Enabling Real-Time Remote Monitoring of Ships by Lossless Protocol Transformations

Giacomo Longo [1,1], Alessandro Orlich [2], Alessio Merlo [2], and Enrico Russo [2]

[1]University of Genova
[2]Affiliation not available

October 31, 2023

Abstract

This paper uses data processing techniques to reduce the required transmission bandwidth in ship-to-shore communications. The proposed framework (ONline Efficient Sources Transmission Optimizer - ONESTO) leverages state-of-the-art technologies and novel algorithms to automatically optimize transmissions under structural (e.g., available bandwidth, fixed packet overhead) and user-defined (e.g., maximum latency) constraints. In addition, ONESTO authenticates and encrypts the communication between the ship and the shore via mainstream free and open-source software components. Initially, we present the abstract mathematical formulation of the problem, with its assumptions, goal function, constraints, and significant quantities. Then, we introduce the architecture of a system capable of continuously estimating the compressibility, processing and transmission time of streaming data. Such estimations allow ONESTO to calculate and apply optimal parameters for achieving the best compression ratio. Lastly, using a prototypical implementation, we evaluate the system performance with a Class B ship simulator on two realistic use cases. Our experiments show an excellent compression ratio with maritime protocols (more than 40:1) and a limited latency impact, demonstrating the approach's viability.

Enabling Real-Time Remote Monitoring of Ships by Lossless Protocol Transformations

Giacomo Longo, Alessandro Orlich, Alessio Merlo, and Enrico Russo

***Abstract*—This paper uses data processing techniques to reduce the required transmission bandwidth in ship-to-shore communications. The proposed framework (ONline Efficient Sources Transmission Optimizer - ONESTO) leverages state-of-the-art technologies and novel algorithms to automatically optimize transmissions under structural (e.g., available bandwidth, fixed packet overhead) and user-defined (e.g., maximum latency) constraints. In addition, ONESTO authenticates and encrypts the communication between the ship and the shore via mainstream free and open-source software components. Initially, we present the abstract mathematical formulation of the problem, with its assumptions, goal function, constraints, and significant quantities. Then, we introduce the architecture of a system capable of continuously estimating the compressibility, processing and transmission time of streaming data. Such estimations allow ONESTO to calculate and apply optimal parameters for achieving the best compression ratio. Lastly, using a prototypical implementation, we evaluate the system performance with a Class B ship simulator on two realistic use cases. Our experiments show an excellent compression ratio with maritime protocols (more than 40:1) and a limited latency impact, demonstrating the approach's viability.**

***Index Terms*—Autonomous vessels, Data compression, Data transfer, Maritime communications**

I. Introduction

DIGITALIZATION of the industry is catching pace in the maritime sector, and modern ships play an essential role in this process. Nowadays, they embed digital enabling technologies and hardware components, e.g., sensors and actuators, increasing more and more their efficiency and safety. Also, new opportunities and challenges arise around the above technologies and the large amount of data they produce.

For example, data fusion and intelligent analytics enable the development of even more effective algorithms to boost the spread of semi-autonomous and autonomous vessels [1]. Another opportunity is feeding data into a digital model, namely a *Digital Twin* (DT), to mirror and predict the behaviors of a monitored ship [2]. Instead, a major challenge is providing the facilities to investigate such data to detect and prevent cyberattacks that the new digital assets might suffer from [3].

It is worth noting that all of the above cannot or can only be partially carried out onboard and require resources and personnel from dedicated shore side centers. In this regard, the maritime industry plans to use *Remote Operation Centers* (ROC) [4] for semi-autonomous and autonomous vessels. Such centers can also provide resources for implementing the functionalities of DTs [5] or host specialized *Security Operation Centers* (SOC) [6] for monitoring and reacting to cybersecurity threats.

In all these scenarios, the connectivity between ships and ROCs is vital and represents the most critical component. However, ship-to-shore communication is still an open issue today [7], [8], [9].

In particular, available technologies cannot yet fulfill the requirement of sufficient communication link capacity, enabling ROCs to receive enough data from a ship on the sea for the above-mentioned shore side operations. A common proposal relies on increasing the link capacity by leveraging multiple connection technologies [10], [11], [12] with a *vertical handover* mechanism [9], [13], but it can work only partially. Once the ship is out of sight of land, signals of faster connections fade, and the only way to connect is by satellite, i.e., a link with low bandwidth, high latency, and elevated costs.

Data compression has become a standard feature to improve transmission capacity in network environments with such features. Also, it is essential in ship-to-shore communications, but effectiveness may be low (or even counterproductive) if applied as is. Compression efficiency [14] tends to be specific to each application and requires a proper balance between improving the compression ratio with aggregated data and the computational latencies it introduces at both the transmitting and receiving sides. Nevertheless, ships are comparable to complex Information Technology (IT) infrastructures with Operational Technology (OT) systems [15] that can host any application. For this reason, compression efficiency requires to keep searching for a reasonable trade-off among the requirements of a plethora of heterogeneous data sources that transmit at a variable rate.

Lastly, ship-to-shore communication has also to deal with cybersecurity issues. In particular, a ship and ROC must communicate over insecure networks [16] without any malicious actor being able to eavesdrop or impersonate any of the parties.

Due to the complexities introduced above, we argue that an open research challenge is to build a *ship-to-shore communication framework* (SSCF) that may be at the same time:

P-1 Self-adaptive. It adapts its processing in real-time to changes in specific transmitting and receiving conditions (see below). The aim is to ensure compression efficiency and the continuity of data reception at the shore side.

P-2 Source agnostic. It works without any prior knowledge about data sources.

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.; Manuscript received ...; G. Longo, A. Orlich, A. Merlo, and E. Russo (Corresponding author) are with the Department of Informatics, Bioengineering, Robotics, and Systems Engineering (DIBRIS), University of Genoa, Italy. Email: {name.surname}@dibris.unige.it.

P-3 Lossless. It ensures that the ROC receives a *bit-perfect* representation of the data stream. For example, this feature is essential for SOC duties, like anomalies detection or forensic investigations.

P-4 Secure. It can authenticate the ROC with the connected fleet (and vice-versa) and provide secure channels for data transmission.

Moreover, we argue that the **P-1** property of an SSCF —tailored for remote monitoring of ships— has to *continuously* and *simultaneously* deal with the conditions below.

C-1 End-to-End latency, i.e., given the requirements of the remote service type, e.g., ROC or SOC, it ensures that the shore side always receives within an upper-bounded latency.

C-2 Bandwidth usage, i.e., considering the coexistence with other ship facilities that use the connectivity, it spares as much headroom as possible on the shared communication channel.

C-3 Source variability, i.e, it is able of multiplexing multiple heterogeneous onboard data sources, handling changes in their rate and contents induced by the distinct phases of navigation.

In this paper, we present the design and implementation of a novel SSCF, namely ONline Efficient Sources Transmission Optimizer (ONESTO), which satisfies all previous requirements. In particular, the inspiring principle of ONESTO is combining an innovative self-adaptive algorithm with a state-of-the-art lossless compression algorithm and a lightweight, high-performance, and secure message-oriented middleware. We empirically evaluated ONESTO under ROC and SOC requirements, and with a bridge simulator providing the data sources of a realistic Integrated Navigation System (INS) [17].

This paper presents the following key contributions:

- A self-adaptive algorithm designed to optimize data transmission efficiency and reliability, able to adapt to varying communication conditions without relying on specific data sources, specifically tailored for ship-to-shore communications.
- A cutting-edge SSCF that seamlessly integrates state-of-the-art technologies to achieve top-notch performance and enhance security. Mainly due to the compression ratio achieved (more than 40 to 1), we prove that a remote center can also monitor high-resolution radar data.
- An extensive assessment of the proposed methodologies using real-world case studies and data sources.

