IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON NETWORKS OF UNMANNED AERIAL VEHICLES: WIRELESS COMMUNICATIONS, APPLICATIONS, CONTROL AND MODELLING

Received September 28, 2017, accepted October 24, 2017, date of publication November 2, 2017, date of current version November 28, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2769658

Unmanned Aerial Vehicles as Data Mules: An Experimental Assessment

DAVID PALMA[1], ARTUR ZOLICH[2], YUMING JIANG[1], AND TOR ARNE JOHANSEN[2]

[1]Department of Information Security and Communication Technology–NTNU, Norwegian University of Science and Technology, 7491 Trondheim, Norway
[2]Department of Engineering Cybernetics–NTNU, Norwegian University of Science and Technology, 7491 Trondheim, Norway

Corresponding author: David Palma (david.palma@ntnu.no).

This work was partially supported by the Research Council of Norway Centre for Autonomous Marine Operations and Systems NTNU under Grant 223254, and in part by the project Hybrid Operations in Maritime Environments through the MAROFF Programme under Grant 269480. The work of D. Palma was supported by the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie (SINet) Grant 699924, SINet.

ABSTRACT Communication in remote locations, specially in high-latitude regions, such as the Arctic, is challenged by the lack of infrastructures and by the limited availability of resources. However, these regions have high scientific importance and require efficient ways of transferring research data from different missions and deployed equipment. For this purpose, unmanned aerial vehicles (UAVs) can be used as data mules, capable of flying over large distances and retrieving data from remote locations. Despite being a well-known concept, its performance has not been thoroughly evaluated in realistic settings. In this paper, such a solution is evaluated through a field-experiment, exploiting the obtained results to define and implement an emulator for intermittent links. This emulator was designed as a mission planning tool, where we further analyze the impact of different flight trajectories when retrieving data. Additionally, we study the overall performance of 4 well-known file-transferring protocols suitable for a UAV being used as a data mule. Our analysis shows that trajectories at higher altitudes, despite increasing distance between nodes, improves communication performance. Moreover, the obtained results demonstrate that DTN2, using the bundle protocol, outperforms FTP, Rsync, and SCP, and that all these protocols are affected by the size of the files being transferred. These results suggest that, in order for the scientific community to practically use UAVs as data mules, further studies are required, namely on how different UAV trajectories can be combined with efficient file-transferring network protocols and well organized data structures.

INDEX TERMS Emulation, hardware, IP networks, protocols, prototypes, sea surface, software, unmanned aerial vehicles, wireless communication.

I. INTRODUCTION

Scientific expeditions have an important role in the research process conducted in many fields of science. This often results in the deployment of equipment for extended periods of time, which may involve a large number of scientists and high costs. Therefore, it is common for distinct research groups to join efforts and collaborate in missions or expeditions, specially in challenging locations such as the Arctic. Research cruises are an example of such joint initiatives (e.g. N-ICE 2015[1] and MOSAIC).[2] However, berth cost alone may exceed the expedition's budget for the entire mission, limiting the available number of participants. This typically results in a few scientists on board the cruise ship being responsible for deploying scientific equipment, as well as for performing measurements on behalf of their colleagues.

Limitations in the research crew, and the possible lack of familiarity with equipment and procedures, require support and verification of collected data, which in turn implies frequent communication between colleagues in the field and at their institutions. However, at high-latitude locations no high-speed communication link is available and scientists have to resort to satellite communication (e.g. Iridium). This represents additional costs, and limited bit-rate links, which may make the data exchange unfeasible. In the worst case scenario, measurements can only be verified by specialists after the expedition is over, often several months later, without any possibility to adapt procedures in case anything goes wrong.

A solution for the lack of high-speed communication options could be using Unmanned Aerial Vehicles (UAVs) as data-mules (FIGURE 1). Data-muling UAVs are networking nodes capable of collecting data, keeping it and delivering it when a destination is reached. This approach is studied

[1]http://www.npolar.no/en/projects/n-ice2015.html
[2]http://www.mosaicobservatory.org/

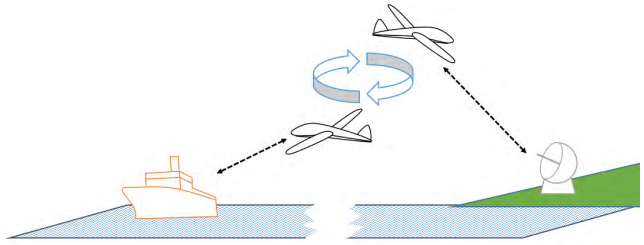

FIGURE 1. Data-muling using an unmanned aerial vehicle.

by researchers taking into account different aspects concerning the use of unmanned aerial systems and existing challenges [1]–[3]. Such aspects and challenges include the control of flying nodes and their trajectory [4], [5], or the study of communication and networking issues [1], [6]. However, these works focus only on a subset of existing challenges individually, such as how to improve connection quality or which routing protocols to be used, ignoring the dependence between them.

Other works provide a more practical approach, describing the technical details and performance of specific solutions, such as the use of data-mules in wireless sensor networks [7], [8]. However, to the best of our knowledge, no work considers the full stack of components comprising unmanned aerial systems as data mules. In particular, the users' perspective on the performance of UAV data mules is often disregarded, giving no insights on what can be expected in realistic settings.

This paper presents an evaluation of data-muling in remote oceanic scenarios, where scientific data should be retrieved from a Research Vessel (R/V). It considers not only practical aspects of such scenario, but also different available research directions in system and mission optimisation for improving the overall performance of using UAVs as data mules. Therefore, the presented field experiment focuses on a common setup, where a transceiver is located on the deck of a ship, resulting in obstructions to the Line-of-Sight (LOS) due to the ship's structure (e.g. bridges, cranes and others). In these conditions the communication quality may be intermittent and, therefore, it is not trivial to determine how, and how efficiently, data-muling UAVs can be used. To answer this issue, this paper describes a field-experiment where a UAV flew next to a ship using 3 different trajectories, measuring networking performance and WiFi link parameters to later be used in an emulated environment.

An evaluation of the overall data-muling performance is provided, resorting to existing and well-known networking solutions, such as the file-transferring protocols FTP, SCP, Rsync and DTN.[3] However, in order to conduct a valid comparison of these protocols, they need to be tested in the same conditions. Since this may be impossible during a flight due to several changing variables, despite repeatedly following the same trajectory, a new network emulator is proposed. This emulator uses the field-experiment results as input, combining aspects such as the Signal to Noise Ratio (SNR), and provides a verifiable platform for testing UAVs as data mules, using existing protocols in realistic conditions.

The main contributions of this paper include:

- Experimental assessment of UAV data-muling and its performance in maritime conditions;
- Network emulator design, development and validation;
- Comparison of file-transferring protocols in 3 realistic, dynamically-changing environments.

