Ad Hoc Networks 82 (2019) 134–145

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

MARLIN-Q: Multi-modal communications for reliable and low-latency underwater data delivery☆

Stefano Basagni[a], Valerio Di Valerio[b], Petrika Gjanci[b,*], Chiara Petrioli[b]

[a] ECE Department Northeastern University, Boston, MA, United States
[b] Dipartimento di Informatica Università di Roma "La Sapienza," Roma, Italy

ARTICLE INFO

Article history:
Received 25 March 2018
Revised 27 July 2018
Accepted 6 August 2018
Available online 7 August 2018

Keywords:
Underwater wireless sensor networks
Multi-modal communications
Reinforcement learning-based routing
Soft QoS

ABSTRACT

This paper explores the smart exploitation of multi-modal communication capabilities of underwater nodes to enable reliable and swift underwater networking. Following a model-based reinforcement learning approach, we define a framework allowing senders to select the best forwarding relay for its data jointly with the best communication device to reach that relay. The choice is also driven by the quality of the communication to neighboring nodes over time, thus allowing nodes to adapt to the highly adverse and swiftly varying conditions of the underwater channel. The resulting forwarding method allows applications to choose among different classes of soft Quality of Service (QoS), favoring, for instance, reliable routes to the destination, or seeking faster packet delivery. We name our forwarding method MARLIN-Q, for Multi-modAl Reinforcement Learning-based RoutINg with soft QoS. We evaluate the performance of MARLIN-Q in varying networking scenarios where nodes communicate through two acoustic modems with widely different characteristics. MARLIN-Q is compared to state-of-the-art forwarding protocols, including a channel-aware solution, and a machine learning-based solution. Our results show that a smartly learned selection of relay and modem is key to obtain a packet delivery ratio that is twice as much that of other protocols, while maintaining low latency and energy consumption.

1. Introduction

The past decade witnessed a marked shift in underwater sensing and communication capabilities. The resulting variety of research and development activities has propelled underwater wireless sensor network (UWSN) technology to new levels of possibilities and usage. Applications such as underwater monitoring, surveillance, discovery, exploration, aquaculture and coastal protection, are becoming increasingly sophisticated and produce more and more complex data in need of reliable and swift delivery to their destination, whether a collection point on the surface or the final user. Examples include pictures and video streams from underwater cameras, as well as data from sonars and other high data rate sensors [1,2]. To best respond to the challenges imposed by underwater communications, which are beset by extreme short-term channel variability and by environmental noise at different frequencies, a recent trend is that of allowing sensor nodes to communicate through multiple devices at the same time, which is also known as *multi-modal communications*. Switching among multiple devices allows nodes to quickly adapt to variations, changing frequencies, and therefore power levels, bit rates and ranges, avoiding noise and other sudden impairments. For instance, endowing a node with optical and acoustic modems provides flexible ways to combine short range, high bit rate data transfer with long range, low bit rate, robust communication capabilities. Applications that take advantage from multi-modal communication include networking with Autonomous Underwater Vehicles (AUVs) visiting sensors to retrieve data optically at high rate and coordinating with nodes and other AUVs over long range acoustic links [3–8]. Multiple acoustic modems also provide great adaptability to channel variability and noise. In fact, acoustic modems with widely different characteristics are already available whose frequencies are centered from 24 kHz [9] to 100 kHz [10], up to 160 kHz [11]. These devices obtain bandwidth, bit rates and ranges that greatly differ from each other. For example, commercial modems from Teledyne Benthos, which communicate on the band centered at 24 kHz, allow nodes to transmit data at 4 kilo bit per second (kb/s) to receivers up to 2 km away [9]. Higher data rate modems, transmitting in the 100 kHz band at up to 28 kb/s, reliably deliver data to nodes no more than 100 m away [10]. This variety of

☆ Results included in this paper are partially covered by a patent pending application.

* Corresponding author.
E-mail addresses: basagni@ece.neu.edu (S. Basagni), divalerio@di.uniroma1.it (V.D. Valerio), gjanci@di.uniroma1.it (P. Gjanci), petrioli@di.uniroma1.it (C. Petrioli).

https://doi.org/10.1016/j.adhoc.2018.08.003
1570-8705/

technologies promptly at disposal of a node provides the per-link reliability unavailable so far to underwater communication.

In this paper we aim at demonstrating how multi-modal communications can be cleverly exploited for reliable and low latency underwater networking that meets the stringent requirements of key underwater applications. Particularly, we are interested in investigating how the *link* reliability afforded to a node by communicating through multiple devices can be extended to *network-wide routes*, thus bringing the advances in underwater communications to fruition for networking underwater devices on a larger scale. We explore how nodes can acquire knowledge on the quality of the links to neighboring nodes through each of their communication devices, and how this knowledge can be used for selecting reliable multi-link, multi-modal routes to the d0ata final destination. To this aim, we define a *model-based reinforcement learning* framework through which senders are able to select the best forwarding relay for their data packets jointly with the best communication device to reach that relay. Our framework is built to make routing decisions that support multiple service classes, through which applications can seek reliable routes to their data final destination, preferring packet delivery ratio (*reliability class*), or routes that provide faster packet delivery at the expense of moderate packet loss (*urgent class*). We name the resulting forwarding method *MARLIN-Q*, for *Multi-modAl Reinforcement Learning-based RoutINg* with soft *Quality of Service*. MARLIN-Q is flexibly defined to consider recent channel quality over each communication device, as well as information on routing reliability and delivery times from neighboring nodes, thus addressing network wide performance via local information exchange.

The performance of MARLIN-Q is evaluated through simulations with SUNSET SDCS, a tool for underwater networking that models a wide variety of details of the underwater channel and environment realistically [12]. We consider compelling underwater scenarios, with different network size, varying traffic and varying amount of urgent traffic, which represent the variety of settings suitable to a large number of applications. In network with nodes with two underwater acoustic modems with different characteristics we investigate key metrics such as packet delivery ratio (PDR), end-to-end latency, and energy consumption. In these settings we compare MARLIN-Q to the following state-of-the-art protocols: (i) CARP, a cross-layer solution designed to be reliable, channel aware and energy efficient [13], and (ii) QELAR, a machine learning-based protocol designed for minimizing and balancing node energy consumption [14]. We choose CARP and QELAR as they have been shown to outperform previous solutions for underwater routing and machine learning-based routing, respectively.

Our results show that MARLIN-Q obtains remarkable packet delivery ratio for both reliable and urgent traffic, irrespective of varying network size, and amount and type of traffic. Latency is kept remarkably low: Even in the most challenging settings—networks with 40 nodes and high traffic—urgent traffic is always delivered within 9 s, while incurring a noticeably low energy consumption. MARLIN-Q also achieves remarkable fairness to different types of traffic, delivering steady amount of packets to the sink independently of their type (either urgent of reliable). We observe that our protocol outperforms CARP and QELAR in all scenarios. Particularly, in the most challenging settings, MARLIN-Q obtains a PDR that is twice as much that of the second best performing protocol. Because of the clever definition of the cost optimization function of its core framework, despite delivering a higher number of packets MARLIN-Q is the fastest of all considered protocols, with improvements over the second fastest of up to 32%. Energy consumption is also kept at bay: Albeit delivering twice as many packets, the energy performance of MARLIN-Q is at par with that of CARP, which uses channel reservation and saves energy on packet re-transmission.

The results in this paper confirm that the trend of using multi-modal communication for link reliability extends to network-wide performance provided that route selection is driven by a smart choice of best relays and communication devices. This makes MARLIN-Q a solution for future underwater networking, achieving performance levels needed by key underwater applications, and especially those with soft QoS requirements.

The rest of the paper is organized as follows. Section 2 introduces the scenarios considered in our investigation, along with notation and preliminaries on model-based reinforcement learning, the core of MARLIN-Q routing. Section 3 defines MARLIN-Q in details. Section 4 reports results on the performance evaluation of MARLIN-Q, and its comparison with previous solutions. In Section 5 we survey previous works on multi-modal communications and on underwater reinforcement learning-based routing. Section 6 concludes the paper.

2. Network scenario and preliminaries

We introduce scenario, concepts and notation that are preliminary to the protocol description. We also provide a brief introduction to model-based reinforcement learning, whose methods constitute the core of MARLIN-Q.

Multi-modal scenario. We consider a static underwater wireless sensor network (UWSN) made up of N nodes whose sensors produce data that need to be delivered to the network data collection point (the *sink*), for processing and/or further forwarding. Given the geographic extent of node deployment, not every node is able to communicate directly with the sink. Therefore, packets may have to travel multi-hop routes.

The scenario we consider is *multi-modal* in the precise sense that each underwater node is equipped with multiple wireless communication devices (modems) operating on different frequencies, at different bandwidths, obtaining different bit rates, and with various communication ranges and power consumption. Nodes are generically indicated as i and j. A specific modem is denoted by m, and $\mathcal{M}$ indicates the set of all modems available at each node. With $\mathcal{M}_{idle}$ we indicate the subset of the modems of a node that are idle at a certain point in time, i.e., that are ready to be used for transmission.

A sketch of a multi-modal underwater scenario is depicted in Fig. 1. Every node is equipped with two communication devices. The sink is shown close to surface, in the upper right corner of the picture, and has capabilities to report data from the UWSN to stations on shore.

