CARP: A Channel-aware Routing Protocol for Underwater Acoustic Wireless Networks

Stefano Basagni[a], Chiara Petrioli[b,c], Roberto Petroccia[b,c], Daniele Spaccini[b]

[a]*ECE Dept., Northeastern University, 360 Huntington Ave., Boston, MA, 02115, U.S.A*
Email: basagni@ece.neu.edu
[b]*Dipartimento di Informatica, Università di Roma "La Sapienza," Via Salaria, 113, 00198, Roma, Italy.*
E-mail: {petrioli,petroccia,spaccini}@di.uniroma1.it
[c]*WSENSE s.r.l., Rome, Italy.*

Abstract

The paper concerns the definition and performance evaluation of a new multi-hop routing protocol for underwater wireless sensor networks. Our solution, termed CARP for Channel-aware Routing Protocol, exploits link quality information for data forwarding, in that nodes are selected as relays if they exhibit recent history of successful transmissions to their neighbors. CARP avoids loops and can successfully route around connectivity voids and shadow zones by using simple topology information, such as hop count. The protocol is also designed to take advantage of power control for selecting robust links. The performance of CARP has been compared with that of two other protocols for underwater routing, namely, the Focused Beam Routing (FBR) and a flooding-based solution (EFlood). Metrics of interest include packet delivery ratio, end-to-end packet latency and energy consumption, which have been investigated through ns2-based simulations and experiments at sea. The in-field trials have been conducted at two European locations, namely, a Norwegian fjord and the Mediterranean Sea. The tests in the Mediterranean Sea have been performed jointly with the NATO Science and Technology Organization Centre for Maritime Research and Experimentation (STO CMRE), under a collaboration agreement between the University of Roma and CMRE.[1] Our results show that CARP robust mechanism for relay selection doubles the packet delivery ratio of FBR and EFlood. CARP also outperforms FBR and EFlood in terms of energy consumption, and delivers packets significantly faster than FBR.

Keywords: Underwater wireless acoustic networks, Cross-layer design, Channel-aware communications, In-field experimentation.

---

[1] We gratefully acknowledge CMRE for their invaluable feedback and for providing expertise and logistics during experimentation.

1. Introduction

Underwater wireless sensor networks (UWSNs) are an enabling technology for many autonomous systems that find application to science, security, and industry. The diversity of aquatic applications ranges from environmental monitoring, pollution control, prediction of and reaction to natural disasters, eco-system analysis, surveillance for defense applications and port safety, off-shore oil and gas industry, aquaculture, geological and oceanographic science, marine, biology and archaeology [1, 2]. Most of these applications are currently served by piecemeal networking infrastructures, mostly based on costly underwater cabling or limited, single hop acoustic communications. Recently, however, research on UWSNs has focused on the design of more flexible and performant networks that can cover larger areas of the submarine environment. This is made possible by deploying a variety of sensor nodes capable to transmit their data wirelessly to collections points (*sinks*) through multi-hop communications. Sinks may then forward the received information to onshore control stations, usually via RF transmissions.

Challenges to the design of communication protocols for the new, multi-hop UWSN architecture come from the very specific environment where they operate. Long propagation delays, low bandwidth, sound speed variability, slow power signal attenuation, and the rapidly changing conditions of the underwater channel affect UWSNs at all levels of their design and deployment. In particular, MAC and routing protocols face new challenges with respect to their terrestrial counterparts, in that channel access mechanisms and routing techniques are highly affected by the link quality whose variations are less predictable and manageable than that of wireless radio links. Recent trends in protocol design show that cross-layer techniques can impact protocol performance positively, especially in networks with limited resources and/or deployed in challenging environments, like UWSNs [3]. An example of cross-layer routing is provided by the Focused Beam Routing (FBR) protocol, proposed by Jornet et al. in [4]. Control packets (RTS and CTS as in CSMA/CA-like channel access) are used to reserve the channel, a typical MAC function. The control packets, however, carry also node location information that is used for routing purposes, namely, to select as next-hop relay the node that is closest to the sink. The use of short packets for channel access and relay selection results particularly effective for routing in the challenged UWSNs environment. This is because their transmissions are relatively fast even at the low rates of acoustic modems, and because they are less subject to bit errors and interferences. However, in the quickly varying conditions of the underwater channel, the fact that two nodes can exchange short control packets correctly, may not be sufficient to guarantee that longer data packets are also going to be safely delivered. In other words, protocols like FBR that do not consider link quality information as a criterion for relay selection are still heavily prone to bit errors. This is a problem that affects many other cross-layer solutions (see Section 2).

In this paper, we present a new distributed cross-layer Channel-aware Routing Protocol (CARP) for UWSNs for multi-hop delivery of data to the sink. While still reaping the benefits of cross-layer design (short packets for robust channel access and relay selection), CARP obviates to the drawbacks of other solutions in that link quality is explicitly taken into account for selecting the next-hop node on a route to the sink. History of successful transmissions to neighboring nodes is maintained and used in relay selection. Other characteristics that make CARP relay selection particularly suitable for implementing multi-hop routing in UWSNs include the following: (i) The use of simple topology information (hop count) for routing around connectivity holes and shadow zones, thus avoiding the well-known pitfalls of geographic routing; (ii) considering residual energy and buffer space, and (iii) taking advantage of power control, if available, for selecting transmission powers so that shorter control packets experience a similar Packet Error Rate (PER) of longer data packets.

The performance of CARP has been evaluated through simulations and through trials at sea, measuring metrics such as packet delivery ratio, packet latency and energy consumption. Simulations have been performed by implementing CARP in SUNSET [5], our extension to the ns-2 simulator specifically designed for underwater networking. We used the Bellhop ray tracer [6] and historical environmental data for more realistic modeling of the underwater acoustic channel. Through Bellhop we have been able to compute the frequency-dependent acoustic path loss of each source-destination pair at a given location, as well as the spatially-varying interference induced by all active nodes. The performance of CARP has been compared to that of FBR [4] and of an enhanced version of flooding, called EFlood, where nodes wait for a random time before forwarding the packet (this desynchronizes node transmissions and reduces interferences, improving the performance of common flooding). We consider networks with desirable size (20 nodes), varying traffic and packet size. Experiments at sea have been performed at two European locations, namely, a Norwegian fjord and the Mediterranean Sea, in multi-hop network of realistic size (6 and 8 nodes, respectively). In the trials, the performance of CARP has been compared to that of EFlood. Both simulations and in-field tests show that CARP is an effective solution for transmitting packets through time-varying channels. The protocol is capable of routing around connectivity holes and shadow zones, and of maintaining a high packet delivery ratio for increasing traffic, irrespective of the ratio between transmission and propagation times. Our simulation results show that CARP outperforms both FBR and EFlood with respect to packet delivery ratio, being always able to deliver 80% of the packets, independently of the wide variety of traffic and data packet sizes we considered, in simulations and at sea. At higher traffic FBR and EFlood show instead decreasing performance: EFlood can deliver only up to 35% of the generated packets, and FBR only up to 40%. CARP also consistently outperforms EFlood in trials at sea. The energy saved by CARP is up to 100% than that of FBR, and up to 400% (simulations) and 63% (in-field) than the energy saved by EFlood. In terms of end-to-end packet latency, our simulations show that both CARP and FBR incur longer delays with respect to EFlood, since they both use an handshaking mechanism for channel reservation. CARP, however, significantly outperforms FBR, proving suitable for actual underwater applications.

