A platform for wireless maritime networking experimentation

Esmerald Aliaj, Georgia Dimaki, Petros Getsopoulos, Yannis Thomas, Nikos Fotiou,
Stavros Toumpis, Iordanis Koutsopoulos, Vasilios Siris, George C. Polyzos
Department of Informatics, Athens University of Economics and Business, Athens, Greece

***Abstract*—We present a software platform for performing wireless maritime networking experiments, with the aim of paving the way for the deployment of IoT applications that support the coast guard in its two main roles, i.e., border surveillance and search and rescue. The platform brings together, on one hand, a novel middleware application, called Dedalus, used for monitoring and controlling experiments, and, on the other hand, a number of implementations of popular protocols that perform ad hoc routing, delay-tolerant routing, information-centric networking, and encryption. The platform has been designed in a modular manner that enhances its scalability and adaptability by the wider research community. We also present preliminary experimental results that shed light on the properties of the wireless channel and the behavior of ad hoc routing protocols in uncluttered settings.**

I. Introduction

The RAWFIE (Road-, Air- and Water- based Future Internet Experimentation) 4-year-long (2015-2018) project is part of the wider EU FIRE+ (Future Internet Research Experimentation) initiative. It aims at providing a federation of testbeds for experimenting with Internet of Things (IoT) applications based on Unmanned Aerial Vehicles (UAVs), Unmanned Surface Vehicles (USVs), i.e., unmanned, autonomous boats, and Unmanned Ground Vehicles (UGVs), collectively referred to as UxVs [1]. As of now, it operates 6 testbeds in 4 countries; these testbeds are equipped with a variety of UxVs that offer outside experimenters the opportunity to conduct IoT research without having to invest the significant amounts of resources required to procure and operate UxVs.

Among the research fronts it opens, RAWFIE provides a unique opportunity for experimenters to advance the state of the art in IoT applications based on the wireless maritime networking of USVs. Indeed, there is a relative dearth of recent research in wireless maritime environments, due to a number of factors. Firstly, whereas UAVs and UGVs can be deployed practically everywhere, deploying USVs requires ready access to a large body of water, which is not possible for most research groups. Secondly, retrieving malfunctioning USVs is much harder than retrieving malfunctioning UGVs, and even UAVs, as in the former case at the very least a boat and its operator are required. Finally, in maritime settings there can be no wired infrastructure except along the coastline.

On the other hand, maritime networking offers unique opportunities for new IoT applications with significant gains to society. In particular, it can help the coast guard efficiently fulfill its two main tasks, i.e., border surveillance and search and rescue. Imagine, for example, a large fleet comprised of hundreds of small autonomous boats, patrolling a large sea border. The boats continuously act as a large sensor network, monitoring the area of interest but also weather conditions, the status of the boats, and the network as a whole (in order, for example, to identify holes in the coverage, etc.). Every now and then, one of the boats discovers a vessel in distress, or a possibly unauthorized vessel. In this case, the whole network will have to switch gears: the command center will have to be notified, using robust multihop links, perhaps some of the nodes will change their trajectory in order to serve the communication needs of the network better and/or to scour the vicinity of the vessel for persons in distress and provide assistance, etc.

For such a network to be realized, a number of wireless networking technologies must be brought together. First of all, in order to conserve energy and make the detection of the network operations harder, the communication would have to be over low-powered radios, resulting in a multihop topology, necessitating the use of ad hoc routing protocols. Furthermore, delay-tolerant routing protocols will also have to be used, firstly in order to give the capability to the nodes to act as data mules (thus avoiding detection) and secondly in order to provide robustness against network partitioning. Also, many of the application scenarios that the network will have to support can be carried efficiently using information-centric networking (ICN) protocols. As an example, an application (user) might be interested in asking for specific information, no matter the node, (e.g., "temperature at area X"), or it may be desirable to multicast questions, such as "which node currently has critical battery levels", which could be distributed in the network using ICN protocols. Finally, intruders would naturally be interested in intercepting the network traffic; it is also conceivable that an intruder will be able to get hold of a node and use it to eavesdrop and/or corrupt the operation of the whole network. This calls for using encryption protocols, such as Identity-Based Encryption (IBE), a solution that relieves network operators from the need to manage security certificates and fits very well ICN architectures.