Classes of service. Nodes are deployed to monitor assets or events through their sensors. When something that needs reporting happens a node produces data that need to be delivered to the sink. Depending on the application, these data need to reach the sink as soon as possible (*urgent data*) or reliably (*reliable data*). To this aim, MARLIN-Q supports different classes of service. In particular, in this work we consider a *reliability class* r for reliable packets, and a *low latency class* u, for urgent ones. If an application requires reliable delivery, MARLIN-Q seeks to deliver the highest number of packets to the sink. Otherwise, for an application that produces packets of type u, MARLIN-Q does its best to be fast, at the expense of tolerating some packet loss. As nodes may run multiple applications, each node may generate packets of both types.

Use of implicit ACKs. In order to reduce overhead—and latency, and energy consumption—each packet is acknowledged implicitly, leveraging the broadcast nature of the wireless channel. Specifically, after transmitting a packet, the sender starts listening to the channel on any of its modems. If it overhears the packet being retransmitted by the chosen relay within a given time, it considers the packet transmitted successfully. If it does not, the packet is considered lost. (Only the sink sends explicit ACKs back to its

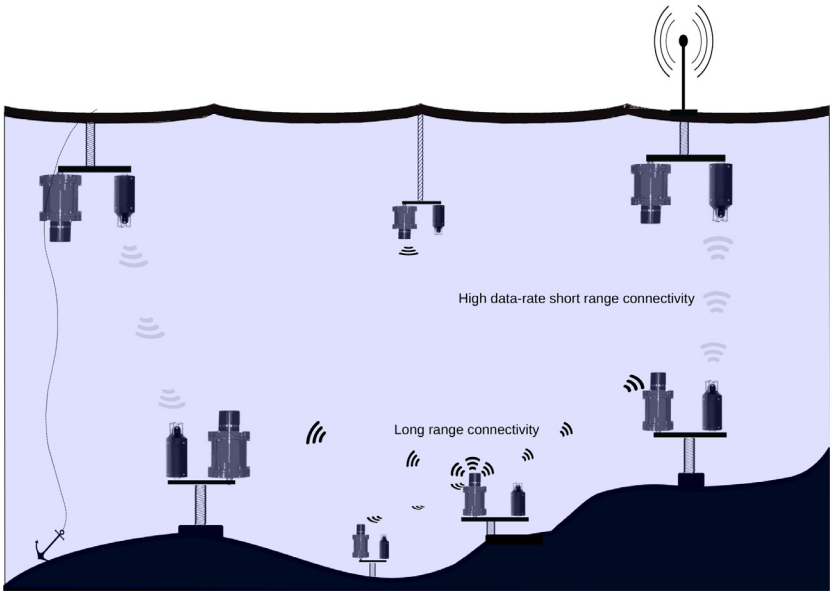


Fig. 1. Multi-modal UWSN. Each node has two communication devices.

senders, as it does not forward the packet further underwater). The node behavior after packet loss depends on the class of service of the packet that has been transmitted, as described in details below.

A brief primer on reinforcement learning. Reinforcement learning concerns how some *agents* take *actions* in a given environment for optimizing some notion of cumulative cost [15]. To this purpose, agents *learn* and optimize *policies* online through direct experience rather than computing them a priori. Given the set $\mathcal{S}$ of possible states of an agent, and the set $A(s)$ of the actions available at each state, a policy is a function π that associates each state $s \in \mathcal{S}$ with the action $a \in A(s)$ that the agent should take towards cost minimization. In the context of our work, agents correspond to underwater nodes handling packets, while the policy corresponds to the forwarding strategy, namely to the choice of a relay and of a modem to communicate with that relay.

In order for an agent to establish how good it is to be in a given state, a value function $V^{\pi}(s)$ is defined as the expected *infinite-horizon discounted cost* starting from s as initial state and using a given policy π as follows:

$$V^{\pi}(s) = E_s^{\pi}\left\{\sum_{t=0}^{\infty}\gamma^t c(s_t, a_t)|s_0 = s\right\}, \quad (1)$$

where $0 \leq \gamma < 1$ is the discount factor, s_t and a_t are the system state and the action taken at time t, respectively, and $c(s_t, a_t)$ is the expected cost associated to state s_t and decision a_t. For each state $s \in \mathcal{S}$, the optimal policy π^* minimizing the value functions satisfies the Bellman optimality equation:

$$V^{\pi^*}(s) = \min_{a \in A(s)}\left\{c(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{s \to s'} V^{\pi^*}(s')\right\}, \quad (2)$$

where $P^a_{s \to s'}$ represents the transition probability from state s to state s' after action a has been taken. Eq. (2) highlight that the policy π^* that minimizes the cost depends on the immediate cost of taking the action a from state s and on the expected discounted cost from the next state s' onward.

In order to measure the costs of taking different actions a from state s to different states s' we define the function Q as the expected infinite horizon discounted cost of taking an action a in state s and then following the policy π:

$$Q^{\pi}(s,a) = c(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{s \to s'} V^{\pi}(s'), \ \ \forall s \in \mathcal{S}. \quad (3)$$

For each state $s \in \mathcal{S}$ the Bellman equation and the function Q allow us to greedily compute the optimal policy as $\pi^*(s) = \arg\min_{a \in A(s)} Q^{\pi^*}(s,a)$.

Solving the Bellman Eq. (2) depends on knowing the cost $c(s, a)$ and the transition probabilities $P^a_{s \to s'}$. In scenarios where these parameters are not known a priori, models can be provided for their estimation, so that functions V and Q can be learned online, by interacting with the agent environment. This interaction usually takes the form of *exploiting* the knowledge acquired so far by the agent, and of *exploring* the agent environment to gain new knowledge. Learning techniques following this methodology are called *model-based* [15]. In the description of our protocol (Section 3), these *ingredients* of reinforcement learning are tailored to our underwater multi-modal scenario to constitute the core of how MARLIN-Q learns how to route, i.e., how nodes choose the best relays and modems to reliably and quickly deliver data packets to the sink.

3. The MARLIN-Q protocol

In this section we describe MARLIN-Q in details, and particularly, the machine learning framework at its core.

3.1. Protocol description

Each node manages two output transmission queues: One for packets of type r ("reliability queue"), and one for those of type u ("urgent queue"). Urgent packets take priority over reliability packets, i.e., are always transmitted first. Reliability packets are transmitted only when the urgent queue is empty.[1]

When a node i has a packet p to forward, it executes the following algorithm SendPacket(p, c), where p is the packet to be transmitted and c is its type, i.e., either r or u.

Forwarding packets depends on their type. Reliability is fostered by having node i re-transmitting packet p until its success-

[1] Although unlikely, there could be cases in which urgent packets are always present at a node that would prevent transmission of reliable packets, which would then loose they "reliability." In order to avoid the occurrence of such an extreme case, MARLIN-Q can be extended to include a "failsafe mode" that would ensure transmission of one (or more) reliability packet(s) every ℓ urgent packets, with ℓ to be selected, possibly as a function of traffic. For ease of protocol description, we have not detailed this mechanism in the current description of the protocol.

Algorithm 1 SENDPACKET(p, c).

```
1: if (c = r) then
2:     k = 0
3:     done = false
4:     while (not done) do
5:         ⟨j, m⟩ = ComputeR&M(r, k, M_idle)
6:         if Channel Idle then
7:             TransmitPacket(p, j, m)
8:             if (j is heard to transmit p within δ time units)
   then
9:                 done = true
10:            else
11:                if k > K − 1 then
12:                    Drop p
13:                    done = true
14:                else
15:                    k = k + 1
16:        else
17:            set Backoff([δ^u, δ^r])
18: else
19:    ⟨j, m⟩ = ComputeR&M(u, 0, M_idle)
20:    if Channel Idle then
21:        Compute&InsertBackUpRelays(p)
22:        TransmitPacket(p, j, m)
23:    else
24:        set Backoff(r ∈ [0, δ^u])
```

ful reception at the selected neighbor j. Success is recognized by node i overhearing node j forwarding the packet within a certain time δ (implicit acknowledgment), or by explicit acknowledgment if node j is the packet final destination (i.e., the sink). We allow up to $K \geq 1$ transmissions, after which the packet is discarded. (Parameters K and δ are global to the algorithm. They are set at the start of node operations.) Low latency for urgent packets is obtained by avoiding packet re-transmissions altogether. A main relay is chosen along with a list of back-up relays, which are in charge to forward the packet if the main relay is not heard to forward it.

In details, the operations of SENDPACKET are as follows. If packet p comes from the reliability queue, node i chooses the *best* relay j and modem m for forwarding packet p by running algorithm ComputeR&M (line 5; detailed below). Subsequently, it checks whether the channel is idle or not. If it is, node i transmits p to the chosen relay j using the selected modem m (line 7). After transmission, node i awaits to overhear the forwarding of p by relay j (implicit ACK) or to receive an ACK from the sink. If the ACK is received within a pre-defined time δ, the transmission of packet p is successfully concluded, and the algorithm terminates (line 9). If that does not happen, and the packet has been already transmitted for $K \geq 1$ times, then the packet is dropped (line 12) and the algorithm terminates (line 13); otherwise, the number of re-transmission attempts is increased by 1 (line 15) and the process is repeated with the selection of new relay and modem, and so on.