The rest of the paper is organized as follows. Previous work on underwater multi-hop routing is summarized in Section 2. In Section 3 we describe CARP in detail. Section 4 illustrates experimental results. Finally, Section 5 concludes the paper.

2. Related Works

Protocols for underwater communications have recently received noticeable attention [7, 8, 9, 10]. Among the first protocols to tackle the problem of finding routes from an underwater sensor node to the sink is the Vector-Based Forwarding (VBF) protocol of Xie et al. [11]. Nodes forward packets by broadcasting them to nodes residing in a constrained "pipe" of given width in the direction of the sink. The pipe surrounds a virtual line (a *vector*) between the packet source node and the sink. The efficiency of the protocol, especially its throughput, depends on the critical determination of the radius of the pipe: If the radius is too small, few or no relays can be found in the pipe; if it is too large, too many nodes might receive the packet, whose retransmission increases interference, overhead, and duplicate packets. VBF has been enhanced in [12] and [13]. Nicolaous et al. define an hop-by-hop version of VBF, HH-VBF [12], introducing multiple vectors from each relay to the sink for increasing the probability to find a relay in the pipe, especially in sparse networks. The LE-VBF protocol of Xiao et al. extends VBF to include energy information in the relay selection mechanism, so to extend the network lifetime [13]. The performance of both protocols depends on the correct determination of the radius of the forwarding pipe, and suffers from high overhead. A different flooding-based geographic solution is investigated by Shin et al. through their *Directional Flooding-based Routing* (DFR) [14]. Data packets are broadcast by each node S to all its neighbors. Based on directional information, namely, by the angle SFD between the sender S, the forwarder F and the destination D (i.e., the sink), a node decides whether to forward the packet or not. The decision is made by comparing the angle SFD with a reference angle carried by the packet. The varying conditions of the underwater channel are addressed by changing the reference angle on a hop-by-hop basis according to the link quality: The better the latter, the smaller the flooding zone. Since determining geographic information underwater can be difficult, or could require high cost/overhead, some protocols use partial geographic information. This kind of information, such as *depth* (distance from the surface), can be easily determined with high accuracy [15]. This is the case of DBR, the *Depth Based Routing* protocol, presented by Yan et al. [16]. Each node that receives a data packet forwards it only if its depth is less than that of the sender. Before forwarding the data packet, a node holds the packet for a time that depends on the difference between its own depth and that of the sender. In particular, the larger the vertical distance, the smaller the holding time, so that nodes that are closer to the surface (i.e., to the sink) are the first to forward the data packet. While holding, a node listens to the transmissions on the channel. If it overhears that the packet that it is about to broadcast is transmitted by another node, the node drops the packet. Protocols such as HydroCast (Lee et al. [17]) and VAPR (Noh et al. [18]) also base forwarding decisions based on a node depth. The idea of these solutions is similar to that of DBR: A node will forward a packet only if other nodes closer to the sink cannot send it. HydroCast tries to find a set of possible relays that maximize "expected packet advance" [19], while limiting the number of forwarding nodes to reduce redundant transmissions, packet collisions and therefore co-channel interference and the impact of hidden terminal phenomena. Recovery route strategies are provided where alternative routes are discovered through a limited hop search of a 2D surface of a convex hull around a void zone. VAPR uses the same forwarding set selection algorithm of HydroCast. Nodes know their next-hop neighbor towards the sink thanks to the surface reachability information flooded starting from the sink via periodic beaconing. Using the created directional trails the protocol performs a *local opportunistic directional flooding* to deliver the data. Simulation results show that VAPR outperforms HydroCast, DBR and a generic routing protocol where routes are only based on the distance in hops from the sink. Depth as expressed by hop count (distance from the sink) is also the main criteria for relay selection in the tier-based underwater acoustic routing protocol proposed by Kuo and Melodia [20]. Next hop relays are selected and transmission power is set with the aim of reducing energy consumption subject to requirements on end-to-end PER and delay. Toso et al. propose a modification of the Dynamic Source Routing (DSR) protocol [21] that makes it suitable for UWSNs [22]. The protocol, termed SUN for Source routing for Underwater Networks, determines routes to the sink based on the hop distance of a node from the sink and on the Signal to Noise Ratio (SNR) of links in the route. Choices are made according to the actual application and network scenario.

Relevant contributions to the cross-layer approach to data forwarding, proven effective for multi-hop wireless radio networks [23], have been made for UWSNs by Pompili et al. [24], Jornet at al. [4] and by Pompili and Akyildiz [25]. Pompili et al. present two distributed protocols for delay-insensitive and delay-sensitive applications, respectively, where each node jointly select its best next hop, the transmitted power, and the forward error correction (FEC) rate for each packet in a cross-layer fashion [24]. The objective is that of minimizing energy consumption, taking the condition of the underwater channel and application requirements into account. More recently, Pompili and Akyildiz explore the cross-layer interactions of key underwater functionalities such as modulation, FEC, MAC and routing to develop a communication protocol for the efficient and fair share of the bandwidth-limited high-delay underwater acoustic medium [25]. In particular, their attention is to devise methods that minimize delays, and therefore result in a protocol suitable for underwater multimedia communications. Jornet at al. introduce the *Focused Beam Routing* (FBR) [4]. Nodes know their own location and the location of the sink (geographic routing). When a node has a data packet to send, it first transmits a control packet (e.g., a request-to-send, RTS, packet) and waits for replies from nodes in a cone centered on the line between itself and the sink. Each node that receives an RTS packets replies with a clear-to-send (CTS) packet containing its position along with other information. Among all nodes that reply, the closer to the sink is chosen as next-hop relay. FBR also assumes that nodes can choose their transmission power within a set of different power levels, P_1 through P_n. If there are no nodes within the hearing range at power P_i, the transmitter increases its power to P_{i+1}, and tries again. The process continues until a relay is found or the packet is discarded. (Further details on the cited protocols and on additional ones can be found in recent surveys on the subject [8, 9, 10].)