Responding to this vision for an IoT-enabled coast guard, and as part of the RAWFIE framework, UNSURPASSED (Unmanned Surface Vehicles as Primary Assets for the Coast Guard), an ongoing one-year-long project, was initiated, after an open call for projects that will enhance the scope of RAWFIE. UNSURPASSED has two aims: Firstly, to enhance

the capabilities of the RAWFIE testbeds in terms of wireless networking, by integrating a variety of wireless networking protocols in their framework. Secondly, to conduct experiments, using these protocols, specifically in a maritime setting, in order to ascertain the maturity of the available technology to support IoT applications for the coast guard [2].

UNSURPASSED differs from most other efforts at developing wireless networking testbeds, such as those described in [3], [4], [5], [6], [7], in two respects. Firstly, its software platform is compatible with the wider RAWFIE architecture, allowing experimenters to either use it independently, or as part of a RAWFIE testbed, in which case the wider RAWFIE infrastructure becomes readily available. Secondly, no kind of wired infrastructure is needed, apart from APs connected to the internet for use by the control plane, thus making the platform suitable for use in maritime environments.

The rest of this work is organized as follows: in Section II we present the system architecture that UNSURPASSED adopted, which, in particular, takes into account the wider RAWFIE architecture. In Section III we present the wireless networking protocols thus far integrated in the system architecture. In Section IV we present results on throughput measurements, which are preliminary (as the project is ongoing) but shed light on the properties of the maritime wireless channel and the effects of the uncluttered environment. We conclude in Section V. We note that a demonstration of the system architecture and its capabilities will take place in ACM WiNTECH 2018 [8].

II. System Architecture

As the aim of UNSURPASSED is to conduct research using jointly a number of different wireless protocols, it is necessary to develop middleware that will allow their robust and efficient coordination and monitoring. To this effect, we have developed *Dedalus*, a network monitoring and controlling application named after the mythical architect Daedalus. Dedalus is designed for use as part of RAWFIE testbeds, in which case it is seamlessly integrated with them, taking advantage of some of their features, as we explain later on. However, it can also be used outside of the RAWFIE infrastructure.

As shown in Fig. 1, Dedalus is comprised of three types of components, i.e., *workers*, *clients*, and *brokers*. In short, clients issue commands to the workers about creating traffic loads in the network and request information about its performance from them. Communication between the workers and the clients is mediated by the brokers. Below, we discuss the role of each of these components in more detail.

There is a single worker for each wireless node, acting as middleware between the experimenters and the various wireless protocols installed at that node. Workers interpret the commands for creating and routing traffic, which are given by the clients in a unified manner, irrespective of the protocols in operation; they scan the software and hardware on their nodes, making sure, for example, that all services are running; they report usage statistics and specific protocol information, etc.

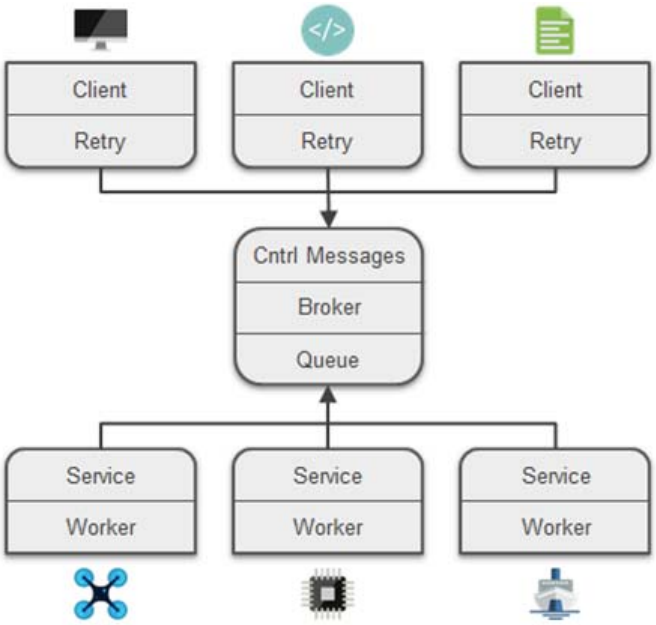

Fig. 1. The Dedalus architecture.