If the channel is found busy, instead, node i re-enqueues the packet and postpones its transmission by setting a backoff timer to a value chosen randomly and uniformly in the interval $[\delta^u, \delta^r]$ (line 17). We stipulate that $\delta^u < \delta^r$. Both parameters are global to the algorithm. They are set at the start of the node operation. Upon timer expiration, operations resume from line 5.

If packet p comes from the urgent queue, node i chooses the *best* relay j and modem m for forwarding packet p by running the algorithm ComputeR&M (line 19; detailed below). We notice that in this case, the parameter k concerning the number of re-transmissions is set to 0, as urgent packets are not re-transmitted. Subsequently, node i checks whether the channel is idle or not. If it is, node i computes a set of back-up relays that are in charge of forwarding p if the main relay j is not heard to transmit it, and attaches the list to the header of packet p (Compute&InsertBackUpRelays(p); line 21. Details on the selection of back-up relays are provided below.) Node i then transmits p to the chosen relay j using the selected modem m (line 22). If the channel is busy, node i re-enqueues the packet and postpones its transmission by setting a backoff timer to a value chosen randomly and uniformly in the interval $[0, \delta^u]$ (line 24). Upon timer expiration, operations resume from line 19. We notice that by choosing $\delta^u < \delta^r$ we favor the re-transmission of urgent packets before that of reliability ones.

The crux of every node operation, i.e., the heart of algorithm SENDPACKET, is choosing of the next hop relay j and of the best idle modem m for transmitting packet p. This choice is performed by running the algorithm ComputeR&M (lines 5 and 19). Choosing relays and modem for packet forwarding, as well as back-up relays for urgent packet, is supported by a sophisticated reinforcement learning-based framework, described in the following.

3.2. Learning how to route

Each node i acts as an agent that, for each packet p, determines the best among a set of forwarding decisions, namely an action a representing a relay and a modem. The model for routing packets in MARLIN-Q is characterized by defining the following: (i) The *state space*, (ii) the *actions*, (iii) the state *transition dynamics*, and (iv) the *cost function*. Based on these four components, model-based reinforcements learning allows nodes to determine (v) *optimal forwarding* decisions. In the rest of this section we first describe each of the ingredients of our framework, and we then use them to define algorithm ComputeR&M.

(i) States. A node i handling a packet p can be in one of $k \leq K-1$ representing the number of times that it has transmitted p unsuccessfully. The state space $\mathcal{S}$ is therefore the set $\{0, \ldots, K-1\} \cup \{rcv, drop\}$, where rcv identifies successful packet transmission and $drop$ identifies the case when the maximum number K of transmissions has been exceeded and the packet is discarded.

(ii) Actions. Actions concern forwarding decisions, i.e., the joint selection of a relay among the sender neighbors and of the modem to that relay. Let us denote with $\mathcal{N}_i^m$ the set of nodes that a node i can reach using modem m. (Different modems correspond to different sets, although a neighbor can be reached by multiple modems.) For each node i, state s and set of modems $M \subseteq \mathcal{M}$, the set $A_i^M(s)$ of available actions is:

$$A_i^M(s) = \{a = \langle j, m\rangle \mid m \in M,\ \ j \in \mathcal{N}_i^m\}, \tag{4}$$

where $a = \langle j, m\rangle$ is the action of forwarding a packet to neighbor j using modem m. Since no action can take place when $s \in \{rcv, drop\}$, it is $A_i^M(rcv) = A_i^M(drop) = \emptyset$.

(iii) Transitions. The transition from one state to another depends on the current state s and on the performed action $a = \langle j, m\rangle$. Let us denote with $P_{i,j}^m$ the probability of correct packet transmission on the link from node i to node j using modem m. When the transmission of p succeeds after k unsuccessful attempts, node i transitions from state $s = k$ to state $s' = rcv$. The transition probability is the following:

$$P_{i,s\to rcv}^{\langle j,m\rangle} = \begin{cases} P_{i,j}^m P_{j,i}^{m'} & \text{if } 0 \leq k < K-1 \\ P_{i,j}^m & \text{if } k = K-1. \end{cases} \tag{5}$$

When p can be (re)transmitted, i.e, when the number k of re-transmission is $< K-1$, successful transmission depends on the following probabilities: (i) The probability $P_{i,j}^m$ that the packet is received by node j on modem m. This probability is computed by

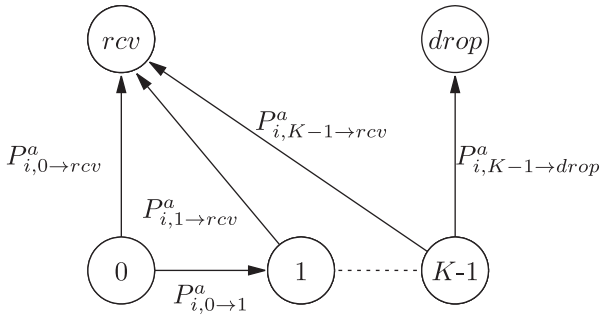


Fig. 2. States and transitions of node i handling packet p. The initial state is 0; *rcv* and *drop* are the final absorbing states.

node j and broadcast in the header of its packets. (ii) The probability $P^{m'}_{j,i}$ that the packet p, forwarded by node j on best modem m', is successfully overheard by node i. This probability is computed by node i based on overhearing node j transmissions on modem m'. (Details on determining best modems and probabilities are provided below.) In case node j is the sink, $P^{m'}_{j,i}$ expresses the probability that node i receives the ACK explicitly sent by the sink. (In this case, we stipulate that the sink sends the ACK using the same modem it received the packet on, i.e., $m = m'$.) When the number of transmissions has reached its last value K, there is no need for i to listen for an ACK, and the state transition depends only on the link from i to j. When the transmission of p fails, we have two possible transitions. If $k < K-1$ we just increase the number of re-transmissions, and the next state is $s' = k+1$. Otherwise, if $k = K-1$, the packet p is dropped and the next state is $s' = drop$. In both cases, the transition probability can be defined simply as $P^{\langle j,m\rangle}_{i,s\to s'} = 1 - P^{\langle j,m\rangle}_{i,s\to rcv}$. The state diagram of a node i handling a packet p is depicted in Fig. 2. *(iv) Costs.* In a reinforcement learning approach, the inner logic of a protocol resides in the structure of the cost function c_i to be optimized. MARLIN-Q focuses on minimizing the *network-wide time needed by node i to deliver packet p to the sink*, i.e., to its final destination. In order to express the whole routing time, we associate each state-action pair (s, a) with a cost reflective of the time needed to forward the packet by one hop, to a selected relay, and of the time needed to forward it from that relay to the sink. Equally important, we increase reliability by associating a high penalty time to transitions to the *drop* state. As we seek to minimize time, this discourages packet loss. More formally:

$$c_i(s,a) = \begin{cases} t_i(s,a) + d_i(s,a) & \text{if } 0 \le k < K-1 \\ t_i(s,a) + d_i(s,a) + l_i(s,a) & \text{if } k = K-1. \end{cases} \quad (6)$$

The cost $t_i(s, a)$ indicates the time needed for p to be delivered to neighbor j. It is defined as follows:

$$t_i(s,a) = t_m + p_{i,j} + \begin{cases} w_i & k = 0 \\ b_i & \text{otherwise,} \end{cases} \quad (7)$$

where t_m is the time needed to transmit p on modem m, $p_{i,j}$ is the propagation time between the two nodes, w_i is the average time spent by packet p in the queue of node i (before the first transmission of the packet), and b_i is the time spent waiting for the implicit ACK (subsequent re-transmissions). The $d_i(s, a)$ component of the cost in Eq. (6) is given by:

$$d_i(s,a) = V_j(0) P^m_{i,j}, \quad (8)$$

where $V_j(0)$ is the value of the function V associated to the state $s = 0$ of relay node j. This cost is, by definition, a measure of the minimum time needed to reach the sink starting from node j. (It is available to node i as it is broadcast by node j in the header of its packets.) The cost $V_j(0)$ is multiplied by the probability $P^m_{i,j}$ as node j will forward the packet only in case it correctly receives it from node i. Finally, in case a packet has been unsuccessfully transmitted for $k = K-1$ times, we associate to the action $a = \langle j, m\rangle$ of the last re-transmission the penalty time $l_i(s, a)$ aimed at discouraging node i to drop the packet. We can think of dropped packets as packets that are delivered to the sink arbitrarily late in time. As such, the cost $l_i(s, a)$ associated to "delivering the packet that late" is defined as:

$$l_i(s,a) = L(1 - P^m_{i,j}), \quad (9)$$

where $(1 - P^m_{i,j})$ is the probability of dropping the packet, and L is set to a value greater than the highest cost of delivering the packet to the sink through any of the neighbors of node i (i.e., $L > \max_{j\in\bigcup_m \mathcal{N}^m_i} V_j(0)$). In other words, as node i approaches the maximum number K of transmission attempts, its actions tend to favor the reliability of the transmission to the next hop.

