This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications

CARMA: Channel-aware Reinforcement Learning-based Multi-path Adaptive Routing for Underwater Wireless Sensor Networks

Valerio Di Valerio, Francesco Lo Presti, Chiara Petrioli, Luigi Picari, Daniele Spaccini, and Stefano Basagni

***Abstract*—Routing solutions for multi-hop underwater wireless sensor networks suffer significant performance degradation as they fail to adapt to the overwhelming dynamics of underwater environments. To respond to this challenge, we propose a new data forwarding scheme where relay selection swiftly adapts to the varying conditions of the underwater channel. Our protocol, termed CARMA for Channel-aware Reinforcement learning-based Multi-path Adaptive routing, adaptively switches between single-path and multi-path routing guided by a distributed reinforcement learning framework that jointly optimizes route-long energy consumption and packet delivery ratio. We compare the performance of CARMA with that of three other routing solutions, namely, CARP, QELAR and EFlood, through SUNSET-based simulations and experiments at sea. Our results show that CARMA obtains a packet delivery ratio that is up to 40% higher than that of all other protocols. CARMA also delivers packets significantly faster than CARP, QELAR and EFlood, while keeping network energy consumption at bay.**

***Index Terms*—Underwater Wireless Sensor Networks, multi-path routing, reinforcement learning, in-field experiments.**

I. Introduction

UNDERWATER Wireless Sensor Networks (UWSNs) have garnered remarkable attention as a viable alternative to cable-based underwater deployments. While the latter are notoriously very expensive, and therefore used mostly by the primary telecommunication industry, UWSNs leverage the more and more reasonable pricing and increasing reliability of underwater wireless communication technologies, ranging from acoustic to optical. Many applications, including marine monitoring, port surveillance and security, underwater biology and discovery and protection of marine archaeology, among others [1], are all possible because of the new design of high performance devices and protocols for UWSNs capable of covering large regions of the underwater world [2], [3].

Designing reliable and efficient solutions for underwater multi-hop wireless networking is quite the new challenge, in that solutions for terrestrial networks cannot be adapted to work underwater. The main culprit is the very nature of the prevailing underwater channel, namely, the acoustic channel, which is beset by long propagation delays, low bandwidth, overwhelmingly fast dynamics, slow signal attenuation, and asymmetric links, among other impairments. In actual UWSN deployments, for instance, it is common to observe that, independently of the device transmission power and of their vicinity, two nodes might not be able to communicate at all, or that the channel is clearly asymmetrical. What makes matters even more complicated, is that these conditions change quickly. As such, designing protocols for underwater requires techniques very different from those successfully used for terrestrial networking. The networking protocols for UWSNs proposed so far attempt at addressing these unique features of underwater communications [3]. However, they fall short of successfully tackling the ever changing conditions of the network, including the varying traffic and, even more compelling, link quality and general environmental settings [4], [5].

In this paper, we propose a new design for routing aimed at tackling the different forms of variability of the underwater world directly. Our protocol is termed CARMA for Channel-aware Reinforcement learning-based Multi-path Adaptive routing. CARMA is guided by a reinforcement learning framework carefully defined to include network and environmental dynamics in the selection of the relays designated to forward a packet. By dynamically adapting the size of the set of relays to current channel conditions, CARMA automatically switches between fast and energy efficient single-path routing (in favorable network conditions) and more robust multi-path routing (unfavorable forwarding). The framework cost function is designed for optimizing route-long energy consumption without sacrificing packet delivery ratio, thus providing long network lifetime together with reliability and robustness.

The multifold contributions of this paper include the following.

- We define CARMA as a multi-path routing scheme that adaptively determines the set of next hop relays depending on channel conditions and route-long costs. Size and composition of the relay set change not only on a per packet basis, but also at each transmission attempt. In other words, if a packet needs to be retransmitted multiple times—an indication of unfavorable channel conditions—the size of the relay set is increased to favor packet delivery and its composition is geared to minimize energy consumption, *every time* the packet is retransmitted. As we will demonstrate later in the paper, the very idea of dynamically determining both size and composition of the relay set affords CARMA remarkable packet delivery ratio at a reasonable energy cost.

Valerio Di Valerio, Chiara Petrioli, Luigi Picari and Daniele Spaccini are with the Department of Computer Science, University of Rome "La Sapienza," Rome, Italy. E-mail: {divalerio, petrioli, picari, spaccini}@di.uniroma1.it

Francesco Lo Presti is with the Department of Civil Engineering and Computer Science, University of Rome "Tor Vergata," Rome, Italy. E-mail: lopresti@info.uniroma2.it

Stefano Basagni is with the Institute for the Wireless Internet of Things at Northeastern University, Boston, MA, U.S.A. E-mail: basagni@ece.neu.edu

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications

- Choosing size and composition of the relay set is formulated as a decentralized reinforcement learning problem. Its solution enables nodes to make optimal routing decisions at runtime locally. The learning problem at each node is based on a computationally efficient model that considers the number of retransmissions of each packet and link transmission probabilities determined by efficiently monitoring link quality. The cost function of our learning framework explicitly considers energy consumption (for minimizing route-long energy cost) and includes a penalty for dropping packets (for increasing packet delivery ratio). CARMA is amenable to efficient implementation as it requires limited exchange of information to keep estimates of model parameters. Particularly, information needed by the learning machinery is piggybacked to the data (in the packet header) and it is distributed among neighboring nodes via overhearing.
- We evaluate the performance of CARMA through simulations in realistic scenarios, and compare its performance to state-of-the-art routing for UWSNs. We select routing solutions that are exemplary of different approaches to underwater routing: CARP represents single-path, cross layer protocols that are channel aware [6]. QELAR is an example of single-path solutions for underwater routing based on machine learning [7], and EFlood, an enhancement of plain flooding specific for UWSNs, represents multi-path routing [6]. CARMA, CARP, QELAR and EFlood have been implemented in SUNSET SDCS, a software suite specifically designed for underwater protocol design and testing [8]. Results in networks of different size and varying traffic show that CARMA outperforms all other protocols in all considered metrics. Particularly, it obtains a packet delivery ratio that is up to 40% higher than that of all the other protocols, delivers packets significantly faster than CARP, QELAR and EFlood, and keeps the network energy consumption reasonably low.
- The performance of CARMA, CARP and EFlood has been also compared through experiments at sea. Results show that CARMA achieves better packet delivery ratio, energy consumption, end-to-end latency and data throughput than CARP and EFlood. It is also robust in that it obtains consistently good performance in spite of time varying channel conditions. To the best of our knowledge, our campaign of experiments is the first to provide a comparative performance evaluation of protocols with different design characteristics. Results do not only offer quantitative evidence of the superiority of CARMA over other approaches, but also testify to the practicality of our reinforcement learning framework.

The reminder of the paper is organized as follows. Section II summarizes state-of-art on routing for underwater wireless sensor networks. In Section III we present CARMA detailing the network scenario, packet handling operations and the reinforcement learning-based routing model that drives protocol operations. Section IV illustrates experimental results. Finally, Section V concludes the paper.

II. Related Works

We report on previous works related to concepts and techniques used in this paper, namely, routing protocols for UWSNs and underwater solutions for data delivery that make machine learning-based choices.[1]

Routing protocols for UWSNs have been proposed for over a decade now. Recent solutions include [6], [12], [13], [14]. For details on these protocols, and several more, the reader is referred to surveys on the subject, such as those by Li et al. [15] and more recently by Khan et al. [16]. Among the many solutions, one that stands out in terms of overall remarkable performance is the Channel-aware Routing Protocol (CARP), exploiting link quality information for successful data delivery to the sink [6]. Nodes are selected as relays based on their link quality, hop count and residual energy. CARP utilizes a channel reservation mechanism à la RTS/CTS for channel access and for selecting packet relays (cross layer design). For this reason, while being reliable and limiting packet collisions, it incurs noticeable latency. Also, in networks with high traffic, nodes often fail to obtain rights to access the channel, which results in low packet delivery ratio. As described in this paper, CARMA uses a different approach to routing, smartly choosing multiple relays to jointly optimize route-long energy consumption and packet delivery. Our protocol equals the performance of CARP on collisions through its reinforcement learning framework for dynamic relay set selection, while at the same time achieving noticeably better PDR, latency and energy consumption (Section IV-C and Section IV-D).