The organisation of this paper is depicted by FIGURE 2, starting with a field experiment based on commercial-off-the-shelf (COTS) components, from which performance data is gathered. This data supports the creation and validation of a network emulator for intermittent links, as well as of an in-depth analysis of common file-transferring network protocols. Finally, the obtained results motivate a discussion about how to setup UAVs as data mules and the main concluding thoughts.

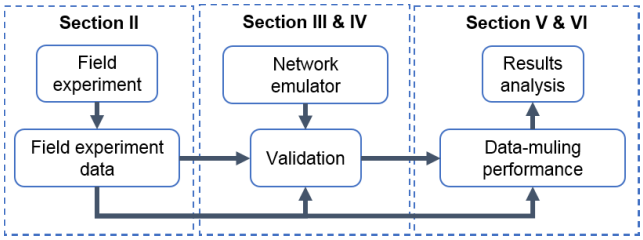

FIGURE 2. Paper's structure.

II. FIELD EXPERIMENT

In this section, we describe a field-experiment conducted in order to obtain performance data on a WiFi link between a UAV and a vessel. An important goal of this experiment was to provide realistic input for further studies and, therefore, COTS hardware was used. This included using real UAV flights with different trajectories and a ship from where data was to be collected.

A. HARDWARE DESCRIPTION

In order to conduct the proposed experiment, dedicated ship-deck nodes were developed, the Coastal and Arctic Maritime Operations and Surveillance (CAMOS) nodes (Fig. 3). The nodes are compatible with a UAV architecture currently used by the UAV-lab at the Norwegian University of Science and Technology (NTNU). Core elements of nodes and the UAV are presented in FIGURE 4.

A CAMOS node is an independent, waterproof, carry-on unit equipped with a computer, radio transceivers, other peripherals and a power source. A unit can be powered from the grid or from internal its battery, providing a few hours lifetime. It can be a standalone node, or it can be connected to external network using Ethernet cable or WiFi. CAMOS nodes have a flexible architecture that allows to test a variety

[3]Based on DTN2 implementation of the Bundle Protocol

FIGURE 3. Transceivers (during the experiment different antennas were used).

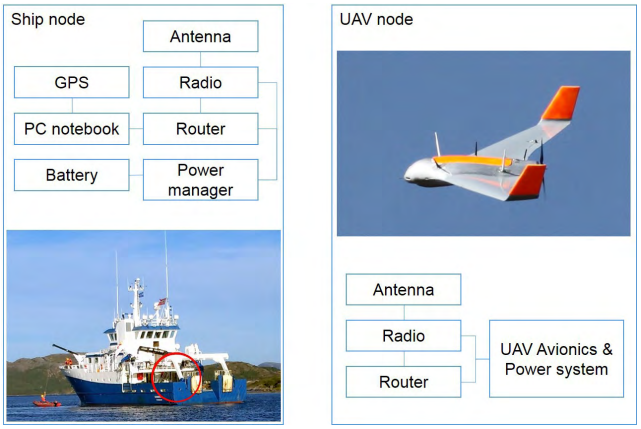

FIGURE 4. Nodes: Skywalker X8 based UAV, Research Vessel "Gunnerus" (transceiver position marked with a red circle).

of radio transceivers and network technologies. The major components of the node are presented in TABLE 1.

TABLE 1. Major experiment components.

Element	Description
Transceiver on ship deck	
PC notebook	HP x360 – Celeron N3050
PC operating system	Ubuntu 16.04 LTS
Radio	Ubiquiti Rocket M5
Radio firmware	XW.ar934x.v5.6.9.29546.160819.1146
Antenna	L-COM HG245806U-4RSP
Router	Ubiquiti EdgeRouterX SFP
Router firmware	EdgeOS v1.6.7
UAV equipment	
Radio	Ubiquiti Rocket M5
Radio firmware	XM.ar7240.v5.6.9.29546.160819.1157
Antenna	WiMo Antennen & Elektronik GmbH 18720.3
Router	OpenEmbed SOM9331
Router operating system	OpenWRT 14.07 Barrier Breaker

Similar to the CAMOS node architecture, the UAV architecture is modular, open and flexible, based on NTNU's X8 model [9]. For evaluating networking performance, the UAV used in this experiment was setup with an instance of an *iperf* server, running on its on-board router. The CAMOS node was setup with a script spawning *iperf* clients, sequentially, with different parameters. Additionally, during the entire experiment, an extensive set of data was collected from both the CAMOS and UAV nodes. This data contained:

- UAV autopilot log (e.g. position, attitude);
- Radio log (e.g. TX rate, SNR, CCQ, Retransmissions);
- GPS log (i.e. time and coordinates);
- *iperf* transfer reports.

B. FIELD EXPERIMENT EXECUTION AND RESULTS

The field experiment involved the cooperation of the R/V "Gunnerus" with a CAMOS node, the X8 UAV, and three teams of engineers and researchers. The CAMOS node was installed on-board of the R/V, which, before the experiment start, was positioned at a target destination ≈ 6 km away from the UAV take-off area. Once at the targeted destination, the ship used its Dynamic Positioning (DP) system in order to maintain its position regardless of sea currents or wind.

The X8 UAV node performed a 40 min flight, during which flight parameters, ship position and communication performance were recorded. For example, FIGURE 5 shows the transmission rate (TX) registered by the radio during the entire flight. This value is an outcome from the used modulation coding scheme (MCS), taking into account the link quality and measured SNR, and has a correlation to the ship's bearing to the UAV.

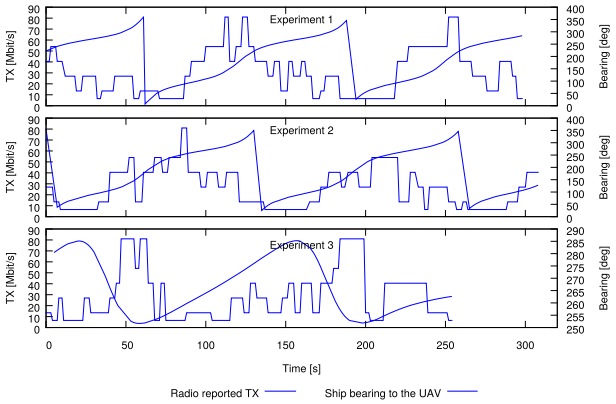

FIGURE 5. Example of data recorded during the flight.

