Communication Infrastructure and Cloud Computing in Robotic Vessel as-a-Service Application

Cosmin Delea[♯], Emanuele Coccolo[‡], Salvador Fernandez Covarrubias[♯], Filippo Campagnaro[‡], Federico Favaro[‡], Roberto Francescon[⋆], Vincent Schneider[♯], Johannes Oeffner[♯], Michele Zorzi[‡]

[‡] Department of Information Engineering, University of Padova, via Gradenigo 6/B, 35131 Padova, Italy,
[♯] Fraunhofer Center for Maritime Logistics and Service, Am Schwarzenberg-Campus 4, 21073 Hamburg, Germany,
[⋆] Wireless and More srl, Via della Croce Rossa 112, 35129 Padova, Italy
[‡]{coccoloe,campagn1,zorzi}@dei.unipd.it, fedefava86@gmail.com
[♯]{cdelea,sfernandez,vfocke,johannes.oeffner}@cml.fraunhofer.de
[⋆]roberto.francescon@wirelessandmore.it

***Abstract*—The current trend in robotics is to enhance robustness against uncertainties through complex modern control methods, applied to either single- or multi-agent systems. While indeed these approaches provide noticeable performance improvements, their implementation requires, among others, enclosed environments, where the operator has full access to the control systems of the robot during execution. With constant improvement of the Internet of Things, real-time systems needed different software architectures and communication systems to enable their remote operation, with the same degree of flexibility when operating the systems in-situ. The paradigm shifted towards cloud computing, which moved the computational effort needed for various real-time processing algorithms on the server-sided application and focused on having lightweight interfaces between components, with the exception of the lower-level control loops, which remained within the processing units of the robotic systems. This paper describes the software architecture and communication system for a state-of-the-art service to enable, support and monitor different robotics applications. The case study presents all features of a small-sized but scalable robotics application, having multiple challenges when building a service oriented platform: autonomous robots, over- and underwater long-distance communication, sensor fusion and command and control of multiple users. The goal is to create a centralised cloud computing environment, with decentralised microservices and redundant resources, supporting a plethora of port-specific operations conducted with the help of waterborne robotics with different equipment configurations, that can be remotely operated and monitored in real time over cross-platform, lightweight applications.**

I. Introduction

A. Overall goal and concept

The usage of waterborne robotics for completing specific tasks in port and coastal areas is gaining momentum, with different levels of autonomy being accomplished by autonomous surface and underwater vehicle (ASVs and AUVs). Furthermore, service-oriented architecture (SOA) and cloud computing are established infrastructure paradigms in the development of software systems [1]. Even in the maritime domain, frameworks such as the maritime connectivity platform were established in order to allow fast efficient, secure, reliable and seamless electronic information exchange between authorised maritime stakeholders [2]. Maritime robotic solutions require well established shore control systems which can be implemented by means of a centralised data exchange server allowing to set up modular testbeds with clients in different locations [3]. By deliberately exposing only the high-level control loops of the robotic system through performant communication protocols, such as web sockets, the environment for running the robotic application expands to web applications, as proven in [4]. Moreover, as described in [5] and in [6], a client-server architecture would focus the computational workload and resources into microservices, which can be supported on different computers (server) inside the application's network. This combined approach could leverage the computational loads from robotics systems when wanting to extend the software solution to accept multiple users and remote operations, thereby promoting scalability.

The Robotic Vessels as-a-Service (RoboVaaS) project aims at demonstrating innovative on-demand robotic aided services via autonomous surface vehicles (ASVs) as well as remotely operated underwater vehicles (ROVs). The novelty of the RoboVaaS project [7] is in showing the feasibility of offering various on-demand services in a port or coastal environment. Therefore, a profound framework was set up that allows reliable data transfer between above- and underwater entities, a shore based monitoring station and a real-time web-based user interface. Specifically, the project aims to exploit the most innovative communication technologies and waterborne robotics to improve shipping operations, offering on-demand and robotic-aided services. To enhance harbour activities, robotic vessels with different levels of autonomy will assist and complete various on-demand services. These services are usually performed by human operators or are not available at all times and require significant resources, such as time, money and availability. Within RoboVaaS five use-cases were

This work has been partially supported by the Italian Ministry of Education, Universities and Research (MIUR), the German Federal Ministry of Economy (BMWi) and ERA-NET Cofund MarTERA (contract 728053).