Reinforcement learning has been extensively used for routing in multi-hop wireless networks, including wireless ad hoc networks, wireless sensor networks and cognitive radio networks and more recently for routing in UWSNs [17], [18], [19], [20], [7], [21]. The advantage of learning-based routing stems from its determining optimal routing policies online, thus achieving and keeping route (semi) optimality in a dynamic environment [22]. Furthermore, learning algorithms are often amenable to distributed implementation and their communication requirements can be made relatively small (e.g., through overhearing). These are all critical as well as desirable features for the highly variable and resource constrained UWSNs environment. Solutions for underwater networks such as those presented in [18], [19], [17], [20] concern specific scenarios that are not similar to the scenario considered here. Particularly, the protocol MARLIN-Q presented in [18] proposes a solution for quality-of-service-based data delivery in multi-modal networks, namely, in networks whose nodes use multiple communication devices, such as multiple acoustic modems and optical transceivers. The HYDRO protocol concerns single-route data forwarding

[1] We acknowledge that machine learning in its many incarnations has been applied to terrestrial wireless networking in general, and to routing problems in particular, for a long time now—surveys abound [9], [10], [11]. However, based on our research experience on both terrestrial and underwater networks, we have learned that the design of solutions for these two very different environments is necessarily different. So much, in fact, that comparison between underwater and terrestrial solutions would be scarcely informative and devoid of insights. For this reason, in this paper we focus our literature review on machine learning-based solutions that explicitly address the challenges of underwater wireless networking.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications

in networks whose nodes are powered by different forms of energy harvesting [19]. As such, the main concern of that routing strategy is to find routes of nodes with the smallest probability of running out of energy. Solutions presented in [17], [20] are also for networks with intermittent connectivity. A learning-informed routing protocol designed for scenarios similar to those considered in this paper is QELAR, by Hu and Fei [7]. Based on a model-based *Q-learning* approach, QELAR aims at maximizing the residual energy of the network nodes. Relays are chosen depending on the energy they can save by locally running a learning framework whose cost function accounts for the residual energy of each node as well as for the energy distribution among neighboring nodes. QELAR is thus a solution that compares well with previous protocols, especially in terms of network lifetime. However, its Q-learning model leads to routing decisions that are prone to packet loss and to unfairness, especially to nodes far away from the sink. As observed in Section IV-C, this leads to degraded performance, especially in larger networks.

CARMA reaps the joint benefit of adaptive multi-relay routing and machine learning-based optimal routing policy for overcoming the limitation of both CARP and QELAR. The performance evaluation provided in this paper clearly demonstrates the effectiveness of CARMA in obtaining reliable and energy efficient data delivery in a set of scenarios considerably wider than that of CARP and QELAR.

III. CARMA

A. Network scenario

We consider a static underwater wireless sensor network (UWSN) made up of N nodes whose sensors produce data to be delivered to the network data collection point (the *sink*), for processing and/or further forwarding. Nodes are generically indicated by i and j. For each node i, with N_i we indicate the set of its $n_i = |N_i|$ neighbors, i.e., the nodes that can receive node i transmissions. We notice that, given the notorious asymmetry of the underwater acoustic channel, the fact that node $j \in N_i$ does not imply that node $i \in N_j$. Because of the extent of node deployment and the time-dependent dynamics of the underwater channel, not all nodes can directly communicate with the network sink, i.e., data packets may travel multi-hop routes. A sketch of a UWSN scenario is depicted in Figure 1.

Every node is equipped with a half-duplex omnidirectional acoustic modem for data transmission. The sink is shown close to surface, in the upper right corner of the picture, and has capabilities to transmit data to stations on shore.

B. Packet handling

Routing according to CARMA happens through a flexible, smart multi-path scheme. When a node has a packet to transmit, it chooses the most suitable *set* of relay nodes among its neighbors and it transmits the packet to all of them. Size and composition of the relay set can change at each transmission attempt, in pursuit of the specific objective of optimizing energy efficiency and packet delivery ratio. The range of choices varies from forwarding the packet to a single relay, minimizing energy consumption and network traffic at the cost of decreased end-to-end delivery, to broadcasting the packet to all the sender's neighbors, thus maximizing transmission reliability at the cost of increased energy consumption and traffic. In between, by carefully adjusting the number of relays based on current conditions and on the number of retransmissions left before discarding the packet, CARMA attempts to combine the best of two worlds.

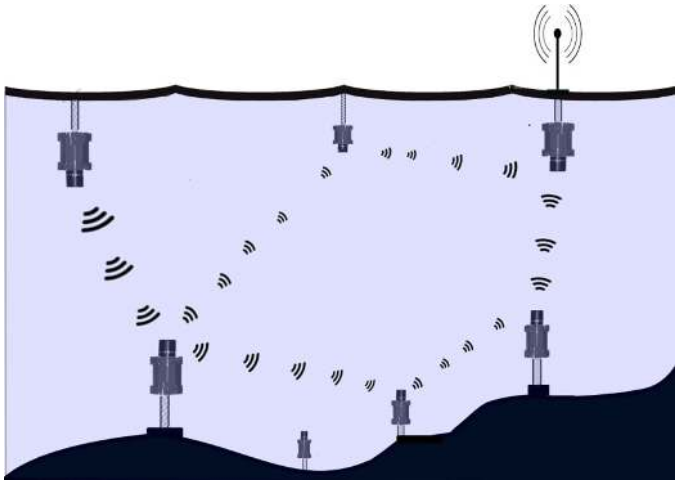


Figure 1: A UWSN scenario.

The presence of at least a packet p in the transmission queue of a node i triggers the packet transmission operations described in Algorithm 1.[2]

Algorithm 1 Packet Forwarding

```
1:  p = dequeue packet
2:  Add header information to p
3:  if there are known neighbors then
4:      k = 0
5:      while k < K do
6:          A = ComputeRelays(k)
7:          Forward packet p to set A
8:          if transmission of p is overheard by time τ then
9:              break
10:         else
11:             k = k + 1
12:     Discard packet
13: else
14:     Broadcast p
```

Packet p is dequeued (line 1) and required information is added to its header (line 2). If node i has known neighbors it executes the following steps: Sets the total number k of transmission attempts (capped at $K \geq 1$) to 0 (line 4); computes the set $\mathcal{A}$ of neighbors to which packet p will be transmitted (function ComputeRelays(k); line 6), and transmits packet p to all (and only) the neighbors listed in set $\mathcal{A}$ (line 7). After the packet has been transmitted, node i awaits τ time units to overhear the retransmission of

[2] Multiple packets are handled one at a time, "First-In First-Out."

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications

packet p by one of the selected neighbors (*implicit acknowledgment*;[3]line 8). (The value of τ is a function of the maximum round trip time.) If so happens, packet transmission operations are considered finished with success. Otherwise, the number of retransmissions is increased by 1 (line 11) and, if $k < K$ the whole set of operation is performed again. We remark that for each retransmission attempt, the node recomputes the set of relays, possibly choosing different sets of relays as the number k of transmission attempts increases. When $k = K$ the packet is discarded (line 12). (The value K is dynamically set as detailed in Section III-E.)

If node i has no information about any of its neighbors, it just broadcasts packet p (line 14). This failsafe behavior is a key feature of CARMA: It allows nodes to discover each other; it enables packet forwarding when no neighboring relays are known, and it is functional to propagate channel, node and route related information for learning purposes.

When not busy with transmitting packets, a node is in "listening mode," awaiting to hear the transmission of packets from neighboring nodes. Overhearing packet transmission triggers the execution of the following Algorithm 2.

Algorithm 2 Packet Overhearing & Reception

1: *Extract header information and store it*
2: **if** *packet destination list includes node* i **then**
3: *Receive entire packet and enqueue it*
4: **else**
5: *Abort packet reception*

Particularly, when node i overhears a packet p transmitted by a neighboring node j, it extracts information from the packet header and stores it (line 1). This information will be used to learn how to route packets, as detailed later in this section. If node i is one of the intended relays of packet p, the packet is entirely received and stored in a local buffer for further forwarding (line 3). Otherwise, the reception of packet p is aborted (line 5).

C. A reinforcement learning framework for CARMA

The heart of CARMA packet forwarding resides in the function COMPUTERELAYS, executed by node i to select the set $\mathcal{A}$ of neighbors to which packet p is transmitted (line 6). The algorithm is driven by a reinforcement learning framework allowing node i to learn from its current environment, namely, local channel quality and route-long energy consumption and reliability. In this section we provide details on the learning framework, namely, the state space, the actions, the state transition dynamics and the cost function. We then define function COMPUTERELAYS of Algorithm 1 that keeps the learning machinery updated and computes the set of relays $\mathcal{A}$ that optimizes routing in terms of energy consumption and packet delivery ratio. (For generalities on reinforcement learning the reader is referred to the extensive literature on this subject [23].)

States: Each node handling a packet p is in a state representing the number of times that p has been unsuccessfully transmitted. Node i is in state $s = k$ if it has transmitted packet p already k times. Node i transits to state rcv if, after a transmission attempt, packet p is correctly received by at least one of its neighbors. If the transmission fails, the state of node i becomes $k+1$. In case all the transmission attempts have failed the packet is dropped, which is modeled by the transition to state $drop$. The state space $\mathcal{S}$ is thus:

$$\mathcal{S} = \{0, \ldots, K-1\} \cup \{drop, rcv\}, \tag{1}$$

where $s = 0$ is the initial state, corresponding to node i first transmission attempt of packet p.

Actions: Node i makes forwarding decisions depending on the set of possible actions it can take from a state s. Each time packet p is (re)transmitted, node i determines the forwarding set for that transmission attempt. Therefore, for each node i and state $s = 0, \ldots, K-1$, the set of possible actions $A_i(s)$ is the set of non-empty subsets of N_i, namely, $A_i(s) \in \mathcal{P}(N_i) \setminus \emptyset$, with $\mathcal{P}(N_i)$ being the power set of the set N_i of all node i neighbors. Since no action can take place when $s \in \{drop, rcv\}$, we have $A(s_{drop}) = A(s_{rcv}) = \{\emptyset\}$.[4]

Transitions: Packet transitions from a state s to the next one depend on the forwarding decision $a \in A_i(s)$ and on the neighboring nodes that correctly receive the packet after a transmission attempt. Let us consider the probabilities $P_{i,j}$ and $P_{j,i}$ of correct reception on the links from node i to node j and viceversa (defined below), with node $j \in N_i$. For each state $s = 0, \ldots, K-1$ we consider two cases depending on whether the packet has been successfully transmitted or not.