Whether using a cross-layer approach to routing or not, the cited solutions focus on energy consumption and delay minimization devising methods that, while considering many of the characteristics of the underwater channel, do not take fully into account the quality of the channel and its variation in time. Our proposed protocol, CARP, chooses next-hop relays depending on recent information on link quality, achieving packet delivery ratios and delays that are consistently better than those obtained by previous solutions.

3. CARP: Protocol Description

We start by defining the operation performed at network initialization and then we describe data forwarding.

3.1. Network initialization and hop count setting

Every node x stores its hop distance from the sink in the variable $HC(x)$, initially set to a very large value (higher than the longest possible route in the network). The sink HC variable is set to 0. At the start of network operations the sink floods a HELLO packet throughout the network. This packet carries the unique ID (src) of the node forwarding it (initially, it will be the sink ID) and the hop count HC value: HELLO = $\langle src, HC \rangle$. Upon receiving a HELLO packet a node x checks whether its hop count variable $HC(x)$ is greater than the hop count HC carried by the packet plus 1. If it is, the node updates its $HC(x)$ to the value in the HELLO packet plus 1 and retransmits it updating the packet hop count field to $HC(x)$; otherwise, the node drops the packet. When this set up phase has finished each node knows its hop distance from the sink.

3.2. Data forwarding

When a node x has one or more data packets to forward it sends a broadcast control packet, called PING, searching for a suitable relay in its neighborhood. The PING packet carries the following information: PING = $\langle src, HC, pid, L_{pkt} \rangle$. The field src is node x unique identifier; HC is the hop count value of x; pid is the unique identifier of the current PING packet (the pair $\langle src, pid \rangle$ identifies uniquely each PING packet); L_{pkt} is a list of pairs $\langle pkt_{src}, pkt_{id} \rangle$ each identifying one data packet that x has to forward. Particularly, pkt_{src} is the unique identifier of the node that generated the packet whose ID is pkt_{id}. If the number of these pairs is > 1 then a train of data packets is transmitted back to back. We stipulate that the maximum number of packets transmitted back to back, i.e., the length of a train, is $\tau \geq 1$.

A node y that receives a PING packet from node x and such that $HC(y) \leq HC(x)$ replies to x transmitting a PONG packet. The PONG packet carries the following information.

$$\texttt{PONG} = \langle src, dst, pid, HC, queue, energy, lq, bit\_mask_{L_{pkt}}, bit\_mask_{JR} \rangle.$$

The fields src and dst contain the identifiers of nodes y and x, respectively. The pid parameter is that of the received PING: The PING and PONG packets are always paired by this parameter. The field HC contains $HC(y)$. The field $queue$ indicates the available buffer space at y, i.e., the number of packets that y can store in its incoming data queue. The parameter $energy$ indicates the residual energy available at node y. The parameter lq is an indication of the quality of the links outgoing from y (see detailed description below). The fields $bit\_mask_{L_{pkt}}$ and bit_mask_{JR} are used for handling link asymmetries, i.e., unidirectional links, as detailed below, and interference. In particular, they report about packets already received by the sink and by the selected relay, respectively. Upon receiving a PING packet from x, a node y executes Algorithm 1.

Algorithm 1 UponReceivingPing(PING$\langle src, HC, pid, L_{pkt}\rangle$) **from** x

if $(HC(x) < HC(y))$ **then**
  drop PING
else
  $bit\_mask_{L_{pkt}} \leftarrow \emptyset$
  $bit\_mask_{JR} \leftarrow \emptyset$
  for all packet $pkt \in L_{pkt}$ **do**
    if (sink has already received pkt) **then**
      Set pkt position in $bit\_mask_{L_{pkt}}$ to 1
    if (I have already received pkt from x) **then**
      Set pkt position in bit_mask_{JR} to 1
  send PONG = $\langle src, x, pid, HC(y), queue, energy, lq, bit\_mask_{L_{pkt}}, bit\_mask_{JR}\rangle$

If the hop count value of node y is greater than that carried by the PING packet, y drops the PING. Otherwise, it first checks which L_{pkt} data packets have been already received by the sink and sets the $bit\_mask_{L_{pkt}}$ bits of the delivered packets to 1. It then checks if it has received some of the L_{pkt} data packets from x, and sets the corresponding bits of bit_mask_{JR} to 1 accordingly. Finally, the PONG packet is sent to node x.[2]

Relay selection happens as follows. After sending a PING, node x awaits for PONG replies from its neighbors for a time δ. The value of δ is set to the round trip time to the farther neighbor plus the transmission delays of a PING and of a PONG packet. If x does not receive any replies from its neighbors, it keeps sending the same PING for at most k times. After these k attempts, it increases the PING HC field to the maximum hop count value, so that every one of its neighbors (if any) will reply with a PONG. (If, again, no PONG is received, the data packets are dropped.) After the time δ node x executes Algorithm 2. For each PONG received, it checks which data packets can be removed from its queue because they have been already received by the sink (their bits in $bit\_mask_{L_{pkt}}$ are set to 1). Then it chooses the best relay between the nodes

[2] In our implementation of CARP used in the trials at sea the PONG packet is transmitted after a time selected randomly and uniformly in an interval whose range depends on the maximum propagation delay and on the transmission delay. This is for reducing the probability of collisions of PONG packets. In both simulations and trials at sea, the occurrence of PONG packet collisions was observed to be negligible, which is consistent with the distances between nodes in the selected network topologies, and with the small size of PONG packets (7 bytes).