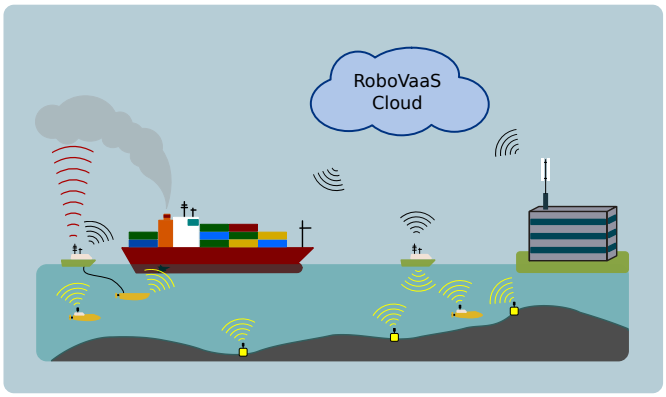

Fig. 1. RoboVaaS envisioned example scenario showing ship-hull inspection, anti-grounding and data collection services enabled through a fleet of ASVs and ROVs connected with acoustic underwater and wireless communication.

identified that are expected to have a high potential: inspection services of quay-walls (1) and ship hulls (2), a newly designed anti-grounding service (3), bathymetry data collection (4) and environmental data collection (5). The RoboVaaS project, as depicted in Figure 1, envisions a fleet of heterogeneous robotic nodes, each permanently connected to an onshore station via over- and underwater communication links and capable of completing a plethora of port-specific operations.

An envisioned user of the RoboVaaS system (such as a ship owner, a navigator or a port authority) can use one of the aforementioned on-demand services by placing specific requests and follow their progress through a standard web browser. Once the service request is issued, the service provider will assign the request in the form of a task or mission to a single or several available Autonomous Surface Vehicles (ASVs) or Remotely Operated Vehicles (ROVs), commanded and supervised by a shore-based control centre. During a mission, its progress and the acquired data are transmitted in real time or - in some cases - after post-processing, to the users. Once the mission is completed, the results of the service can be recalled by the user at any point in time, in order to visualise them through a web browser or to generate a report, based on the acquired data. This report contains different types of information depending on the service: a report of a quay wall inspection service will present the status and pictures of the quay wall along with coordinates of the most damaged areas. A ship hull inspection can provide information on the fouling status of the ship hull. It could alert the user if a specific area requires service due to unwanted marine growth or damage as well as provide the valuable information that hull cleaning is not yet required. In the data collection services, the entities would perform a cost-efficient autonomous survey of waterways. The processed bathymetry and/or environmental sensor data collected during the mission could be reviewed in the user portal. The anti-grounding service, described in [8], will provide real-time multi-beam echo-sounder data directly to the end-user. In this service, an ASV travelling ahead of a merchant vessel collects and sends bathymetry data in real time, which is subsequently used to display areas of danger due to insufficient water depth while considering the specific

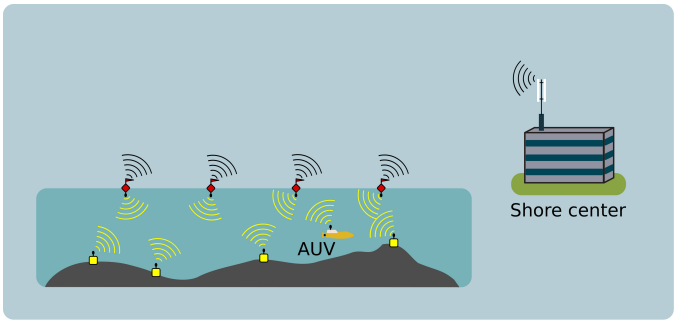

Fig. 2. RoboVaaS environmental data collection scenario: an AUV collects data from static underwater sensor nodes, and forwards the data to the surface vehicle connected to shore via WiFi.

draught of the trailing vessel. The distance and hence the lead time of the ASV depends on parameters such as minimum stopping distance of the merchant vessel.

This paper will describe in depth the system architecture, together with the communication interfaces and protocols used by the major subsystems, but will use the environmental data collection use-case to specifically describe the complete information workflow. In this scenario, we use the underwater polling protocol to gather underwater sensor information from floating buoys and transmit it to the shore station, where it can be visualised on the web user interface. This scenario will be a solid and sufficient basis for extending the overall concept to the other envisioned services. The aforementioned scenario is depicted in Fig 2.

B. *Past projects and related works*

Using as-a-service concepts for ports and coastal areas is a new trend offering a certain service on demand without the need to purchase the whole product entities involved. The Hamburg Port Authority and the Fraunhofer CML for example drafted a vision of "interconnected smart ports," which includes such services for port areas [9]. Currently, the Digital Ocean Lab (DOL), a core project of the large-scale project Ocean Technology Campus Rostock, will set up an undersea site for testing ideas and simulations under controlled conditions in a real-world environment, where efficient connection of different robotic entities will be demonstrated [10]. Combining unmanned systems for joint operations have been investigated in MORUS [11] by combining AUVs and unmanned aerial vehicles for security and environmental monitoring or in SWARMS [12] by combining AUVs, ROVs and ASVs to facilitate offshore operations.