In the first case, node i transits from state s to state rcv. The transition probability to state rcv is then:

$$P^a_{i,s\to rcv} = 1 - \prod_{j\in a}(1 - P_{i,j}P_{j,i}),\ a \in A_i(s). \tag{2}$$

Equation (2) indicates the joint probability that at least one neighbor of node i receives the packet, and that node i overhears its retransmission. It is critical to consider both probabilities $P_{i,j}$ and $P_{j,i}$ since the transmission of a packet is considered successful only when the packet is implicitly acknowledged, i.e., when its sender overhears at least one relay forwarding it. As underwater links can be highly asymmetric, the two probabilities can significantly differ, and therefore should be both taken into account.

In case the packet is not successfully transmitted we have to further consider two cases. If $s < K-1$, we just need to increase the number of retransmissions, and the next state of node i is $s' = k+1$. Otherwise, if the maximum number of retransmissions has been reached ($s = K-1$), the packet is dropped and the next state of node i is $s' = drop$. In both cases the transition probability is:

[3] To reduce both network traffic and energy consumption each packet is acknowledged implicitly: After a packet is sent, its sender listens to the channel awaiting to overhear the retransmission of the packet by at least one of the nodes that have received it. If such a retransmission is heard within τ time units, the sender considers that packet transmission to be successful. If not, the sender retransmits the packet after a pre-set amount of time. Only the sink sends explicit acknowledgments to its senders, as it does not forward the packet further underwater.

[4] As a state is an abstraction to model single packet forwarding, if a node has no packet to transmit, it has no state.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications

$$P^a_{i,s\to s'} = 1 - P^a_{i,s\to rcv} = \prod_{j\in a}(1 - P_{i,j}P_{j,i}), a \in A_i(s). \quad (3)$$

Costs: CARMA routes packets with the goal of minimizing the energy consumption on the *whole route* towards the sink, and of maximizing packet delivery to the sink. In order to model route-long energy consumption each state-action pair (s, a) is associated with a *cost function* c_i reflective of the energy spent by sender node i to transmit packet p and of the energy spent for transmission on routes from the selected relays to the sink. For maximizing packet delivery the cost function should also include a penalty for dropping the packet. Formally:

$$c_i(s,a) = \begin{cases} e_i(s,a) + n_i(s,a) & s < K-1 \\ e_i(s,a) + n_i(s,a) + l_i(s,a) & s = K-1, \end{cases} \quad (4)$$

where $e_i(s, a)$ is the energy spent for transmitting the packet, $n_i(s, a)$ is the cost incurred by downstream nodes to forward a copy of the packet, and $l_i(s, a)$ is the penalty associated to dropping the packet (which can only occur in state $s = K-1$).

Energy cost $e_i(s, a)$ does not depend on the number of chosen relays. For the sake of simplicity we assume that all nodes use the same transmission power and that therefore the cost for transmitting a packet is constant, namely, $e_i(s, a) = E$. The network component $n_i(s, a)$ of the cost is provided by:

$$n_i(s,a) = \sum_{j\in a} V_j P_{i,j}, \quad (5)$$

where V_j is the cost to forward a packet from node j to the sink along a whole route. This cost includes node j multiple retransmissions of the packet and the aggregate energy expenditure of all nodes from node j to the sink. Cost V_j is multiplied by probability $P_{i,j}$ since node j will forward the packet only in case it successfully receives it from node i.[5]

Finally, in case a packet has been unsuccessfully retransmitted for $K-1$ times, we associate the action $a = \mathcal{A}$ of the last set of retransmissions with the energy penalty $l_i(s, a) > 0$. This penalty aims at discouraging node i to drop the packet, i.e., transitions to the *drop* state. As such, $l_i(s, a)$ is defined as:

$$l_i(s,a) = L\prod_{j\in a}(1 - P_{i,j}), \quad (6)$$

where $(1 - P_{i,j})$ is the probability of unsuccessful delivery of the packet to node $j \in a = \mathcal{A}$, $\prod_{j\in a}(1 - P_{i,j})$ is the overall probability of dropping the packet, and L is set to an arbitrarily large value.

We finally have all the "ingredients" to describe how a node i that has a packet p to transmit learns how to optimally select relays to route it to the sink. Each node starts with no knowledge of its surrounding environment. Interacting with its neighbors, it learns and updates this knowledge over time. According to the reinforcement learning methodology [23], a *value function* V_i is approximated and updated relying on current estimations of the transition probabilities $P^a_{i,s\to s'}$, and on the estimated value of the functions V_j from neighboring nodes j, needed to estimate the cost $c_i(s, a)$ (Equations (4) and (5)). Algorithm 3 describes the learning process of node i and the corresponding determination of the best relays for packet p.

Algorithm 3 Learning Algorithm

1: **function** COMPUTERELAYS(current state k)
2: **for all** $s \in \mathcal{S}$ **do**
3: **for all** $a \in A_i(s)$ **do**
4: $Q_i(s,a) = c_i(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{i,s\to s'} V_i(s')$
5: $V_i(s) = \min_{a\in A_i(s)} Q_i(s,a)$
6: $V_i = V_i(0)$
7: $a = \arg\min_{a\in A_i(k)} Q_i(k,a)$
8: **return** a

Every time a node has to (re)transmit a packet, it computes the *Q value function* Q_i [23] for each possible state and action pair (line 4). (The *discount factor* γ, $0 \le \gamma \le 1$ is used to provide a way of deciding the importance of future retransmission costs.) The estimate of the value function V_i is then updated to represent the most current cost to route from node i to the sink (line 6). The action to be returned, namely, the set of the neighbors of node i selected as relays on optimal routes to the sink, is chosen as the set that minimizes the current value of the function Q (line 7), and returned to the node (line 8 and line 6 of Algorithm 1). We observe that Algorithm 3 is the model-based reinforcement learning Full Backup Algorithm (Section 9.5 of [23]), corresponding to one iteration of the Value Iteration Algorithm. Sutton and Barto show that this algorithm enjoys faster convergence than the Q-learning because at each iteration the latter only updates the entry $Q(s, a)$ associated to the current state s and action a, while the former updates the entire Q-table at once [23].

D. A dynamic programming perspective

We provide a more in-depth understanding of the CARMA learning machinery by rewriting the expression for $V_i(s)$ (line 5 of Algorithm 3) by replacing cost function and transitions probabilities (Equations (2) through (6)) in the computation of Q (line 4):

$$\begin{aligned} V_i(k) = \;& \min_{a\in A(k)}\{e_i(k,a) + \textstyle\sum_{j\in a} V_j(0)P_{i,j} + \\ & \gamma V_i(k+1)\textstyle\prod_{j\in a}(1 - P_{i,j}P_{j,i})\}, \quad k < K-1 \\ V_i(k) = \;& \min_{a\in A(k)}\{e_i(k,a) + \textstyle\sum_{j\in a} V_j(0)P_{i,j} + \\ & L\textstyle\prod_{j\in a}(1 - P_{i,j})\}, \quad k = K-1. \end{aligned} \quad (7)$$

Written in this format, and setting $\gamma = 1$, it is easier to see $V_i(k)$ as the *overall* cost to send a packet (already transmitted k times) to the sink. In fact, this formulation explicitly comprises: 1) the cost $e_i(k, a)$ of the k-th retransmission from node i; 2) the cost $\sum_{j\in a} V_j(0)P_{i,j}$ incurred

[5] For the machine learning literati, we note that V_j is node j value function in the initial state $s = 0$, that is, $V_j = V_j(0) = \min_{a\in A_j(0)} Q_j(0, a)$, as described later in the section. This value represents the current minimum cost incurred by node j to forward a data packet to the sink. As described below this information is included in the header of the packets transmitted by node j (Figure 2).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications

by the relay nodes $j \in a$ for successfully delivering the packet all the way to the sink, and 3) the cost $V_i(k+1)$ of node i possible additional retransmissions, which occur with probability $\prod_{j \in a}(1 - P_{i,j}P_{j,i})$, namely, the probability that node i does not overhear any relay forwarding the packet. In the last retransmission attempt ($k = K - 1$), the last term is replaced by the penalty L for dropping the packet in case of transmission failure.

Equations 7 show that the reinforcement learning approach of CARMA can be regarded as the solution of a distributed multipath routing problem with retransmission-dependent routing decisions that uses the current estimates of the link transmission probabilities. Correspondingly, Equations 7 take the typical form of a dynamic programming problem.

E. Parameter computation and packet format

The execution of the algorithms described in this section relies on the knowledge of the following.

The transition probabilities $P^a_{i,s \to s'}$. The estimation of the transition probabilities is based on the estimation of the link probabilities $P_{i,j}$, a measure of link quality in that they keep an estimate of the probability of successful packet reception at node j. Each node j keeps count of the number $p_{i,j}$ of packets received from each neighbor i, regardless of whether it is an intended receiver of the packets from those neighbors. The total number p_i of packets sent by node i is deduced by information in the header of the packets sent by that node. This allows node j to estimate the incoming link probability simply as $P_{i,j} = \dfrac{p_{i,j}}{p_i}$. These estimates are then inserted into the header of packets transmitted by node j to be received/overheard by all its neighbors (Figure 2). To account for varying link conditions counters p_i and $p_{i,j}$ are computed over a sliding window.