Experimental results on maritime protocols demonstrate that ONESTO can effectively transmit crucial data to reconstruct the ship's situational awareness onshore, even under the most constrained bandwidth conditions and with a latency of no more than 7 seconds. We claim that the above capability is of great interest to the research community working on the design of onshore solutions to improve ships' safety and security.

Structure of the paper. The rest of the paper is organized as follows. In Section II, we review the related work. In Section III, we discuss the mathematical formulation of our self-adaptive algorithm. In Section IV, we introduce the architecture of the framework and evaluate its implementation in Section V. We conclude the paper in Section VI.

II. Related work

Our work aims to develop an SSCF for the remote monitoring of ships that meets **P-1**, **P-2**, **P-3**, and **P-4** properties. Related work includes solutions that optimize data transmission in contexts with similar criticalities (e.g., low-bandwidth or high-latency links) or ship-to-shore network communications.

TABLE I: Comparison between ONESTO and other literature works.

	Self-adaptive P-1	**Source agnostic P-2**	**Lossless P-3**	**Secure P-4**
Wiseman et al. [18]	◑	◑	●	○
Krintz et al. [19]	◑	◑	●	○
Ling Sun et al. [20]	○	●	●	○
Berni et al. [21]	◑	●	●	○
Yang et al [22]	○	○	●	○
Ifrim et al. [23]	○	○	○	○
Ferreira et al. [24]	○	○	○	○
Perera et al. [25]	○	○	○	○
ONESTO	●	●	●	●

In Table I, we summarize features of considered literature works w.r.t. the four properties met by ONESTO. For each work, we use ● and ○ to denote whether the property is satisfied or not, respectively. Moreover, we use ◑ to indicate that the property is only partially met. Partially fulfilling **P-1** means that the proposal addresses some, but not all, of the conditions **C-1**, **C-2**, and **C-3**.

A research topic that is not specific to ship-to-shore communication but shares with us the general schema is adaptive compression. The most relevant works are from Wiseman et al. [18] and Krintz et al. [19].

The solution proposed in [18] relies on a fixed block size and hard-coded coefficients derived from a chosen dataset. ACE [19] similarly relies on a statically determined comparison between different compression algorithms, performed again on a generalist compression corpus. Using a predetermined dataset makes them partially satisfy **P-2**, and the precalculated parameters do not fit **C-3**. Lastly, they both work as exclusive transmitters, contrasting with our **C-2**.

Ling Sun et al. [20] propose a real-time adaptive packet compression scheme improving bandwidth in scenarios with satellite networks. They use a fixed-size buffer that collects packets until it is full or a specific interval expires. Then, the buffer is compressed with a lossless algorithm before being sent. Since their adaptiveness does not consider our conditions, it does not represent an alternative SSCF to ONESTO.

Berni et al. [21] present a solution to support sea trials requiring a large shore–ship–shore data exchange. Unlike ONESTO, they increase bandwidth not by using compression but by aggregating multiple links, which is not always viable.

Their approach to adaptiveness uses Quality of Service (QoS) technology to allocate bandwidth for specific protocols while ensuring a portion remains for other resources, meeting condition **C-2** but only partially **P-1**.

Yang et al. [22], Ifrim et al. [23], Ferreira et al. [24], and Perera et al. [25] share with ONESTO a similar context and objective in terms of maritime communication and the transmission of data from ships to shore. Proposal from [22] applies to video data and uses real-time genetic algorithms to implement a scheduler that optimizes the transmission considering the tardiness and weights of jobs. The one from [23] applies to Automatic Identification System (AIS) [26] data and reduces them in real-time by extracting and transmitting only the essential information from sentences. Ferreira et al. [24] consider data from onboard sensors and, similarly to [23], propose an automated process that decides whether the collected data needs to be transmitted or not. Instead, Perera et al. [25] propose a solution to reduce data transmitted through an autoencoder trained with a set of predefined data sources. None of the above works meet the required properties.

In summary, ONESTO offers a more robust self-adaptive solution by utilizing both compression to increase bandwidth and real-time tuning of algorithms to maintain transmission efficiency in changing conditions.

Lastly, ONESTO is the only solution that (i) looks at the security issues of communication channels by supporting authentication and link encryption, (ii) provides multi-tenancy natively for fleet monitoring, and (iii) includes the Zstandard [27] compression algorithm.

III. Mathematical Formulation

In this section, we describe the mathematical formulation behind ONESTO.

A. Background

Onboard, multiple data sources transmit packets to the receiving end of the system. Each of these packets has length n_p and is comprised of a sequence of bytes $\langle b_1, \dots, b_{n_p} \rangle$ where $b_j \in [0, 2^8)$ for $j = 1, \dots, n_p$.

As a result of **P-2**, we make no assumptions on the process originating b_js.

For the i-th data source, we model the arrival of such packets as a Poisson process with parameter λ_i, and their size n_p as a random variable sampled from a normal distribution with unknown mean μ_i and variance σ_i^2.

Those packets enter the system and are batched together in aggregations of size n_i. Such aggregations are subsequently subject to a lossless compression operation. We model that operation as a pair of actions $\langle C_{c_i}, D \rangle$ with C being the *compression* action parameterized by its settings c_i and D being the *decompression* action. We make the following three assumptions about such an operation:

1) $M = D(C_{c_i}(M))$ holds for any possible admissible input M and setting c_i, i.e. that it is *lossless*.
   This assumption implies property **P-3** by construction.
2) for the majority of inputs $|M| > |C_{c_i}(M)|$ for an admissible input M and a proper compression setting c_i, i.e. that it is *size reducing*.
3) the time taken for compressing a message M is greater or equal that the time taken for decompressing it, i.e., that it is *compression-heavy*.

If desired, any invertible transformation F can be used to define a derived compression operation in which the original compression actions are replaced by $\langle F(C_{C_i}), D(F^{-1}) \rangle$. We leverage this property to meet **P-4** by including an authenticated encryption function F_e. Since the choice of an encryption algorithm directly influences the security profile, F_e is parameterless, fixed, and not tied to the optimization process.

Then, the compressed result with a size overhead of O is sent on a channel having a head-of-line latency of T_{lat} and an available bandwidth B. Lastly, the receiving side decompresses the aggregation and replays it at the original source rate.

B. Optimization goal and constraints

To meet **P-1**, the system must automatically and continuously comply with the set constraints and targets. For that purpose, the system reacts to the provided conditions by constantly solving a constrained optimization problem, updating the found solution as the underlying conditions or constraints change.

$$\underset{n_i, c_i}{\operatorname{argmin}} \sum_i \frac{D_{tx_i}(n_i, c_i)}{D_{proc_i}(n_i)} \tag{1}$$

Equation 1 expresses the optimisation goal: finding for each transmitting source an aggregation batch size n_i and compression parameters c_i so that the ratio between the transmitted data size D_{tx_i} and the aggregated data size D_{proc_i} is minimized, i.e., maximizing the global compression ratio to achieve minimal bandwidth usage, as per **C-2**. Calculation of D_{tx_i} and D_{proc_i} will be explained in Section III-C.

$$D_{tx_i}(n_i, c_i) \leq D_{proc_i}(n_i) \qquad \forall i \tag{2}$$

In addition, we impose the set of constraints found in Equation 2: the size of transmitted data in any solution should be *equal* or *smaller* than the corresponding original data.

$$t_{proc_i} + T_{lat} + t_{tx_i} \leq t_{gen_i} \qquad \forall i \tag{3}$$

Finally, Equation 3 bounds any found solution to have finished processing (t_{proc_i}), and sending ($T_{lat} + t_{tx_i}$) data before the next aggregation has been generated (t_{gen_i}). Section III-D will explain the composition of t_{proc_i}, t_{tx_i}, and t_{gen_i}.

