Robot Operating System (ROS) talks underwater: an open-source communication middleware to control underwater vehicles

Davide Costa
davide.costa@unipd.it
University of Padova - Department of Information Engineering
Padova, PD, Italy

Filippo Campagnaro
filippo.campagnaro@unipd.it
University of Padova - Department of Information Engineering
SubSeaPulse SRL
Padova, PD, Italy

Michele Zorzi
michele.zorzi@unipd.it
University of Padova - Department of Information Engineering
SubSeaPulse SRL
Padova, PD, Italy

***Abstract*—Underwater robots are usually remotely controlled from the base station using umbilical cables due to the propagation problems of electromagnetic waves in the water, which are strongly attenuated after a few centimeters. It is possible to overcome this constraint by employing acoustic networks to communicate with underwater vehicles to send commands and receive sensor informations, by creating an ad hoc system tailored for each specific scenario.**

Advancements in underwater acoustic transmissions allowed us to introduce a general approach which can be exploited to use generic robot applications both with standard Ethernet networks and with acoustic underwater networks. In this paper we present a middleware layer module that enables any Robot Operating System (ROS) application to exchange informations through the underwater acoustic channel, without making any change in the user-level control logic.

***Index Terms*—Robot Operating System, DESERT Underwater, underwater acoustic networks.**

I. Introduction

Underwater Unmanned Vehicles (UUVs) and sensor stations are widely used devices across different sectors like oceanographic research, infrastructure inspection, offshore oil and gas industry, and defense [1]. Moreover, specific types of unmanned, self-propelled underwater drones known as Autonomous Underwater Vehicles (AUVs) are often employed to operate in extreme scenarios and follow pre-programmed missions [2]. In contrast, the Remotely Operated Vehicles (ROVs) category refers to all drones which are linked to a surface vessel with a cable, allowing for real-time remote operation by a human pilot. This physical connection enables ROVs to perform intervention tasks requiring high maneuverability and manipulation, though their range is limited. To avoid the constraints due to cable length, as well as the hazards related to cable entanglement, wireless communications would be desirable. However, when an electromagnetic wave travels through a conductive medium, the oscillating electric field of the wave causes the free ions to move, creating an electric current. This movement of charge encounters resistance, leading to the conversion of the EM wave's energy into heat. Furthermore, water molecules are polar, meaning they have a positive and a negative end. The oscillating electric field of an EM wave causes these molecules to rotate and align with the field. As the electric field oscillates rapidly, the water molecules continuously try to reorient themselves, and this constant movement creates friction, which again converts the EM wave's energy into heat [3].

A possible solution to overcome the constraints of physical cables is to use acoustic waves. The process involves converting data into sound waves, transmitting such waves through water using acoustic transducers, and employing sophisticated signal processing and modulation techniques to recover the transmitted information [4]. To achieve the goal of remotely controlling a UUV through acoustic transmissions, we chose to leverage on already existing systems used for both robotics and underwater applications, combining those two fields together without creating new applications from scratch.

The main issue is the lack of a single, universally adopted standard for underwater networks, but there are emerging systems, best practices, and common architectural approaches being developed and used for creating underwater acoustic networks (UANs). Different communication protocols for UANs are developed by a diverse range of entities, including academic researchers and universities, industry development teams and military research institutions, so the field is constantly evolving with new techniques emerging as the understanding of the underwater communications improves. These efforts have the goal to address the unique challenges of the acoustic channel, such as low bandwidth, long propagation delay, low bitrate and poor performance in shallow water and in the presence of shipping and wind noise [5].

JANUS [6] represents a key initiative to standardize underwater acoustic communications, which aims to achieve interoperability across different military and civilian underwater platforms and equipment from multiple manufacturers. Its dis-

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of Next Generation EU, partnership on "Telecommunications of the Future" (PE0000001-program "RESTART").

tinct open and public specification enables wide utilization by academia, industry, and governments, with the fully described details on signal encoding and message format allowing the development of universally compatible transmitters and receivers.