In the performed experiment, the vessel's position was used as the centre of a 400 m radius circumference, for which the UAV-operators team programmed the UAV's loiter-waypoint at an altitude of 300 m. The UAV track was modified twice by changing either the centre or altitude (FIGURE 6). The first track begun when the UAV was loitering above the ship at 300 m altitude, for which *iperf* tests were performed (Experiment 1). When these tests were completed, the altitude was changed to 200 m and the *iperf* tests repeated (Experiment 2). Finally, keeping altitude, the loiter-waypoint

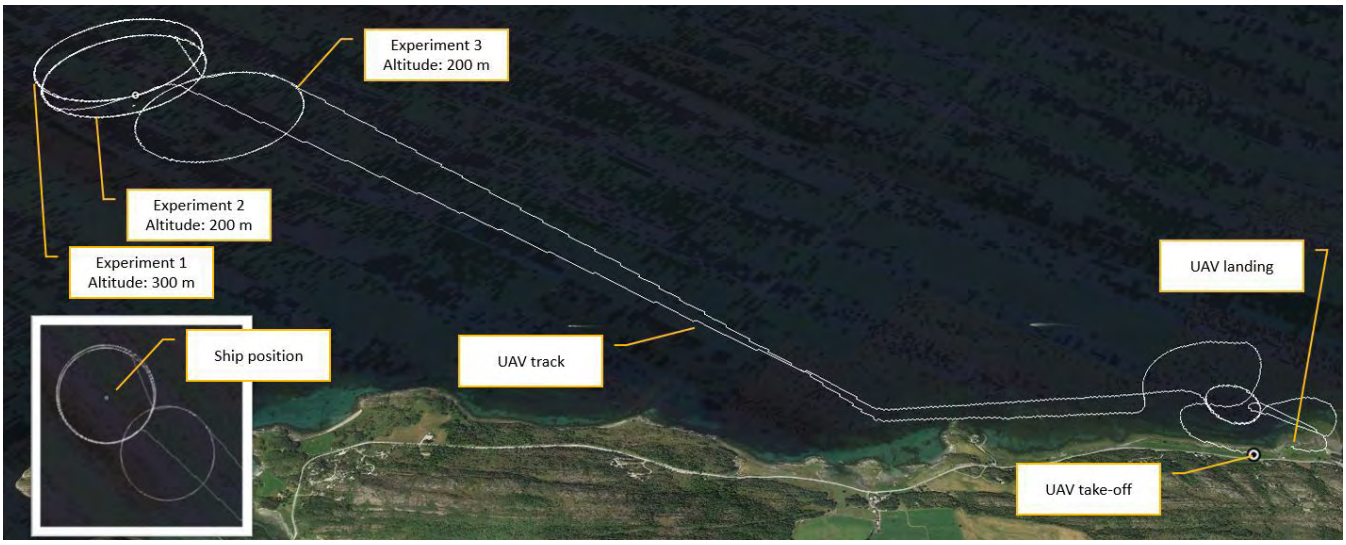

FIGURE 6. Field experiment: UAV flight overview.

was moved by ≈ 800 m away from the ship, repeating once again the network-performance tests (Experiment 3). These different tracks, and collected data, are depicted in FIGURE 5 and were used both for the validation of a new network emulator (Section IV) and of different protocols for file transferring (Section V).

III. NETWORK EMULATOR FOR INTERMITTENT LINKS

Experts in the field of autonomous, or remotely-operated, unmanned vehicles have dedicated tools for analysing scenarios' topology and trajectories, following known models to determine the quality of used communication links. However, this approach lacks a full understanding of the networking capabilities, typically ignoring the requirements and subtleties of protocols that can interconnect different systems. Alternatively, field experiments can be conducted to realistically characterise the environment, but they involve high costs and therefore limited availability of resources, which hinder the quality of such characterisation. For this reason, we developed an emulator capable of fully supporting the network stack and varying communication links, as well as the execution of real tools and components. This tool allows researchers to plan their missions without the costs and complexity of testing in the field, giving full control over all the actors and available options. With such control and flexibility it will be possible to proceed with the planning of trajectories and deployment of vehicles.

A. ARCHITECTURE

The main goal of the proposed emulator is to evaluate the performance of common networking protocols, used for connecting heterogeneous devices and technologies available in maritime scenarios. Bearing this in mind, the emulator allows shaping connectivity between different nodes in a scenario, as well as the use of real software implementations (e.g. network protocols, monitoring tools, etc.). Full emulation is achieved by using operating-system-level virtualisation, also known as *containers*, combined with Linux's process network namespace management (*netns*) and traffic control (*tc*) tools.

The shaping of communication links is derived from performance traces used as input, allowing the support of different technologies. This input may result either from field experiments or from existing tools and models. In fact, several communication technologies are likely to be interdisciplinary and highly specialised, having their own established tools and mechanisms for performance evaluation. These are perfect candidates to serve as input for the link configuration process.

FIGURE 7 illustrates the emulation tool and its components, where the input from specialised interdisciplinary tools is used for creating an emulated topology and links. Using this information, the emulation starts at a host machine, which in turn spawns as many *containers* as there are nodes in the provided topology, which in our experiment corresponds to 2 containers (i.e. 2 nodes, the vessel and the UAV). This approach allows emulating the Operating System of each node, which in our experiment was Ubuntu 16.04 LTS, as well as any other running software, in a distributed and isolated fashion.

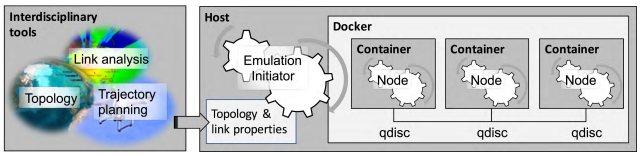

FIGURE 7. Architecture of the emulation tool.

Each emulated node has a dedicated and isolated link to each other, dynamically configured by using Linux *qdiscs*. A *qdisc* is a scheduler at the output of each network interface,

or link, that will allow traffic control by shaping flows, delaying them, or even dropping them completely. Traffic control management occurs throughout the entire emulation process.

B. IMPLEMENTATION

A key component of the developed simulator is the *container* management software. This is supported by the open-source software Docker [10], a widely used solution for the development of applications and management of dependencies. The creation of the *containers* is triggered by a *Startup* script that also initiates the *Core* script in each *container*. The *Startup* script is based on the Imunes emulator [11], which offers similar emulation possibilities. However, Imunes does not provide the support required for the continuous management of *containers*, links' characteristics and topologies.

As previously mentioned, Linux *qdiscs* are used to configure each link and shape it. This process, however, is heavily influenced by the host machine running the emulator and its kernel configuration of timer events. In particular, the resolution of the used software clock, or *jiffy*, depends on the value of the kernel-constant *HZ*, defined by the configuration parameter CONFIG_HZ, which sets the interval between each packet being shaped. With the purpose of adequately managing traffic control, the *burst* and *buffer* of each *qdisc* are respectively defined in (1) and (2).

$$burst = MTU + \frac{bitrate \cdot jiffy}{8} \tag{1}$$

$$buffer = c \cdot \left\lceil \frac{bitrate}{8 \cdot MPU} \right\rceil, \quad \forall c \in \mathbb{Z}_{>0} \tag{2}$$

These take into account the Maximum Transmission Unit (*MTU*), the Minimum Packet Unit (*MPU*), the available *bitrate*, and the system's *jiffy*. However, it is important to note that these configurations may vary on some Linux systems, in particular ones using high-resolution timers (HRTs). This depends on the used kernel version and hardware architecture, which in our system are 4.11.7 and *x86_64* respectively.