Clients request information from workers about their node status and issue commands regarding traffic that must be created. Clients can run on any kind of hardware and be written in any kind of programming language. They can be anything from a simple logger to a fully-fledged application with its own GUI. In order to control and monitor our experiments, we have developed such a GUI-enabled client, which allows the experimenters to monitor the topology of the network online, issue commands to the workers, and monitor their progress and status. A snapshot of the GUI appears in Fig. 2.

Finally, brokers handle the communication between clients and workers. Notably, they maintain queues of messages destined for the various workers and clients, making sure that no message is lost, and do basic book-keeping, such as keeping a list of the available workers and their current status, for reporting to the clients. In order to provide robustness, there can be multiple brokers at any time, however one broker per experiment is sufficient.

The architecture is modular, with scalability in mind. For example, if an experimenter using Dedalus would like to experiment with a new ad hoc routing protocol, she would have to create an interface between the routing protocol and the worker application, so that workers can activate the protocol, but no other modification to code would be needed.

Clients, brokers, and workers communicate through a dedicated control plane that has one of the following two arrangements: for experiments not in a RAWFIE testbed, communication is through a wireless LAN, which may comprise of multiple APs and a single SSID. Therefore, the experiment area is not geographically limited, and could cover, for example, a whole city, provided a correspondingly large network of APs with a common SSID exists. For experiments in a RAWFIE testbed, communication is through the wireless LAN of the

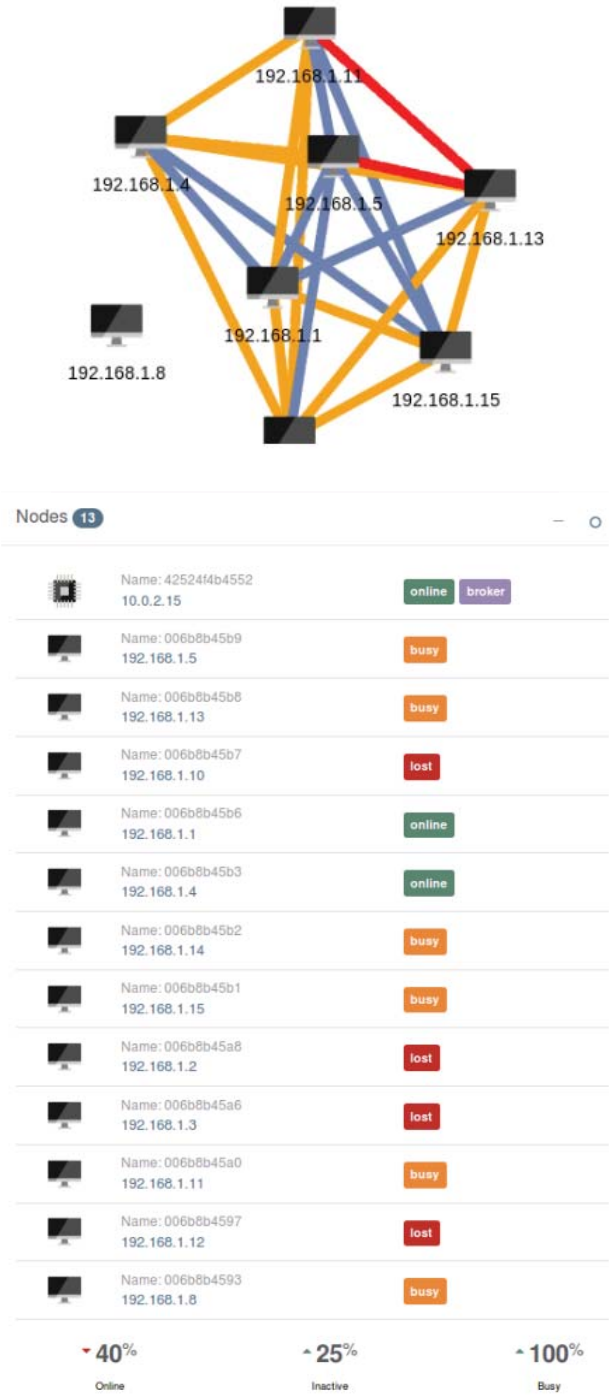

Fig. 2. Snapshots of the Dedalus client GUI. In the upper snapshot, the color of the edges provides a quick visualization of the quality of the respective wireless link