Until now, the presented equations concerned structural properties of the problem, i.e., necessary conditions for validating a given solution. Yet, the system might also need to meet other constraints given by the user in order to tailor the system operation to its own needs. In this work, we use a user constraint derived from **C-1** and relevant to multiple use-cases: ensuring that the maximum end-to-end system traversal time T is kept below a chosen maximum latency L_{max} ($T \leq L_{max}$). Definition of T will be provided in Section III-D.

Choice of L_{max} has the intuitive task of constraining the maximum admissible data delay, possibly by influencing the latency-compression ratio trade-off. For instance, a ROC might want to use a low maximum latency value (at most tens of seconds) to ensure near real-time observability. Instead, a SOC concerned with threat intelligence and forensic analysis can tolerate optimization solutions associated with higher delays, gaining a potentially higher compression ratio and the capability of receiving more data.

C. Size-related quantities

$$D_{proc_i}(n_i) = n_i \cdot \nu_i \tag{4}$$

D_{proc_i} is obtained with Equation 4, i.e. by multiplying the number of packets n_i belonging to an aggregation and the estimate of an individual packet size ν_i.

$$D_{tx_i}(n_i, c_i) = h_i(n_i, c_i) \cdot D_{proc_i}(n_i) \tag{5}$$

In Equation 5 we obtain the compressed size as some unknown function h_i of the original size, influenced by the number of elements being batched n_i and the compression operation parameters c_i.

D. Latency-related quantities

The system maximum end-to-end traversal time T is the maximum latency between a packet ingress into the system on the ship side and its egress on the shore side.

$$\max_i t_{gen_i} + \max_i t_{proc_i} + T_{lat} + \sum_i t_{tx_i} + \max_i t_{rx_i} \tag{6}$$

Equation 6 shows the calculation of the maximum latency figure T. Its components are as follows:

- $\max_i t_{gen_i}$ is the maximum time taken by a source for generating the packets belonging to an aggregation.
- $\max_i t_{proc_i}$ is the maximum time taken to compress an aggregation.
- T_{lat} is the head-of-line latency of the communication channel.
- $\sum_i t_{tx_i}$ is the sum of utilization time of the communication channel.
- $\max_i t_{rx_i}$ is the maximum time taken to decompress a received aggregation.

We detail their computation below.

$$t_{gen_i} = \frac{n_i}{\lambda_i} \tag{7}$$

The time taken by a source for generating the n_i packets for an aggregation t_{gen_i} can be calculated as consequence of the arrival rate λ_i, as shown in Equation 7.

$$t_{proc_i} = g_i(n_i, c_i) \cdot D_{proc_i}(n_i) \tag{8}$$

The duration of the compression operation t_{proc_i} is calculated in Equation 8 as some coefficient g_i applied to original size $D_{proc_i}(n_i)$, influenced by the number of elements being batched n_i and the compression operation parameters c_i.

$$t_{tx_i} = \frac{D_{tx_i}(n_i, c_i) + O}{B} \tag{9}$$

The individual source utilization time of the channel t_{tx_i} is modeled in Equation 9 as the ratio between the transmitted size $D_{tx_i}(n_i, c_i)$ plus overhead O and the available bandwidth B. By utilizing the sum in Equation 6, we model the sources as sharing the channel via time-division.

$$t_{rx_i} = \begin{cases} v_i(n_i, c_i) \cdot D_{tx_i}(n_i, c_i) & v_i(n_i, c_i) \text{ known} \\ t_{proc_i} & \text{otherwise} \end{cases} \tag{10}$$

The time taken for decompressing the aggregation in Equation 10 is calculated either as a coefficient v_i of the transmitted size $D_{tx_i}(n_i, c_i)$ or as equal to t_{proc_i}. This latter equality stems from Assumption 3 made in III-A on the compression operation and allows to perform the calculation whenever v_i is unknown.

E. Estimation of properties

The previous calculations rely on perfect knowledge about the behavior of each source, i.e., their stochastic process parameters, and the chosen compression algorithm.

Such knowledge is not available in a practical case. To lift this assumption, each parameter appearing in the equations has to be replaced with its estimate, which is calculable. These estimates are also to be updated over time in response to **C-3**.

1) Arrival rate λ_i: As in [28], the Poisson arrival rate process parameter λ_i can be estimated with its maximum likelihood estimator (MLE).

$$\hat{\lambda_i} = \frac{A}{P} \tag{11}$$

The MLE, shown in Equation 11 is an efficient, unbiased estimator for λ_i in which A is the sum of arrivals in the period and P is the length of the period.

$$\lambda_{i_L} = \frac{\chi^2_{\alpha/2}(2A)}{2P} \qquad \lambda_{i_U} = \frac{\chi^2_{1-\alpha/2}(2A+2)}{2P} \tag{12}$$

For a given confidence $100(1-\alpha)$ the estimator upper and lower confidence bounds are given by Equation 12, where $\chi^2_q(\xi)$ is the q-th quantile of the χ^2 distribution with ξ degrees of freedom. Such bounds indicate that the true value lies between them with $100(1-\alpha)$ confidence and can be used to bound the estimated value in calculations.

In order to overestimate t_{gen_i}, we use λ_{i_L} as λ.

2) Packet size estimate ν_i: An estimation of the underlying normal distribution parameters (μ_i, σ_i) is required to calculate the estimated packet size for a source ν_i. Such parameters can be derived from n samples of individual packet size measurements x_i.

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (13) \qquad s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})^2 \quad (14)$$

As in [29], Equation 13 depicts the MLE for the mean $\hat{\mu}$.

Equation 14 depicts the *sample variance* s^2, an unbiased estimator of the underlying distribution parameter σ^2.

Like in the previous case, confidence intervals of such quantities are of interest to bound the estimates. We calculate them as shown in [30].

$$\mu_c = t_{1-\alpha/2}(n-1)\frac{s}{\sqrt{n}} \qquad \mu_L = \hat{\mu} - \mu_c \quad \mu_U = \hat{\mu} + \mu_c \tag{15}$$

In particular, Equation 15 shows the confidence intervals for the mean, where $t_p(\xi)$ is the q-th quantile of the Student's t-distribution with ξ degrees of freedom.

$$\sigma^2_L = \frac{(n-1)s^2}{\chi^2_{1-\alpha/2}(n-1)} \qquad \sigma^2_U = \frac{(n-1)s^2}{\chi^2_{\alpha/2}(n-1)} \tag{16}$$

Likewise, Equation 16 illustrates the confidence intervals for the variance.

In the following, we use $\nu_i = \mu_U + 3\sigma_U$ to overestimate the aggregated data size.

3) Compression algorithm coefficients g_i, h_i, v_i: Excluding Assumption 3 made in Section III-A, which produces the inequality $v_i \leq g_i$, no analytical derivation of these coefficients is possible without further assumptions on the used algorithm.

As a result, their values have to be extracted from a Look-Up Table (LUT).

IV. Proposed framework

This section describes the architecture of our framework.

A. Overview

Figure 1 details the architecture of ONESTO. For the sake of presentation, the figure depicts a one-way message flow, i.e., the ship acting as the transmitter and the shore side as the receiver. Instead, the ship-to-shore transmission must be considered bidirectional, and the two communicating entities can simultaneously perform the transmitter and receiver functions. In such a one-way representation, *data* from monitored sources reach three tasks, i.e., *Sampler*, *Estimator*, and *Compressor*.

Sampler and Estimator continuously monitor the *compression statistics* and *source properties*, e.g., the data rate of each source. They generate outputs that, together with *user constraints* and *decompression statistics* coming from the shore side, represent the input of the *Optimizer* task.