In the presented emulator, two different *qdiscs* are used, the hierarchical token bucket (HTB) and the network emulator (NetEm). These are created immediately after spawning the necessary containers and their respective interfaces. The calculated *burst* value can be used to configure the used HTB leaf class, even though it is omitted in our HRT system. The *buffer* is used to define the *limit* parameter of NetEm, with $c = 1$, representing a one second buffer. Moreover, the maximum value of *buffer* is limited to 256, corresponding to the real buffer size available in the hardware used in our experiments. The value of *limit* directly affects the HTB's buffer and represents the number of packets available at a given moment. These configurations are changed throughout the emulation, according to the used input, and define the performance of each considered communication link.

FIGURE 8 depicts the interactions between the different components run by the emulator in one *container*. After initiating the *Core* in each *container* an initial *setUp* script is run. This script can be edited to initialise any parameters or software required to perform the desired experiments (e.g. a logging tool or tcpdump). After completing the initial setup the *Scheduler* process begins. This component is responsible for parsing the input from existing communication-performance traces. As previously mentioned, these traces may result from previous in-field experiments or from simulation experiments using specialised tools such as SPLAT! [12]. It is important to note that in order to synchronise all the *containers* and processes, a time barrier is defined for all them. This barrier is based on the clock shared by all *containers* and the host computer. However, other inter-process communication (IPC) mechanisms, such as *signals*, could also have been used.

When processing the information related to topology and link characteristics, modifying relevant *qdiscs*, the *Scheduler* may also spawn an additional *Process* script that can be edited to run any required software (e.g. start a specific file-transfer). Upon completion of the *Scheduler* process, which is reported back to the core, a *tearDown* task is initiated, terminating any other processes that may have started for the experiment, as well as the entire experiment.

Apart from the inner-workings of the emulation tool itself, additional software may be required for specific scenarios. For example, data collection and delivery, or the coordinated operation of unmanned vehicles may require custom frameworks such as the LSTS Toolchain [13] or MOOS-IvP [14]. Fortunately, all this software can be installed and run simply by creating the desired *container* image and starting the software either at the *setUp* stage or during predetermined instants triggered by the *Scheduler*.

IV. EMULATOR VALIDATION

Despite being mostly based on already existing software, before confidently using emulation data, the created emulator was validated against real data. The validation process and its results, which replicate real experiments, are described in the following subsections.

A. METHODOLOGY

As described in Section II, we conducted a field experiment consisting of 3 sub-experiments, where a UAV flew in 3 different trajectories. In those experiments, network and communication performance-data was gathered, which was used as input for validating the emulator. This input allows configuring the *qdisc* parameters in real-time, shaping the traffic going through the defined links.

The validation process, for all the performed experiments, executed the same script used in the field experiment, which ran a predefined sequence of network-performance tests. These tests were based on *iperf 2.0.5* software, using the UDP protocol through the following command:

```
$ iperf -f k -y C -V -u -c 10.0.1.254 \
        -b BANDWIDTH -l PACKET_SIZE \
        -t DURATION
```

First, the script ran 10 tests, each 10 s long (*DURATION*), with a 5 s interval between each test.

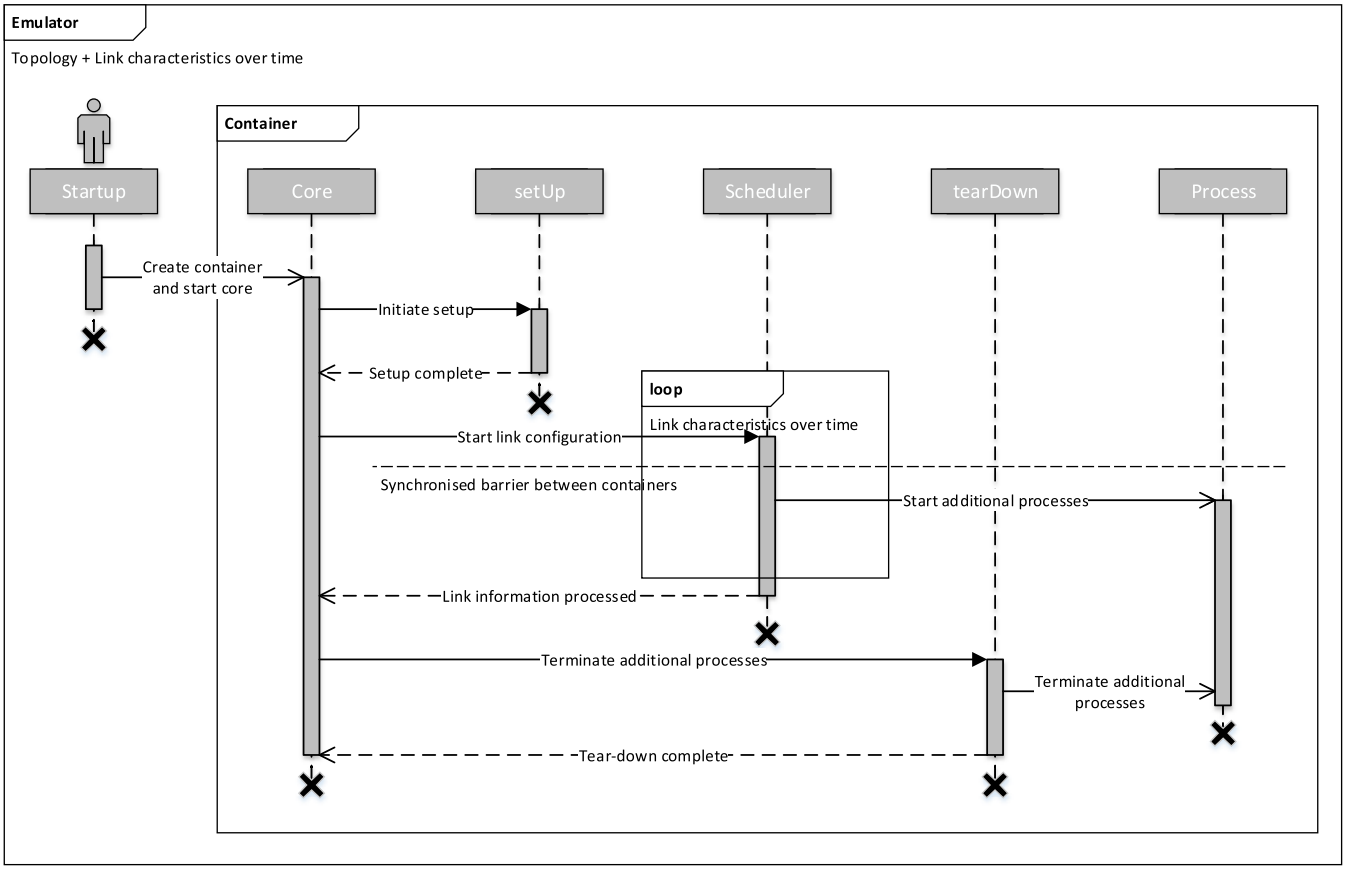

FIGURE 8. Sequence diagram for the emulator (one-node perspective).