testbed and then an Apache Kafka server. According to the Kafka architecture, different software components communicate with each other by posting and consuming messages in specific topics maintained by the server. In our case, there is a single topic for each worker through which workers are issued commands and are requested information. An important advantage of using this arrangement is the fact that with a minimal programming cost the various Dedalus components can access information available in other Kafka topics maintained in the RAWFIE testbed, by other experimenters. For example, the wireless nodes can learn of their physical location by consuming GPS information published by the USVs on which they are attached.

In order to evaluate the performance of the wireless network, experimenters instruct the nodes to create traffic, using TCP or UDP connections, or single packets. This can happen in three distinct ways. Firstly, but only for experiments in a RAWFIE testbed, the instructions can be given using a script in the Experiment Description Language (EDL), developed by RAWFIE; the same script also specifies the mobility of the USVs. Secondly, the instructions can be given online, using the client GUI (see Fig. 2). Thirdly, scripts can be installed on the nodes, either before the experiment, or while the experiment is running.

III. Implemented Protocols

In this section, we discuss wireless protocols already integrated at the UNSURPASSED platform. It should be stressed that a main goal of the platform is to ensure that more protocols can be integrated at a latter date, also by other researchers, in a manner that is as straightforward and efficient as possible.

A. Ad hoc Routing

The lowest-level protocols supported by Dedalus are ad hoc routing protocols. Their aim is to provide connectivity to nodes that are in the same network partition. Once the experimenter specifies the ad hoc routing protocol that nodes will use, either online, through the GUI, or offline, through a script, the protocols work by expanding the IP tables of the nodes, enabling multihop communication.

Two ad hoc routing protocols have been integrated to the platform so far. The first one is the BatMan-eXperimental version 7 (BMX7) routing protocol [9]. As with other routing protocols, topology information is announced by nodes, however an effort is made to keep these announcements at a minimum level, by using aggregation and broadcasting, and avoiding replication. BMX7 makes exclusive use of IPv6 addresses and operates on layer 3. The second protocol we have integrated in our testbed is Babel [10], a distance-vector routing protocol that contains a number of features aimed at avoiding the formation of loops and black holes due to node mobility, through the use of sequenced routes. Babel is proactive, in the sense that routes are maintained by periodic advertisements, even when not in use. Both protocols have received significant amounts of attention by researchers

Fig. 3. One of the USVs used in the experiments. Note the zip-locked clear plastic bag containing a wireless node hanging from the upper horizontal bar.

and experimenters [11], and thus were natural choices for integrating in Dedalus.

B. Delay-Tolerant Routing

The second-lowest level protocols supported by Dedalus are delay-tolerant networking (DTN) protocols. Their aim is to provide connectivity between nodes even when these are not in the same partition. Note that these protocols work seamlessly with the ad hoc routing protocols at the lower level: if nodes also use an ad hoc routing protocol, the DTN protocol treats all nodes in the same partition as direct neighbors, as all of the nodes of the partition appear in the IP table of the current packet holder; if no ad hoc protocol is used, packets destined to nodes in the same partition that are not direct neighbors of their current holder are routed using the DTN protocol, similarly to packets destined for other partitions of the network.

One DTN protocol has been integrated to the platform so far, i.e., the IBR-DTN routing protocol [12], [13], which implements both the Bundle DTN protocol (RFC 5050) as well as the Bundle Security Protocol (RFC 6257). An important feature of IBR-DTN is the fact that the experimenters can select a variety of DTN routing methods, notably the ProPHET protocol, flooding, static routing, and epidemic routing using Bloom filters.