The setup of the system makes use of the DESERT Underwater protocols stack [7], which enables the deployment of a simulated channel to verify the correct operation of the network prior to the sea trial.

For robotics applications the situation is similar, but the field is converging towards a set of widely adopted frameworks and emerging standards. Hence, we are going to use the Robot Operating System (ROS) [8], the most widely employed open-source solution available for building robotic applications that includes extensive features, a large community, and growing industrial support.

The purpose of this work is to show how robotic frameworks can be used to remotely control underwater vehicles using acoustic waves, that are well suited for specific situations where employing cables is not a viable option like deep-sea operations, long-range underwater communications and various military scenarios. Especially, we will focus on presenting an open-source system capable of controlling AUVs in a simulation environment, with the goal of preparing ourselves for a future trial with real hardware at sea.

This paper is structured as follows. Section II describes the system architecture with a description of the software components used, for both the robotic and the acoustic parts. Section III deepens the concept of middleware, fundamental for our control purposes, while the results of the simulation are discussed in Section IV. Finally Section V concludes the paper.

II. System Architecture

In this section we present the architecture of the system developed to control AUVs with acoustic transmissions: the most basic setup is composed of two nodes, equipped with ROS (Section II-A) and the DESERT Underwater framework (Section II-B), responsible for creating the network protocol stack. With this configuration, robotic applications built with ROS can easily become control stations using acoustic transmissions rather than the standard network, by simply changing one parameter at runtime.

A. Robot Operating System

Initially, various software platforms emerged, offering modularity and flexibility to simplify robot construction. Some of these evolved into full ecosystems of tools and algorithms. However, the original Robot Operating System (ROS 1), fueled by Willow Garage's innovation, became the main solution employed in robotics research and industrial developments [9]. Today, the second major release known as ROS 2 replaced the original in most use cases, for both the industrial and research fields.

The set of ROS 2 open source software libraries provides a communication infrastructure, a common API to build robot

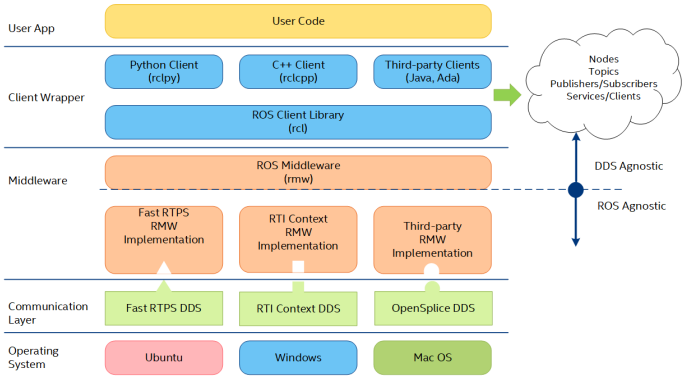

Fig. 1. Architecture of ROS2.

applications, hardware abstraction, and advanced simulation capabilities [8]. As depicted in Figure 1, it employs a layered architecture to manage its complexity and promote modularity, making the system more maintainable and adaptable.

The User Application Layer is the topmost one, where users develop their specific robot functionalities. It consists of the nodes that implement the robot's behavior, algorithms, sensors drivers, and control logic. Developers use the client libraries to create these nodes and employ the communication mechanisms provided by ROS 2. Examples of applications in this layer include:

- navigation systems;
- perception algorithms;
- robot control systems;
- user interfaces;
- task planning modules.

The Client Wrapper Layer provides higher-level access to the core communication APIs. Its libraries are customized for each programming language and depend on a common interface called ROS Client Library (`rcl`), which provides access to the ROS 2 concepts [8]. `rcl` is in charge of adapting the specific language implementation with the common C++ underlying layers of the framework, offering convenient abstractions for tasks such as creating nodes, publishing and subscribing to topics and managing services. The most widely used client libraries are `rclcpp`, the C++ client library, offering high performance and direct access to ROS 2 features, and `rclpy`, the Python client library, which provides a more accessible and rapid development environment. Other client libraries such as `rclc` exist, often used in resource-constrained environments such as micro-ROS.