The concept of robotics delivering services and being commanded through a web interface has been investigated in [13], the use-case being computer science education. With the aim of extending the tools provided by Robot Operating System-(ROS) compliant robotics, the Robot Web Tools project [6] created an open-source framework for communicating with such systems over web sockets, using *rosbridge* servers. With a robotics competition as use-case, another Robots-as-a-Service (RaaS) system has been developed in [14], which proposes web-based virtual machines for clients interacting with various ROS tools. Its successful implementation has led

to the development of ROS Developent Studio [15], which offers on-demand ROS tools, such as Gazebo Simulator, for users, with the computational workload left on the server-side application. As in the current work, the end-user requires only a web browser to access the whole array of services offered by the service provider. In [16], the concept of cloud robotics is extended to service robots, that assist the elderly and the disabled with daily activities, with the added benefit that the robots are permanently connected to the cloud, where developers can deliver their updates and operators can monitor and maintain them. The main differences from the past projects are the use-cases that are closely related to port-specific operations and the focus on waterborne robotics. Moreover, except for the anti-grounding use-case, all other services have been tested using both simulations and real testbeds.

C. Structure of the paper

The rest of the paper is organised as follows: In Section II we describe the system architecture, and how the RoboVaaS applications are managed. In Section III we present the details of the underwater and the above water network that enables the connectivity between all the RoboVaaS components. Section IV presents the results obtained with some preliminary tests performed with the whole system, and, finally, Section V concludes the paper.

II. System Architecture

A. Overall system, main actors and their interfaces

The overall system architecture, depicted in Figure 3, can be broken down into two main components: **Shore System** and **Robotic System**. The first entity hosts a web-based application, mainly served for all registered users having access to the RoboVaaS system with the content set by the **Shore System** considering the user roles. The **Shore System** is a collection of servers and client applications with different purposes, that enable a two-way communication channel with the **Robotic System**. The **Client Web Interface** is the actual web user interface from where different port-specific requests can be initiated, administered and supervised without the need for the users to be present, in the area where the port-specific service takes place. The **Shore Operations Centre** is another component of the **Shore System** and is the actor directly giving the higher level commands to the **Robotic System**. This link has the topmost priority in terms of communication and serves as an access point for the professional human intervention required, when operating the RoboVaaS system. Additionally, it can be coupled with a digital twin of the overall system, in order to perform a large-scale simulation. Lastly, the **Robotic System** represents the set of heterogeneous robots along with their interfaces to interact with the aforementioned actors.

B. Application management (users, live data management, data flow)

Within the RoboVaaS network, there are five users that can access the web application, each having different sets of permissions:

- client (or user): can place requests and visualise real-time data belonging to him/her.
- robotic vessel (or robot): can be mounted and commanded by the operator and emits real-time data.
- operator: can take up a job, select a robot to command and consume real-time data from the selected robot.
- system broker: can assign requests to an operator in form of a job.
- administrator: can modify roles and apps available for each user type.

Real-time information is handled through the RabbitMQ Broker (Cloud), which is capable of handling various messaging protocols, such as AMQP, MQTT or STOMP [17]. Each subsystem will have its own exchanges, which will enable the flow of information between actors. Moreover, during runtime they will be able to subscribe with their own queues to the exchanges of the other subsystems. Their rights will be granted by the **Back Server**, which can verify their identity and validate their requests. Multimedia data will be routed by the **Video-Streaming Server**, which collects multimedia packets sent over from the Robotic System and routes them to the web browser clients. Additionally, it will compress, decompress and transcode the video packets if clients cannot read the format encoded by the cameras by default. The main entities and their interfaces are depicted in Fig 4.

C. Software pipeline of the overall system

Each user will register and authenticate through a standalone application or through the browser-based client, which will send a request to the central **Back Server**, having exclusive reading and writing access to the **Database**. The request is executed by using a RESTful API. Once authenticated, the user will receive a unique user identification number (*userID*). The *userID* will be stored together with the credentials into the *Jobs* table and will be used whenever the user wants to access stored data. All data is first filtered out by *userID*, in order to prevent information disclosure. The connection between the back-end server and the source of the messages (i.e., robotic systems) is managed by a unique user with administration rights, called **System Broker**. The latter will match all incoming mission requests with an available robotic system (i.e. **USV**) and an **Operator**. The **Realtime-Data Server** will use this input to enable a communication channel between them, by either forwarding the messages to each other, or sending the channel details to the parties' back-ends. Once the mission is requested by the user, it will receive a unique job identification number (*jobID*), also known by the robotic system and the Operator, which will be used in turn to sign the generated data.