(v) Optimal forwarding. To compute optimal forwarding decisions, every time a packet p is ready to be transmitted, node i executes an algorithm for determining the action $a = \langle j, m\rangle$ for p, i.e., the best relay j and the best modem m to reach it. Each node starts with no knowledge of its surrounding environment. Interacting with its neighbors, it iteratively acquires and updates its knowledge over time. In particular, the 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 $V_j(0)$ from neighboring nodes j, needed to estimate the cost $c_i(s, a)$. (From now on, all values of the transition probabilities, and of functions V and c are to be intended estimates changing over time.)

We are finally able to describe the core component of MARLIN-Q forwarding process, namely, algorithm ComputeR&M. This is the algorithm through which node i executes the learning framework and uses it to determine the best *relay* and *modem* (R&M) every time there is a packet p to (re-)transmit.

Algorithm 2 ComputeR&M(c, k, $\mathcal{M}_{idle}$).

1: **for all** $(s \in \mathcal{S})$ **do**
2: **for all** $a \in A^{\mathcal{M}}_i(s)$ **do**
3: $Q^c_i(s,a) = c_i(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{i,s\to s'} V^c_i(s')$
4: $V^c_i(s) = \min_{a\in A^{\mathcal{M}}_i(s)} Q^c_i(s,a)$ *#Update*
5: $\tau = random\_number(0,1)$
6: **if** $\tau < 1-\epsilon$ **then** *#Exploitation*
7: $\langle j, m\rangle = arg\,min_{a\in A^{\mathcal{M}_{idle}}_i(k)} Q^c_i(k,a)$
8: **else** *#Exploration*
9: $m =$ random modem in $\mathcal{M}_{idle}$
10: $\langle j, m\rangle = arg\,min_{a\in A^{\{m\}}_i(k)} Q^c_i(k,a)$
11: **return** a = $\langle j, m\rangle$ *#Forwarding decision*

The algorithm takes as input the type c of packet p, the current state k and the set of idle modems $\mathcal{M}_{idle}$. When packet p is ready for transmission, node i updates the model and computes the new value function (line 1to 4). Once the model has been updated, a forwarding action can be selected. To balance between exploitation of the acquired knowledge and exploration of new one, we use the ϵ-greedy heuristic, which is well-known for robustness and effectiveness [15]. Specifically, each time, the best neighbor on the best idle modem is selected with probability $1-\epsilon$ (line 7), exploiting the knowledge just updated. Conversely, with probability ϵ we explore new solutions by selecting a random modem among those available (line 9), and we forward the packet to the best relay we can reach using that modem (line 10). Even when exploration may produce a suboptimal choice of modem and relay, the broadcast

nature of the transmission to this relay allows nodes to acquire key statistics about their neighbors. (Parameter ϵ is global to the algorithm. It is set at the start of the node operations.)

It is worth pointing out that algorithm ComputeR&M handles both types of packets simultaneously: Two instances of our learning framework are executed in parallel based on the type of the packet to be transmitted. Specifically, depending on the flow type c, functions Q^c and V^c are updated and used to determine the best relay and modem for packet p. In order for a node to understand which functions to update upon receiving a packet p, we stipulate that the header of p carries an extra bit indicating its type.

The execution of Algorithm ComputeR&M relies on the knowledge of the transition probabilities $P^a_{i,s\to s'}$ and on the packet forwarding cost $c_i(s, a)$, which in turn depends on the value function $V^c_j(0)$ of each neighbor j of node i. Here is how these probabilities and function values are determined. The estimation of the transition probabilities is based on the estimation of the link probabilities $P^m_{i,j}$ (Eq. (5)). Nodes estimate link quality upon receiving a packet. In particular, a receiver j keeps count of the number of packets $n^m_{i,j}$ received from each neighbor i on modem m, regardless of whether node j is the packet intended destination. The incoming link probability is estimated as $P^m_{i,j} = n^m_{i,j}/n^m_i$, where n^m_i is the total number of packets sent by node i, an information that node i broadcasts in the header of its packets. These estimates are then broadcast by node j into its packet headers, to be overheard by its neighbors. In order to keep track of the varying link conditions, the counts n^m_i and $n^m_{i,j}$ are computed over a sliding window. If node i fails to overhear transmissions from a neighbor j within a given time it automatically updates its own link transmission probability. In particular, node i "degrades" $P^m_{i,j}$ to $(n^m_i/(n^m_i+1))P^m_{i,j}$. If node i does not hear packets from node j for a given time, it removes node j from the list of its neighbors until node j is heard again. The determination of $c_i(s, a)$ is based on information available locally at node i and on values broadcast by node j in the header of its packets.

Finally, we describe how a node that is forwarding an urgent packet p computes the list of back-up relays, namely, we detail the operations of Compute&InsertBackUpRelays(p) (line 21 of algorithm SendPacket). Let us assume that Algorithm ComputeR&M outputs node j as the main relay for forwarding packet p to be reached using modem m. Let us also assume that node j uses modem m' to forward the packet p. The first back-up relay h is selected by node i using the following rule:

$$\langle h, m\rangle = \arg\min_{a\in A_i^{\{m\}}(0)\setminus\{\langle j,m\rangle\}} \frac{Q_i(0,a)}{P^{m'}_{j,h}}. \tag{10}$$

This rule aims at selecting a node that is not only a good forwarder (low $Q_i(0, a)$) but also a node that is "well connected" to the main relay j, i.e., with high probability of overhearing j forwarding p (high $P^{m'}_{j,h}$). In so doing, node i tries to avoid redundant re-transmissions that would increase network traffic and waste energy. Applying rule (10) with h in place of j produces the second best back-up relay, and so on.

The length of the list is set to a number $\ell_i \le |\mathcal{N}^m_i|$, which may vary dynamically, depending on the network load. In fact, we keep adding back-up relays to the list until either the probability that none of them receives the packet is lower than a given threshold P_{lost} or the size of the list ℓ_i is reached.

Upon receiving the packet, the back-up relays set a back-up timer and store the packet in their queues. The timer is set to a time that is inversely proportional to the node position in the prioritized list of back-up relays: The higher its position, the shorter the time. When the timer goes off, the node checks if it has overheard the packet transmission by higher priority nodes. If so, it discards the packet; otherwise it transmits it. For example, the first back-up relay forwards p only if it does not overhear its transmission by the main relay; the second back-up relay transmits p only if it does not overhear its transmission from either the main relay or the first back-up node, and so on. By not re-transmitting a packet for which there is no implicit ACK, the original sender keeps transmitting other packets, whose queuing delay is thus shorter. This is a key feature of MARLIN-Q, enabling overall faster delivery of urgent packets.

4. Performance evaluation

We demonstrate the effectiveness of our approach to multimodal networking through a simulation-based performance evaluation of MARLIN-Q in realistic underwater channel conditions and scenarios. The performance of our protocol is also compared to that of two solutions that represent current state-of-the-art routing for UWSNs. The two protocols are: (i) CARP, a cross-layer solution designed to be reliable, channel aware and energy efficient. CARP is characterized by a channel reservation phase through control packets (named PING and PONG) that also carry routing information [13], and (ii) QELAR, a machine learning-based protocol designed for minimizing and balancing node energy consumption [14]. All routing protocols have been implemented in SUNSET SDCS [12]. We used the Bellhop ray tracing tool via the WOSS interface [16] for accurate modeling of the underwater acoustic channel. The environmental data input to Bellhop refer to an area located in the Norwegian fjord off the coast of Trondheim, with the coordinates (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, from the General Bathymetric Chart of the Oceans (GEBCO), and from the National Geophysical Data Center's Deck41 data-base, respectively [17]. In the following we first describe the selected scenarios and protocol parameters settings (Section 4.1), we then introduce the investigated metrics (Section 4.2), and we finally report the results of our experiments (Section 4.3).

4.1. Simulation scenarios and settings

We consider UWSNs with $N = 6$, 20 and 40 nodes deployed according to a uniform random distribution in rectangular regions with surface of 1 km^2, 2 km^2 and 4 km^2, respectively. This allows us to investigate networks with size ranging from that of current deployments (6 nodes) to that of larger (20) and desirable (40) networks. In all scenarios nodes are deployed at different depths, ranging from 10 to 240 m, while the sink is located at one of the corners of the deployment area, 10 m below surface. Topology construction ensures that each node has at least one route to the sink.

We consider underwater nodes equipped with two acoustic modems with different characteristics. The first modem carrier frequency is set to 25.6 kHz for a bandwidth of 4 kHz, resulting in a bit rate of 4000 b/s. The second modem carrier frequency is higher, 63 kHz, for a bandwidth of 30 kHz and a bit rate of 9000 b/s. We assume a BPSK modulation for both modems. For the selected value of the bandwidth and of the carrier frequency the transmission power of the two modems is set to 2.8 W and 5.5 W. Their reception power is set to 0.5 W. These values are consistent with those of commercial modems by Teledyne Benthos [9] and Evologics [11].