The accuracy of probability estimates depends on successful reception of packets in that node i needs to wait for packets from node j to compute its own estimate. If packets from node i to node j fail to be implicitly acknowledged, namely, after node i has awaited for τ time units to overhear their retransmissions, node i "degrades" $P_{i,j}$ to $\frac{p_i}{(p_i+1)}P_{i,j}$. Eventually, if node i does not receive any transmissions from node j for a given time interval, it removes node j from the list of its neighbors, until node j is heard again.

The packet format of CARMA is shown in Figure 2.

i		V_i		*packet ID*	
# Relays	*Relay*$_1$	*Relay*$_2$	...		*Relay*$_m$
n_i	j_1	$P_{j_1,i}$	...	j_{n_i}	$P_{j_{n_i},i}$
Data					

Figure 2: The CARMA packet format.

The first row contains information related to the sender node: i is the node unique identifier; V_i is the node current *value function*, expressing the current minimum cost incurred by node i to forward a data packet to the sink, and *packet ID* is the packet identifier (a positive number initially set to 1; this field can be used to infer the total number p_i of packets sent by the node.) The second row contains information about those neighbors of node i that have been selected as relays for the kth transmission attempt of the packet, $k < K$. Particularly, *# Relays* is the number m of the selected relays, and *Relay*$_1$, *Relay*$_2$, ..., *Relay*$_m$ are their unique identifiers. The third row contains information concerning the quality of incoming links as estimated by their senders, namely, all the neighbors of node i. Particularly, n_i is the number of the neighbors of node i, $j_1, \ldots, j_{n_i}$ are their unique identifiers, and $P_{j_1,i}, \ldots, P_{j_{n_i},i}$ indicate the quality of links incoming to node i. The last row is the packet payload, namely, the data.

The maximum number K of transmission attempts. The maximum number of transmissions of each packet affects network traffic, and therefore network performance. As a consequence, the value of the maximum number K of transmission attempts of a packet p by a node i should be set dynamically, according to the current traffic. To determine the value of K, we approximate the network throughout S of CARMA using the well-known ALOHA closed-form expression $S = Ge^{-2G}$, where G indicates the average number of transmission attempts in a time interval equal to the time needed to transmit one packet. If we define t_p as the collision window, i.e., the time needed to transmit a packet plus the propagation time, and λ as the aggregate (network-wide) packet arrival rate, G can be approximated as $G = t_p \lambda K$. Since the maximum throughput of an ALOHA network is achieved when $G = 0.5$, we can compute the maximum number K of retransmissions as follows: $K = \lceil 0.5/(t_p\lambda) \rceil$. For example, if $t_p = 2.3$s and $\lambda = 0.1$ pkt/s, we obtain a maximum of $K = 3$ retransmissions. This value grows up to 22 if λ decreases to 0.01 pkt/s. We observe that K can assume arbitrarily large or small values, as it depends on highly varying network parameters. Therefore, we restrict the range of feasible values between the two thresholds K_{min} and K_{max}. This allows a node to avoid an unnecessary large number of retransmissions while ensuring that a minimum number of transmission attempts is guaranteed. We have determined K_{min} and K_{max} through a wide set of field trials (Section IV-D), and we have set them to 3 and 8, respectively.

IV. Performance Evaluation

We demonstrate the effectiveness of our reinforcement learning-based approach to multi-path underwater routing through a comparative performance evaluation of CARMA and three state-of-art solutions for underwater routing, namely, CARP [6], QELAR [7] and EFlood [6].

A. Benchmark protocols

CARMA is compared to three previously proposed underwater routing techniques each presenting a different and paradigmatic approach to underwater routing. CARP represents the category of cross layer protocols designed to be aware of the current quality of the underwater channel; QELAR shows how to route smartly by using a machine learning technique to optimize energy consumption, and EFlood represents a multi-path approach to underwater routing. In the following,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications

we briefly describe these three solutions, highlighting those features that make them some of the best performing solutions currently available.

CARP is a *cross layer single-path* routing solution designed to be reliable, channel aware and energy efficient [6]. CARP is characterized by a channel reservation phase via the exchange of control packets, named PING and PONG, which also determine relay selection, namely, perform routing operations. CARP takes advantage of its channel reservation mechanism to transmit *trains* of consecutive packets, which are transmitted back to back and acknowledged cumulatively. This maximizes channel utilization, reduces control packet overhead and improves overall network performance.

QELAR is a *single-path* routing protocol for underwater networks based on a model-based Q-learning approach aimed at maximizing residual energy at the network nodes [7]. Its learning cost function takes into account the residual energy of each node as well as the energy distribution among neighboring nodes. Relays are chosen depending on the energy they can save. It uses the well-known CSMA protocol at the MAC layer. The maximum number R of retransmissions of a packet is statically set to a pre-defined value.

EFlood is a *multi-path* approach to routing obtained by enhancing common flooding using a simple de-synchronization scheme to randomize transmission attempts [6]. This reduces collisions and increases robustness. It uses the CSMA for channel access. In field deployment has shown that EFlood achieves high reliability and low latency, incurring much less collisions than plain flooding.

B. Investigated metrics

Protocol performance is assessed through the investigation of the following metrics.

- *Packet delivery ratio* (PDR), defined as the ratio between the number of packets correctly received by the sink and the number of packets generated by all nodes.
- *End-to-end latency*, defined as the time it takes by a packet to be delivered to the sink, namely, from when it is generated at a node to when it is received at the sink for the first time.
- *Energy per bit*, defined as the energy consumed by the network to correctly deliver a bit of data to the sink.

C. SUNSET-based simulations

All routing protocols have been implemented in SUNSET SDCS [8], a framework specifically designed for underwater network simulations. SUNSET has been connected to the Bellhop ray tracing tool [24] via the WOSS interface [25] to accurately model the underwater acoustic channel. Bellhop is used to compute acoustic path loss at a given location, as well as the spatially-varying interference induced by node transmissions given as input a specific sea profile. In the experiments, the environmental data input to Bellhop refers to an area located in the Norwegian fjord off the coast of Trondheim, with the coordinate $(0, 0, 0)$ of the surface located at $63°, 29', 1.0752''N$ and $10°, 32', 46.6728''E$. Sound speed profiles, bathymetry profiles and information on the type of bottom sediments of the selected area are obtained from the World Ocean Database [26], from the General Bathymetric Chart of the Oceans (GEBCO) [27], and from the National Geophysical Data Center's Deck41 data base [28].

Simulation scenarios and settings: We consider UWSNs with 6, 20 and 40 nodes, which are representative of current, larger, and desirable deployments, respectively. Nodes are statically positioned at different depths, ranging from 10m to 240m, in rectangular regions with surface of 1km^2, 2km^2 and 4km^2, respectively. The sink is placed at the left corner of the deployment area, 10m under the surface.

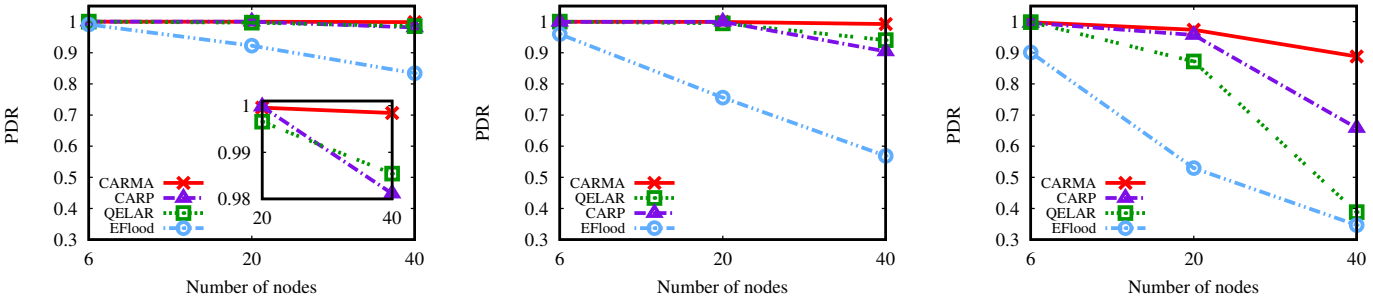
Network traffic is generated according to a Poisson process with aggregate rate λ packets per second, with $\lambda \in \{0.01, 0.04, 0.1\}$, corresponding to low, medium and high traffic, respectively. Once a packet is generated, it is assigned to a source node randomly selected among all nodes but the sink.[6] The destination of all packets is the sink.

The data packet size is set to 1000 bytes. The total size of a data packet is given by the size of the payload plus that of the headers added by the different layers, computed as follows. The physical header overhead changes according to the data rate. It is dominated by a 10ms synchronization preamble. At the MAC layer, the header size depends on the protocols. EFlood and QELAR use a CSMA-based MAC protocol without acknowledgments. The CSMA header contains the sender and the destination addresses, and the packet type for a total length of 3B. QELAR requires 6 additional bytes for the routing header that carries information on the node state space and residual energy. CARP is a cross layer protocol with its own MAC protocol also performing routing duties. The size of its PING and PONG control packets is 10B and 6B, respectively. Acknowledgments and "HELLO" packets (for node discovery) are 6B long. The size of the CARP MAC header is 4B. CARMA carries a host of information in the packet header, including the node value function, list of selected relays, and $P_{i,j}$ estimates for all neighbors (Figure 2). As a consequence, the size of the header depends on the number of a node neighbors, i.e., it is variable. As the protocol tends to favor smaller set (at least at first), keeping the size of the header variable allows to have smaller headers. The maximum size of headers in our experiments was 30B, which occurred in networks with 40 nodes. Channel access is performed according to a CSMA scheme. Whether CSMA-based or cross layer, all channel access methods considered here implement backoff mechanisms to reduce collisions. Particularly, CARMA estimates its backoff time depending on the current forwarding set and transmission retries.