C. Information-Centric Networking

The next-level protocols perform Information-Centric Networking (ICN). The ICN paradigm advocates the use of information as the main primitive of all networking functions; for a survey on ICN architectures, interested users are referred to [14]. In this work, we are focused on a particular ICN architecture, namely Content Centric Networking (CCN). In CCN users issue "interest" messages to request content items. Interest messages are forwarded hop-by-hop by intermediate nodes; each node maintains three data structures: the Forwarding Information Base (FIB), the Pending Interest Table (PIT) and the Content Store (CS). The FIB maps content names to network interface(s) that should be used to forward interest messages, the PIT tracks the incoming interface(s) from which pending interest messages have arrived, and, finally, the CS acts as a local cache. When an interest message arrives, a forwarding node examines if the requested item exists in its CS (i.e., the local cache). If something is found, it is immediately sent back to the user. Otherwise the interest is forwarded based on the information included in the FIB.

Regarding the particular implementation of CCN, we are using a modified version of CCN-lite [15], a lightweight implementation of the CCN architecture. In our test scenarios CCN-lite is used on top of IP, as an overlay network. We assume the content prefixed with a USV-specific identifier. Furthermore, all USVs are configured with a static IP address and the mapping from an identifier to the corresponding IP address is considered well known. For our experiments we have adapted CCN-lite so as to take advantage of the underlay ad hoc protocol. In particular, initially we populate the FIB entries of all nodes with all content name prefixes. A daemon periodically polls the Dedalus worker and learns the next hop towards a name prefix. Then, this daemon populates the FIB table accordingly. Therefore, even if there is node mobility, the FIB table is updated with a correct, reachable next hop. Whenever a node receives an interest packet, it follows the standard CCN approach, i.e., if it has the requested content cached, it responds with a data packet, otherwise it forwards the interest towards the appropriate node.

D. Identity-Based Encryption

Finally, on the highest level, and in order to secure the underlay communication, we employ the Identity-Based Encryption (IBE) scheme of Green and Ateniese [16]. An IBE scheme is a public key encryption scheme in which an arbitrary string can be used as a public key.

For our security experiments we are using the implementation of the Green-Ateniese scheme included in the Charm crypto cryptographic library [17] (implemented by our team). Each boat in our experiments is identified by a unique ID: the keys that correspond to each ID have been generated and distributed to the boats offline (i.e., the Setup and Extract algorithms). Just like any other public key encryption keys, the Encryption and the Decryption IBE algorithms are computational intensive; for this reason in order to encrypt a file F the following approach is used:

1) A random symmetric encryption K is generated.
2) F is encrypted using a symmetric encryption algorithm (e.g., AES) and K.
3) K is encrypted using the IBE Encrypt algorithm.

For decrypting the ciphertext the reverse process is followed.

In our experiments we are using 128-bits symmetric encryption keys. Encrypting such a key using IBE results in a 546 bytes ciphertext (base-64 encoded). The encryption algorithm in our PIs requires 156 ms, whereas the decryption algorithm requires 96 ms.

IV. Preliminary Experiments in the Skaramagas Testbed

In this section, we describe preliminary experimental results obtained in one of the RAWFIE maritime testbeds, i.e., the Skaramagas testbed.

Regarding the hardware used, we have installed Dedalus workers and the experimental protocols in 10 nodes, each comprised of a Raspberry Pi 3 Model B, an external Dual-Band WiFi Dongle, and a power bank. Six of these nodes were placed inside water-tight plastic zip-locked bags, and were fastened on Flexus USVs, as shown in Fig. 3; the rest were kept at the shoreline. The boats were then put to sea, and controlled either through the EDL language, or remote control.

Regarding communication, the external WiFi Dongle is used for communication in the control plane; at any time, it is connected to one of the approximately ten APs of the testbed, which are spread over the shoreline; the ten APs operate in both the 2.4 GHz and 5 GHz bands, each AP using one channel of either of the two bands, automatically selecting that channel in order to minimize interference from adjoining APs. On the other hand, the internal 2.4 GHz WiFi network interface is used (with a channel left unused by the control plane) in order to support traffic in the data plane, so that traffic carried by experimental protocols is carried by that interface.