Finally, the Middleware Layer is a crucial abstraction that isolates the core framework functionalities from the specifics of the underlying communication infrastructure. The ROS Middleware (`rmw`) interface defines a set of C APIs that higher-level libraries employ for the physical data exchange, enabling the usage of multiple `rmw` implementations and making ROS 2 potentially compatible with different communication technologies [8].

B. DESERT Underwater

The DESERT Underwater Framework [7] is an open-source software tool specifically designed for the development and testing of underwater acoustic networks. Built upon the NS-MIRACLE network simulator, it offers a customizable environment for researchers and developers in underwater communication. DESERT facilitates the design and implementation of various layers of the underwater network protocol stack, from the physical to the application layer, allowing for the integration of custom protocols and algorithms.

The simulated channel within the framework provides a way to model the complex environment of underwater acoustic communication, computing the propagation delay between the transmitter and the other nodes, and using noise, path loss, and interference to obtain the bit error rate. In this way, high-level applications can be tested locally without employing a modem, in order to perform first-stage validations on the functionalities implemented. Another significant strength of DESERT lies in its emulation capabilities, that enable users to connect to physical underwater acoustic modems via serial or Ethernet connections for hardware testing and real-time interaction. By supporting real modems and providing experimentation tools, DESERT facilitates the transition of developed applications to actual underwater deployments.

Finally the upper layer called `uwApplication` is used to interface the protocol stack with user applications, creating a TCP socket on the machine running DESERT. By interacting with the transport layer or lower layers, it sends and receives the application data that drives the network's purpose. A key characteristic is its role in abstracting the complexities of the underwater communication medium and network protocols, allowing application developers to focus on their specific functionalities. It will be employed in our architecture to create the bridge between ROS 2 and DESERT Underwater.

III. Middleware Structure

This section describes the RMW interface's implementation called ROS Middleware for DESERT (`rmw_desert`), which enables ROS applications to exchange data through underwater acoustic networks. The layer handles essential communication tasks like:

- discovery: how nodes find each other in the network;
- publish/subscribe: the mechanism for one-to-many asynchronous communication via topics;
- request/response: the mechanism for synchronous service calls;
- actions: a more complex request-response pattern with feedback and goals.

To understand how those operations are performed, we have to focus on the ROS modules stack and zoom in on the central part, as depicted in Figure 2. When `rcl` generates some data, it passes a pointer through the `rmw` interface indicating the start location of the message, with an additional parameter called type support that is used to understand the structure of the message itself. Using the above information, the implementation splits the data into various fields by scanning the memory locations, leading back to a list of fundamental data types such as integers, strings or floats.

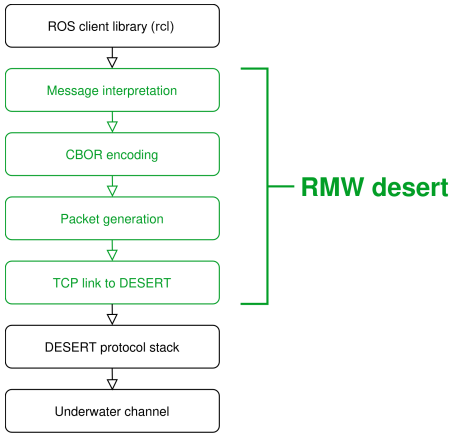

Fig. 2. Architecture of rmw_desert.

Having extracted the basic data types within the message, the middleware encodes and serializes them into a unified data stream. We selected the Concise Binary Object Representation (CBOR) [10] for this purpose due to its efficient serialization and deserialization capabilities. This choice ensures a compact binary representation, which mitigates the risk of saturating the limited bandwidth of the underwater channel. When encoding data, CBOR first identifies the type of data it is dealing with, such as an integer, a text string, or an array. It then uses a small header, typically just one to a few bytes, to indicate this data type and its length or the number of items it contains [10]. For example, a small positive integer might be encoded with a single byte, where the first few bits indicate it is positive, and the remaining bits represent the value. This binary nature of CBOR leads to faster parsing compared to text-based formats like JSON, as there is no need for extensive text processing.