In our simulations, we assume BPSK modulation (consistent with the type of Evologics modem we used in experiments at sea [29]). The carrier frequency is 25.6kHz for a bandwidth of 4000Hz. Bandwidth efficiency is set to 1bps/Hz, resulting in a data rate R_b of 4000b/s. For the selected value of the bandwidth and of the carrier frequency the transmission power is set to 2.8W, resulting in an average BER of 10^{-6} on the routes. The reception power consumption is set to 0.5W. The

[6] This corresponds to each node i generating packets according to an independent Poisson process with rate $\lambda_i = \frac{\lambda}{N-1}$ where N is the number of nodes in the network.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications



(a) Low traffic: $\lambda = 0.01$ (b) Medium traffic: $\lambda = 0.04$ (c) High traffic: $\lambda = 0.1$

Figure 3: Packet delivery ratio.

estimation of the reception and transmission powers is based on the energy consumption of existing acoustic modems [29].

The maximum size of a train of packets transmitted by CARP is set to 8. In other words, once reserved the channel through a PING/PONG handshake, a node can send up to 8 packets back to back. The maximum number R of packet retransmissions in QELAR is set to 5. Based on our investigation, this value maximizes the performance of the protocol. The maximum number K of retransmissions in CARMA is dynamically adjusted by each node according to traffic and packet transmission time. The values for K_{min} and K_{max} have been determined through a wide set of field trials. They are set to 3 and 8, respectively.

Finally, the discount factor γ used by CARMA was set to 1 in all our experiments (including those on the testbed at sea—Section IV-D). This is because we want to minimize the overall end-to-end cost of delivering a packet to the sink, and there is no reason to discount future retransmission costs.

Simulation results: The performance of the four protocols is summarized in Figure 3 (packet delivery ratio), Figure 4 (end-to-end latency), and Figure 5 (energy per bit).

Packet delivery ratio. Figure 3 shows the PDR for different network sizes and traffic rates. As a general trend across all protocol, the PDR decreases with increasing traffic and network size because of the higher number of interference and the higher probability to find the channel busy.

CARMA shows the best performance in all scenarios, showing a PDR of 100% at low and medium traffic, which decreases to 88% only in scenarios with high traffic. The advantage over the other protocols increases with traffic, indicating higher scalability. At high traffic, CARMA delivers an average of 20% more packets than CARP and doubles the PDR of QELAR and EFlood. The causes of this are multifold. First and foremost, the CARMA cost function accounts for the end-to-end transmission cost, which is an energy cost plus a penalty for failing to deliver a packet to the sink (Equations 4, 5, and 6). This allows CARMA to find the paths with the highest probability to success delivery. Secondly, multi-path transmissions increase protocol robustness. We observe that CARMA resorts to multi-path transmissions judiciously, using multiple relays exclusively as the number of retransmissions increases. This benefits the PDR without being detrimental to network traffic and energy consumption. Lack of judicious resorting to multi-path does not allow the other multi-path solution, EFlood, to obtain acceptable PDR. Finally, in CARMA the maximum number K of retransmissions is dynamically set depending on traffic as perceived by each node (Section III-E). This keeps the network traffic below a level that would cause excessive interference and noticeable performance degradation.

CARP delivers 100% of generated packets at low traffic. Its performance decreases to 65% at the highest traffic. In CARP, the channel reservation handshake allows the transmitter to determine the best candidate relay in terms of link quality and residual energy. As such, a relay is chosen that maximizes the probability of successful one-hop transmission. Moreover, the use of packet trains reduces network overhead and optimizes channel utilization. Nevertheless, at high traffic, CARP delivers only 70% of the packets delivered by CARMA. This is because when the traffic increases, the number of nodes involved in channel contention increases as well, thus hindering the access to the channel.

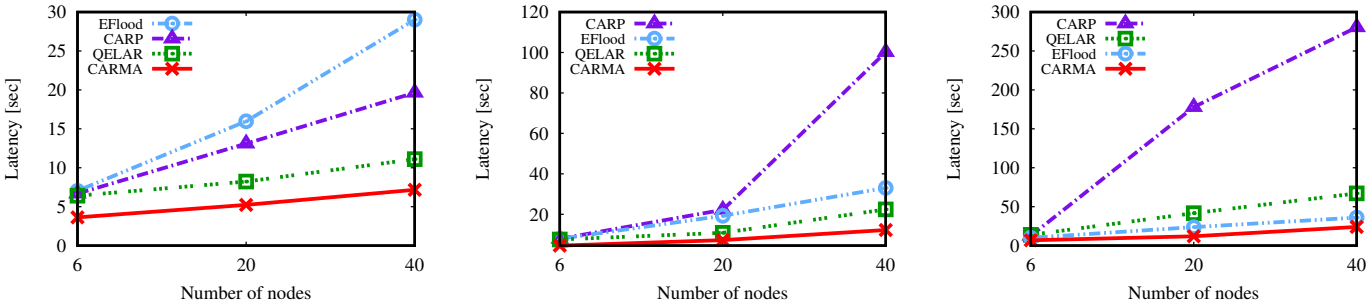
In QELAR, the learning algorithm cost function is based on the relay node residual energy. This favors energy consumption over packet delivery ratio as packets are dropped whenever discarding them will cost less energy than that needed to deliver them to the sink. Furthermore, at high traffic the use of a fixed number R of retransmissions leads to excessive offered load. This increases the number of collisions, negatively impacting successful packet reception and hence the PDR. When congestion builds up, the mechanism used by QELAR to estimate the transmission probability looses accuracy. This impairs the learning mechanism and leads to the selection of suboptimal relay nodes.

Finally, it is no surprise that EFlood, which is based on a flooding-based multi-path scheme, exhibits the worst performance, achieving a PDR below 40% in the large network scenario with high traffic. As mentioned, this is because of the indiscriminate use of multi-path routing when instead a smaller number of relays—as dynamically chosen by CARMA—would be beneficial.

End-to-end latency. Figure 4 shows the average end-to-end latency for packets successfully delivered to the sink. As a general trend, latency increases with traffic and network size, for increased number of collisions and retransmissions.

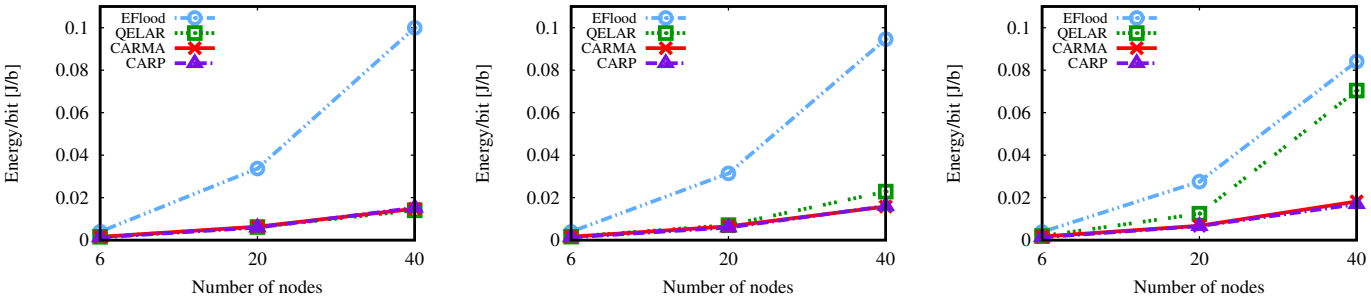
Results show that CARMA achieves the lowest latency.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications



(a) Low traffic: $\lambda = 0.01$ (b) Medium traffic: $\lambda = 0.04$ (c) High traffic: $\lambda = 0.1$

Figure 4: End-to-end latency.



(a) Low traffic: $\lambda = 0.01$ (b) Medium traffic: $\lambda = 0.04$ (c) High traffic: $\lambda = 0.1$

Figure 5: Energy per bit.

This is yet another beneficial consequence of the definition of the CARMA cost function that, by explicitly seeking to minimize route-long energy consumption, succeeds in also minimizing the average route length traveled by packets, which corresponds to lower travel time.

Not surprisingly, because of its channel reservation mechanism, CARP experiences high end-to-end latency. This severely limits CARP scalability. As shown in Figure 4 (c), in the 40 nodes topology, at high traffic, packets sent by CARP incur a tenfold latency increase over that incurred by packets sent by CARMA, and latency that is five times higher than that of the other two protocols. The performance of QELAR is closer to that of CARMA, for basically the same reason: Minimizing energy consumption leads to shorter routes and hence to lower latency. The fact that QELAR obtains higher latency than CARMA is mainly due to the pre-set number R of retransmissions.

EFlood latency increases with network size because of the longer routes. However, latency does not vary noticeably with increasing traffic. This can be explained by observing that since each packet is transmitted only once by a node, the latency component due to retransmissions is avoided here.