Optimizer executes the self-adaptiveness algorithm and outputs the *solution* to the *Compressor*, i.e., the optimal parameters the task needs to use for aggregate and compress data. The self-adaptiveness algorithm uses a stateful approach and stores historical data in the *Context* database.

Optimizer creates *aggregated packets* consisting of aggregated and compressed data for each source that *Transmitter* transmits to the shore side.

Receiver at the shore side receives the aggregated packets that forward to the *Replayer* task.

Replayer retransmits data from the aggregated packets as if the original onboard sources had transmitted them.

Periodically, Optimizer can perform a *benchmark request*. Such a request is sent along with the aggregated packet to the *Benchmarker* task.

Benchmarker measures latencies due to the shore side procedures and sends the results via the decompression statistics. Such activity is an example of the shore side acting as the transmitter, as mentioned above.

Below, we detail each task at the ship and shore side.

B. Ship Side

The ship side of the architecture is where data gathering, compression, and transmission occur. We designed it as the cooperation of 5 types of modules, as described below.

1) Sampler: As mentioned in Section III-E3, the self-adaptiveness algorithm requires values for g_i and h_i LUT. For each data source, Sampler continuously estimates these parameters by aggregating $\bar{n}_i$ packets. It applies to such an aggregation the compression operation parameterized by $\bar{c}_i$. $\bar{n}_i$ and $\bar{c}_i$ have been sampled randomly from the range of their admissible values.

In particular, sampling of $\bar{n}_i$ is performed on $\mathcal{U}_{1,\lfloor\lambda_i \cdot L_{max}\rfloor}$ where $\mathcal{U}_{L,U}$ indicates the uniform distribution with values $\in [L, U]$. $\lfloor\lambda_i \cdot L_{max}\rfloor$ is the number of elements generated by a given source across the entire latency budget.

2) Estimator: For each data source involved in the transmission, a dedicated Estimator estimates the arrival rate λ_i and the packet size of messages as shown in Equations 11-16.

3) Optimizer: This component receives statistics from the Sampler and Benchmarker (see Section IV-C) and decides for every data source what is the best number of packets to aggregate and the best compression level to solve the problem described in Section III.

Solution of the problem amounts to periodically reevaluating the best n_is and c_is whenever any of these four events happen: (i) estimator changes its estimates for a source, (ii) the user constraints are updated, (iii) sampler has added a new data point, (iv) benchmarker has added a new data point.

The *best* solution will be the one satisfying every constraint and fitting as the argmin in Equation 1.

It is worth noting that situations may arise in which there are no admissible solutions, e.g., a user constraint cannot be met with the current structural properties. A "*no solution found*" exception is sent to the Compressor in this case.

4) Compressor: Parallel to the abovementioned components, Compressor receives packets. It applies to them the aggregation and compression operations, as indicated in the current solution by the Optimizer.

Additionally, it prefixes the sent data with the time taken by the source to generate the aggregation $\bar{t}_{gen_i}$ that will be used by the Replayer at shore.

If it receives an exception from Optimizer, ONESTO drops data transmission and triggers an alarm. It is up to administrators to decide how to handle this scenario. For example, they can relax user constraints to allow the Optimizer to find a solution.

5) Transmitter: As the last stage for the ship side of the architecture, the Transmitter sends the Compressor-generated packets to shore. This module is also responsible for providing the authentication of communicating entities and activating a secure channel for data transmission.

C. Shore Side

The framework at the shore side has two tasks: reconverting packets to their original form and participating in the optimization problem. The following components achieve these duties.

1) Receiver: This component acquires aggregated packets, distinguishing from which source they were generated, and feeds them to the Decompressor.

2) Decompressor: This component receives compressed batches of packets from a data source and decompresses them back to their original form, leaving them grouped.

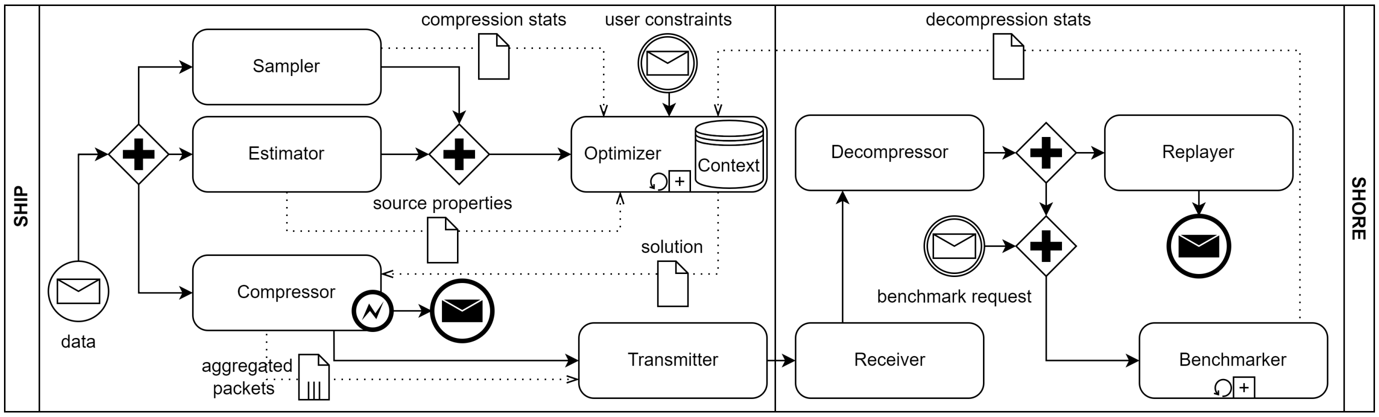


Fig. 1: The architecture of ONESTO.

3) Replayer: This component aims to divide the batches into single packets and transmit them so that the software listening locally can receive them. It must ensure that packets arrive in the same order and rate as they were originally sent. It performs such pacing by leveraging the $\bar{t}_{gen_i}$ value found in the batch.

4) Benchmarker: It is analogous to the Sampler component for the shore side. It receives from the latter requests to calculate decompression performance on a given aggregation size and compression parameters. Thus, it can send back $v_i(n_i, c_i)$ to Optimizer to more precisely estimate the values in Formula 10.

V. Implementation and results

In this section, we describe our implementation of the architecture presented in Section IV. Then, we test the ONESTO tool on two realistic scenarios and discuss the results.

A. Implementation Details

Our Proof-of-Concept implementation consists of four ad-hoc programs and a Free and Open Source Software (FOSS) component:

1) A general purpose TCP/UDP receiver (one for each source) acquires data to transmit.
2) A monolithic process undertakes the roles of *Estimator*, *Sampler*, *Optimizer*, and *Compressor* modules.
3) A monolithic process undertakes the roles of *Decompressor* and *Replayer*.
4) A process carries out the function for the *Benchmarker* module.
5) A FOSS message broker provides data exchange between local modules and remote transmission by implementing the publish/subscribe paradigm [31] (pub/sub).

In particular, the ship instance comprises items 1, 2, and 5, and the shore one comprises items 3, 4, and 5.

We wrote each of these ad-hoc programs in the Rust programming language (version 1.63) [32], totaling 1933 lines of non-library code.

For the *Compressor* module, we use Zstandard, a fast lossless compression algorithm with outstanding compression ratio [33]. In particular, we leverage its reference implementation (zstd [34], version 1.5.2), and we use the compression level as our compression operation parameter c_i (see Section III-B)

Concerning the communication between ship and shore, we selected NATS (version 2.8) [35], a multi-tenant messaging system based on pub/sub. In particular, modules subscribe and publish to topics of interest to interact with each other and transmit data. Each topic also distinguishes messages based on different properties, e.g., from which data source they are coming or whether they are in their original form, grouped, or compressed. We use a federation between NATS instances to handle data transmission. In detail, the shore side subscribes to the topics of interest for its activity (in our case, the ones containing the compressed data sources), and NATS transmits only them with *exactly-once* delivery.