Algorithm 2 UponReceivingPong(PONG$\langle src, dst, pid, HC, queue, energy, lq, bit_mask_{L_{pkt}}, bit_mask_{JR}\rangle$**) from neighbors** y

for all received $\texttt{PONG}_\texttt{y}$ **do**
    Remove packets from L_{pkt} according to $bit\_mask_{L_{pkt_y}}$
if $L_{pkt} \neq \emptyset$ **then**
    relay ← chooseBestRelay() according to lq_y, $energy_y$ and $queue_y$
    Remove packets from L_{pkt} according to $bit\_mask_{JR_{\text{relay}}}$
if $L_{pkt} \neq \emptyset$ **then**
    send L_{pkt} DATA packet(s) to relay

that replied with a PONG packet. Packets already received by the selected relay (recognized by the bits set to 1 in the bit mask bit_mask_{JR} carried by the PONG) are removed from x queue. After this update, node x transmits to the selected relay as many packets as the relay can get (as indicated in the field *queue* carried by the PONG), if any (i.e., if $L_{pkt} \neq \emptyset$). When selecting the best relay node, for each neighbor y replying with a PONG node x computes that neighbor goodness considering the quality $lq_{x,y}$ of the link from x to y, and the quality of the best link among those from y to its neighbors z, as follows:

$$goodness_y = lq_{x,y} lq_{y,z}.$$

The node y with the highest $goodness_y + \frac{1}{HC(y)}$ is chosen as relay. Nodes with a lower hop distance or higher goodness from the sink are preferred. If there are ties, priority is given to the node with the highest energy, and then to the node with the higher available buffer space. Further ties are broken by using the node unique identifiers. Upon receiving a train of data packets, a node y replies with a cumulative ACK to x (unicast), acknowledging each single packet in the train that has been received correctly. In particular: ACK $= \langle src, dst, pid, HC, bit\_mask_{L_{pkt}}\rangle$, where *src* and *dst* are the IDs of the ACK sender (y) and of its destination (x), *pid* is the ID of the PING/PONG handshake, HC is the hop count of the sender, and the bit mask $bit\_mask_{L_{pkt}}$ encodes the packets that are being acknowledged. Once an acknowledgment from y is received, node x updates its hop count to reflect possible changes in the network topology (see details below). When node y has received a train of one or more data packets, it checks whether it has received them previously in order to retransmit only those that have not been sent before. In order to increase robustness, the sink always broadcasts its acknowledgments (instead of using unicast communications as the other nodes). As a consequence, the neighbors of the sink that are aware of the packets successfully delivered can inform the other nodes by piggybacking this information in the field $bit\_mask_{L_{pkt}}$ of their PONG packets.

Computing the link quality lq. Assessing the link quality lq is the fundamental step for computing the *goodness* of the neighbors of a sender x seeking for a relay. In particular, the $goodness_y$ of a neighbor y of x aims at representing an estimate of the quality of the channel from x to y ($lq_{x,y}$) and from y to its best reachable neighbor z in a route to the sink ($lq_{y,z}$). The quality $lq_{y,z}$ of the links outgoing from y is computed by y based on the success of past transmissions to its neighbors z. It is defined as an exponential moving average, where the weight of transmissions back in the past is less influential than that of recent ones in assessing the goodness of the link for transmissions. This enables CARP to take into account the time varying nature of the channel, giving more importance to recent performance. More formally, we consider two kinds of transmissions on the channel: 1) The PING/PONG handshake, and 2) the transmission of a (train of) data packet(s) and the corresponding ACK. For each transmission t of either kind from node y to one of its neighbors z, node y computes:

$$lq_{y,z}^{t} = \alpha Y_{y,z}^{t} + (1 - \alpha) lq_{y,z}^{t-1}. \tag{1}$$

The coefficient $\alpha \in (0, 1)$ is the constant smoothing factor through which we can control how quickly the influence of older transmissions decreases. For instance, a higher α could be used for highly variable underwater channels, as it discounts older transmissions faster. The term $Y_{y,z}^{t}$ is the success ratio of the tth transmission from y to z. The term $lq_{y,z}^{t-1}$ is the value of the moving average after $t - 1$ transmissions from y to z. Since the definition of Equation (1) is recursive, we define $lq_{y,z}^{1}$ as the success ratio of the first transmission. The value lq that node y transmits in its PONG packet to x is the best among the $lq_{y,z}$ to all its neighbors z that could further relay the packet towards the sink, based on the (different) transmissions with each of them. The value $lq_{x,y}$ used by x for computing $goodness_y$ is determined similarly, via transmissions from x to y.

Updating the hop count. Given the extreme variability of the underwater channel, determining the hop count is made more dependent on the channel quality than on the actual (topology-based) hop distance from the sink. Therefore, node x continuously monitors the links to its neighbors and sets its hop count to a value that depends on the quality of data transmission to them. In order to implement this mechanism, each node maintains a data structure storing L_{ratio_i} for each hop count i, where $1 \leq i \leq maxHC$, being $maxHC$ the maximum possible hop distance of a node from the sink. The L_{ratio_i} is the value of the exponential moving average of the ratio between the number of data packets acknowledged to x by each neighbor with hop count

i and the number of packets sent by x to those nodes. A threshold is set that "filters out" all those links that are not deemed fit for successful transmissions. Finally, x sets its hop count $HC(x)$ to the smallest value i such that L_{ratio_i} is above a threshold.

Handling link asymmetry. Channel dynamics can result in links that are temporarily unidirectional. These links can affect the performance of CARP depending on whether the unidirectional link is to the sink, or to a neighbor. *Unidirectional links to the sink*. Suppose that a neighbor x of the sink sends a PING to initiate relay selection. This PING is received by all its neighbors, including the sink. Due the unidirectional link to the sink, x cannot receive the PONG from the sink. The next hop relay will have to be another node y that will be the recipient of the packet. The sink, however, receives the data packet correctly, even if the packet is not directed to it. In order to keep forwarding the packet(s), node y will broadcast a PING with the *id*(s) of the packet(s) it wants to transmit. At this point, the sink is able to identify that it has already received that data packets (from x), and sets the $bit\_mask_{L_{pkt}}$ field of the PONG to y accordingly. As a consequence, node y deletes the packet from its queue, avoiding its retransmission. *Neighbor asymmetry*. In order to avoid unnecessary retransmissions of packets successfully received but whose ACK went lost, if a node receives a PING referring to already received data packets, it warns the sender by using the bit_mask_{JR} of its PONG appropriately. The bits corresponding to the packets that have been correctly received are set to one, which allows the sender to drop them.