We consider a scenario where nodes generate traffic based on the occurrence of events that are of interest to applications running on the nodes. As soon as an event happens, it is sensed by a node that starts producing packets with a given type, at a specific rate and for a variable period of time. Multiple events can happen

concurrently, and can be sensed by different nodes in the network. As a consequence, different types of packets are generated in the network at the same time. The inter arrival time of occurrences of events is exponentially distributed, with mean μ_a seconds. The event duration is also distributed exponentially, with mean μ_d seconds. When an event happens at a given location (chosen according to a random uniform distribution within the network deployment area) the node closest to it senses the event and generates the corresponding packets for the entire duration of the event. All packets from a sensed event are either urgent or reliable. Selection of the type of all packets from an event is controlled by the probability P_u that the event is urgent. Finally, the packet generation rate of an event is Constant Bit Rate (CBR). For instance, a CBR rate of one packet every 10 seconds for an event that lasts 200 seconds will generate 20 packets. In our simulations, we consider the following parameter setting: $\mu_a \in \{100, 200\}$ s, $\mu_d = 100$ s, $P_u \in \{0, 0.25, 0.5, 0.75, 1\}$, and CBR $\in \{10, 20\}$ s. Mixing these parameters up allows us to generate different network-wide aggregated traffic rates. Particularly, in this paper we consider the following rates.

1. *High traffic* (H): $\mu_a = 100$, $\mu_d = 100$ and CBR $= 10$ s.
2. *Medium-High traffic* (MH): $\mu_a = 200$, $\mu_d = 100$, and CBR $= 10$ s.
3. *Medium-Low traffic* (ML): $\mu_a = 100$, $\mu_d = 100$, and CBR $= 20$ s.
4. *Low traffic* (L): $\mu_a = 200$, $\mu_d = 100$, and CBR $= 20$ s.

The destination of all packets is the sink. The data packet payload size is set to 1000 bytes (B). The actual size of a transmitted data packet is given by its payload plus the bytes of the headers added at different layers. The physical header overhead changes according to the data rate but is dominated by a 10 ms synchronization preamble. At the MAC layer, the header size depends on the protocol. QELAR uses CSMA, whose header contains the sender and the destination addresses, and the packet type. Its length is 3 B. QELAR also needs 6 B extra for information on the residual energy and for the state space of the node. Being a cross layer protocol, CARP implements its own MAC, and the header of its MAC packets also carries routing information. As such, the size of its PING and PONG control packets is 10 B and 6 B, respectively. Its ACK and HELLO packets are 6 B long. The CARP MAC data packet header is 4 B long. Finally, as MARLIN-Q carries a number of information in the packet header, including the value function, $P_{i,j}$ estimates, list of back-up relays, etc., its size varies with the network size. In our implementation the MARLIN-Q header size was 7 B, 15 B and 30 B, depending on the network size. In our experiment, we have set the parameter ϵ of Algorithm ComputeR&M to 0.1, as typical [15]. The number of re-transmissions K used by MARLIN-Q for reliability packets, QELAR and CARP is set to 4 (low traffic), 3 (medium traffic) and to 2 (high traffic). The maximum length ℓ_i of the prioritized list of back-up relays of node i is set to 4 (low traffic), 3 (medium traffic) and to 2 (high traffic), and P_{lost} is set to 0.05.

4.2. Simulation metrics

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

- *Packet delivery ratio* (PDR), defined as the fraction of packets correctly received by the sink over the total number of packets generated by the nodes.
- *End-to-end latency*, defined as the time between packet generation and the time of its correct delivery to the sink.
- *Energy*, defined as the overall energy consumed by the network to correctly deliver all the data belonging to a specific type (i.e., reliable or urgent).

4.3. Performance results

We start by describing the performance of MARLIN-Q in networks with different size, traffic, and amount of urgent traffic. We finish our investigation with a comparative evaluation of the performance of MARLIN-Q, CARP and QELAR. For CARP and QELAR we show results obtained by using the modem that produces their best performance. All results are obtained by averaging over data from a number of simulation runs that achieves a statistical confidence of 95% within a 5% precision.

4.3.1. MARLIN-Q vs. network size

Fig. 3 shows the performance of MARLIN-Q for increasing network size. We show results from the most challenging scenario, namely, at the highest traffic: $\mu_a = 100$, $\mu_d = 100$, CBR $= 10$ s. The number of reliable packets generated in the network is the same as the one of urgent packets ($P_u = 0.5$).

As expected, the PDR decreases with increasing network size (Fig. 3a). This is due to the fact that in larger networks, packets travel longer routes (the average number of hops per route is 1.2 for small networks, which is almost half of that in networks with 40 nodes), and therefore incur a higher level of interference/re-transmissions. The PDR of reliable traffic is always higher than that of urgent packets, because of the re-transmission mechanism that MARLIN-Q implements for this kind of traffic. This is particularly noticeable in large networks ($N = 40$), where the PDR of reliable traffic is, on average, 13% higher than that of urgent packets.

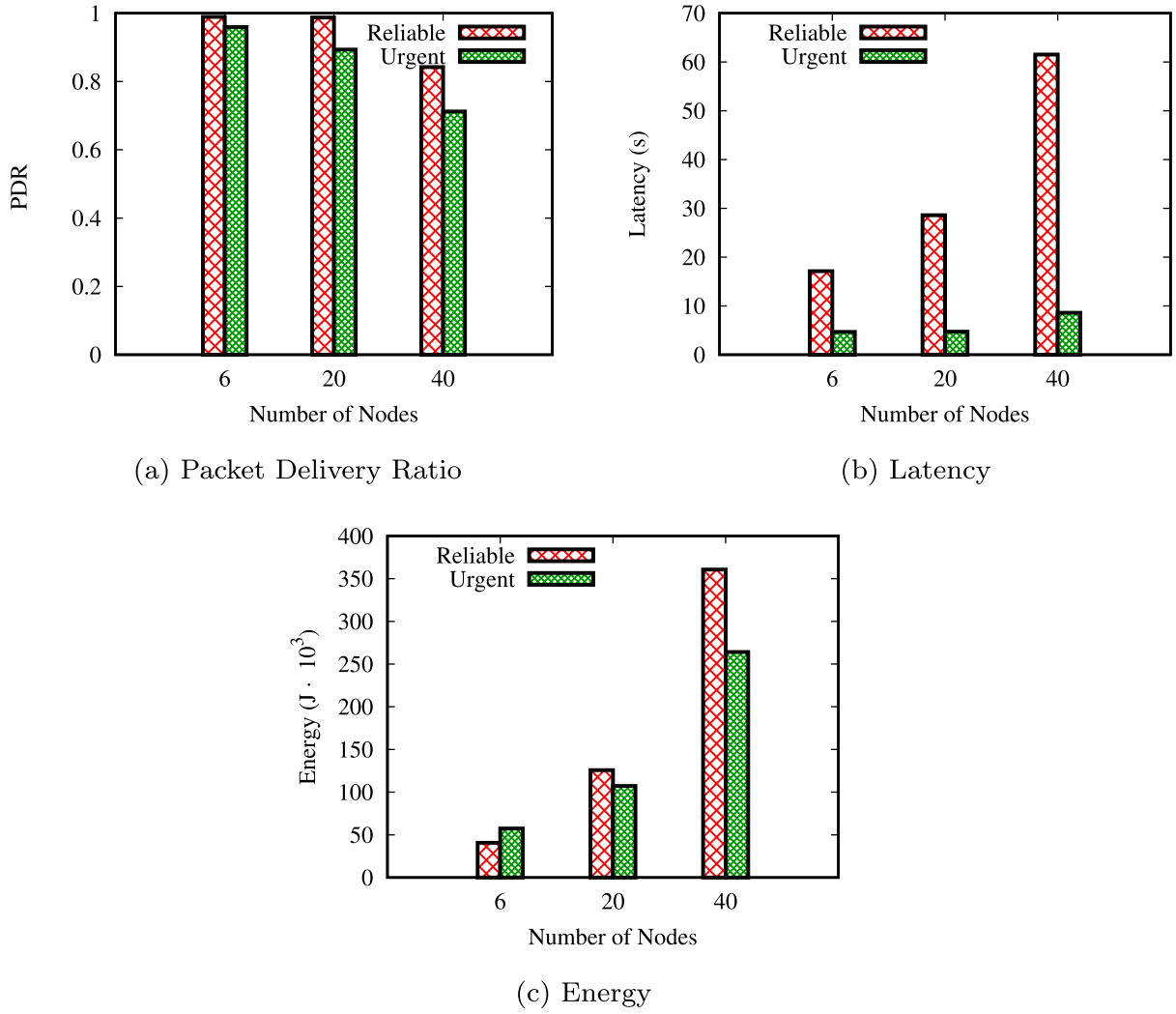
Latency increases with network size, also because of longer routes and increased interference (Fig. 3b). This is particularly evident for reliable traffic. Reliable packets are re-transmitted (while urgent packets are not), and re-transmissions occur more often in larger networks. Finally, as reliable packets are transmitted only if there are no urgent packets to be forwarded, they have to wait longer. Urgent packets incur the lowest latency, improving on the latency of reliable packets by up to 600% (large networks). Their latency is pretty stable: Independently of the network size it is always < 9 s.

Finally, energy also increases with increasing network size (Fig. 3b). This is because of the higher number of interfering transmissions and of the longer routes taken by the packets. In general, the energy spent to deliver reliable packets is higher than that needed for urgent ones, because of the re-transmissions allowed for reliable packets. This is particularly evident in large networks, where the number of interference is higher, therefore requesting a higher number of re-transmissions. In case of small networks, however, we notice that it costs more to deliver urgent packets (around 30% more). This is because the topology of small networks is so sparse that when node i chooses node j as main relay and node h as back-up relay, node h is so far from node j that it cannot overhear node j transmission. As a consequence, node h (also) transmits the packet to the sink. We have indeed observed that in this scenario the sink received most of the packets twice, which implies higher energy consumption. This clearly does not happen for reliable traffic, for which we observed that there are very few re-transmissions (because of the short routes of these networks, whose number of hops per route is slightly in excess of 1), and a very high PDR (almost 100%).