The tests used a 20 Mbit s^{-1} bitrate (*BANDWIDTH*) and a packet size of 1000 B for every odd, and 1500 B (*PACKET_SIZE*) for every even, test. Finally, the script ran 10 tests in the same conditions but with an 80 Mbit s^{-1} bitrate. These tests were repeated for each of the 3 experiments, every time the UAV changed its trajectory. However, due to endurance and cost constraints, the performance evaluation for experiment 3 could not be fully executed.

By using the performance data obtained in the field experiment as input to the network emulator, we expect the results obtained by the network emulator to be similar. However, during the flight in the field experiment, some *iperf* reports were lost due to the intermittent behaviour of the network and to bufferbloat effects [15], [16]. Nonetheless, by combining information reported by the radio, namely the "total number of bytes transmitted", together with *iperf* data, we were able to accurately characterise the link's performance for the duration of the entire flight. With this, the emulator's *Scheduler* component was responsible to configure the previously described *qdiscs* (i.e. HTB and NetEm), namely the rate, delay and loss parameters.

B. VALIDATION RESULTS

The outcome of the validation process is depicted by FIGURE 9. In this figure the dashed line represents the

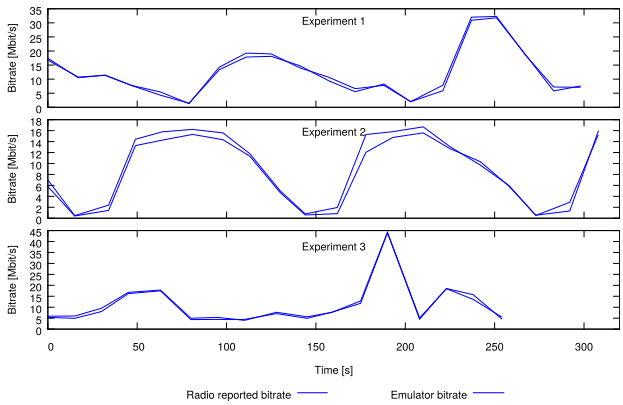

FIGURE 9. Emulator validation results.

performance results obtained from the field-experiment, while the solid line corresponds to performance registered by our emulator. These results show that the emulator performs as expected, following the trends in link quality and almost overlapping the real data, obtaining a mean absolute error (MAE) of 0.79 Mbit s^{-1}, which corresponds to 7.15% of the average bitrate. Additionally, registered packet losses are within the same level of confidence, as they are directly dependent on the achieved transmission rate, and transmission delay results directly from the distance between nodes.

The design of the emulator allows using the same software as used in real life. Consequently, variations due to the specifics of the emulator are mitigated. This means that no unexpected behaviours, such as additional overhead, or different handling of data, will ever occur.

Further analysing the obtained validation results we verify that, despite the different trajectories between each experiment, the emulator is able to reproduce the network conditions observed in the real experiment. In particular, the registered *iperf* behaviour was also matched, which resulted in the loss of the final reports for some tests. Once again, the results from those tests were obtained from monitoring the network, confirming the quality of the emulation.

V. PERFORMANCE OF DATA-MULING UAVS

An important contribution of this work is to provide a better understanding on how well UAVs perform when collecting data in remote locations. Bearing this in mind, we performed mock field-experiments and developed an emulator that allows recreating realistic networking conditions and the execution of real software. With this emulator it is possible to thoroughly assess protocols and procedures used in real hardware without the costs involved in real missions. The following subsections present an evaluation of 4 file-transferring networking protocols commonly used for the exchange of data between two nodes.

A. METHODOLOGY

Using the collected flight data for each trajectory, and replicating the observed conditions using our emulator, it is possible to evaluate different configurations and protocols for the exchange of data between two nodes. Focusing on the well-known and widely used FTP, SCP, Rsync and DTN protocols, we compare their performance under the same conditions, using the real communication-link behaviour registered during each experiment as input for the emulation. A limitation of this approach is that, while replicating realistic conditions, these actually vary in every occasion, for different vessels, different UAVs, among other factors. However, while the overall data-transfer performance might not be representative of all likely scenarios, the variations in communication and network performance due to intermittent links, and their impact on data-exchange protocols, will be similar for most cases.

The communication-performance data, in the form of input files for the emulator, was generated from the monitored radio-reports, which registered, every second, the used data rate (TX) and the Client Connection Quality (CCQ). The TX value is the same as presented in Section II, while the CCQ value, also extracted from the field experiment, indicates the quality of the link – with 100% being a perfect connection and lower values resulting from errors in transmission. In our emulation tool these values were used to define the maximum link rate, configuring the *rate* parameter from the HTB *qdisc* to match the registered TX values, as well as the number of random losses in the link, configuring the *loss random* parameter from the NetEm *qdisc* as 100% − CCQ. It is important, however, to notice that the TX value expresses a theoretical bitrate expected at a given moment, which may not always be achievable. In fact, by analysing FIGURE 10, which depicts the raw TX and CCQ values (i.e. $TX \cdot CCQ$) and the bitrate obtained by re-running the validation scripts, it is possible to verify that the theoretical values are higher than what could actually be achieved with real experiments (FIGURE 9).

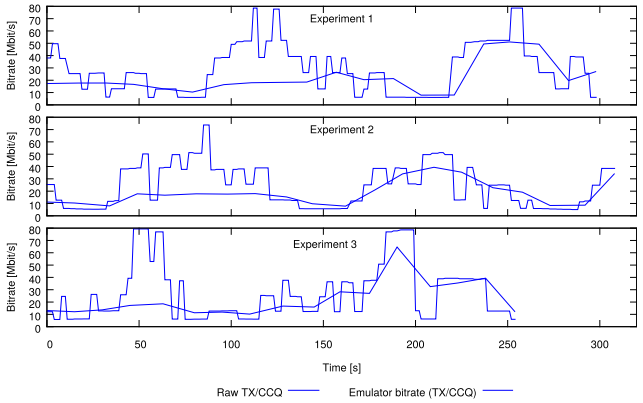

FIGURE 10. Raw TX/CCQ data and emulator results with UDP.