The multi-tenancy support also enables multiple ship-side nodes to connect to the shore side and extends our architecture to a fleet monitoring solution.

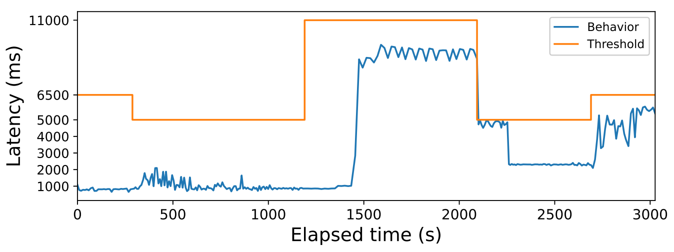
Moreover, NATS provides internally mutual Transport Layer Security (TLS) [36] authentication and encryption. We leverage the above facility as our authenticated encryption function F_e (see Section III-A) Lastly, using a common standard as pub/sub and such a configuration of topics ease the extension of ONESTO with new modules and functionalities. For example, we can add a module that filters source-specific data and further reduces bandwidth demand (implementing a solution as proposed in [24], [23], see Section II). Briefly, integration involves subscribing the module to the data source topic and publishing its output on a new one to distinguish the filtered source to be compressed and transmitted.S

B. Experimental Settings

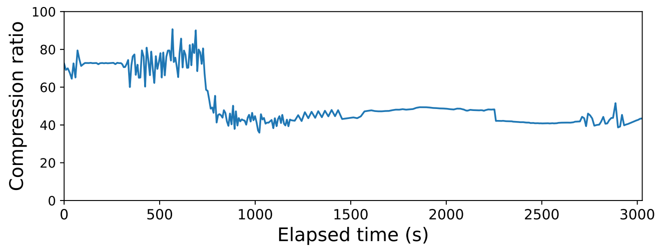
Our test environment runs on Ubuntu GNU/Linux 22.04, installed on a Virtual Machine (VM) hosted by VMWare ESXi 7.0U3 and configured with 16 Intel Xeon Gold 6252N vCPUs at 2.3GHz and 64 GB of RAM. The above VM hosts the instances of ONESTO for the ship and shore entities.

We enabled TLS on the two federated instances of NATS. We used X.509 certificates [37] to identify and authenticate the two entities and ChaCha20 [38] as the encryption algorithm.

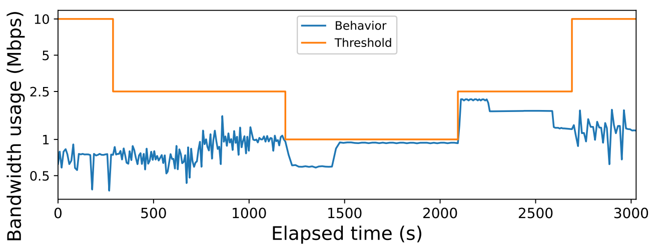
Simulation of the network link was performed by delaying packets before their arrival to the receiver component of ONESTO. Each transmitted packet was serialized inside of a FIFO



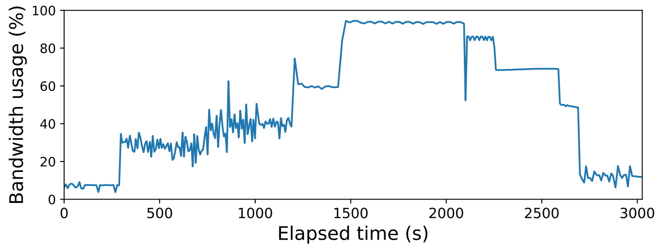
(a) End-to-End latency.



(b) Compression ratio.

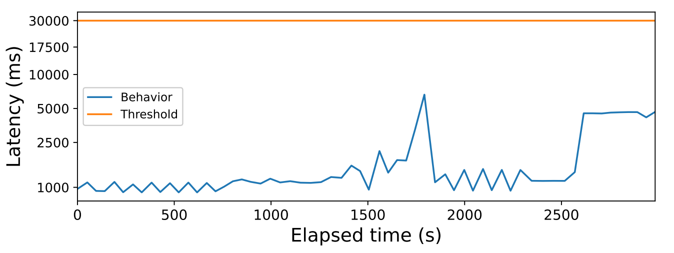


(c) Bandwidth usage (Kbps).

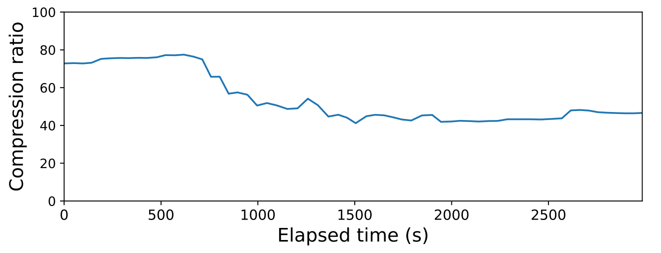


(d) Bandwidth usage (%).

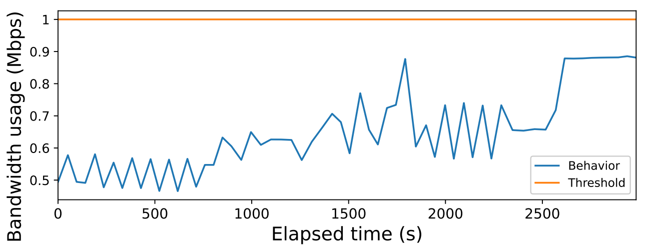
Fig. 2: Results during the scenario presented in V-C.



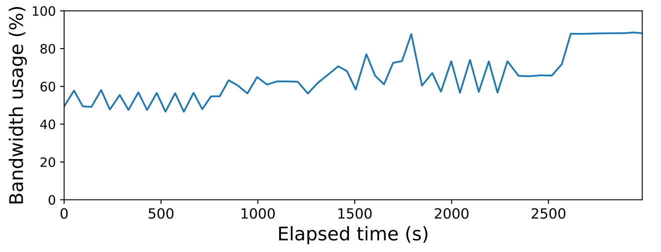
(a) End-to-End latency.



(b) Compression ratio.



(c) Bandwidth usage (Kbps).



(d) Bandwidth usage (%).

Fig. 3: Results during the scenario presented in V-D.

queue and subsequently extracted after a time corresponding to the simulated round-trip latency T_{lat} plus the transmission time t_{tx} (see section III-D). Both T_{lat} and the bandwidth B are perturbed in this link simulator by an additive Gaussian noise with a standard deviation corresponding to 10% of their value. The per-packet overhead O was instead set to 46 bytes, i.e., the size of a packet containing Ethernet, IPv4, and UDP headers.

We modeled the characteristics of satellite connectivity based on the datasheet of a leading provider [39] and by considering the worst conditions for geostationary configuration, i.e., a B between 1-100Mbps and a T_{lat} of 160ms.

A commercial full bridge simulator [40] simulates the ship underway and provides data sources to be transmitted. Our installation provides the features of a Class B console simulator [41] and is depicted in Figure 4. In particular, we consider as sources the data that are carried by two standard protocols running on INSs: *NMEA 0183* [42] and *ASTERIX CAT240* [43]. The first carries data from navigation sensors and equipment, and the latter carries data from radar antennas.

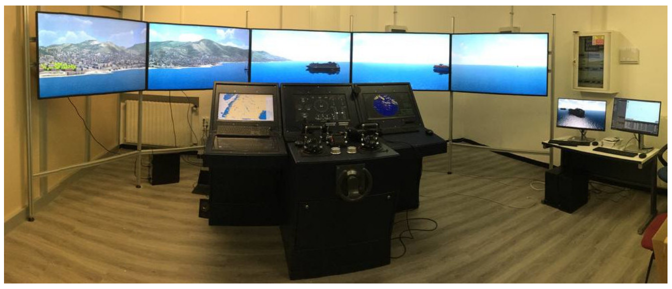


Fig. 4: The simulator used in the experiments.