By decoupling the overall system in a series of decentralised microservices, with a common node for user authentication and data allocation, the system is capable of escalating to handle multiple users, missions and even applications. Once the user is registered into the overall system, all related information will be handled on a unique path, managed by the System Broker(s). The user can request multiple missions,

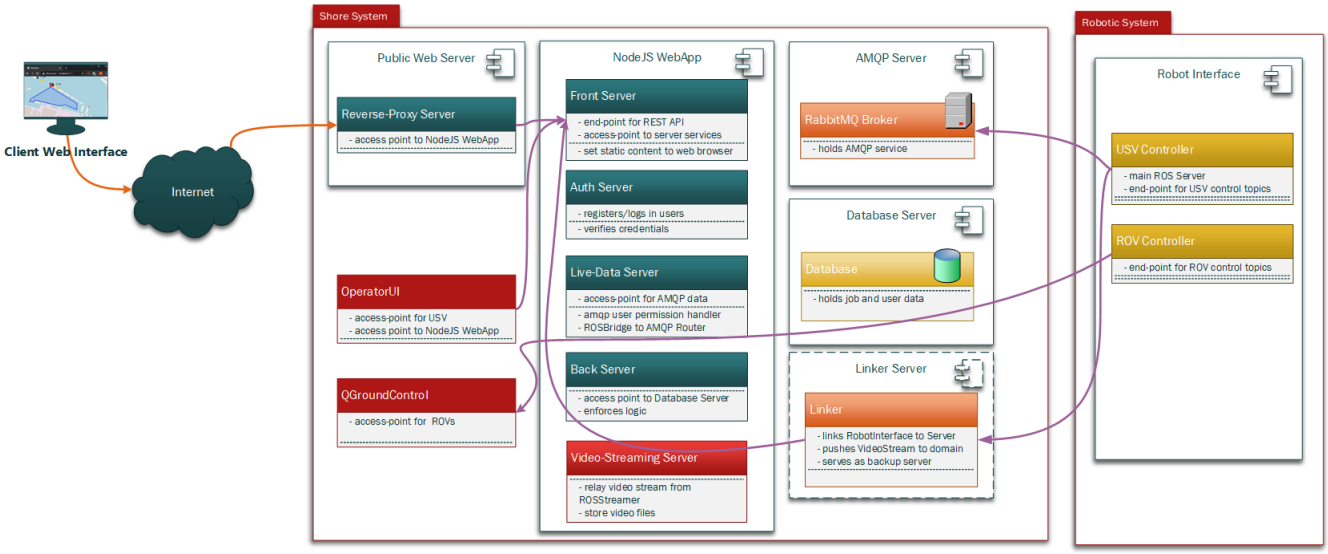

Fig. 3. RoboVaaS System Architecture

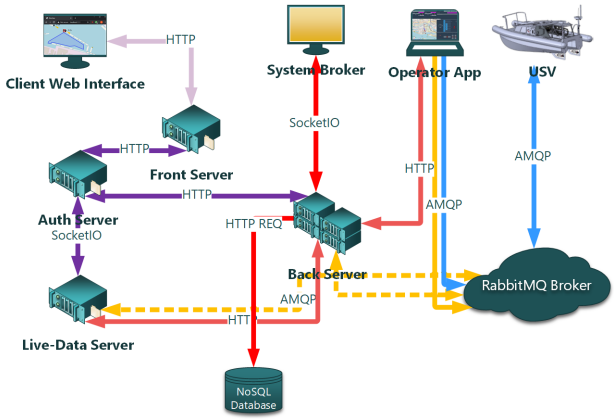

Fig. 4. Service Layout

each being identified by a unique *jobID* and linked to its respective *userID*. In this way, the closed missions can be analysed by the user that requested them or by the one having the rights to view the data. Using the exact same pipeline, further applications requiring on-demand services, performed by robotic nodes at different autonomy levels, can be envisioned.

III. Communication Network

This section presents the different parts of the telecommunication pipeline from the actual sensor nodes to the server. The complete pipeline can be divided in two main sections, specifically:

1) the underwater section that uses acoustic telecommunication technologies;
2) the above water section, that uses Ethernet, long range wireless antennas and connection to the cellular network.

Since the two technologies have different characteristics in terms of performance and functioning, we implemented the sections with distinct protocols to optimise the communications.