Starting from the payload just built, the middleware creates a packet with additional fields, as depicted in Figure 3. A predefined bit sequence marks the start of the packet, while another sequence denotes its end, then a byte with the payload dimension is included to check the integrity of the packet. A stream type field is added to the CBOR encoded part, used to detect if the sending entity is a publisher, a service or a client. In addition, a stream identifier field is used to encode the topic name or the service name.

In the last step, the packet is delivered to the upper layer of DESERT through a TCP socket created by the `uwApplication` module, that sends the packet down to the lower layers responsible for routing it to the correct destination through the acoustic channel.

Starting Sequence	Starting Sequence	Payload Dimension	Stream Type	Stream Identifier	Remaining Payload	Ending Sequence
1 byte	1 byte	1 byte	1 byte	1 byte	n bytes	1 byte

Fig. 3. Packet structure.

IV. Simulation results

In this section, we test the middleware described in Section III using an acoustic modem emulator. The Evologics S2C acoustic modems come with an emulator called S2C DMACE [11], provided by the manufacturer to test all the software components before actual deployment. The aim is to create two ROS nodes, one for the controller station and the other for a remote robot, communicating through the DESERT protocol stack rather than the standard network.

The scenario was created using the open source Gazebo simulator, designed to generate a 3D dynamic multi-robot environment capable of building complex worlds [12]. The purpose of those worlds is to contain all the objects related to a robotic system, but also to recreate the physical dynamics to which the elements are subject. For example, the water viscosity and the underwater drone inertia are computed to simulate the behavior of a real environment. In our scenario, depicted in Figure 4, we deployed an AUV and added two dynamics capabilities:

- buoyancy: since the AUV should maintain depth when stationary;
- hydrodynamics: using Fossen's equations which describe the motion of a craft through the water.

The fully configured environment used for our simulations, including all relevant parameter settings, is available in the Gazebo Simulator API Reference, which also provides detailed installation instructions for Linux systems [13].

Buoyancy in the simulation is modeled as a force that counteracts the weight of the vehicle. To determine the required buoyant force for neutral buoyancy, the total weight of the AUV must first be calculated. The total mass of the vehicle is 148.3571 kg, which includes the chassis (147.8671 kg), fins (0.2 kg each), and the propeller (0.09 kg). The buoyant force is directly proportional to the displaced volume of water according to Archimedes' principle, which must equal the weight force:

$$\rho_{water} \cdot V_{neutral} \cdot g = m_{vehicle} \cdot g \tag{1}$$

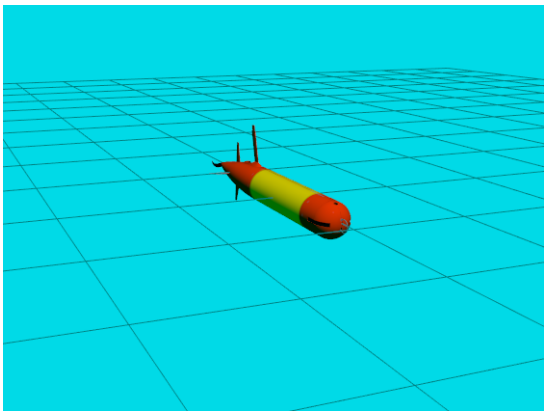

Fig. 4. Underwater world with an AUV.

$$V_{neutral} = \frac{m_{vehicle}}{\rho_{water}} = \frac{148.3571kg}{1000\frac{kg}{m^3}} \tag{2}$$