4.3.2. MARLIN-Q vs. traffic

Fig. 4 shows the performance of MARLIN-Q for increasing traffic. We show results for network with 40 nodes (results for smaller networks show similar trends and provide no further insights), and $P_u = 0.5$.

As expected, as the traffic increases the PDR decreases for both types of packets: The higher the traffic, the higher the interference and packet loss (Fig. 4a). Reliable packets are delivered more than



(a) Packet Delivery Ratio

(b) Latency

(c) Energy

Fig. 3. MARLIN-Q in networks with varying size. Scenario with high traffic.

urgent packets, because of the re-transmissions allowed for reliable data.

Latency also increases with increased traffic (Fig. 4b). Urgent packets are always considerably faster at reaching the sink than reliable packets. The improvement ranges from 100% (low and medium-low traffic) to 600% (high traffic). As noticed in Section 4.3.1, this is because higher traffic causes higher interference and hence it takes more time to forward reliable traffic. The latency of urgent packets is instead quite low independently of the traffic, consistently remaining under 9 s.

Finally, increasing the traffic increases the level of congestion in the network, which corresponds to increased energy consumption (Fig. 4c). The energy consumption of urgent traffic is consistently less than that of reliable traffic (up to 26% improvement, obtained at high traffic). This is because of the re-transmissions of reliable packets, which clearly impose higher energy requirements.

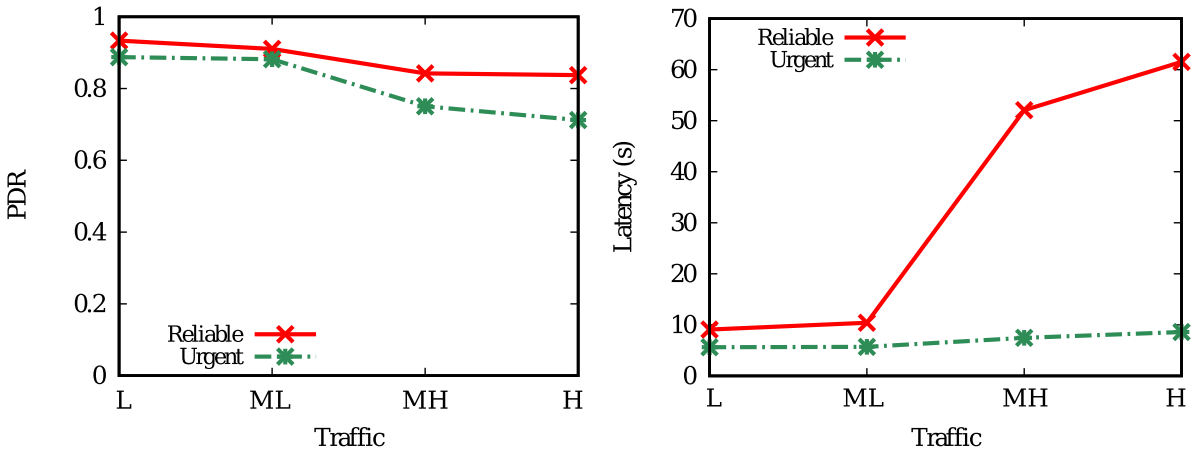
4.3.3. *MARLIN-Q vs. varying* P_u

In Fig. 5 we investigate the performance of MARLIN-Q in scenarios where we vary the amount of urgent traffic from 0 to 100%, i.e., P_u is varied from 0 to 1. We show results for large networks with high traffic.

The PDR of both types of traffic is fairly consistent irrespective of the value of P_u (Fig. 5a). The PDR of reliable packets is consistently higher (around 13%) than that to urgent traffic, due to the re-transmission mechanism that MARLIN-Q allows for reliable packets.

The latency of reliable packets decreases by increasing the amount of urgent traffic (Fig. 5b). We observe that the major factor affecting the latency of these packets is the overall number of re-transmissions in the network, which is clearly higher when "re-transmittable" (i.e., reliable) traffic is higher. Latency of urgent packets is instead consistently kept at bay by swift transmission and smart selection of best relay and modem. We note that the latency of urgent packet is not affected by the varying amount of reliable traffic, as independently of how many reliable packets there are, urgent packets are always transmitted before reliable packets. As mentioned, urgent packets find their way to the sink always in less than 9 s.

As the number of urgent packets increases the energy consumed to correctly deliver them increases proportionally, simply because there are more urgent packets to transmit (Fig. 5c). This same trend is observed for reliable packets: The more the packets of this type, the more energy is consumed to deliver them to the sink. As noticed, on average, the energy consumed for delivering reliable packets is higher than that needed for urgent packets because of the re-transmission mechanism that MARLIN-Q allows for reliable traffic.



(a) Packet Delivery Ratio

(b) Latency

(c) Energy

Fig. 4. MARLIN-Q in networks with varying traffic. Networks with 40 nodes and $P_u = 0.5$.

4.3.4. MARLIN-Q vs. CARP and QELAR

Results of the comparative performance evaluation of MARLIN-Q, CARP and QELAR are shown in Fig. 6. We consider the most challenging scenarios with large networks, $P_u = 0.5$ and vary the traffic from low to high. All metrics refer to averages from values for reliable and urgent traffic.

We observe that MARLIN-Q obtains the highest PDR independently of traffic (Fig. 6a). This is because of the smart way with which MARLIN-Q chooses the best modem on a per-link basis, each time selecting the device that provides the best forwarding. Improvements vary from 20% (low traffic) up to 100% (high traffic) over the second best performing protocol, CARP. We notice that CARP outperforms QELAR because of its channel reservation mechanism, which, especially at lower traffic, provides higher probability of collision free channel access.

In spite of a PDR significantly higher than that of the other two protocols, MARLIN-Q exhibits always the lowest latency. This is because the learning-based framework that governs MARLIN-Q operations explicitly takes latency (and link robustness) into account in its cost function (Section 3). Not surprisingly, CARP experiences the highest latency because of the channel reservation phase needed prior to packet transmission.

By choosing relays and modems smartly (which include the ability of a link to forward packets successfully), MARLIN-Q always exhibits excellent performance by consuming the lowest amount of energy. CARP also avoids spending extra energy for retransmission through its channel reservation phase, which results in fewer packet collisions. However, the energy spent for control packets raises its energy toll. This is particularly evident at low traffic, where CARP spends up to 44% more energy than that spent by MARLIN-Q. The two protocols exhibit similar performance only at high traffic, which is where the CARP channel reservation is at its most effective.

5. Related work

In this section we review solutions in the realm of UWSNs that use the concepts and techniques used in this paper, namely, multi-modal communication for underwater networking, data delivery in UWSNs, and underwater solutions for data delivery that use machine learning-guided ideas.

Multi-modal communications have emerged as a means to enhance UWSN reliability and performance in a variety of scenarios. Some of the existing works concern combining communication technologies at the extremes of the transmission range spectrum and of the spectrum of data rates. An example is provided by the joint usage of acoustic communication for the long haul, more robust, low data rate exchanges, and of short-range, high data rate optical packet transfer [3–8]. While these works show that concurrent use of multiple communicating devices overcomes engaging challenges of underwater data transfer, they do not concern data delivery, as we do in this paper. To the best of our knowl-