The underwater network (presented in Section III-A) uses very low-cost and low-power acoustic modem, the smartPORT Acoustic Underwater Modem (AHOI), while the above water network (described in Section III-B), is composed by 802.11 WiFi and Ethernet technologies manufactured by Mikrotik, plus an LTE antenna to interface the network with the Internet.

A. Underwater communication network

As previously mentioned, in the underwater section the transmissions use the acoustic water channel, since it allows to communicate wirelessly, where the usual electromagnetic waves suffer severe attenuation, at the cost of low transmission bitrates and large propagation delay.

The concept followed for this architecture is the data muling approach, that consists in having one or more nodes which travel in the area where the network is deployed and trigger the transmissions between all the nodes. In particular, there are two types of nodes: one category includes the sensor nodes which generate the data from environmental sensors and the other one is the central node, that gathers the data generated by the nodes.

The data muling approach has several advantages in a network where the nodes are deployed in clusters or the transmission range is short, so a multi-hop network should be implemented. In the case of RoboVaas, nodes are expected to be deployed in areas of interest inside ports and harbours, possibly appearing in clusters due to physical obstacles in the communication range, and the data collection is done with one device that collects all the data packets from the sensor nodes and later interfaces with the Shore System through the above water communication network.

The network management and the protocol functioning are performed in the DESERT Underwater Framework [18],

which is a network simulator based on ns2 with the option of running with a *real-time scheduler* [19]. This enables the protocols implemented for simulations to be used in full-scale applications in real life scenarios.

For the present scenario, a MAC protocol [20] has been developed with a polling policy, where a central node coordinates the communications from the sensor nodes avoiding overlapping and packet collisions. In particular, the protocol presents two main phases: the first is a *discovery phase*, where the central node (an ASV) sends a TRIGGER packet, expecting the sensor nodes within transmission range to reply with the number of packets that every single sensor has to upload; then the second phase includes the creation of a priority list, by the central node, for the sensor nodes. The order of the nodes in the list is based on the number of packets that has already been sent by every node in previous polls and those expected for the current cycle, and then the central node polls each node following this list. The cycle terminates when all the nodes in the list have sent their packets or the back-off time for the second phase expires.

Moreover, the packets exchanged in the underwater network have to be as small as possible to overcome the challenges introduced by the water as a medium. Indeed, the high bit error rate, and the low bitrate of the acoustic modems allow the successful transmission of only packets with a limited overall length, therefore we introduced a known format at the application layer for the data to be transmitted, so that the basic information is contained in a string Sx with a low number of characters. The first two values are reserved for the node ID, the subsequent six contain the timestamp in the format *HHmmss*, then one character indicates the data type and all the remaining characters until the end are the actual value. With this general format, the packet length can be kept under 32 bytes, including the header introduced by DESERT.

Furthermore, the packets are retrieved at the application layer by another program that translates the string Sx into a JSON file. Indeed, in the DESERT Framework at the application layer there is the option of using a module called *uwApplication*, which opens a local socket connection available for whichever program external to DESERT. This allows the implementation of programs that manage or create buffers of characters, which DESERT encapsulates by following the user-defined protocol stack. The packets can be later retrieved at the end of the communication through another socket connection. Thus, the second program we introduced has the main objective of translating the basic information contained in the string Sx into a format that is compatible with the **Shore System**.

Once the file has been created, it is stored in a specific folder where another program, an AMQP client, scans for new files and publishes their content in the RabbitMQ server. This approach of transmitting information through files between different applications is useful in case of crash of the AMQP client, or in case of a missing link in the above water network.

Regarding the equipment used for this application, all the software is lightweight and can be run on Single Board Computers (SBC), like Raspberry Pi or ASUS Tinkerboard, the only requirement in terms of software being a Debian-based Operating System installed on the board. The underwater transmissions are performed with the AHOI modems [21], which operate in a high frequency band (50-75 kHz) with a nominal data rate of 2.35 kb/s. The maximum payload available for this modems is 96 bytes, but to limit as much as possible the packet error rate caused by the errors introduced by the underwater channel, we decided to keep the maximum payload usable in DESERT to 32 bytes.