Despite the overestimation resulting from using optimistic performance values, these correspond to actual measurements obtained from the radio, which are commonly used in similar evaluations. Moreover, these values follow the pattern observed in the real experiment, where in the first half of each experiment the 20 Mbit s^{-1} rate tends to be more stable, and where the second half shows a more accentuated curve, for the 80 Mbit s^{-1} bitrate. Nonetheless, a second input file for the characterisation of the link was generated, increasing the error rate by a factor of 2. Since all the selected protocols rely on TCP, FIGURE 11 presents a comparison between these 2 inputs, generated from the same *iperf* script but using TCP as the transport protocol. This figure effectively shows the impact of an increased CCQ, while at the same time providing a first insight on the expected file-transfer performance.

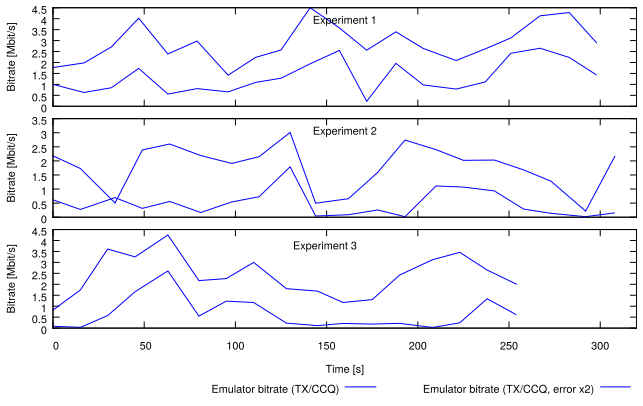

FIGURE 11. Emulator results with TCP using the measured TX and CCQ.

As previously mentioned, each phase of field-experiment (i.e. Experiment 1 to 3), lasted only a few minutes due to cost constraints and endurance limitations. However, having registered the details about the trajectory followed by the UAV, which consisted of more than one full revolution during the loitering, we were able to replicate the trajectory and obtained link performance values for each experiment. This allowed the execution of longer emulation tests without loss of generality and without the additional cost of performing longer flights. By carefully appending more loitering revolutions to each input file, without breaking the trajectory followed by the UAV, the total emulation time was of $\approx$ 30 min per experiment.

In addition to studying the impact of different trajectories on the selected networking protocols, we have also evaluated their performance when transferring different types of files. For that purpose, two sets of files were used. The first one contained 14 files of 40 MB each, totalling 560 MB, which corresponds to the amount of data to be generated by two Acoustic Zoo-plankton Fish Profilers (AZFPs), over a one week period as predicted for the ArcticABC project [17], [18]. The second set contained 560 files of 1 MB each, intended to evaluate the impact of size and number of files in the performance of each data-exchange method.

As with the validation process, the file transfer was entirely scripted, guaranteeing equal conditions between all the tested protocols. The node representing the vessel was setup with the necessary services, or daemons, which were *ssh*, *vsftpd*, *rsync* and *dtnd*. It is important to note that the SCP protocol is the only protocol operating over an encrypted connection and that Rsync uses its native protocol. In addition to *dtnd*, the DTN approach required a dedicated application for transferring files, which was *dtncp*. For the UAV node, the file transfer processes were initiated at the same instant as the *iperf* tests, for which a link is known to exist. The transfer of the files included a loop for verifying that all the files were correctly received, avoiding terminating before time due to temporary link failures. Additionally, the "partial-transfer" features offered by the FTP and Rsync protocols were enabled, namely the *continue* and *partial* options, respectively. The SCP protocol does not provide such feature and for the DTN protocol this is already a built-in functionality.

B. RESULTS

The performance of the selected protocols will be evaluated considering 2 key metrics for each experiment. The overall performance will be represented by the total number of completed file transfers (see TABLE 2 and TABLE 4), while the peak performance will be reflected by the highest achieved bitrate transfer (see TABLE 3 and TABLE 5). In this section, the results referred to as error factor x1 (EF_{x1}), are the results obtained from the input file using the raw TX and CCQ radio measurements. Similarly, results from using the input file with increased link errors will be referred to as error factor x2 (EF_{x2}).

TABLE 2. Transmission of 40 MB files.

Exp.	Maximum number of completed file transfers	Percentage of the best result			
		DTN	FTP	Rsync	SCP
Error factor x1 (EF_{x1})					
1	14	100%	100%	100%	100%
2	11	100%	100%	100%	100%
3	11	100%	91%	100%	100%
Error factor x2 (EF_{x2})					
1	9	100%	78%	89%	89%
2	4	100%	75%	50%	50%
3	4	100%	75%	75%	50%

1) DATA-SET WITH 40 MB FILES

The results obtained for the maximum number of transmitted files, presented in TABLE 2, show that the trajectory used in experiment 1 favours all protocols. This is verified both for EF_{x1} and EF_{x2}. Nonetheless, as expected, increased communication errors result in reduced complete file-transmissions. Moreover, disparities between protocols become clearer, with the DTN protocol outperforming all others in every experiment. In fact, while for EF_{x1} only FTP failed to receive 11 files in experiment 3, when link errors increase, FTP outperforms both Rsync and SCP in experiments 2 and 3, being close in experiment 1. The poor performance of SCP for EF_{x2} can be justified by the use of an encrypted connection, which requires a more complex session initiation that can be hampered by lost packets.

Regarding the peak performance of each protocol, detailed in TABLE 3, no significant differences were registered for EF_{x1}. Despite the computational overhead of encrypting each session, the SCP protocol achieved an overall higher transfer rate, even though this might not be the case in computationally-constrained settings. Following the trend of the number of transmitted files, the maximum transfer rate performance reduces to nearly a third with EF_{x2}. The SCP protocol is again the most affected, in particular for experiment 2, while Rsync and FTP have similar performances with none of them standing out.

TABLE 3. Transfer rate for 40 MB files.

Exp.	Maximum transfer bitrate [Mbit/s]	Percentage of the best result			
		DTN	FTP	Rsync	SCP
Error factor x1 (EF_{x1})					
1	2.81	94%	93%	100%	100%
2	1.75	100%	94%	98%	99%
3	2.06	98%	96%	97%	100%
Error factor x2 (EF_{x2})					
1	1.36	100%	85%	95%	94%
2	0.63	100%	72%	67%	57%
3	0.76	100%	88%	90%	70%

2) DATA-SET WITH 1 MB FILES

In order to better understand the impact of different file sizes in intermittent communication links, the performance evaluation of UAVs as data-mules was repeated with 1 MB files for the same amount of data transfer 560 MB. In particular, this gives insights about how the collected sensor data should be organised, as it typically can be composed of several small

TABLE 4. Transmission of 1 MB files.