A primary concern in maritime networking is the quality of the links over salt water. Towards this goal, and as a preliminary step, we have performed the following experiment: we placed one node on a boat, and another node on the shoreline. We then instructed the boat to distance itself from the shoreline, while measuring the throughput between the two. We then repeated the experiment, but this time between a node placed on the shoreline and another node moving over the ground, and parallel to the shoreline. Throughput was measured by instructing Dedalus, over the client GUI, to open a TCP connection and flood it with packets, for approximately 15 seconds, i.e., enough time for TCP to converge to the maximum sustainable throughput.

The results of this experiment appear in Fig. 4. Two features of this figure stand out: firstly, the sea link is much more volatile, and thus less dependable; we attribute this partly to the rocking movement of the boat. On the other hand, the average throughput measurements of the two links are roughly the same. These observations, taken jointly, imply that a maritime network will perform just as well as a terrestrial network, provided there is enough redundancy built in to cancel out the effects of deep fades.

Another observation to come out of the preliminary experiments was the fact that the wireless network avoided the use of multiple-hop links, preferring the use of a minimum

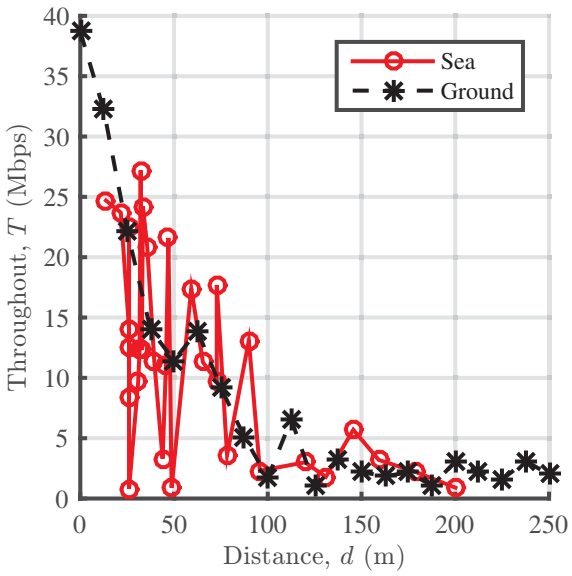

Fig. 4. Throughput versus distance in the case of a sea and a ground link.

number of links, even at the cost of significantly reduced throughput. In Fig. 6 we plot throughput measurements, again of TCP connections, versus the number of hops that the TCP connection spans. For these experiments, there were at most 6 nodes on the sea (see Fig. 5) and at most 4 more nodes spread on the shoreline, for a maximum of 10 nodes at any given time. It is evident that connections with 3 hops were fewer and had a much slower throughput with respect to connections with 1 or 2 hops: The average throughout of 1-hop connections was 8.53 Mbps, the average throughout of 2-hop connections was 4.57 Mbps, and the average throughout of 3-hop connections was 0.29 Mbps. No 4-hop connections were measured, despite the fact that efforts were made to spread out the 10 nodes as much as possible.

We conjecture that the behavior is explained as follows: each network interface can transmit in any of a fixed number of modulation schemes, selecting lower-rate ones if packets are lost. Therefore, as the nodes spread out, all interfaces resort to using the lowest-rate modulation scheme, in an effort by each node to listen to all other nodes, however distant. As a result, nodes never switch to using high-rate routes with many hops as opposed to low-rate routes with fewer hops. Also, 3-hop routes are invariably low-throughput as the nodes participating in the route always use low-rate modulation schemes. This situation is exacerbated by the fact that the environment is uncluttered, meaning that all nodes are continuously within line-of-sight of each other; if, on the other hand, the environment was more congested, each node would only be able to communicate with a subset of nearby nodes, making it easier both to resort to higher data rates and observe routes with numerous hops.

V. Conclusions

In this work we presented a software platform under development for performing wireless networking experiments

Fig. 5. A snapshot of an experiment in progress, with a total of 6 USVs.