Energy per bit. Figure 5 shows the energy consumed to deliver a bit of data to the sink. As expected, energy consumption increases with traffic and network size. CARMA, CARP and QELAR obtain very similar performance at low traffic.

CARMA reaps the benefit of its cost function being designed explicitly to minimize energy consumption, which obtains low consumption irrespective of traffic and network size.

CARP saves energy by reducing excessive retransmissions via effective channel reservation and transmitting multiple packets back to back.

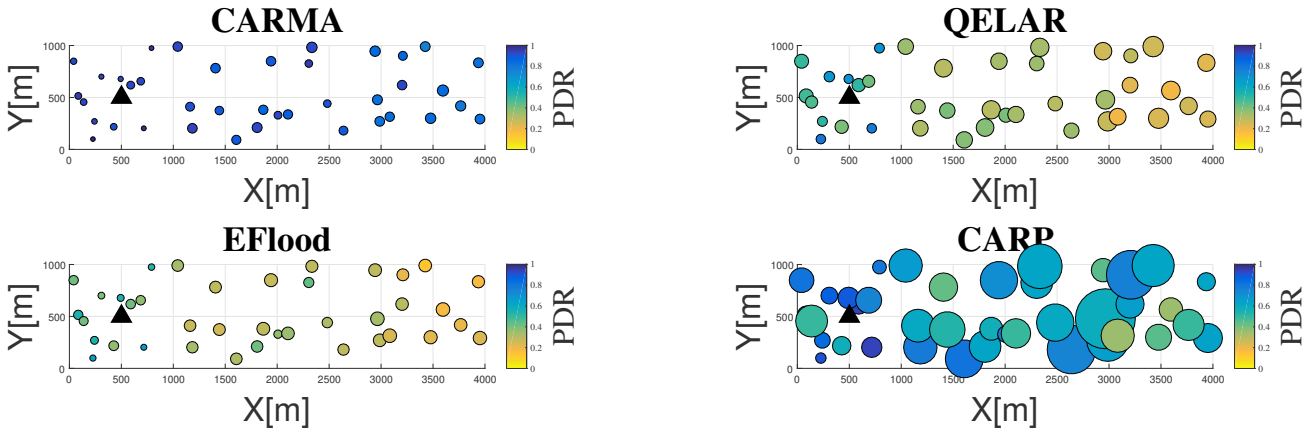
As the traffic increases, the performance of QELAR degrades because of the higher number of packet retransmissions and the lower number of bits correctly delivered to the sink.

Because of its flooding-based nature, EFlood shows the worst performance in all scenarios.

We conclude our journey through the performance of the selected protocols by taking a look at their performance from a different perspective. Specifically, we provide a spatial view of the three considered performance metrics at the node level. Figure 6 and Figure 7 show results for networks with 40 nodes and high traffic ($\lambda = 0.1$ pkt/s). In both figures, the sink is depicted as a black triangle. In Figure 6 nodes are depicted as circles whose radius is proportional to the latency of packets from that node (the smaller the better). The color of the node indicates its PDR: The darker the color the higher the PDR.

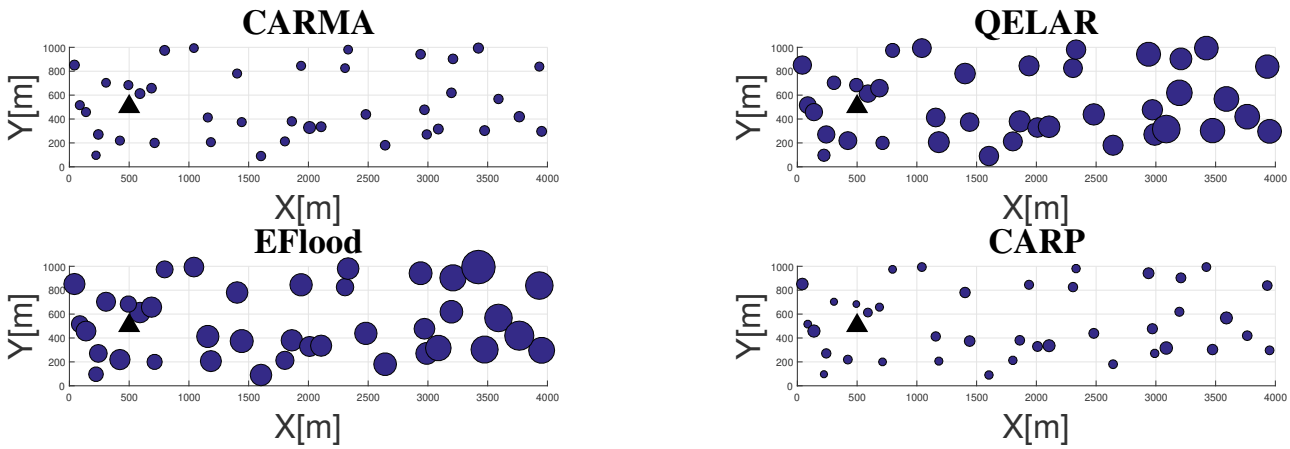
In Figure 7 each node is depicted as a circle whose radius is proportional to the energy per bit consumed at that node (again, the smaller the better). These results clearly indicate that all CARMA nodes perform remarkably well, including those far away from the sink. With QELAR, node performance is uneven, with the PDR rapidly degrading as we move farther away from the sink. As mentioned earlier, this is due to QELAR pre-set, high number R of retransmissions and to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications



(a) CARMA and EFlood (b) QELAR and CARP

Figure 6: A joint snapshot of PDR and end-to-end latency in networks with 40 nodes and high traffic.



(a) CARMA and EFlood (b) QELAR and CARP

Figure 7: A snapshot of the energy consumed per bit in networks with 40 nodes and high traffic.

its cost function, that favors energy consumption over packet delivery. With CARP, nodes obtain good PDR and energy consumption performance. However, they suffer from high end-to-end latency, independently of the distance from the sink, because of the channel reservation mechanism.

D. Experiments at sea

We tested the performance of CARMA at sea through a series of experiments off the coast of Calabria, at Vibo Valentia Marina (South of Italy). The six-node network topology is depicted in Figure 8. The distances (in meters) between each pair of nodes are shown in Table I.

Table I: Distances between pairs of nodes (meters).

	1	2	3	4	5	6
1	-	110	142	185	312	96
2	110	-	121	95	214	220
3	142	121	-	102	213	230
4	185	95	102	-	129	293
5	312	214	213	129	-	420
6	96	220	230	293	420	-

Figure 8: Network topology used for coastal monitoring.

We deployed six Evologics S2C/18 acoustic modems [29] in a shallow water environment with a maximum depth of 10 meters to reproduce a coastal monitoring scenario. Five nodes (with IDs from 1 to 5) were deployed at different depths ranging from 1 to 3 meters. One modem acted as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications

the sink (node 6). It was deployed on the side of a boat working as control station. All nodes but the sink generate packets. In this setting, we compared the performance of CARMA with that of CARP and EFlood. We conducted two 90-minute experiments two hours apart from each other (referred below as Experiment 1 and Experiment 2). In each experiment, the three protocols were tested for a period of 30 minutes each.[7] The packet size is set to 64B and the bit rate to 480bps.[8] Given the small size of the network, the MAC portion of the header of CARMA packets was 9B long. That of the header of CARP and EFlood was 5B and 3B long, respectively. (The part of the header concerning the physical layer has the same size for all protocols.) Traffic was generated according to a constant bit rate process at an aggregate rate of $\lambda = 0.05$ packets per second (3 packets per minute) and $\lambda = 0.066$ packets per second (4 packets per minute) in Experiment 1 and 2, respectively. It is worth observing that, due to the time consuming handshake phase, $\lambda = 0.066$ is the highest load that CARP was able to sustain in the considered scenario. We set the Evologics modem to operate at the lowest power level possible, resulting in a power consumption for transmissions of 2.8W. Power consumption for reception was 0.5W. Since the acoustic modems we used do not feature power control with the fine granularity required by CARP, the same transmission power level for control and data packets was used for each experiment.

Table II shows the link Packet Error Rate (PER) measured during the two experiments.

We observe that link quality was in general rather poor. It varies significantly between the two experiments. Links were also highly asymmetric. Aside very few exceptions, link quality during Experiment 2 was worse than that of links in Experiment 1. Particularly, the PER of the link from node 3 to node 5 decreased from 87% to 30%. It is interesting to observe that node 5, which is the farthest node from the sink, had the second best direct transmission performance to the sink itself (PER=19%) during Experiment 1, but completely lost connectivity to the sink during Experiment 2 (PER=100%).

Table III summarizes the performance of the three protocols. Given the relatively small size of packets transmitted by the Evologics modems (64B), besides investigating PDR, end-to-end latency, and energy per bit, we also show results on data throughput (data bits/second), to measure the effect of the (MAC component of the) packet header size on the amount of information delivered to the sink per time unit.

