the following sections.

A. Cloud Layer

Cloud computing and the Internet of Things (IoT) have become two closely connected future internet technologies with one providing a platform for the success of the other. As it is shown in Fig. 1, the cloud software design is split fourfold. The first layer is the communication layer between the drone and the cloud which uses web sockets and web services. The second layer is the analytics engine, which is used for data analysis. The third layer is the virtual environment, which is utilized for creating the Kubernetes cluster and Docker images, and the fourth layer is used for data storage such as saving images.

1) Communication: In the context of cloud robotics many researchers and engineers are attempting to integrate robots with the cloud through a network. There exist two types of protocols for interconnection with devices [3]. These protocols are publish-subscribe based messaging protocol such as Advanced Message Queuing Protocol (AMQP) and a request-reply based messaging for client-server applications such as Hypertext Transfer Protocol (HTTP). HTTP is a message protocol based on Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). However, AMQP is a message protocol based on the TCP. AMQP makes it easy to establish connections between multiple devices, which is a good solution for IoT applications.

2) Analytics engine: UAVs have limited energy, which prevents them from running computationally intensive machine learning algorithms onboard. Therefore, we need to develop a cloud-based solution for a fault detection algorithm. Machine learning algorithms such as YOLO (You Only Look Once) and R-CNN (Regions Convolution Neural Network) together with the deep neural network for image classification such as RestNet are performed in this layer for object detection during power line inspection.

3) Virtual environment: A CaaS cloud infrastructure is built based on the Docker and the Kubernetes concepts. All of the application code such as libraries and dependencies are packed into a container. A Kubernetes cluster [4] includes container applications and services are defined by YAML file. To automate create and update pods in the Kubernetes cluster, Jenkins is selected as a Continuous Integration (CI) tool. There are several reasons for choosing Jenkins such as it is user-friendly, easy to install, does not require additional installations or components, and it is free and open source.

4) Data Storage: This layer is used as a repository for saving our data such as inspection images in the cloud.

B. Network Layer

The network layer manages network connectivity of individual UAVs. The data of each UAV will be packaged and abstracted with a unique identifier (ID). The mission planning and operations, information gathering and sharing among UAVs will be coordinated in this layer. For maintaining reliable operation, it evaluates the communication environment and status in real-time. Further network architecture and communication protocols will be addressed in future work.

C. UAV Layer

The UAV layer contains modules that support autonomous UAV inspection services on both practical environments and simulated platforms. It contains a high-level controller (HC) and a low-level controller (LC), which are designed to be deployed on separated hardware. The LC deals with standard flight control by using the PX4 autopilot (PX4) or ArduPilot Mega (APM) flight stack, middleware and Real-Time Operating System (RTOS). The HC provides autonomous flight and inspection services. The host system of HC uses Ubuntu Linux, which is compatible with the Robot Operating System (ROS). Docker is deployed on top of the host OS for providing a cross-platform, isolated and consistent developing environment. Containers provide different services with specific hardware resource assignments, according to services requirements on computing power, latency and memory. With the support of the Nvidia-docker2 container runtime, GPU accelerated containers can be used for the service that requires intensive image processing, e.g. Simultaneous Localization And Mapping (SLAM). On top of the runtime, there is a ROS, e.g. ROS Kinetic Kame, which provides libraries and tools to create robot functions. ROS2 is a promising technology for multi-UAV collaboration, because of its distributed architecture and improvements on network connectivity. For the simulation of UAV inspection operations, Gazebo has been adopted (Fig. 2). Challenges on path planning, synchronization and swarming will be further investigated. For providing parallel, isolated, and flexible simulations, we are also building a cloud-based simulation platform [5] using container packaged simulation engine and web client for Gazebo (Gzweb).

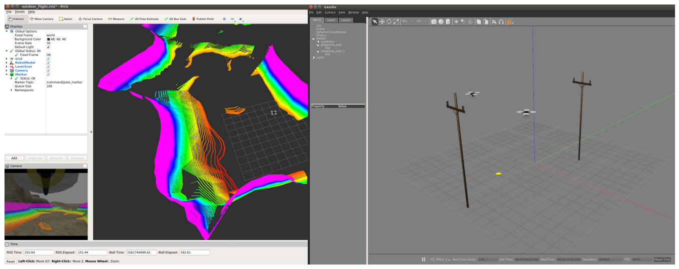

Fig. 2. RViz (ROS visualization) and Gazebo for UAV simulations

III. Future Plans and Challenges

We are working on architecture validation. On the cloud layer, Kubernets and Jenkins are being integrated for automated deployment. For data analytics, we are investigating structures for training machine learning algorithms with limited labeled data. On the network layer, the structure needs to balance the complexity of the protocol and the requirements of the connection. Organizing methods [6] including centralized, decentralized, cloud-based and UAV-based structure will be evaluated. On the UAV layer, challenges come from software functions management subject to different tasks, priorities and hardware resources.

IV. Acknowledgment

This research was supported by the Innovation Fund Denmark, Grand Solutions, under grant agreement No. 8057-00038A Drones4Energy project. https://drones4energy.dk/.

References

[1] L. Shi, N. J. H. Marcano, and R. H. Jacobsen, “A Survey on Multi-Unmanned Aerial Vehicle Communications for Autonomous Inspections,” Euromicro DSD/SEAA conference, August 2019.

[2] A. Balalaie, A. Heydarnoori, and P. Jamshidi, “Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture,” *IEEE Software*, vol. 33, no. 3, pp. 42–52, May 2016.

[3] S. Kim, H. Choi, and W. Rhee, “Iot home gateway for auto-configuration and management of mqtt devices,” in *2015 IEEE Conference on Wireless Sensors (ICWiSe)*, August 2015, pp. 12–17.

[4] G.Mehrooz, E.Ebeid, and P.Schneider Kamp, “System design of an open-source cloud-based framework for internet of drones applications,” Euromicro DSD/SEAA, August 2019.

[5] M. Schmittle and et al., “OpenUAV: A UAV Testbed for the CPS and Robotics Community,” in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, April 2018, pp. 130–139.

[6] S. Mahmoud and N. Mohamed, “Collaborative UAVs Cloud,” in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, May 2014, pp. 365–373.