B. Above water communication network

The above water section of the network have been designed and implemented using equipment manufactured by Mikrotik [22], which provides a vast variety of reliable, powerful and cost-effective network devices. All Mikrotik devices run the RouterOS operating system, which provides an easy-to-use, yet powerful configuration interface, able to exploit any ISO/OSI Layer 2 and Layer 3 technologies, such as Ethernet Bridging, 802.1Q VLANs, static and dynamic routing and DHCP. On WiFi capable devices, WiFi parameters such as SSID, ACLs, WPA2 and some physical layer parameters can be easily configured. Moreover, on the AP/Base Station side, an ACL based on clients' MAC address can be set up, in order to add another layer of security over the WPA2 protocol. If a back-haul is needed with internet connection, several Mikrotik devices can mount an LTE/UMTS modem with usim slot. Moreover, RouterOS provides basic firewall capabilities, in order to prevent unauthorised access. For remote users to access securely and privately, a VPN can be configured. Several VPN protocols, such as OpenVPN, L2TP, PPTP are natively supported. Hereafter, we describe the network topology designed and implemented for the above water section. On the ASV, a Mikrotik Metal 52AC WiFi CPE [23] has been adopted. The Metal 52AC device is a full-fledged router, mounting a Gigabit Ethernet port, a WiFi 802.11b/g/n and 802.11ac compliant WiFi radio (configured to act as a CPE client) and a 6 dBi omnidirectional antenna. On the piers, we adopted mANTBox 12s as the WiFi AP [24], which mounts a 12 dBi 120 degrees directional antenna and a 802.11b/g/n WiFi modem, able to cover long distances and challenging radio channels. We adopted a 5 Ethernet ports Mikrotik hEX [25] as the core router, which connects the mANTBox, the RabbitMQ servers and optionally laptops for troubleshooting and monitoring. If an internet connection or inbound remote connection is needed via a back-haul link, a wAP R LTE Mikrotik device, mounting an LTE modem and a Gigabit Ethernet port can be connected to the hEX [26]. For what concerns IP connectivity, the network has been designed in order to have a single LAN from the ASV to the RabbitMQ server. Mikrotik hEX will act as a core router providing a DHCP server for any client that is allowed to connect. IP Addressing of the LAN will be on a single CIDR with a /24 subnet. This way, we will be able to have a full layer 2 connectivity between any client connected (both on Ethernet and WiFi) without any static routing usage. For

remote inbound connection, hEX provides some basic firewall rules and operator authentication and authorisation. A visual scheme of the network just described can be seen in Fig. 5

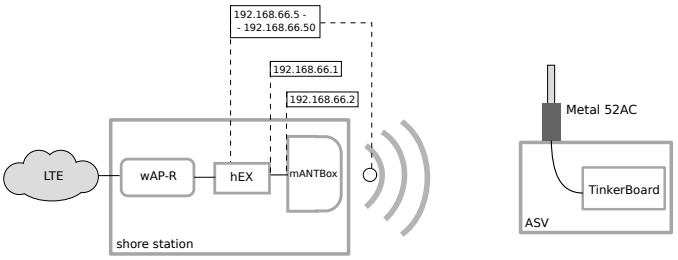

Fig. 5. Above Water Network scheme

IV. Results

Following the intended usage of the application, the registered end-user generates a request on the web user interface and awaits its assignation to an operator from the system broker. Once the operator books an available ASV and confirms the job request, the end-user will receive all the required updates on the web user interface. The operator gives the path the ASV needs to follow and manages possible hazards, such as incoming vessel traffic, unexpected obstacles or events, such as sudden changes in the water depth that can potentially ground the ASV, or component changes in the ASV, such as sensing equipment or battery swapping. During the entire operation, it is assumed that the corresponding sensory equipment is correctly fitted on the ASVs and/or ROVs. When the goal of the task given by the end-user is considered to be fulfilled, the operator marks the event closure, which reverts the statuses of both the operator and the ASV to "available" in order for new tasks to be assigned to them.

By following the above described procedure, a method to manage port-specific operations with waterborne robotics is established. By making use of the microservices, multiple tasks run independently of each other and at the same time, and all actors involved in the operations are informed in real time. Due to the nature of SOA, the server-side infrastructure and services require more work for deployment, but once finalised, clients can gain access to the several end-points of the overall system with very little effort. Moreover, as shown in Figure 6, the service client, or end-user, can reanalyse the data obtained for a specific job also, subsequently, after finalisation. In this figure, the path followed by ASV during the mission is represented by the grey line, while the green points mark the positions of the ASV at the moment when a packet is received by the central node and forwarded to the server.

The data collected from a sensor node and acquired with the RoboVaaS network can then be retrieved, visualised and further processed by the user on the web interface, as depicted in Figure 7.