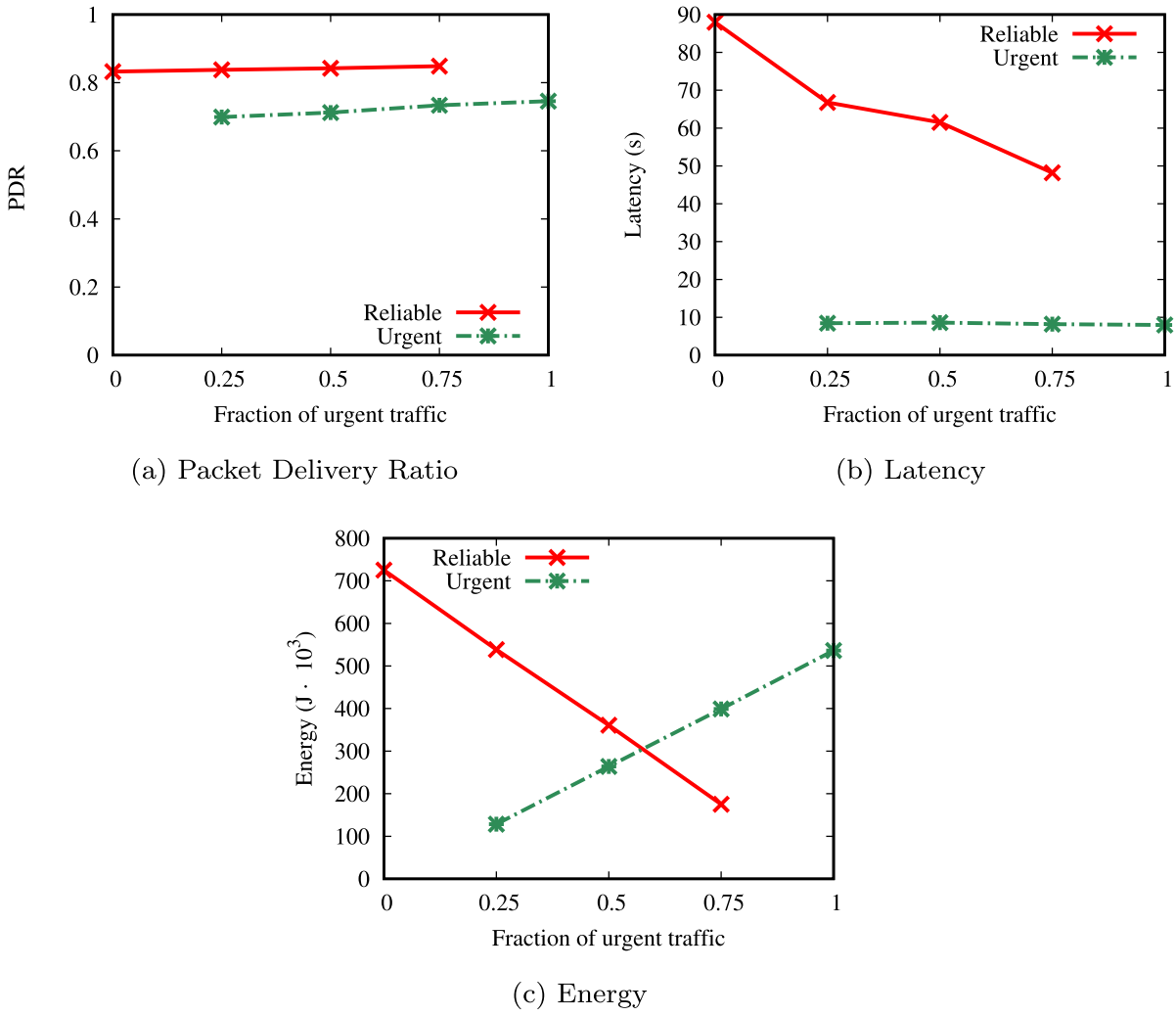
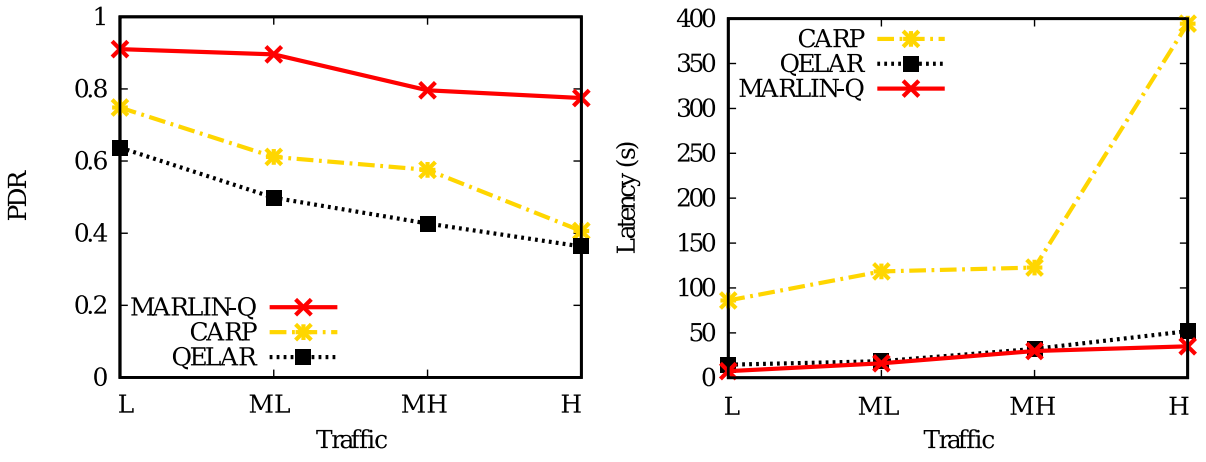


Fig. 5. MARLIN-Q in networks with varying amount of urgent traffic. Networks with 40 nodes and high traffic.

edge, a first attempt at underwater multi-modal routing is provided by Hu and Fei's MUlti-level Routing protocol for Acoustic-Optical UWSNs (MURAO) [6]. This solution concerns partitioning the network nodes into two layers. Lower layer nodes are responsible for multi-hop data forwarding over optical channels. Nodes in the upper layer use long range/low bandwidth acoustic communication to coordinate the routing of the lower level nodes. Actual data routing within the two layers is performed by QELAR, a machine learning-based routing protocol for single-modem UWSNs (described below). MURAO requires nodes to be deployed densely enough to obtain a connected topology over the optical links. Given the considerably short range of optical communication, however, MURAO can be costly and even impracticable for applications requiring coverage of large areas.

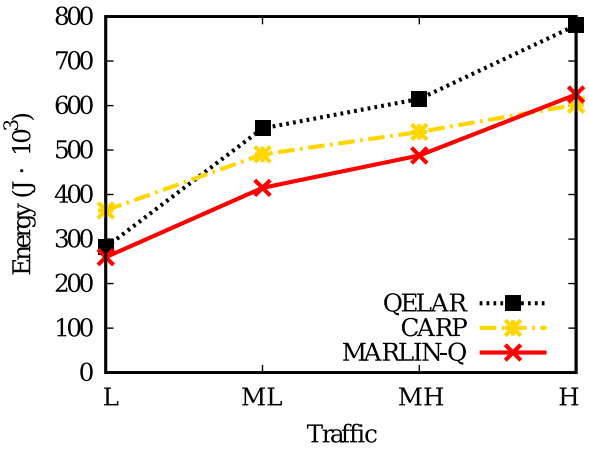
While multi-modal routing is still quite the unexplored topic, routing protocols for UWSNs with single-mode acoustic modems have been proposed for over a decade now, and include remarkably effective solutions, including [13,18–21] and those surveyed by Ayaz et al. [22] and by Li et al. [23]. A solution that stands out in terms of enhanced performance is the Channel-aware Routing Protocol (CARP) by Basagni et al., which exploits link quality information for data forwarding [13]. 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 achieving reliability and suffering from few packet collisions, it incurs remarkable latency. Furthermore, in networks with high traffic nodes have troubles in gaining rights to the channel, which results in quickly decreasing PDR. MARLIN-Q uses a channel aware approach similar to that used by CARP for selecting the next hop relay for a packet. However, instead of selecting a relay through channel reservation, it smartly uses our reinforcement learning-based framework for both relay and modem selection, obtaining CARP performance on collisions, but also remarkably better PDR, latency and energy consumption (Section 4.3.4).

Reinforcement learning has been already successfully applied to routing problems in multi-hop wireless networks, including wireless ad hoc networks, wireless sensor networks and cognitive radio networks (see [24] for an extensive survey) and more recently to UWSNs [6,14,25,26]. The advantage of learning-based routing algorithms is that they are able to learn optimal routing policies online, and are thus capable to remain optimal or near optimal in a dynamic environment. Furthermore, learning-informed algorithms are often amenable to distributed implementation and require relatively small communication and computation overhead. These are all essential and desirable features for the resource constrained UWSNs environment. Solutions presented in [25,26] concern the specific scenario of networks with intermittent connectivity, which is not similar to the scenario considered here. The QELAR protocol by Hu and Fei [14] has been introduced for routing in scenarios similar to those considered in this paper. QELAR is based on a model-based *Q-learning* approach aimed at maximizing the resid-



(a) Packet Delivery Ratio

(b) Latency



(c) Energy

Fig. 6. MARLIN-Q vs. CARP and QELAR. Large networks, and $P_u = 0.5$.

ual energy among nodes. The learning cost function accounts for the residual energy of each node as well as for the energy distribution among neighboring nodes, and relays are chosen depending on the energy they can save. This makes QELAR a solution that compares well with previous protocols, especially in terms of network lifetime. However, the QELAR model leads to routing decisions that are prone to packet loss and to unfairness, especially to nodes far away from the sink. This, as observed in Section 4.3.4, leads to degraded performance, especially in larger networks.

To the best of our knowledge, the only solution that reaps the joint positive effects of multi-modality and machine learning-based routing is provided by the *Multi-modAl Reinforcement Learning-based RoutINg* (MARLIN) protocol [27], which provides the design platform from which we built MARLIN-Q. The core difference between the two protocols concerns the fact that MARLIN could be configured at the nodes to support only one soft QoS class. For instance, concerning our performance investigation, a node using MARLIN could run either applications requesting reliability or applications concerned with low-latency delivery, but not both. Our MARLIN-Q, instead, provides nodes with the capability to run the reinforcement learning framework for both types of traffic, thus allowing them to run multiple applications with different QoS requirements. The simulation-based performance evaluation provided in this paper clearly demonstrates the heightened effectiveness of MARLIN-Q in obtaining reliable and low-latency data delivery in a wider class of scenarios with respect to those where nodes could use MARLIN.