CARP and power control. Protocols such as CARP determine the next hop relay based on the correct exchange of control packets. Once a neighbor has been selected as relay, the channel is reserved and used for data transmission. However, data packets are usually considerably longer than control packets. Therefore, what is considered an acceptable PER for control packets might result in a (too) high PER for data packets. CARP takes advantage of modem power control, if available, to obtain similar desirable PER for both control and data packets. The power used to transmit PING and PONG packets is computed so to obtain the desired PER corresponding to a given channel bit error rate (BER). Once a relay has been selected, the power for sending data packets is increased so that the corresponding PER matches that of the PING/PONG handshake. More precisely, the transmission power P for transmitting packets at a given PER is computed as follows. Assuming a BPSK modulation and no forward error correction, the probability to transmit correctly a packet with l bits is equal to $(1 - BER)^l$. The BER is computed as $\frac{1}{2}\text{erfc}(\sqrt{SNR})$, where *erfc()* is the complementary error function. The SNR is given by $\frac{P/A(r,f)}{N(f)\Delta f}$, where P is the required transmission power, $A(r, f)$ is the attenuation in the underwater channel over a distance r for a signal of frequency f, $N(f)$ is the noise power spectral density, and Δf is the receiver noise bandwidth [26].

4. Performance Evaluation

The performance of CARP has been evaluated through an extensive set of simulations as well as with experiments at sea in two very different scenarios, namely, a fjord in Norway and in the Mediterranean coast of Italy. Simulation scenarios consider multi-hop UWSNs with desirable size (20 nodes, not currently available at sea). We compare the performance of our protocol with that of FBR and of an enhanced version of the common flooding protocol, termed EFlood. In the two sets of experiments at sea we have compared CARP and EFlood in networks with up to 8 nodes.

4.1. Investigated metrics

Effectiveness and costs of delivering bits to the sink are assessed by investigating the following metrics.
• *Packet delivery ratio* (PDR) at the sink, defined as the ratio between the packets correctly received by the sink and the packets generated by the nodes.
• *End-to-end latency per meter*, defined as the time between the packet generation and the time of its correct delivery at the sink, divided by the distance between source and destination. Normalization by distance is used so as to unify the performance over varying deployment areas.
• *Energy per bit*, i.e., the energy consumed by the network to correctly deliver a bit of data to the sink.

4.2. Simulations experiments

This section describes the simulation-based comparative performance evaluation of CARP and two other proposed protocols for UWSN routing, namely, FBR [4] and EFlood, an enhanced version of the common flooding protocol. The performance of common flooding, where a node immediately retransmits a received packet (unless it is a duplicate), is enhanced by EFlood by letting a node wait for a random time before forwarding the packet. This random time depends on the network topology and on the packet transmission delay. Through extensive trials (both simulation-based and in-field) we have observed that EFlood remarkably outperforms common flooding. For instance, the packet delivery ratio increases from 5% to 60% in network with 20 nodes and varying traffic (simulations) and from 58% to 80% in network with 8 nodes (in-field experiments). CARP, FBR and EFlood have been implemented in SUNSET [5], our extension of ns-2 [27] for underwater networking, connected to the Bellhop software [6] via the WOSS interface [28].

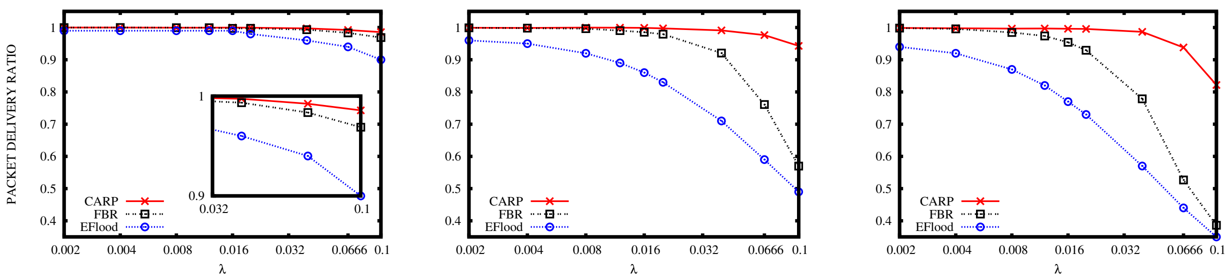
Bellhop is used to compute acoustic path loss at a given location, as well as the spatially-varying interference induced by node transmissions. The environmental data refer to an area located in the Norwegian fjord off the coast of Trondheim, with the coordinate $(0, 0, 0)$ of the surface located at $63°, 29', 1.0752''N$ and $10°, 32', 46.6728''E$. Sound speed profiles (SSP), bathymetry profiles and information on the type of bottom sediments of the selected area are obtained from the World Ocean Database [29], from the General Bathymetric Chart of the Oceans (GEBCO) [30] and from the National Geophysical Data Center Deck41 data-base [31], respectively. The SSP is retrieved by WOSS from the World Ocean Database (average of measurements from September 2009).

Simulation scenarios and settings. Our simulations concern UWSNs with 20 nodes (19 nodes plus the sink) statically placed in a region with surface 1km $\times$ 2km. Sensor nodes are placed randomly and uniformly in the region at different depths, ranging from 10 to 240m. Each packet that makes it to the sink travels an average of 1.8 hops (the maximum number of hops is 4). The data packet payload size (in bytes) varies in the set $\{100, 512, 1000\}$. The carrier frequency is 25.6kHz for a bandwidth of 4000Hz. Bandwidth efficiency is set to 1bps/Hz. Since we selected a modem transmission rate of 4000bps, the transmission time of the three packet sizes of 100B, 512B and 1000B is smaller than, comparable to, and bigger than maximum propagation delay, respectively.[3] We assume BPSK modulation. Topology construction ensures that each node has at least one route to the sink going through robust links (with respect to SNR). 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. Reception and transmission power were estimated based on the energy consumption of existing acoustic modems. In order to have similar PERs for control packets and data packets, CARP control packets are transmitted with a transmission power of 1.5W. The FBR protocol uses 3 levels of transmission power corresponding to 2.8W, 8W, and 35W. Traffic is generated according to a Poisson process with aggregate (network-wide) rate of λ packets per second, where λ takes values in the set $\{0.002, 0.004, 0.008, 0.012, 0.016, 0.02, 0.04, 0.0666, 0.1\}$. Once a packet is generated, it is associated with a source selected randomly among all nodes (except the sink). The destination of all packets is the sink. The total size of a data packet is given by the selected payloads plus the headers added by the different layers. The physical layer header contains all the information needed by the modem to correctly start receiving a packet (synchronization preamble, delimiters, etc.). At the physical layer, nodes need about 10ms for synchronization (the physical header overhead changes according to the data rate). EFlood uses CSMA

[3] The maximum propagation delay is the time it takes a bit to travel a distance equals to the modem nominal transmission range.