These two feeds comprise most of the information used for navigation and allow the shore side to obtain situational awareness comparable with that on board.

Navigation is set in a scenario reproducing the Ligurian Sea and the Port of Genoa (Coordinates: $44^\circ\ 24'\ 10''$ N, $8^\circ\ 55'\ 0''$ E). In particular, we simulate a ship leaving the harbor, sailing in the open sea, and going back, switching communication links as it steps away from the shore. Such a switch changes the connectivity from ground-based wireless communication systems to satellite-based ones.

The same scenario is tested against the constraints of a ROC and SOC (see below). In both cases, the shore side receives data and transmits them back to two software reproducing a radar plan position indicator and an electronic chart display. We use RadarView-240 [44] (version 1.89.2, see Figure 5), a free ASTERIX CAT-240 viewer from Cambridge Pixel, and OpenCPN [45] (version 5.6.2, see Figure 6b), an open-source Chart Plotter Navigation software.

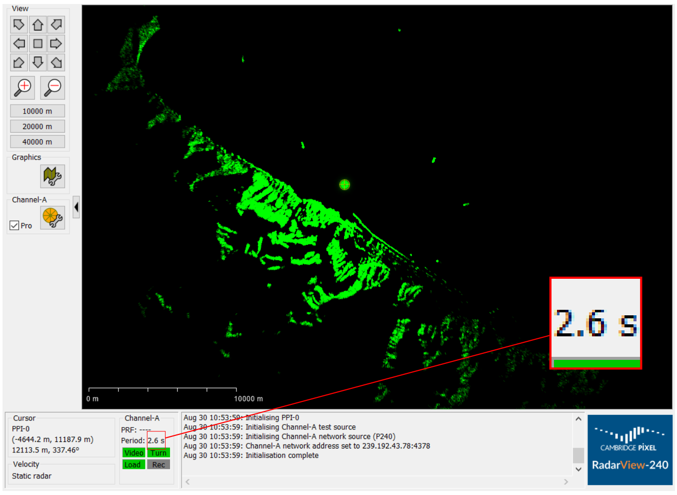
Baseline: Before assessing the properties of the framework, Table II presents the original data rates for each data source we considered.

TABLE II: Baseline data rates.

Source	λ $[\frac{n}{s}]$	ν $[Byte]$	$\lambda \cdot \nu$ $[Bps]$	$\lambda \cdot \nu$ $[Mbps]$
NMEA 0183	36	77	2772	≈ 0.022
ASTERIX CAT240	1638	4160	6814080	≈ 54.51

A digital twin of a commercial antenna generates the radar image and related ASTERIX packets. Its features are 0.88° of angular resolution, 4096 as the number of sweeps, 8 bits of resolution, $5.42m$ of range resolution, 4096 as the number of cells, and $24rpm$ as the rotation speed.

Due to the high resolution, the radar video traffic is much more bandwidth-intensive w.r.t. to the other data source. As summarized in the table, it requires more than 50 Mbps to be transmitted in the original form.

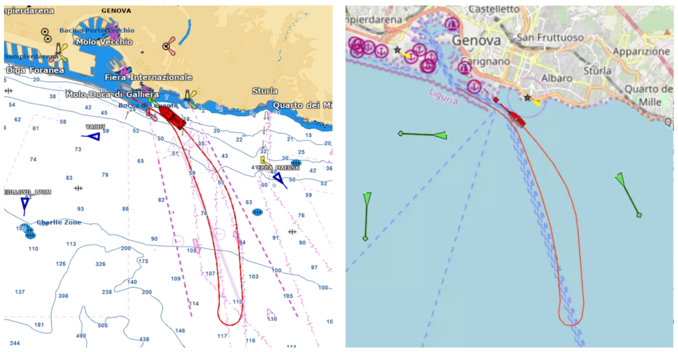


```
Cursor
PPI-0
(-4644.2 m, 11187.9 m)
12113.5 m, 337.46°
Velocity
Static radar

Channel-A
PRF: ----
Period: 2.6 s

Aug 30 10:53:59: Initialising PPI-0
Aug 30 10:53:59: Initialising Channel-A test source
Aug 30 10:53:59: Initialising Channel-A network source (P240)
Aug 30 10:53:59: Channel-A network address set to 239.192.43.78:4378
Aug 30 10:53:59: Initialisation complete
```

Fig. 5: Radar display on the shore side.



(a) Shipboard. (b) Shore side.

Fig. 6: Comparison between the cartographic system onboard and the shore side.

C. *Real-time monitoring of a ship underway*

In this scenario, a ROC wants to monitor the navigational situation of a ship by receiving INS data in *real-time*. We decline the real-time monitoring by fixing a *minimum precision*, i.e., the maximum distance the ship can cover from the last received position. In particular, the maximum latency constraint L_{max} can force the above precision by calculating it as a function of the required minimum value and the current ship's speed (Equation 17).

$$L_{max} = \frac{\text{precision}}{\text{speed}} \tag{17}$$

Moreover, we change this constraint when the ship is underway. We base on the available link capacity and leverage the self-adaptiveness capability to keep a lower but tolerable precision during limited bandwidth conditions.

TABLE III: ROC scenario.

Phase	Duration	Speed	L_{max}	Precision	B
Exiting harbor	300s	3kn	6.5s	≈10m	10Mbit
Maneuvering out	900s	10kn	5s	≈25m	2.5Mbit
At sea	900s	18kn	11s	≈100m	1.0Mbit
Maneuvering in	600s	10kn	5s	≈25m	2.5Mbit
Entering harbor	300s	3kn	6.5s	≈10m	10Mbit

Table III summarizes how we set L_{max} during the different phases of the scenario. Bandwidth B follows from the vertical handover mechanism.

Figure 2 shows the simulation results. During the experiment, the bandwidth usage and the end-to-end latencies were kept under the preset limit values. Furthermore, the system achieved a compression ratio of more than 40 to 1, realizing significant bandwidth savings.

The radar display at the shore side (see Figure 5) plots data from the ASTERIX protocol with a period of $2.6s$, confirming that the local transmission behaves like the originating antenna rotating at $24rpm$. Lastly, the cartographic system at the shore side (see Figure 6) perfectly reproduces the entire ship's track and AIS targets compared with the remote site. Again, ONESTO transmits NMEA packets locally as if they came from the original sources.

D. Collecting data to a Security Information and Event Management (SIEM) tool

A SOC requires sending data from INS to a SIEM tool so that operators can make *periodic* queries and correlations to check for anomalies or conduct post-mortem analyses [46].

Such an activity places constraints on receiving at the shore side *all* the exchanged data to collect, regardless of the bandwidth capacity of the link. To this aim, we set the maximum latency to the interval between such periodic queries ($30s$). Moreover, fixing a maximum bandwidth constraint to $1Mbps$ allows us to leverage the self-adaptiveness property to ensure data transmission during the worst case of global satellite connectivity.

We depict the simulation results in Figure 3. The experiment confirms the results of the previous scenario. It is worth noting that latency is far below the set limit and never exceeds $7s$.

E. Performance figures

During the scenarios V-C and V-D, we measured the average CPU and memory usage for each system component. On the ship side, the CPU usage amounted to one maxed-out core for each source due to the sampler operations and 1.18 CPUs for the remaining estimation and compression steps. On the shore side, the benchmarker utilized 0.81 CPUs and 0.05 CPUs for the remaining decompression and replay.