A. Performance and limitations

In terms of above water communication, the proposed system was capable of delivering the packets between the

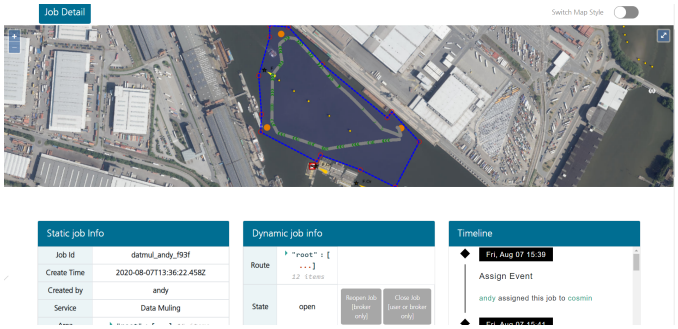

Fig. 6. Environmental Data Collection: Mission details page on web UI

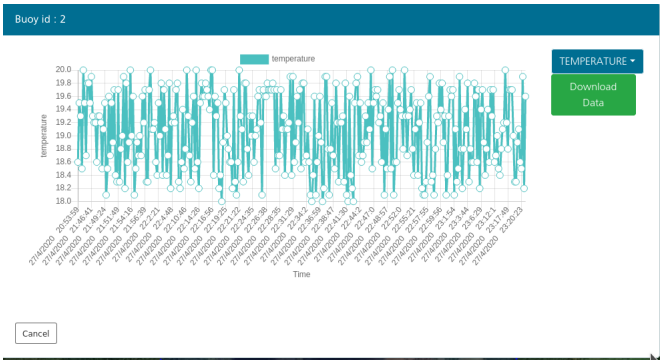

Fig. 7. Environmental Data Collection: data summary for a sensor node on web UI

Robotic System and the **Shore System** in real time. By using 1x Miktrotik Metal 52AC on-board the ASV and having 1x Miktrotik mANTBox 12s installed on the piers, creating a mesh network with 1x Miktrotik wAP R LTE acting as the central hub, excellent results have been obtained. The performance indicators are summarised in Table I.

TABLE I
Communication System Limits (*tests within internal network)

Distance [m]	Area Info	Packet loss [%]	Latency [ms]
427	some trees and above water	0	2.5
533	some trees and above water	16	13
565	several trees and above water	100	-

The results prove the effectiveness of the system in a confined space with a $\leq$ 500 m radius. Considering the safety requirements of port areas, which generally require Line-Of-Sight operations, and the scope of the RoboVaaS application, this limitation is relevant only for the anti-grounding scenario, where larger areas need to be covered. Even there, the operational range can be extended using multiple wireless access points mounted around the harbour and connected together.

The overall system is capable of handling multiple users with different roles at the same time, the only user limitation is the possibility to perform multiple jobs with the same robotic system. In that sense, the number of channels opened for

handling live data is limited to the number of ASVs in the port fleet that can be operated at a time.

Secondly, in the environmental data collection use-case, the infrastructure permits underwater communication only for ASVs carrying an underwater communication node that can operate on the frequency of the buoys.

B. Benefits (and drawbacks) of the service

The proposed system not only satisfies its scientific purpose of building a robust solution for performing above- and underwater port-specific operations using waterborne robots, but also leads the way to a market-ready product, with a coherent, resilient, scalable and state-of-the-art system architecture.

Due to its complexity, each connection between the described subsystems must be tested individually. Gradually, more actors can be involved, up to the point where the overall system is successfully tested in a controlled environment, such as an internal network and/or using the digital twin simulating the ASV.

Secondly, each operation needs highly skilled personnel, i.e., an operator, that can correctly operate, monitor and maintain the unmanned vehicles. In that sense, the proposed application does not face the difficulty of operating robotics, but focuses on distributing some of the tasks to potential customers, working alongside specialists.

V. Conclusions and further works

The overall system was developed within the RoboVaaS Project and used for showcasing the envisioned port-specific on-demand services performed by waterborne robotics with a high level of autonomy, commanded by skilled personnel and demanded by maritime stakeholders. The ambition was to validate the concept using real small-scale demonstrations of three individual use-cases: bathymetry and environmental data collection and quay-wall inspection. The presented layout had to be modular and flexible to the change of equipment needed for each use-case. This meant having a modular design of the hardware and software components on-board the ASV and having a scalable server and communication infrastructure. The overall system will be demonstrated to the public in a specially designated test area in the Port of Hamburg. The aforementioned use-cases will be performed using an in-house developed ASV and a commercial ROV. For each use-case, specific sensing equipment will be mounted on the ASV, the buoys will be deployed on-site and the users will be instructed to use the web interface to request one of the services.

The proposed system architecture and communication infrastructure will be extended and adapted to further projects, such as [27], which will benefit from the added value of the proposed scheme and further test its capabilities against medium-sized industrial robotics.

References

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

[2] The Maritime Cloud Development Forum, "Maritime cloud conceptual model," in *IALA ENAV19 Committee meeting*, St Germain en Laye, France, September 2016.