MAC protocol without ACKs. The CSMA header contains the sender and the destination addresses, and the packet type. Its length is 3B. The RTS and CTS packets used by FBR are 6B long; its ACK and WARNING packets are 3B long. The size of PING and PONG packets is 11B and 7B, respectively; the ACK packets used by CARP are 6B long, while its HELLO packets are 6B long. The CARP MAC header is 4B long. Finally, the smoothing factor α used for computing CARP link quality has been set to 0.9.

Simulation results: Packet delivery ratio. Figure 1 shows the packet delivery ratio of the three protocols for increasing traffic λ and for three different packet payloads.

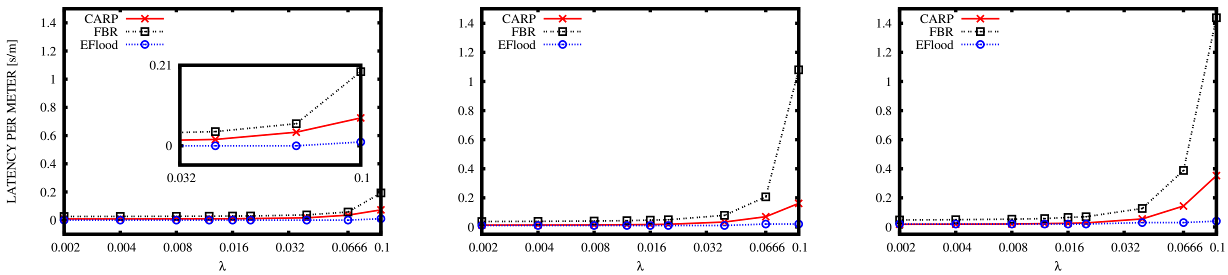


(a) Payload of 100B. (b) Payload of 512B. (c) Payload of 1000B.

Figure 1: Packet delivery ratio.

As expected, for increasing traffic and packet sizes the packet delivery ratio decreases. Higher traffic increases the probability of packet collisions and retransmissions and, as a consequence, the probability of finding the channel busy. Moreover, when the packet size increases the probability of collision becomes higher because of the decreasing ratio between the transmission and propagation delays. CARP clearly outperforms FBR and EFlood because of its link quality-based relay selection and because data are forwarded on links that are robust for both control and data packets. This substantially reduces the number of collisions, with the consequence that a higher number of data packets reaches the sink successfully irrespective of the length of the routes they travel. The packet delivery ratio of FBR drops faster than that of CARP for increasing traffic and packet sizes. This is because 1) FBR does not transmit trains of data packets to reduce the handshaking overhead; 2) FBR uses the same power level for both control and data packets, and 3) FBR selects relays only if they provide positive advancement towards the sink, and as such suffers from the problem of not being able to route around connectivity holes. EFlood shows the worst performance, suffering from congestion and lack of retransmissions.

Simulation results: End-to-end latency per meter. The average latency per meter experienced by data packets successfully delivered to the sink is shown in Figure 2.



(a) Payload of 100B.

(b) Payload of 512B.

(c) Payload of 1000B.

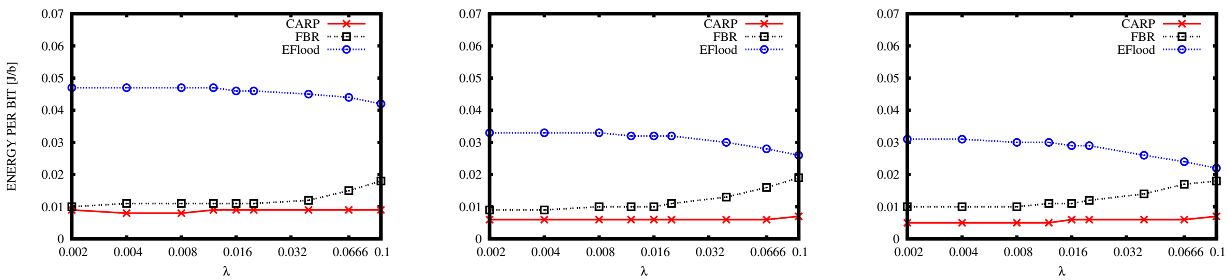
Figure 2: Data packet latency per meter.

Increasing the traffic induces degraded performance, as expected. This happens even if the average number of hops traveled by each packet stays relatively stable. Each hop imposes longer delays because of the increased number of collisions and retransmissions. EFlood shows the best latency per meter performance because of its ability of delivering packets on multiple routes and the absence of the handshake phase. CARP outperforms FBR because of its link quality-based relay selection and of the use of robust links for control and data transmissions, which keep retransmissions at bay. Moreover, CARP transmits trains of data packets whenever possible, thus reducing the handshake overhead of the relay selection phase. The performance of FBR is beset by the many times a node has to increase its power to find a suitable relay. We observe that, however, increasing transmission power allows nodes to select farther relays, i.e., nodes that provide higher advancements toward to the sink. This explains why the average lengths of FBR routes decreases as the number of power increments increases. In particular, the average length of FBR routes is up to 16% (21%) shorter than that of CARP (EFlood).

Simulation results: Energy per bit. Our final set of simulations concerns the energy spent for delivering one bit of data to the sink correctly. Results are shown in Figure 3.

We consider the energy spent for transmitting and receiving a bit of data in networks with nodes equipped with “wake up modems,” where idling effects are almost negligible.[4] We observe that the energy consumed per bit decreases for increasing packet sizes. This is because, despite the lower packet

4 Since idling is a major culprit of energy expenditure, especially at low traffic, recent commercial transducers feature “wake up” capabilities. Modems are now built with very low power devices capable of alerting a node of upcoming transmissions. Considerable results are being seen for terrestrial radio nodes [32, 33], and similar research is ongoing for underwater modems. For instance, Teledyne Benthos [34] and Evologics [35] modems feature low power wake up.



(a) Payload of 100B.

(b) Payload of 512B.

(c) Payload of 1000B.

Figure 3: Energy per bit.