CARMA significantly outperforms both CARP and EFlood in terms of PDR with 40% and 100% more packets delivered to the sink, respectively. Performance is better in the first experiment with 98% of delivered packets vs. the 89% of the second. Here multi-path routing at higher number of retransmissions increases the robustness of CARMA, allowing nodes to deliver significantly more packets. As expected in this scenario EFlood showed the worst PDR performance, because of the high link PER and the highest energy demand. In these experiments, CARP suffers from the high PER that impairs channel reservation and, therefore, actual packet transmission. This offsets the benefit of the use of packets trains and, as a result, CARP has lower PDR than CARMA, and incurs extremely high latency. CARMA exhibits the lowest energy consumption per bit despite the multi-path communication and longer headers. This can be explained by observing that the use of multi-path routing is connected to the number of retransmissions to increase robustness without affecting energy consumption sensibly. The same considerations justify data throughput results: Despite the larger header, CARMA delivers data bits at a rate that is higher than that of the other two protocols. In general, the data throughput of all protocols measured during Experiment 2 is higher than that measured in Experiment 1. This is because of the higher traffic used in the second experiment. Although in this case we observe a higher number of collisions, and therefore a higher number of lost packets (lower PDR), the overall number of bits delivered to the sink per second is higher, whence the higher throughput.

Figure 9 and Figure 10 depict the routes selected by CARMA and CARP in Experiment 1 and Experiment 2, respectively.

Figure 9a and Figure 10a depict the CARMA forwarding set through solid and dashed line: Solid lines indicate the most used paths to the sink; the dashed ones concern the extra relays chosen to support multi-path packet forwarding. In both sets of experiments we clearly observe that CARMA resorts to multi-path routing to deliver data packets to the sink. Figure 9b shows that, when using CARP, packets from node 5 reach the sink via two different paths, namely, directly or via node 1. These are unicast paths: There are times where the direct link between node 5 and the sink is good enough to allow successful transmission, and times when this does not happen. In this case CARP resorts to forward the packets through node 1. CARMA, instead, forwards packets on *both* paths at the same time, increasing robustness and PDR.

As a final note we observe that results from simulations on networks with 6 nodes (Figure 3, Figure 4 and Figure 5) and those from the six-node testbed (Table III) are in reasonable agreement, in that CARMA always shows performance that is superior to that of both CARP and EFlood. Simulated results are better than those from the testbed mainly because our simulator, while implementing typical impairments of the underwater channel (like multi-path and fading), does not fully capture the time varying nature of underwater links.

E. Convergence analysis

In this section we set to investigate the impact of channel variations on learning how to route, i.e., we provide a measure of how fast CARMA is able to converge to optimal relay selection. To this aim, we study the variations of the value function of a node as a function of the number of transmitted packets. We focus on node 2, which, in all our experiments, is the only node that was never able to communicate directly with the sink (farthest node from the sink in terms of number of hops). For the sake of simplicity we assume unitary energy cost and a relative large drop penalty, $L = 100$ (Equation (7)).

[7] Test duration was selected as a trade-off between the need of testing the protocols under the same underwater channel conditions and have each test last enough to collect statistically meaningful results.

[8] These packet size and bit rate are the maximum available on Evologics modems used in the "Instant Messages" mode.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications

Table II: Link quality. Entry (i, j) contains the Packet Error Rate (PER) of the link from i to j.

(a) Experiment 1

	1	2	3	4	5	6
1	-	22%	66%	88%	54%	0%
2	22%	-	75%	25%	69%	100%
3	48%	82%	-	61%	87%	88%
4	73%	42%	82%	-	84%	56%
5	23%	55%	60%	60%	-	19%
6	0%	63%	74%	46%	37%	-

(b) Experiment 2

	1	2	3	4	5	6
1	-	37%	55%	97%	47%	20%
2	47%	-	94%	70%	62%	100%
3	45%	94%	-	59%	30%	92%
4	83%	58%	72%	-	80%	88%
5	52%	79%	46%	60%	-	100%
6	10%	75%	93%	50%	91%	-

Table III: Results from experiments at sea.

(a) Experiment 1: Aggregate rate of 0.05 packets per second

Metric	CARMA	CARP	EFlood
Packet Delivery Ratio [%]	98	72	48
End-to-end latency [s]	5.92	89.27	7.79
Energy per bit [J/b]	0.033	0.044	0.060
Data throughput [b/s]	22.8	17.8	12.2

(b) Experiment 2: Aggregate rate of 0.066 packets per second

Metric	CARMA	CARP	EFlood
Packet Delivery Ratio [%]	89	63	44
End-to-end latency [s]	11.8	81	7.07
Energy per bit [J/b]	0.043	0.047	0.053
Data throughput [b/s]	30.5	23	16

(a) CARMA (b) CARP

Figure 9: Experiment 1: Route selection.

We observe that trends vary significantly from Experiment 1 to Experiment 2. During Experiment 1 the learning algorithm (Algorithm 3) executed by node 2 converged to a solution after less than 20 packets were transmitted. Starting without any knowledge of the surrounding environment, node 2 was able to quickly learn the best forwarding strategy. After converging, the value function $V_2(0)$ remains stable, a further indication that for the duration of the experiment the channel quality was continuously good. This is consistent with our measurements of the PER on the links between node 2 and its chosen relay (node 1), and between node 1 and the sink (entries $(2, 1)$ and $(1, 6)$ of Table IIa). During Experiment 2, instead, CARMA struggles to converge to a stable value. This reflects a greater channel variability, as confirmed by the values in Table IIb. However, despite the worsened channel conditions, CARMA obtains a PDR that is just shy of 90%, showing fast adaptation and robust packet delivery via multi-path routing (Table III).

V. Conclusions

In this paper we have presented CARMA, a Channel-aware Reinforcement learning based Multi-path Adaptive routing protocol for UWSNs. CARMA jointly determines the cardinality and composition of the set of packet relays to maximize packet delivery ratio and minimize energy consumption. Energy savings and robustness stem from a reinforcement learning framework that drives protocol operations by letting each node to react to the ever changing underwater channel conditions. Particularly, a node adaptively switches from swift and energy efficient single-path routing in favorable network conditions to a more robust multi-path routing when single-relay forwarding becomes problematic.

We evaluated the performance of CARMA by means of simulations in networks with increasing size and traffic, and also with actual experiments at sea. The performance of the protocol is compared to that of three among the most efficient solutions available in the literature, namely CARP, QELAR and EFlood. Results show that CARMA achieves remarkable performance improvements in terms of end-to-end latency and energy consumption, and achieves a PDR that is up to 40% higher than that of all other protocols.

Acknowledgments

The work was supported by the following projects: EC EASME ArcheoSub project “Autonomous underwater Robotic and sensing systems for Cultural Heritage discovery Conservation and in situ valorization,” Sapienza’s “IoT4Offshore” and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications

(a) CARMA (b) CARP

Figure 10: Experiment 2: Route selection.

MIUR "Dipartimenti di eccellenza 2018–2022" of the Department of Computer Science of Sapienza University. Stefano Basagni was supported in part by grant NSF CNS 1726512 ("MRI: SEANet: Development of a Software-Defined Networking Testbed for the Internet of Underwater Things").

REFERENCES

[1] J. Heidemann, M. Stojanovic, and M. Zorzi, "Underwater sensor networks: Applications, advances and challenges," *Philosophical Transactions of the Royal Society A*, vol. 370, pp. 158–175, August 2 2012.

[2] E. Demirors, J. Shi, A. Duong, N. Dave, R. Guida, B. Herrera, F. Pop, G. Chen, C. Cassella, S. Tadayon, M. Rinaldi, S. Basagni, M. Stojanovic, and T. Melodia, "The SEANet project: Toward a programmable Internet of Underwater Things," in *Proceedings of IEEE UComms 2018*, Lerici, Italy, August 28–30 2018, pp. 1–5.

[3] T. Melodia, H. Kulhandjian, L.-C. Kuo, and E. Demirors, "Advances in underwater acoustic networking," in *Mobile Ad Hoc Networking: Cutting Edge Directions*, S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, Eds. Hoboken, NJ: John Wiley & Sons, Inc., March 5 2013, ch. 23, pp. 804–852.

[4] B. Tomasi, G. Toso, P. Casari, and M. Zorzi, "Impact of time-varying underwater acoustic channels on the performance of routing protocols," *IEEE Journal of Oceanic Engineering*, vol. 38, no. 4, pp. 772–784, September 2013.

[5] P. Casari, D. Spaccini, G. Toso, B. Tomasi, R. Petroccia, C. Petrioli, and M. Zorzi, "A study on channel dynamics representation and its effects on the performance of routing in underwater networks," in *Proceedings of the IEEE Asilomar Conference on Signals, Systems and Computers 2012*, Pacific Grove, CA, USA, November 4–7 2012, pp. 1536–1540.

[6] S. Basagni, C. Petrioli, R. Petroccia, and D. Spaccini, "CARP: A channel-aware routing protocol for underwater acoustic wireless networks," *Elsevier Ad Hoc Networks and Physical Communication, joint Special Issue on Advances in Underwater Communications and Networks*, vol. 34, pp. 92–104, November 27 2015.

[7] T. Hu and Y. Fei, "QELAR: A machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 6, pp. 796–809, June 2010.

[8] C. Petrioli, R. Petroccia, J. R. Potter, and D. Spaccini, "The SUNSET framework for simulation, emulation and at-sea testing of underwater wireless sensor networks," *Elsevier Ad Hoc Networks*, vol. 34, no. C, pp. 224–238, November 2015.

[9] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2595–2621, 2018.