For memory usage, the process undertaking the rofles of estimator, sampler, optimizer, and compressor consumed a stable 4314MiB. On the shore side, the benchmarker consumed 547MiB, and the receiver consumed 294MiB.

The message broker, acting as the transmitter and receiver components, consumed the same amount of resources on both sides, amounting to 0.21 CPU cores and 60MiB of memory.

F. Discussion

Following the results presented in the previous section, we draw the following considerations related to ONESTO.

As per **P-1**, it succeeded in continuously adapting its parameters so that the resulting bandwidth and **C-1** end-to-end latency always remained under the given constraints. Still, thresholds were never exceeded as the maximum bandwidth and latency changed during execution. The system promptly reacted to these configuration changes with low transient times, even when faced with changes implied by **C-3** in the maximum achievable compression ratio of the underlying sources. Then, the achieved size reduction was constantly maximized by the concerted operation of its components, automatically applying the best parameters for a given data source and condition, and minimizing bandwidth usage as per **C-2**. Observing the two presented scenarios, the one with the lax latency constraint presented more stable compression values, stemming from the system being able to explore a bigger admissible solution space.

Both experiments reveal a significant amount of high-frequency noise in the transmission flow. This results from the radar being the highest volume data source and including numerous high-entropy, i.e., badly compressible, areas, such as thermal noise, terrain scattering, and reflective parts of waves.

In general, considering the achieved compression ratio and results from the self-adaptive algorithm, we argue that ONESTO can support the simultaneous transmission of a more comprehensive set of data sources, e.g., control systems setpoints and measurements or application security logs.

These results were achieved with no specific fine-tuning of the system to any application protocol, thus meeting **P-2**.

Since ONESTO leverages invertible transformations for each step and transmits data with *exactly-once* delivery semantics, it complies with **P-3** by construction.

Considering **P-4**, X.509 certificates provide strong authentication of the parties, and ChaCha20 is one of the most robust forms of encryption available. This setup effectively protects against eavesdropping and tampering attacks, where malicious actors attempt to alter or overhear the communication.

Performance figures indicate that the component with the highest resource usage is the sampler, whose impact could be reduced by throttling its operations.

Nevertheless, a coarser sampling might impair optimization effectiveness. If needed, this reduction could be tied to the availability of an admissible solution and proportional to the number of acquired samples.

The asymmetry in resource consumption between ship and shore facilitates the application of the system in many-to-one contexts, i.e., fleet monitoring solutions. Furthermore, our measurements experimentally verify that the algorithm we used fits with assumption 3 of the compression operation given in section III-A.

VI. Conclusion and future work

In this paper, we presented a novel ship-to-shore communication framework leveraging state-of-the-art technologies for enabling a self-adaptive, source-agnostic, lossless, and secure data transmission of any onboard source. In our experiments, the system rapidly adapted to changes in user and environmental constraints, optimizing and securing data transmission.

The achieved bandwidth savings of more than 40 times and low latency figures suggest that frameworks such as ONESTO will be the enabling technology of upcoming ship-to-shore communication strategies.

Future developments will involve adaptations to the underlying mathematical model to new user constraints. For example, we can consider the inclusion of communication blackouts in the formulas, in-framework estimation for the bandwidth and head-of-line latency, and the inclusion of Quality-of-Service classes between data sources. Lastly, we want to evaluate its applicability in bidirectional use cases like unmanned vehicle remote control.

Acknowledgments

This work was partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU. It was carried out while Giacomo Longo was enrolled in the Italian National Doctorate on Artificial Intelligence run by the Sapienza University of Rome in collaboration with the University of Genoa. The research activities were made possible thanks to the ShIL (Ship-In-the-Loop) research infrastructure, co-funded by Regione Liguria, University of Genova and DLTM under the program POR FESR LIGURIA 2014-2020 ASSE 1 ”Research and Innovation (OT1)” Action 1.5.1 Notice ”Support for research infrastructures considered critical / crucial for regional systems”.

References

[1] M. Martelli, A. Virdis, A. Gotta, P. Cassara, and M. D. Summa, “An outlook on the future marine traffic management system for autonomous ships,” *IEEE Access*, vol. 9, pp. 157 316–157 328, 2021. [Online]. Available: https://doi.org/10.1109/access.2021.3130741

[2] A. Coraddu, L. Oneto, F. Baldi, F. Cipollini, M. Atlar, and S. Savio, “Data-driven ship digital twin for estimating the speed loss caused by the marine fouling,” *Ocean Engineering*, vol. 186, p. 106063, Aug. 2019. [Online]. Available: https://doi.org/10.1016/j.oceaneng.2019.05.045

[3] I. Ashraf, Y. Park, S. Hur, S. W. Kim, R. Alroobaea, Y. B. Zikria, and S. Nosheen, “A survey on cyber security threats in IoT-enabled maritime industry,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2022. [Online]. Available: https://doi.org/10.1109/tits.2022.3164678

[4] K. Magnhild and A. O. Braseth, “Operating autonomous ships remotely from land-based operation centers: The current state-of-the-art,” 2020. [Online]. Available: https://ife.brage.unit.no/ife-xmlui/bitstream/handle/11250/2738266/IFE-E-2020-008.pdf?sequence=1&isAllowed=y

[5] P. Major, G. Li, H. Zhang, and H. P. Hildre, “Real-time digital twin of research vessel for remote monitoring,” *Proceedings of 35th European Council for Modelling and Simulation*, Jun. 2021. [Online]. Available: https://doi.org/10.7148/2021-0159

[6] O. Jacq, X. Boudvin, D. Brosset, Y. Kermarrec, and J. Simonin, “Detecting and hunting cyberthreats in a maritime environment: Specification and experimentation of a maritime cybersecurity operations centre,” in *2018 2nd Cyber Security in Networking Conference (CSNet)*. IEEE, Oct. 2018. [Online]. Available: https://doi.org/10.1109/csnet.2018.8602669

[7] O. J. Rodseth, B. Kvamstad, T. Porathe, and H.-C. Burmeister, “Communication architecture for an unmanned merchant ship,” in *2013 MTS/IEEE OCEANS - Bergen*. IEEE, Jun. 2013. [Online]. Available: https://doi.org/10.1109/oceans-bergen.2013.6608075

[8] G. Aiello, A. Giallanza, and G. Mascarella, “Towards shipping 4.0. a preliminary gap analysis,” *Procedia Manufacturing*, vol. 42, pp. 24–29, 2020, international Conference on Industry 4.0 and Smart Manufacturing (ISM 2019). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978920305588

[9] M. Hoyhtya, “Connectivity manager: Ensuring robust connections for autonomous ships,” in *2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS)*. IEEE, Feb. 2019. [Online]. Available: https://doi.org/10.1109/icoias.2019.00022

[10] J. S. Pathmasuntharam, J. Jurianto, P.-Y. Kong, Y. Ge, M. Zhou, and R. Miura, “High speed maritime ship-to-ship/shore mesh networks,” in *2007 7th International Conference on ITS Telecommunications*. IEEE, Jun. 2007. [Online]. Available: https://doi.org/10.1109/itst.2007.4295914

[11] X. Huang and W. Liu, “Dynamic networking and channel access strategies of hybrid communication network for intelligent ship,” *Journal of Physics: Conference Series*, vol. 1834, no. 1, p. 012011, Mar. 2021. [Online]. Available: https://doi.org/10.1088/1742-6596/1834/1/012011

[12] X. Li, W. Feng, J. Wang, Y. Chen, N. Ge, and C.-X. g Wang, “Enabling 5g on the ocean: A hybrid satellite-UAV-terrestrial network solution,” *IEEE Wireless Communications*, vol. 27, no. 6, pp. 116–121, Dec. 2020. [Online]. Available: https://doi.org/10.1109/mwc.001.2000076