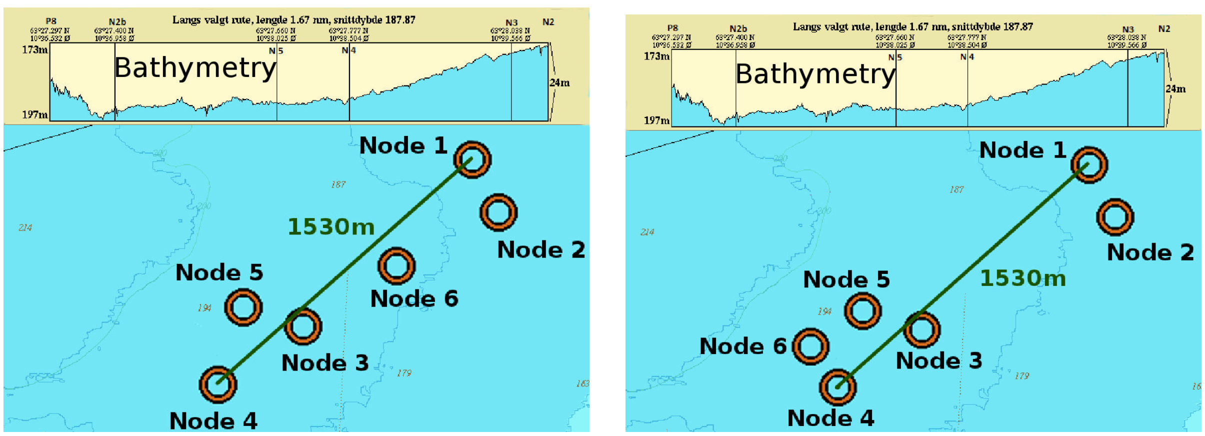
delivery ratio of larger packets, the total number of bits delivered to the sink is still higher than when a greater number of smaller packets is delivered. CARP saves more energy per bit than FBR and EFlood. Its energy per bit remains stable, independently of increasing traffic. This is due to the fact that CARP uses a lower power level for control packet transmissions. Furthermore, CARP delivers more data to the sink, and reduces the number of packet collisions and retransmissions. FBR energy per bit increases with traffic because nodes experience a higher number of retransmissions, each time with a higher transmission power. The energy per data bit of EFlood decreases with increasing traffic. This is due to the lower number of data packets correctly received and broadcast by each node (remember that EFlood does not perform retransmissions).

4.3. Experiments at sea

We have evaluated the performance of CARP through experiments at sea at two different geographic locations with diverse underwater characteristics. Our results refer to two campaigns of experiments performed in the Norwegian fjord of Trondheim in May 2013 and off the coast of the Palmaria island (La Spezia, Italy) in September 2013. In both locations we used modems that do not feature power control with the fine granularity required by CARP. Therefore, the experiments were all performed using the same transmission power level for control and data packets. In the campaigns of experiments at sea we have also collected information on the length of the routes traveled by data packets from their source node to the sink. What reported in the tables below is the average length of the routes of all packets successfully delivered to the sink.

4.3.1. Trondheim experiments

For this set of experiments we deployed six Kongsberg cNODE acoustic modems [36] to mimic a real application scenario of pipeline and oil well monitoring. Four nodes were placed at depths varying from 170 to 190 meters along the pipeline and in the oil well area (nodes with ID 2, 3, 4 and 5). One was connected to a surface station (buoy) at a depth of about 10 meters and acted as a gateway to the control station on shore (node 1, the sink). One modem was deployed on the side of the boat *Gunnerus* at a depth of 15 meters (node 6) acting as mobile node for monitoring along the pipeline (Scenario 1, Figure 4a) and in the oil well area (Scenario 2, Figure 4b). Data were generated at the nodes by a CO_2 Probe performing CO_2 and temperature monitoring [37]. They were delivered to the sink in real time and forwarded to the operator on shore. In



(a) Scenario 1.

(b) Scenario 2.

Figure 4: Network topologies for pipeline and oil well monitoring.

both scenarios, nodes with ID 4, 5 and 6 were selected as data sources. Traffic was generated according to a constant bit rate (CBR) process. In particular, each data source generated one packet every 60s. All other nodes were used as relays. Among the transmission formats (packet size, bit rate and modulation) provided by the Kongsberg nodes, we selected the best fit for the harsh fjord channel. This format supports a maximal packet size of 48B and a bit rate of 200bps. In fact, in order to reduce PER and transmission delays, we used a packet size of 24B. Finally, we selected the lowest transmission power level provided by the cNODE, namely, 173dB re $1\mu Pa$, which corresponds to 3.3W.

Multi-hop communications were obtained by implementing "node blacklisting" in software. In particular, we configured nodes 1 and 6 (labeled BL in the following) so that they drop packets received from each other. This made node 6 a critical source node, which communicates data to the sink through a route at least two hops long. The two test scenarios differ in the relative position of node 6, which impacts link quality. In Scenario 1, the node on the Gunnerus was anchored along the pipeline at a central position 655m away from the sink. In Scenario 2, node 6 was anchored to the oil well, 1500m away from the sink. Scenario 1 lasted 1240s during which 57 packets were generated. Scenario 2 was run for 1330s, with 66 packets. Results of our experiments are shown in Table 1.

Metric	Results
Packet delivery ratio [%]	82
Latency per meter [s/m]	0.08
Route length [hops]	1.3
Energy per bit [J/b]	0.087

(a) Scenario 1.

Metric	Results
Packet delivery ratio [%]	89
Latency per meter [s/m]	0.04
Route length [hops]	1.32
Energy per bit [J/b]	0.088

(b) Scenario 2.

Table 1: Trondheim tests.

Packet delivery ratio. CARP always achieves an average packet delivery ratio greater than 80%. This is because of its link quality-based relay selection and of its feature that allows to transmit trains of multiple packets within a single handshake. We have also observed that the CARP mechanism for handling unidirectional links prevented unnecessary retransmissions of both control and data packets, enhancing packet delivery ratio because of lower network congestion.

End-to-end latency per meter. Packets travel an average of 1.3 hops to get to the sink. Despite the limited route length, packet latency can however be high.[5] The average time it takes a packet to reach the sink exceeds 80s in the first scenario, resulting in a latency per meter of 0.08m/s. In the second scenario packets travel faster, averaging 55s to reach the sink (with a latency per meter of 0.04m/s). The reason is to be found in the poor link quality of the fjord channel, where links are asymmetric and unstable, significantly varying in time and space. This is clearly shown in Table 2 that reports the percentage of `PING` packets successfully delivered per link. Many links are low quality. This is because, in the harsh channel condition typical of these scenarios, handshake-based protocols are more prone to suffer from a quickly varying links. In order to complete the handshake, these protocols need links that are stable and bidirectional. In this case, several

[5] Timeout settings for CARP were conservative, since this was the first CARP test at sea with a quite large number of devices. These settings have been more finely tuned for the CommsNet13 experiments thanks to the lessons learned in Trondheim.