This volume represents the space that the vehicle should take up to achieve neutral buoyancy. If the actual volume is smaller, the vehicle will sink; if larger, it will ascend. The factor g represents the gravitational acceleration, while $m_{vehicle}$ is the total mass of the AUV and ρ_{water} is the density of the water. The Gazebo buoyancy plugin computes the volume using the `collision` elements contained in the model's Simulation Description file. Each `collision` element defines a bounding volume, approximated as a cuboid, specified by a three-dimensional vector representing its side lengths. By adjusting the dimensions of these `collision` elements, it is possible to manipulate the effective volume of individual components and thereby achieve neutral buoyancy for the entire vehicle:

$$\begin{aligned} V_{chassis} &= 2m \cdot 0.3m \cdot 0.246445167m \\ V_{fins} &= 2 \cdot (0.1m \cdot 0.1m \cdot 0.02m) \\ V_{propeller} &= 0.03m \cdot 0.1m \cdot 0.03m \end{aligned} \tag{3}$$

$$\begin{aligned} V_{vehicle} &= V_{chassis} + 2 \cdot V_{fins} + V_{propeller} \\ &= 0.1483571m^3 \end{aligned} \tag{4}$$

Those sides lengths for the volume of the chassis $V_{chassis}$, the volume of the fins V_{fins} and the volume of the propeller $V_{propeller}$ were chosen to ensure that their sum $V_{vehicle}$ corresponds to the value of $V_{neutral}$.

Moreover, to prevent the vehicle from exhibiting a continuous increase in velocity, the simulation includes a hydrodynamic drag opposing the thrust force. The plugin incorporates those drag forces based on the equations presented by Fossen, which describe the dynamics of marine vehicles in fluid environments [14]. These equations model the resistive forces experienced by an underwater vehicle as it moves through water, accounting for factors such as linear and quadratic damping. The coefficients required for these models are typically obtained through computational fluid dynamics (CFD) simulations or empirical testing in controlled environments such as tow tanks. For the purposes of this test, we adopt the parameter values provided in Fossen's research to define the hydrodynamic damping characteristics of the vehicle in our configuration.

To interact with the simulation, sensors can be used to receive data from the environment, and joints can be employed to manipulate the environment. Gazebo's data exchange is based on a publish-subscribe mechanism that can be directly connected to ROS, and so each sensor creates a publisher topic sending periodic data, while each joint creates a subscriber topic listening for commands.

From the ROS point of view of the system, this part forms the remote AUV node. The other node is a controller application used to send commands from the virtual base station, that we built with a graphical interface where the user

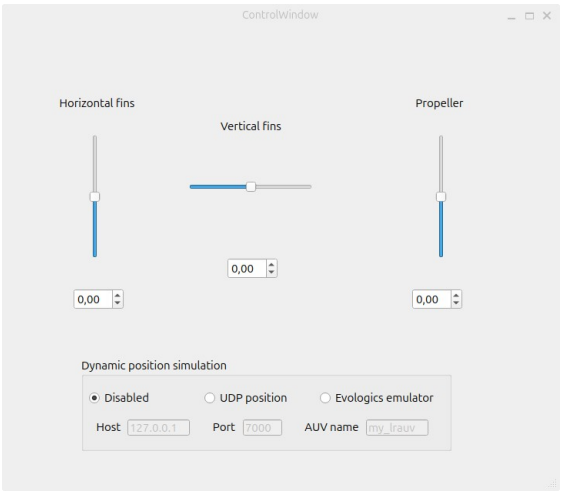

Fig. 5. ROS controller application for the AUV.

can set the horizontal and vertical fins positions as well as the propeller value (Figure 5). In the end, both nodes are ROS user applications running with the `rmw_desert` middleware, so the commands are sent down to the emulated acoustic channel created by the S2C DMACE software by the DESERT stack.

The experiments allowed to test each component of the system architecture, as well as to investigate our scenario of interest in a controlled environment. Moreover, we managed to successfully control the simulated AUV with a negligible packet loss, thanks to the combination of ROS, DESERT and DMACE which provided all the characteristics of the underwater channel and paved the way for future deployments with real hardware.

As part of the evaluation process, it was verified that the ROS middleware correctly interfaces with the DESERT Underwater framework by successfully transmitting and receiving messages between the two systems. Particular attention was given to ensuring that the messages were accurately interpreted on both sides, preserving the intended structure and semantics during the exchange. This confirmation is crucial for the integration of higher-level robotic control with the underlying communication stack provided by DESERT.