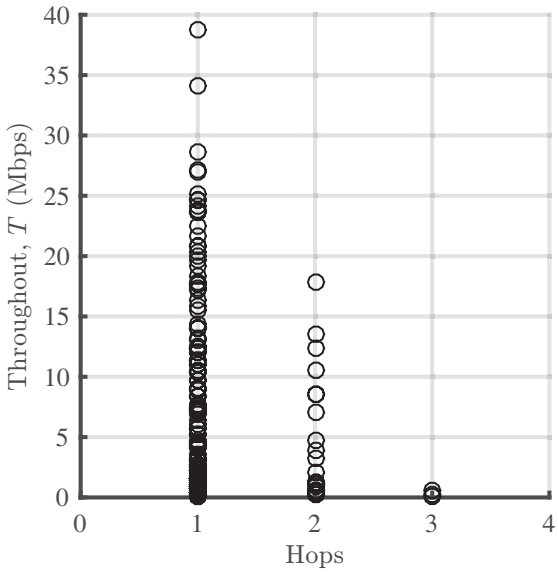

Fig. 6. Throughput versus the number of hops

involving multiple experimental protocols. The platform is designed to operate seamlessly with the wider RAWFIE infrastructure, and especially in its maritime testbeds. It is a tool through which intuition can be gained and our understanding of the challenges that must be addressed by maritime IoT applications can be improved, as our preliminary results and their analysis demonstrate.

Future work includes adding more ad hoc routing protocols to the platform as well as a variety of applications with different Quality of Experience (QoE) requirements.

ACKNOWLEDGEMENTS

This work has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement No. 645220 (Road-, Air- and Water-based Future Internet Experimentation - RAWFIE) through the National and Kapodistrian University of Athens. Administration support was provided by the Research Centre of the Athens University of Economics and Business.

Also, we gratefully acknowledge the help and support of the Hellenic Navy and the National and Kapodistrian University of Athens personnel and researchers participating in RAWFIE.

REFERENCES

[1] www.rawfie.eu.
[2] https://mm.aueb.gr/projects/unsurpassed.
[3] J. Horneber and A. Hergenröder, "A survey on testbeds and experimentation environments for wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1820–1838, 2014.
[4] K. Pechlivanidou, K. Katsalis, I. Igoumenos, D. Katsaros, T. Korakis, and L. Tassiulas, "NITOS testbed: A cloud based wireless experimentation facility," in *Teletraffic Congress (ITC), 2014 26th International*. IEEE, 2014, pp. 1–6.
[5] A.-S. Tonneau, N. Mitton, and J. Vandaele, "How to choose an experimentation platform for wireless sensor networks? A survey on static and mobile wireless sensor network experimentation facilities," *Ad Hoc Networks*, vol. 30, pp. 115–127, 2015.
[6] R. P. Karrer, I. Matyasovszki, A. Botta, and A. Pescapé, "Experimental evaluation and characterization of the magnets wireless backbone," in *ACM WiNTECH*. ACM, 2006, pp. 26–33.
[7] ——, "Magnets-experiences from deploying a joint research-operational next-generation wireless access network testbed," in *Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom 2007. 3rd International Conference on*. IEEE, 2007, pp. 1–10.
[8] E. Aliaj, G. Dimaki, P. Getsopoulos, Y. Thomas, N. Fotiou, S. Toumpis, V. Siris, I. Koutsopoulos, and G. C. Polyzos, "Demo: Wireless maritime networking experiments with Dedalus," in *ACM WiNTECH*. ACM, 2018.
[9] https://github.com/bmx-routing/bmx7.
[10] https://www.rfc-editor.org/info/rfc6126.
[11] https://www.battlemesh.org/.
[12] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf, "IBR-DTN: A lightweight, modular and highly portable bundle protocol implementation," *Electronic Communications of the EASST*, vol. 37, 2011.
[13] https://github.com/ibrdtn/ibrdtn.
[14] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, G. C. Polyzos *et al.*, "A survey of information-centric networking research." *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.
[15] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, "Information centric networking in the IoT: experiments with NDN in the wild," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*. ACM, 2014, pp. 77–86.
[16] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Applied Cryptography and Network Security*. Springer, 2007, pp. 288–306.
[17] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.