Exp.	Maximum number of completed file transfers	Percentage of the best result DTN	FTP	Rsync	SCP
Error factor x1 (EF_{x1})					
1	560	100%	100%	100%	100%
2	475	96%	93%	100%	98%
3	462	92%	91%	100%	95%
Error factor x2 (EF_{x2})					
1	352	100%	76%	97%	95%
2	179	100%	60%	75%	71%
3	180	100%	56%	56%	83%

measurements. TABLE 4 presents the obtained results for this evaluation and the impact of using smaller files is clear. For both experiment 2 and 3, with input EF_{x1}, the percentage of complete file-transfers varies among protocols, being Rsync the best. In absolute terms, and for experiment 2, all the protocols were able to transmit more data than in the 40 MB setup (i.e. > 11 files $\cdot$ 40MB). However, for experiment 3, the performance with 1 MB files is lower than with larger ones, for all protocols except Rsync. This decrease in performance is also verified in EF_{x2}, but it affects differently each protocol. The DTN protocol shows again better resistance to increased error rates, outperforming all other protocols, in particular for experiments 2 and 3. Notably, in these conditions, the DTN protocol was able to transmit more data than in the 40 MB test, while the other protocols seem to be affected by the number of necessary transfers. However, while FTP, Rsync and SCP clearly had a worse performance when compared against using 40 MB files, the increased performance of the DTN is thought to be due to the fact that incomplete files were not considered in the 40 MB setup, even though they had been partially transferred.

TABLE 5. Transfer rate for 1 MB files.

Exp.	Maximum transfer bitrate [Mbit/s]	Percentage of the best result DTN	FTP	RSYNC	SCP
Error factor x1 (EF_{x1})					
1	2.80	91%	91%	100%	94%
2	1.76	96%	93%	100%	98%
3	2.10	92%	91%	100%	95%
Error factor x2 (EF_{x2})					
1	1.31	100%	75%	97%	95%
2	0.67	100%	60%	75%	71%
3	0.82	100%	55%	55%	83%

The results shown in TABLE 5 reveal that the 1 MB files do not have a significant impact on the maximum transfer bitrate, being very similar to the 40 MB set. Nonetheless, an individual analysis of each protocol reveals a decrease of performance for SCP in EF_{x1}, but an increase for EF_{x2}. This suggests that, when using SCP, larger files should be avoided in challenging communication-conditions. On the other hand, FTP performed worse with smaller files both in EF_{x1} and EF_{x2}, being this performance degradation more accentuated in the latter. The DTN and Rsync protocols achieved a similar transfer-rate performance, regardless of the used file size, with a slight improvement registered for Rsync in EF_{x1} and overtaking DTN. However, for EF_{x2}, the DTN protocol takes again the lead in performance, while Rsync has a decrease in performance in experiment 3, when compared against the 40 MB file sizes.

A final outcome from the obtained results concerns the use of *iperf* as a tool for assessing network performance. In FIGURE 11, TCP flows reach up to 4.5 Mbit s^{-1}, while in the file-transfer tests the maximum bitrate was of 2.81 Mbit s^{-1}. This does not invalidate the usefulness of *iperf* as a network performance tool, in a limited number of scenarios, but instead it reinforces the need for more detailed assessments, taking into account the characteristics of the chosen file-transferring network protocols and how they are used.

VI. DISCUSSION

With this paper, we demonstrate the real performance of UAVs as data mules in remote locations, using realistic settings and focusing on maritime environments. This was achieved not only by performing field experiments, but also by developing a mission planning tool for thoroughly conducting research on this topic. The obtained results showed the impact of different UAV trajectories in communication links, as well as the combined effects of the used trajectories with the selected protocols and how scientific data is organised.

Using different flight trajectories can improve results, but available trajectories for data-muling depend on several factors. First of all, the loiter radius of a UAV is defined by physical limitations of the vehicle, mainly the maximum safe *load factor* and lowest possible speed. Conventional fixed-wing UAVs, which are preferred for data-muling due to their range and endurance, require a minimum velocity to fly (stall speed V_S). Their wings create lift-force, perpendicular to their surface and, during loiter, the plane banks. To keep the flight at the same level, the vertical component of the force created by the wings needs to be constant. Therefore, the stall speed in manoeuvre increases and is equal to $V_{SM} = V_S\sqrt{n}$, where n is a Nominal Load Factor (''G''). Another concern is the lowest safe altitude (LSALT). This can be defined by regulations or, if no regulations apply, by the UAV operator's judgement.

The combination between LSALT and loiter radius determines the angle between a node and the data-muling UAV. This angle should match antenna-radiation patterns, in order to efficiently transfer data, but that may not always be possible. Therefore, the definition of flight trajectories typically takes into account the physical limitations of a plane, regulatory/safety aspects and antenna-radiation patterns. However, the obtained results show that the selection of trajectories should also take into account the performance of different networking protocols, since each trajectory affects each protocol in distinct ways.

Transferring data between two network-enabled devices can be achieved in many ways. However, when considering the practical aspects of using a UAV as a data mule in scientific missions, only a few solutions are likely to be used. The performed analysis considered well-known file-transferring

network protocols, each with its advantages and disadvantages, and revealed that the DTN protocol was, overall, the most suitable one.

The popularity of TCP is widely known and seen in all the selected networking protocols, which use it as a transport layer. Nonetheless, TCP is known to under-perform in scenarios with intermittent links, or when networking flows are unidirectional or asymmetric [19], which is likely to happen with data-muling. In fact, extensions to the DTN protocol already exist for supporting datagram-based communications (e.g. UDP or DCCP) [20]. This confirms the need for a complete emulation tool that takes into account not only link characteristics, but also the specifics of data flows and mission objectives

VII. CONCLUSION

The use of real UAVs as data mules is evaluated, flying over a research vessel for retrieving data and using different file-transferring protocols. The presented work includes a field-experiment and the steps required to reproduce the obtained results in emulated settings. The implementation and validation of a mission planning tool is provided, relying entirely on open-source tools, and achieving a mean error of 7.15% of the real bitrate.

Using real data from the field experiment, combined with the developed emulation tool, we show the performance of a UAV as a data mule. This includes an evaluation with 3 flight trajectories using different file sizes and 4 distinct, well-known file-transferring protocols. Our evaluation confirms that each flight trajectory affects the communication link in different ways. For example, higher altitudes, despite increasing the distance between the UAV and the ground node, can actually improve communication performance by avoiding on-board obstacles. Moreover, concerning the performance of the file-transferring network protocols, we verify that the DTN protocol outperforms FTP, Rsync and SCP, with all of them affected by the size of the files being transferred. In particular, the obtained results show that in scenarios with higher error-rates the performance of transferring 40 MB files is higher than when transferring 1 MB files. These results motivate further studies on how different UAV trajectories can be combined with efficient file-transferring network protocols, as well as on which data structures are most suitable for distinct scenarios.

ACKNOWLEDGEMENT

The authors would also like to thank the lighthouse project CAMOS (Coastal and Arctic Maritime Operations and Surveillance) of the Faculty of Information Technology and Electrical Engineering, NTNU.