V. Conclusion and Future Work

In this paper we presented an efficient open-source framework to control underwater vehicles using acoustic waves, where DESERT Underwater and the Robot Operating System are working together to perform a wireless management of submerged drones. So far, the proposed solution was tested with a simulated AUV communicating not only through a simulated acoustic channel, but also through real EvoLogics modems and the Subsea underwater acoustic software-defined Modem [15]. The results are promising, and the work done will soon be exploited to set up a scenario with real vehicles to test the entire system. Moreover, the modularity offered by the two frameworks enables users to customize the infrastructure for different specific use cases, such as cooperative autonomous navigation, environmental monitoring, or swarm coordination, moving a step forward towards the realization of flexible and scalable wireless underwater robotic systems.

References

[1] A. Pal, Filippo Campagnaro, K. Ahraf, R. Rahman, A. Ashok, H. Guo, "Communication for underwater sensor networks: A comprehensive summary," *Transactions on Sensor Networks*, vol. 19, no. 1, pp. 44, Nov. 2022.

[2] J. A. Dowdeswell, J. Evans, R. Mugford, et al., "Autonomous underwater vehicles (AUVs) and investigations of the ice–ocean interface in Antarctic and Arctic waters," *Journal of Glaciology*, vol. 54, no. 187, pp. 661–672, Oct. 2008.

[3] P. Lunkenheimer, S. Emmert, R. Gulich, M. Köhler, M. Wolf, M. Schwab, A. Loidl, "Electromagnetic-radiation absorption of water," *Phys. Rev. E*, vol. 96, no. 6, Dec. 2017.

[4] F. Campagnaro, R. Francescon, E. Coccolo, A. Montanari, M. Zorzi, "A Software-Defined Underwater Acoustic Modem for Everyone: Design and Evaluation," *IEEE Internet of Things Magazine*, vol. 6, no. 1, pp. 102-107, March 2023.

[5] M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," *ACM Mobile Comput. Commun. Rev.*, vol. 11, no. 4, pp. 34–43, Oct. 2007.

[6] J. Potter, J. Alves, D. Green, G. Zappa, I. Nissen, and K. McCoy, "The janus underwater communications standard," *Underwater Communications and Networking (UComms)*, Sestri Levante, Italy, pp. 1-4, Sept. 2014.

[7] F. Campagnaro, R. Francescon, F. Guerra, F. Favaro, P. Casari, R. Diamant, M. Zorzi, "The DESERT underwater framework v2: Improved capabilities and extension tools," *IEEE Third Underwater Communications and Networking Conference (UComms)*, Lerici, Italy, pp. 1-5, Sept. 2016.

[8] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and William Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, May 2022.

[9] M. Albonico, M. Dordević, E. Hamer, I. Malavolta, "Software engineering research on the Robot Operating System: A systematic mapping study," *Journal of Systems and Software*, vol. 197, March 2023.

[10] C. Bormann, P. Hoffman, "RFC 8949 Concise Binary Object Representation (CBOR)," *Request for Comments*, Dec. 2020.

[11] *S2C DMAC Emulator*, Last time accessed: Apr. 2025. [Online]. Available: https://evologics.de/emulator

[12] N. Koenig, A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol.3, pp. 2149-2154, Japan, 2004.

[13] *Simulation of underwater AUV*, Last time accessed: Jun. 2025. [Online]. Available: https://gazebosim.org/api/sim/8/underwater_vehicles.html

[14] T. I. Fossen, "How to incorporate wind, waves and ocean currents in the marine craft equations of motion," *IFAC Proceedings Volumes*, vol. 45, no. 27, pp. 126-131, Sept. 2012.

[15] A. Montanari, V. Cimino, D. Spinosa, F. Donegà, F. Marin, F. Campagnaro, M. Zorzi, "PSK modulation for Underwater Communication and One-Way Travel-Time Ranging with the Low-Cost Subsea Software-Defined Acoustic Modem," *The 18th ACM International Conference on Underwater Networks & Systems (WUWNET '24)*, Šibenik, Croatia, no. 9, pp. 1-8, Jan. 2025.