[13] M. Höyhtyä and J. Martio, “Integrated satellite–terrestrial connectivity for autonomous ships: Survey and future research directions,” *Remote Sensing*, vol. 12, no. 15, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/15/2507

[14] B. Welton, D. Kimpe, J. Cope, C. M. Patrick, K. Iskra, and R. Ross, “Improving i/o forwarding throughput with data compression,” in *2011 IEEE International Conference on Cluster Computing*. IEEE, Sep. 2011. [Online]. Available: https://doi.org/10.1109/cluster.2011.80

[15] G. Kavallieratos and S. Katsikas, “Managing cyber security risks of the cyber-enabled ship,” *Journal of Marine Science and Engineering*, vol. 8, no. 10, 2020. [Online]. Available: https://www.mdpi.com/2077-1312/8/10/768

[16] J. Pavur, D. Moser, M. Strohmeier, V. Lenders, and I. Martinovic, “A tale of sea and sky on the security of maritime VSAT communications,” in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2020. [Online]. Available: https://doi.org/10.1109/sp40000.2020.00056

[17] “Adoption of the revised performance standards for integrated navigation systems (INS),” International Maritime Organization, pp. 1–6, 2017.

[18] Y. Wiseman, K. Schwan, and P. Widener, “Efficient end to end data exchange using configurable compression,” in *24th International Conference on Distributed Computing Systems, 2004. Proceedings.* IEEE, 2004. [Online]. Available: https://doi.org/10.1109/icdcs.2004.1281587

[19] C. Krintz and S. Sucu, “Adaptive on-the-fly compression,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 1, pp. 15–24, Jan. 2006. [Online]. Available: https://doi.org/10.1109/tpds.2006.3

[20] T. L. Sun, T. C. Eng, and L. S. Ping, “Real-time adaptive packet compression scheme for high latency network,” in *2012 IEEE Symposium on Computers & Informatics (ISCI)*. IEEE, Mar. 2012. [Online]. Available: https://doi.org/10.1109/isci.2012.6222676

[21] A. Berni, D. Merani, M. Leonard, and M. Rixen, “Network-enabled rapid environmental assessment: Architectures for near-real-time data collection and fusion,” *Journal of Marine Systems*, vol. 78, pp. S408–S414, 2009, coastal Processes: Challenges for Monitoring and Prediction. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924796309001687

[22] T. Yang, H. Feng, J. Zhao, R. Deng, Y. Wang, and Z. Su, “Genetic optimization–based scheduling in maritime cyber physical systems,” *International Journal of Distributed Sensor Networks*, vol. 13, no. 7, p. 155014771771716, Jul. 2017. [Online]. Available: https://doi.org/10.1177/1550147717717163

[23] C. Ifrim, M. Wallace, V. Poulopoulos, and A. Mourti, *Methods and Techniques for Automatic Identification System Data Reduction*. Cham: Springer International Publishing, 2021, pp. 253–269. [Online]. Available: https://doi.org/10.1007/978-3-030-38836-2_12

[24] J. C. Ferreira and A. L. Martins, “Edge computing approach for vessel monitoring system,” *Energies*, vol. 12, no. 16, 2019. [Online]. Available: https://www.mdpi.com/1996-1073/12/16/3087

[25] L. P. Perera and B. Mo, “Ship performance and navigation data compression and communication under autoencoder system architecture,” *Journal of Ocean Engineering and Science*, vol. 3, no. 2, pp. 133–143, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468013317301109

[26] *M.1371 Technical characteristics for an automatic identification system using time division multiple access in the VHF maritime mobile frequency band*, International Telecommunication Union Std., Rev. 5.

[27] M. K. Y. Collet, “Zstandard Compression and the ’application/zstd’ Media Type,” Internet Requests for Comments, RFC Editor, RFC 8878, July 2021. [Online]. Available: http://www.rfc-editor.org/rfc/rfc8878.txt

[28] M. E. Engelhardt, “Events in time: Basic analysis of poisson data,” 1994. [Online]. Available: https://www.osti.gov/biblio/10191309

[29] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[30] M. Smithson, *Confidence intervals*. Sage, 2003, no. 140.

[31] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, “The many faces of publish/subscribe,” *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, Jun. 2003. [Online]. Available: https://doi.org/10.1145/857076.857078

[32] “Rust Programming Language,” https://www.rust-lang.org/, 2023.

[33] “Zstd reference implementation,” http://www.zstd.net, 2023.

[34] M. Platforms, “Zstandard,” 2022, accessed on 28/08/2022. [Online]. Available: http://facebook.github.io/zstd/

[35] “NATS,” https://nats.io/, 2023.

[36] E. Rescorla, “The transport layer security (tls) protocol version 1.3,” Tech. Rep., 2018.

[37] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” RFC 5280, May 2008. [Online]. Available: https://www.rfc-editor.org/info/rfc5280

[38] Y. Nir and A. Langley, “ChaCha20 and poly1305 for IETF protocols,” Tech. Rep., Jun. 2018. [Online]. Available: https://doi.org/10.17487/rfc8439

[39] SES, “Cruise mpowered - service brief,” accessed on 28/08/2022. [Online]. Available: https://www.ses.com/sites/default/files/2022-06/2022-Service-Brief-Cruise-mPOWERED.pdf

[40] “Multipurpose Advanced Naval Training Architecture (MANTA),” https://www.cetena.it/en/products/simulator/, Cetena S.p.A., 2022.

[41] DNV, “Maritime simulator systems,” 2017, accessed on 28/08/2022. [Online]. Available: https://rules.dnv.com/docs/pdf/DNV/ST/2017-03/DNVGL-ST-0033.pdf

[42] *61162-1 Maritime Navigation and Radiocommunication Equipment and Systems—Digital Interfaces—Part 1: Single Talker and Multiple Listeners*, International Electrotechnical Commission Std., Rev. 2016.

[43] *Specification for Surveillance Data Exchange - Part 1 All Purpose Structured EUROCONTROL Surveillance Information Exchange (ASTERIX)*, 3rd ed., 2020, EUROCONTROL-SPEC-0149. [Online]. Available: https://www.eurocontrol.int/publication/eurocontrol-specification-surveillance-data-exchange-part-i

[44] C. P. Ltd., "RadarView-240," 2022, accessed on 28/08/2022. [Online]. Available: https://cambridgepixel.com/products/display-applications/radarview-240-asterix-cat-240-viewer/

[45] "OpenCPN Chart Plotter Navigation," accessed on 28/08/2022. [Online]. Available: https://opencpn.org

[46] M. Raimondi, G. Longo, A. Merlo, A. Armando, and E. Russo, "Training the maritime security operations centre teams," in *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, Jul. 2022. [Online]. Available: https://doi.org/10.1109/csr54599.2022.9850324

Giacomo Longo received his B.S. and M.S. degrees, in the field of computer engineering from the University of Genoa, in 2018 and 2021. He is currently a student in the Italian national PhD program in AI and Cybersecurity. His research interests lie in the area of simulation, systems security analysis, and virtualization.

Alessandro Orlich received his B.S. and M.S. degrees, in the field of computer engineering from the University of Genova, in 2018 and 2021. He is currently a research fellow at DIBRIS, University of Genova.

Alessio Merlo is an Associate Professor in Computer Engineering at the University of Genova where he leads the Mobile Security research group. His main research interests focus on Mobile and IoT Security. He published more than 100 scientific papers in international conferences and journals.

Enrico Russo received his M.Sc.in Computer Science and Ph.D. in Computer Science and Systems Engineering at University of Genoa in 2001, and 2021. He joined as Assistant Professor at DIBRIS, University of Genova, in 2021. His research activity is focused on cyber range systems, digital twins, and maritime cybersecurity.