[3] J. Oeffner, S. Shetty, and H.-C. Burmeister, "A modular testbed using centralised data exchange for autonomous navigation systems," *International Symposium on Integrated Ship's Information Systems Maritime Traffic Engineering Conference*, 2016.

[4] J. Lee, "Web applications for robots using rosbridge," Brown University, Providence, RI, USA, 2012.

[5] C. Crick, G. Jay, S. Osentoski, and O. C. Jenkins, "Ros and rosbridge: Roboticists out of the loop," in *7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2012, pp. 493–494.

[6] R. Toris, J. Kammerl, D. V. Lu, J. Lee, O. C. Jenkins, S. Osentoski, M. Wills, and S. Chernova, "Robot web tools: Efficient messaging for cloud robotics," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 4530–4537.

[7] "Robotic vessels as-a-service," Last time accessed: Aug. 2020. [Online]. Available: https://www.martera.eu/projects/robovaas

[8] D. Zordan, F. Campagnaro, and M. Zorzi, "On the feasibility of an anti-grounding service with autonomous surface vessels," in *Proc. MTS/IEEE Oceans*, Marseille, France, June 2019.

[9] C. Jahn and S. Saxe, "Digitalization of seaports-visions of the future," *Fraunhofer Center for Maritime Logistics and Services (CML)*, 2017.

[10] "Ocean technology campus rostock," https://www.igd.fraunhofer.de/en/projects/ocean-technology-campus-rostock, accessed: 2020-08-28.

[11] T. Haus, M. Orsag, and S. Bogdan, "Mathematical modelling and control of an unmanned aerial vehicle with moving mass control concept," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 219–246, 2017.

[12] "Smart and networking underwater robots in cooperation meshes," http://swarms.eu/approach.html#technicalapproach., accessed: 2020-08-28.

[13] Y. Chen and H. Hu, "Internet of intelligent things and robot as a service," *Simulation Modelling Practice and Theory*, vol. 34, pp. 159–171, 2013.

[14] E. Cervera, G. Casañ, and R. Tellez, "Cloud simulations for robocup," in *Robot World Cup*. Springer, 2017, pp. 180–189.

[15] "Robot ignite academy," Last time accessed: Aug. 2020. [Online]. Available: https://www.theconstructsim.com/robotigniteacademy_learnros/ros-courses-library/

[16] K. Kamei, S. Nishio, N. Hagita, and M. Sato, "Cloud networked robotics," *IEEE Network*, vol. 26, no. 3, pp. 28–34, 2012.

[17] "Advanced message queuing protocol," Last time accessed: Aug. 2020. [Online]. Available: https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol

[18] R. Masiero, S. Azad, F. Favaro, M. Petrani, G. Toso, F. Guerra, P. Casari, and M. Zorzi, "DESERT Underwater: An NS-Miracle-based framework to design, simulate, emulate and realize test-beds for underwater network protocols," in *Proc. MTS/IEEE Oceans*, Yeosu, Republic of Korea, Aug. 2012.

[19] F. Campagnaro, R. Francescon, F. Guerra, F. Favaro, P. Casari, R. Diamant, and M. Zorzi, "The DESERT underwater framework v2: Improved capabilities and extension tools," in *Proc. Ucomms*, Lerici, Italy, Sep. 2016.

[20] A. Signori, F. Campagnaro, F. Steinmetz, B.-C. Renner, and M. Zorzi, "Data gathering from a multimodal dense underwater acoustic sensor network deployed in shallow fresh water scenarios," *MDPI Journal of Sensor and Actuator Networks*, vol. 8, no. 4, p. 55, 2019.

[21] B.-C. Renner, J. Heitmann, and F. Steinmetz, "AHOI: Inexpensive, Low-power Communication and Localization for Underwater Sensor Networks and micro-AUVs," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, pp. 1–46, 01 2020.

[22] "Mikrotik," accessed: Aug. 2020. [Online]. Available: https://mikrotik.com/

[23] "Mikrotik Metal 52AC," accessed: Aug. 2020. [Online]. Available: https://mikrotik.com/product/RBMetalG-52SHPacn

[24] "Mikrotik mANTBox," accessed: Aug. 2020. [Online]. Available: https://mikrotik.com/product/mantbox_2_12s

[25] Accessed: Aug. 2020. [Online]. Available: https://mikrotik.com/product/RB750Gr3

[26] "Mikrotik wAP R," accessed: Aug. 2020. [Online]. Available: https://mikrotik.com/product/RBwAPR-2nD

[27] "The seaclear project," Last time accessed: Aug. 2020. [Online]. Available: https://seaclear-project.eu/