[10] H. A. Al-Rawi, M. A. Ng, and K.-L. Yau, "Application of reinforcement learning to routing in distributed wireless networks: A review," *Artificial Intelligence Review*, vol. 43, no. 3, pp. 381–416, March 2015.

[11] M. A. Alsheikh, S. Lin, D. Niyato, and H. P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1996–2018, 2014.

[12] R. Petroccia and J. Alves, "A hybrid routing protocol for underwater acoustic networks," in *Proceedings of Med-Hoc-Net 2018*, Capri, Italy, June 20–22 2018, pp. 94–101.

[13] G. Han, L. Liu, N. Bao, J. Jiang, W. Zhang, and J. J. P. C. Rodriguez, "AREP: An asymmetric link-based reverse routing protocol for underwater acoustic sensor networks," *Journal of Networks and Computer Applications*, no. 92, pp. 51–58, August 15 2017.

[14] Y. Noh, U. Lee, P. Wang, B. S. C. Choi, and M. Gerla, "VAPR: Void-aware pressure routing for underwater sensor networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 5, pp. 895–908, 2013.

[15] N. Li, J.-F. Martinez, J. M. Meneses Chaus, and M. Eckert, "A survey on underwater acoustic sensor network routing protocols," *Sensors*, vol. 16, no. 3, pp. 1–28, March 22 2016.

[16] A. Khan, I. Ali, A. Ghani, N. Khan, M. Alsaqer, A. Ur Rahman, and H. Mahmood, "Routing protocols for underwater wireless sensor networks: Taxonomy, research challenges, routing strategies and future directions," *Sensors*, vol. 18, no. 5, pp. 1–30, May 18 2018.

[17] T. Hu and Y. Fei, "An adaptive routing protocol based on connectivity prediction for underwater disruption tolerant networks," in *Proceedings of IEEE Globecom 2013*, Atlanta, GA, December 9–13 2013, pp. 65–71.

[18] S. Basagni, V. Di Valerio, P. Gjanci, and C. Petrioli, "MARLIN-Q: Multi-modal communications for reliable and low-latency underwater data delivery," *Ad Hoc Networks*, no. 82, pp. 134–145, January 2019.

[19] ——, "Harnessing HyDRO: Harvesting-aware Data ROuting for underwater wireless sensor networks," in *Proceedings of ACM MobiHoc 2018*, Los Angeles, CA, June 25–28 2018, pp. 271–279.

[20] R. Plate and C. Wakayama, "Utilizing kinematics and selective sweeping in reinforcement learning-based routing algorithms for underwater networks," *Ad Hoc Networks*, vol. 34, pp. 105–120, 2015.

[21] T. Hu and Y. Fei, "MURAO: A multi-level routing protocol for acoustic-optical hybrid underwater wireless sensor networks," in *Proceedings of IEEE SECON 2012*, Seul, Korea, June 18–21 2012, pp. 218–226.

[22] K.-L. A. Yau, P. Komisarczuk, and P. D. Teal, "Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues," *Journal of Network and Computer Applications*, vol. 35, pp. 253–267, 2012.

[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., ser. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, 2017.

[24] M. Porter *et al.*, "Bellhop code." [Online]. Available: http://oalib.hlsresearch.com/Rays/index.html

[25] F. Guerra, P. Casari, and M. Zorzi, "World Ocean Simulation System (WOSS): A simulation tool for underwater networks with realistic propagation modeling," in *Proceedings of ACM WUWNet 2009*, Berkeley, CA, November 3 2009, pp. 1–8.

[26] "World ocean atlas," www.nodc.noaa.gov/OC5/WOA05/pr_woa05.html.

[27] "General bathymetric chart of the oceans," www.gebco.net.

[28] "National geophysical data center, seafloor surficial sediment descriptions," http://www.ngdc.noaa.gov/mgg/geology/deck41.html.

[29] Evologics, "Evologics S2C acoustic modems." [Online]. Available: http://www.evologics.de/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2933968, IEEE Journal on Selected Areas in Communications

Valerio Di Valerio is a Postdoc researcher at the University of Rome "La Sapienza". He received the master degree in Computer Engineering in 2010 and the Doctorate degree in Computer Science in 2014, both from the University of Rome "Tor Vergata". His research interests concern Service Oriented Architecture, Cloud computing and Underwater Sensors Networks, with special emphasis on modeling, performance evaluation and optimization. In the last two years he has also participated to several experimental campaigns at sea where innovative underwater systems have been extensively tested. He worked on the EU-funded projects TROPIC and SUNRISE and served as a reviewer for several international journals and conferences.

Francesco Lo Presti is Associate Professor in the Department of Civil Engineering and Computer Science of the University of Roma Tor Vergata. He received the Doctorate degree in computer science from the University of Rome Tor Vergata in 1997. His research interests include measurements, modeling and performance evaluation of computer and communications networks. He has more than 70 publications in international conferences and journals. He has served as TPC member of conferences on networking and performance areas, and as reviewer for various international journals.

Chiara Petrioli is professor of Computer Science, director of the Sensor Networks and Embedded Systems laboratory (SENSES lab) in the department of Computer Science of the University of Rome "La Sapienza." She also leads the Cyber Physical System lab of "La Sapienza" center for Cyber Intelligence and Information Security, and is a founding partner of "La Sapienza" spinoff WSENSE S.r.l. She has been member of the academic senate and chair of the PhD program in Computer Science at La Sapienza. Professor Petrioli research interests concern the design and optimization of future wireless, embedded, IoT and cyber physical systems. Prof. Petrioli is chair of the steering committee of IEEE SECON and general chair of ACM MobiHoc 2019, was program co-chair of IEEE INFOCOM 2016 and general chair of ACM SenSys 2013. She has been member of the steering committee and associate editor of IEEE Transactions on Mobile Computing, member of the steering committee of ACM SenSys, associate editor of IEEE Transactions on Vehicular Technology, member of the executive committee of ACM SIGMOBILE, associate editor of Elsevier Computer Communications, guest editor of special issues for IEEE Access, Elsevier Ad Hoc Networks, Elsevier Physical Communications, and has been program co-chair of leading conferences in the field, such as ACM MobiCom and IEEE SECON. She is currently an elected member of the ACM Europe Council. Prof. Petrioli has published over a hundred and fifty papers in prominent international journals and conferences (over 5900 citations; h-index 42). She has been the PI of over twenty national and international research projects, serving as coordinator of three EC projects (FP7 projects GENESI and SUNRISE, EASME ArcheoSub) highlighted as success stories on the Digital Agenda of Europe and featured by international mass media including RAI SuperQuark and Presa Diretta, Wired USA, the Guardian, Bild magazine, and National Geographics. Her research has resulted in international patents and in award-winning innovative technologies. She is a pioneer of the Internet of Underwater Things, an area on which she has led the development of breakthrough technologies listed in the NT100 Top "Social Global Techs changing our lives 2016." Prof. Petrioli was a Fulbright scholar and is one of the Inspiring 50 2018, top women in technology.

Luigi Picari received the master degree in Computer Science from University of Rome "La Sapienza" summa cum laude in 2012 and the PhD in Computer Science from University of Rome "La Sapienza" in 2017. His research interests focus on the design, implementation and in-field evaluation of novel Internet of Things (IoT) solutions for Underwater Wireless Sensor Networks (UWSNs). In the last five years he has participated in dozen at sea experimental campaigns where innovative underwater systems have been tested and validated. He has collaborated in several EU-funded research projects such as FP7 CLAM, FP7 SUNRISE and H2020 EASME ArcheoSub. Luigi Picari is also co-author of two international patents on adaptive underwater communications.

Daniele Spaccini received the Master Degree in Computer Science with the highest honors (2009) and the Ph.D. (2015) from Rome University "La Sapienza," Italy, where he is currently a post-doctoral researcher. His research interests include underwater communications, networking and underwater vehicles localization, to which he has contributed with over two dozen papers published in leading venues. He participated in several EU-funded projects including the FP7 STREP CLAM project, the FP7 SUNRISE project and H2020 EASME ArcheoSub. In SUNRISE, he was in charge of coordinating all underwater experimental activities. In ArcheoSub he is in charge of coordinating the design and development of the software to be used in all the in field experiments. In the last five years he has participated in over twenty experimental campaigns at sea where innovative underwater systems have been extensively tested. During these campaigns he collaborated with research and defence centers, such as CMRE (Centre for Maritime Research and Experimentation) e SPAWAR (Space and Naval Warfare Systems Command). He is also author of two international patents on adaptive underwater communications.

Stefano Basagni is with the Institute for the Wireless Internet of Things and an associate professor at the ECE Department at Northeastern University, in Boston, MA. He holds a Ph.D. in electrical engineering from the University of Texas at Dallas (December 2001) and a Ph.D. in computer science from the University of Milano, Italy (May 1998). Dr. Basagni's current interests concern research and implementation aspects of mobile networks and wireless communications systems, wireless sensor networking for IoT (underwater and terrestrial), definition and performance evaluation of network protocols and theoretical and practical aspects of distributed algorithms. Dr. Basagni has published over nine dozen of highly cited, refereed technical papers and book chapters. His h-index is currently 42 (May 2019). He is also co-editor of three books. Dr. Basagni served as a guest editor of multiple international ACM/IEEE, Wiley and Elsevier journals. He has been the TPC co-chair of international conferences. He is a distinguished scientist of the ACM, a senior member of the IEEE, and a member of CUR (Council for Undergraduate Education).