REFERENCES

[1] O. K. Sahingoz, "Mobile networking with UAVs: Opportunities and challenges," in *Proc. Int. Conf. Unmanned Aircraft Syst. (ICUAS)*, May 2013, pp. 933–941.

[2] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in UAV communication networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1123–1152, 2nd Quart., 2016.

[3] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based Internet of Things services: Comprehensive survey and future perspectives," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 899–922, Dec. 2016.

[4] C. Dixon and E. W. Frew, "Controlling the mobility of network nodes using decentralized extremum seeking," in *Proc. 45th IEEE Conf. Decision Control*, Dec. 2006, pp. 1291–1296.

[5] E. P. de Freitas *et al.*, "UAV relay network to support WSN connectivity," in *Proc. IEEE Int. Congr. Ultra Mod. Telecommun. Control Syst. Workshops (ICUMT)*, Oct. 2010, pp. 309–314.

[6] S. Grasic, "Development and deployment of delay tolerant networks: An arctic village case," Ph.D. dissertation, Dept. Bus. Admin., Technol. Soc. Sci., Human Work Sci., Luleå Univ. Technol., Luleå, Sweden, 2014.

[7] J. Palmer, N. Yuen, J.-P. Ore, C. Detweiler, and E. Basha, "On air-to-water radio communication between UAVs and water sensor networks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 5311–5317.

[8] R. Barbatei, A. Skavhaug, and T. A. Johansen, "Acquisition and relaying of data from a floating wireless sensor node using an unmanned aerial vehicle," in *Proc. Int. Conf. Unmanned Aircraft Syst. (ICUAS)*, Jun. 2015, pp. 677–686.

[9] A. Zolich, T. A. Johansen, K. Cisek, and K. Klausen, "Unmanned aerial system architecture for maritime missions. Design & hardware description," in *Proc. Workshop Res., Edu. Develop. Unmanned Aerial Syst. (RED-UAS)*, Nov. 2015, pp. 342–350.

[10] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, Mar. 2014, Art. no. 2. [Online]. Available: https://dl.acm.org/citation.cfm?id=2600239.2600241

[11] D. Salopek, V. Vasić, M. Zec, M. Mikuc, M. Vašarević, and V. Konačar, "A network testbed for commercial telecommunications product testing," in *Proc. 22nd Int. Conf. Softw., Telecommun. Comput. Netw. (SoftCOM)*, Sep. 2014, pp. 372–377.

[12] J. Magliacane. (2017). *SPLAT! RF Signal Propagation, Loss, And Terrain Analysis Tool*. Accessed: Apr. 2017. [Online]. Available: http://www.qsl.net/kd2bd/splat.html

[13] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, and J. Sousa, "The LSTS toolchain for networked vehicle systems," in *Proc. MTS/IEEE OCEANS Bergen*, Jun. 2013, pp. 1–9.

[14] Madras Institute of Technology. (2017). *MOOS-IVP Home Page*. Accessed: Apr. 2017. [Online]. Available: http://oceanai.mit.edu/moos-ivp

[15] J. Gettys, "Bufferbloat: Dark buffers in the Internet," *IEEE Internet Comput.*, vol. 15, no. 3, p. 96, May/Jun. 2011.

[16] V. G. Cerf, "Bufferbloat and other Internet challenges," *IEEE Internet Comput.*, vol. 18, no. 5, p. 80, Sep./Oct. 2014.

[17] J. Berge, M. Geoffroy, G. Johnsen, F. Cottier, B. Bluhm, and D. Vogedes, "Ice-tethered observational platforms in the Arctic Ocean pack ice," *IFAC-PapersOnLine*, vol. 49, no. 23, pp. 494–499, 2016.

[18] J. Berge, *Mare Incognitum—Arctic ABC Project Summary*, document nFR projects 244319 and 245923, 2017.[Online]. Available: http://www.mare-incognitum.no/index.php/arcticabc

[19] M. Ramadas, S. Farrell, and S. C. Burleigh, *Licklider Transmission Protocol—Motivation*, document RFC 5325, Sep. 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5325.txt

[20] H. Kruse, S. Jero, and S. Ostermann, *Datagram Convergence Layers for the Delay- and Disruption-Tolerant Networking (DTN) Bundle Protocol and Licklider Transmission Protocol (LTP)*, document RFC 7122, Mar. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7122.txt

DAVID PALMA received the Ph.D. degree in information science and technology from the University of Coimbra. He has been a Researcher and Assistant Professor at the University of Coimbra and worked at OneSource as a Project Manager. He is currently an H2020 Marie Curie Post-Doctoral Fellow at the Department of Information Security and Communication Technology, Norwegian University of Science and Technology. His current research interests are on Routing, IoT, and software-defined networking, subjects on which he has authored and co-authored multiple papers in refereed conferences and journals. He has participated in several TPCs, national, and international research projects (FP6, FP7, and H2020), and in the preparation of successful research proposals.

ARTUR ZOLICH received the M.Sc. degree in Robotics from the Wroclaw University of Technology Wroclaw, Poland, in 2010. He is currently pursuing the Ph.D. with the Department of Engineering Cybernetics

From 2009 to 2014, he was involved on unmanned aerial systems development with the WB Group, Poland, and aspects of UAS integration into the European Airspace as a member of an ULTRA consortium, Honeywell Aerospace, Czech Republic. Since 2016, he has been a Senior Engineer with the Department of Biology, Norwegian University of Science and Technology, Trondheim, Norway. His research focuses on control and communication architectures for coordinated operations of maritime autonomous vehicles and ArcticABC project.

YUMING JIANG received the bachelor's degree from Peking University, China, and the Ph.D. degree from the National University of Singapore. Since 2005, he has been a Professor with the Norwegian University of Science and Technology. His main research interest is in the provision and analysis of quality of service guarantees in communication networks. He was the General Chair of the IFIP Networking 2014 Conference and also the author of Stochastic Network Calculus.

TOR ARNE JOHANSEN received the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Norwegian University of Science and Technology, Trondheim, Norway, in 1989 and 1994. From 1995 to 1997, he was with SINTEF as a Researcher. He was also appointed as an Associated Professor with the Norwegian University of Science and Technology, Trondheim, in 1997 and as a Professor in 2001.

He has published several hundred articles in the areas of control, estimation, and optimisation with applications in the marine, automotive, biomedical, and process industries.

Dr Johansen co-founded the company Marine Cybernetics in 2002, As where he was Vice-President until 2008. He received the 2006 Arch T. Colwell Merit Award of the SAE, and is currently a Principal Researcher within the Center of Excellence on Autonomous Marine Operations and Systems and Director of the Unmanned Aerial Vehicle Laboratory, Norwegian University of Science and Technology.