6. Conclusions

This paper concerns UWSNs with nodes with multi-modal communication capabilities. We present a reinforcement learning-based framework for senders to jointly select the best forwarding relay for their data and the best communication device to reach that relay. The resulting forwarding method, named MARLIN-Q for Multi-modAl Reinforcement Learning-based RoutINg with soft QoS capabilities, allows nodes to perform routing depending on the nature of the data-creating application they run. In other words, applications can choose among different soft QoS data delivery services. Through a SUNSET SDCS-based study we show that MARLIN-Q always shows excellent performance in scenarios with varying network size, varying traffic and varying amount of urgent traffic. We also show that it always outperforms state-of-the-art underwater forwarding protocols by delivering more packets, faster, and by spending considerable low energy. Our results clearly show that the smart use of multi-modal communications takes underwater networking to levels of reliability and low latency long demanded by the majority of key underwater applications.

Acknowledgments

The work was performed under the sponsorship of the EC EASME ArcheoSub project “Autonomous underwater Robotic and sensing systems for Cultural Heritage discovery Conservation and in situ valorization.” (Grant Agreement

n. EASME/EMFF/2016/1.2.1.4/01/SI2.749264.) Stefano Basagni was supported in part by grants NSF CNS 1428567 ("MRI: Development of the Northeastern University Marine Observatory NETwork—NU MONET") and NSF CNS 1726512 ("MRI: SEANet: Development of a Software-Defined Networking Testbed for the Internet of Underwater Things"). Part of this work has been performed while Dr. Basagni was visiting the department of Computer Science of the University of Rome "La Sapienza," while on sabbatical leave from Northeastern University. During his stay he was also supported by a grant from the Sapienza Visiting Professor programme (2016).

References

[1] F.S. Rende, A.D. Irving, A. Lagudi, F. Bruno, S. Scalise, P. Cappa, M. Montefalcone, T. Bacci, M. Penna, B. Trabucco, R. Di Mento, A.M. Cicero, Pilot application of 3D underwater imaging techniques for mapping posidonia oceanica (l.) delile meadows, in: ISPRS—International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XL-5, 2015, pp. 177–181.
[2] D. Scaradozzi, L. Sorbi, F. Zoppini, DiRAMa facilitates data gathering and analysis at sea, Sea Technol. Mag. 55 (6) (2014) 19–22.
[3] P. Gjanci, C. Petrioli, S. Basagni, C.A. Phillips, L. Bölöni, D. Turgut, Path finding for maximum value of information in multi-modal underwater wireless sensor networks, IEEE Trans. Mob. Comput. 17 (2) (2018) 404–418.
[4] F. Campagnaro, F. Guerra, P. Casari, R. Diamant, M. Zorzi, Implementation of a multi-modal acoustic-optic underwater network protocol stack, in: Proceedings of MTS/IEEE OCEANS 2016, Shanghai, China, 2016, pp. 1–6.
[5] F. Campagnaro, F. Favaro, F. Guerra, S.V. Calzado, M. Zorzi, P. Casari, Simulation of multimodal optical and acoustic communications in underwater networks, in: Proceedings of the MTS/IEEE OCEANS 2015,Genova, Italy, 2015, pp. 1–6.
[6] T. Hu, Y. Fei, MURAO: a multi-level routing protocol for acoustic-optical hybrid underwater wireless sensor networks, in: Proceedings of SECON 2012, Seul, Korea, 2012, pp. 218–226.
[7] N. Farr, A. Bowen, J. Ware, C. Pontbriand, M. Tivey, An integrated, underwater optical/acoustic communications system, in: Proceedings of the MTS/IEEE OCEANS 2010, Sydney, Australia, 2010, pp. 1–6.
[8] N. Farr, J. Ware, C. Pontbriand, T. Hammar, M. Tivey, Optical communication system expands CORK seafloor observatory's bandwidth, in: Proceedings of the MTS/IEEE OCEANS 2010, Seattle, WA, 2010, pp. 1–6.
[9] The Teledyne Benthos SMART product SM-975, http://teledynebenthos.com/product/smart_products/sm-975.
[10] E. Demirors, G. Sklivanitis, T. Melodia, S.N. Batalama, D.A. Pados, Software-defined underwater acoustic networks: toward a high-rate real-time reconfigurable modem, IEEE Commun. Mag. 53 (11) (2015) 64–71.
[11] Evologics, Evologics S2C acoustic modems, https://www.evologics.de/files/DataSheets/EvoLogics_S2CR_735Modems_a4_WEB.pdf.
[12] C. Petrioli, R. Petroccia, J. Potter, D. Spaccini, The SUNSET framework for simulation, emulation and at-sea testing of underwater wireless sensor networks, Els. Ad Hoc Netw. 34 (C) (2015) 224–238.
[13] S. Basagni, C. Petrioli, R. Petroccia, D. Spaccini, CARP: a channel-aware routing protocol for underwater acoustic wireless networks, Ad Hoc Netw. 34 (2015) 92–104.
[14] T. Hu, Y. Fei, QELAR: a machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks, IEEE Trans. Mob. Comput. 9 (6) (2010) 796–809.
[15] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.
[16] F. Guerra, P. Casari, 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, 2009, pp. 1–8.
[17] WOD, GEBCO, and Deck41, http://www.nodc.noaa.gov/OC5/WOA05/pr_woa05.html, http://www.gebco.net, http://www.ngdc.noaa.gov/mgg/geology/deck41.html.
[18] Y. Noh, U. Lee, P. Wang, B.S.C. Choi, M. Gerla, VAPR: Void-aware pressure routing for underwater sensor networks, IEEE Trans. Mob. Comput. 12 (5) (2013) 895–908.
[19] X. Xiao, X.P. Ji, G. Yang, Y.P. Cong, LE-VBF: lifetime-extended vector-based forwarding routing, in: Proceedings of CSSS 2012, Nanjing, China, 2012, pp. 1201–1203.
[20] D. Shin, D. Hwang, D. Kim, DFR: an efficient directional flooding-based routing protocol in underwater sensor networks, Wireless Commun. Mobile Comput. 12 (17) (2012) 1517–1527.
[21] D. Pompili, I.F. Akyildiz, A multimedia cross-layer protocol for underwater acoustic sensor networks, IEEE Trans. Wireless Commun. 9 (9) (2010). 1536–1276
[22] M. Ayaz, I. Baig, A. Abdullah, I. Faye, A survey on routing techniques in underwater wireless sensor networks, J. Netw. Comput. Appl. 34 (6) (2011) 1908–1927.
[23] N. Li, J.-F. Martnez, J.M. Meneses Chaus, M. Eckert, A survey on underwater acoustic sensor network routing protocols, Sensors 16 (3) (2016) 1–28.
[24] H.A. Al-Rawi, M.A. Ng, K.-L. Yau, Application of reinforcement learning to routing in distributed wireless networks: a review, Artif. Intell. Rev. 43 (3) (2015) 381–416.
[25] R. Plate, C. Wakayama, Utilizing kinematics and selective sweeping in reinforcement learning-based routing algorithms for underwater networks, Ad Hoc Netw. 34 (2015) 105–120.
[26] T. Hu, Y. Fei, An adaptive routing protocol based on connectivity prediction for underwater disruption tolerant networks, in: Proceedings of IEEE Globecom 2013, Atlanta, GA, 2013, pp. 65–71.
[27] S. Basagni, V. Di Valerio, P. Gjanci, C. Petrioli, Finding MARLIN: exploiting multi-modal communications for reliable and low-latency underwater networking, in: Proceedings of IEEE Infocom 2017, Atlanta, GA, 2017, pp. 1–9.

Stefano Basagni is an associate professor at the Department of Electrical and Computer Engineering 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 research 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 dozens of highly cited, refereed technical papers and book chapters. His h-index is currently 40 (August 2018). 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).

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.

Petrika Gjanci is a Postdoc researcher at the University of Rome "La Sapienza". He received the BS and MS degrees in computer engineering from the University of Rome "La Sapienza," in 2009 and 2012, respectively, and the PhD degree in computer science from the University of Rome "La Sapienza", in March 2017. His research interests concern theoretical and experimental aspects of wireless sensor networking, with an emphasis on underwater communication and networking. He has collaborated extensively to large research EU funded projects such as SUNRISE and CLAM and has published over a dozen of papers in the most prominent venues.

Chiara Petrioli received a Ph.D. in Computer Engineering from the University of Rome "La Sapienza" in 1998. She is currently a professor with the Computer Science department of "La Sapienza." Prof. Petrioli is director of the Sensor Networks and Embedded Systems laboratory (SENSES lab) and of the Cyber Physical System lab of "La Sapienza" center for Cyber Intelligence and Information Security. She is also founding partner of "La Sapienza" spinoff WSENSE, S.r.l. Her research interests focus on the design and optimization of wireless, embedded and cyber physical systems. Prof. Petrioli is chair of the steering committee of IEEE SECON, was program co-chair of IEEE INFOCOM 2016, workshop co-chair of ACM MobiCom 2014, and general chair of ACM SenSys 2013, and is general co-chair of ACM MobiHoc 2019. 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. She has been program co-chair of leading conferences including ACM MobiCom and IEEE SECON. Prof. Petrioli has published over a hundred fifty papers in prominent international journals and conferences (with over 5400 citations; h-index 40). She has also been PI of over twenty national and international research projects, serving as coordinator of three EC projects (FP7 projects GENESI and SUNRISE, EASME ArcheoSub). Prof. Petrioli was a Fulbright scholar.