	1	2	3	4	5	6
1	-	75%	75%	38%	77%	BL
2	53%	-	0%	40%	0%	53%
3	100%	28%	-	0%	85%	100%
4	53%	39%	0%	-	65%	34%
5	63%	2%	68%	61%	-	66%
6	BL	63%	61%	55%	52%	-

(a) Scenario 1.

	1	2	3	4	5	6
1	-	40%	42%	50%	67%	BL
2	100%	-	0%	83%	0%	0%
3	66%	11%	-	0%	72%	77%
4	52%	52%	0%	-	54%	67%
5	82%	23%	76%	54%	-	61%
6	BL	48%	71%	71%	71%	-

(b) Scenario 2.

Table 2: Link quality. Entry (i, j) is the percentage of `PING`s successfully transmitted from i to j.

control packets were sent for completing one data transmission successful, which impacted performance, and especially latency, detrimentally. In particular, the average number of `PING` packets needed for one successful data transmission in the first scenario is 3.3. This value drops to 2.7 in the second scenario, and explains the lower average latency per meter in that case.

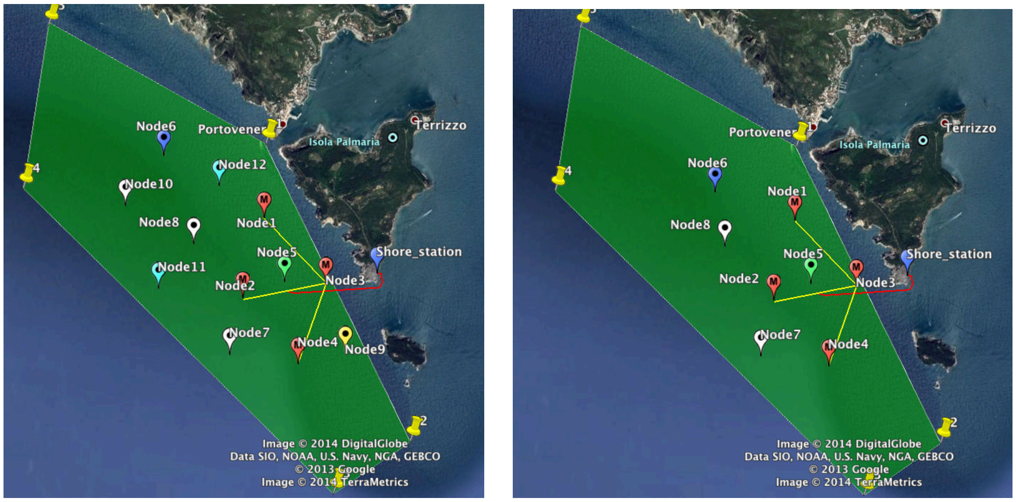
Energy per bit. We computed the energy per bit using the values for energy transmission and reception from the Kongsberg cNode data sheet. In particular, the power needed to transmit a bit is 3.3W and that for receiving a bit is 0.620W. CARP requires low energy per bit. This is because each successful handshake results in the transmission of a train of multiple data packets, which increases channel utilization. Furthermore, the mechanism used by CARP to handle unidirectional links significantly decreases unnecessary retransmissions, which has a beneficial effect on energy consumption.

4.3.2. CommsNet 13

The CommsNet13 experiments have been conducted from September 9 to September 22, 2013 off the coast of the Palmaria island, in the South-West part of the gulf of La Spezia, Italy. The experiments were performed through a collaboration agreement between the University of Roma "La Sapienza" and the NATO Science and Technology Organization Centre for Maritime Research and Experimentation (STO CMRE). Twelve nodes, static and mobile, were deployed as shown in Figure 5a.

Configuration was as follows.

- **LOON.** The Littoral Ocean Observatory Network, consisting of 4 bottom-mounted tripods with acoustic modems. The four nodes were cabled to shore and connectivity and power were provided by



(a) CommsNet13 area and nodes.

(b) Network configuration used.

Figure 5: Setting of the CommsNet13 experiments.

a fiber optic cable (nodes with IDs 1 through 4). A PC on the shore station was used to control the modem operations.

- **Gateway.** One gateway communication buoy, temporarily moored with recoverable mooring tackle (node 5).
- **Alliance.** The NATO Research Vessel Alliance working as a static node with a Manta system [38] connected to a modem deployed on the side (node 6).
- **WaveGlider.** One WaveGlider autonomous surface vehicle (named Carol) with gateway communication payload (node 9).
- **eFolaga.** Three eFolagas [39] Autonomous Underwater Vehicles (AUVs) with communication payload (IGEPv2 board). The eFolagas were used as static nodes, moored for overnight tests (nodes 7, 8 and 10).
- **Manta.** We used Manta portable systems [38] for additional modems, deploying them from Rigid-Hulled Inflatable Boats (RHIBs) to increase the scale of the network temporarily (nodes 11 and 12).

All nodes were equipped with Evologics S2CR 18/34 acoustic modems. The site where the underwater network was deployed is a broad plateau approximately 30m under sea surface. It is located along the major current direction between potential sources of pollution to the South and the marine protected areas to the North. Therefore, the position of the network is ideal for environmental monitoring applications. These applications may include sensing for measuring ship traffic, especially in restricted areas, assessing the presence and amount of chlorophyll and blue-green algae (indicators of eutrophication and toxic algae blooms), conductivity and temperature (to investigate water stratification and movements), dissolved O_2 (for water quality), and water clarity (for suspended sediments and algae formation). The seabed in this area is predominantly mud.

The performance of CARP and EFlood have been compared considering two different setups (referred below as Experiment 1 and Experiment 2) where 8 nodes were statically deployed as depicted in Figure 5b. Node 1 was designated as the sink, while nodes 3, 4 and 8 were used as data sources. The remaining nodes acted as relays to the sink. Traffic generation is CBR. In particular, in Experiment 1 a packet was generated by each source node every 60s. In Experiment 2 we considered a higher traffic, where a packet was generated every 30s by each data source. The duration of Experiment 1 and 2 was 2000s and 1300s, respectively. The packet size was 50B. We used the maximum bit rate offered by the Evologics S2CR 18/34 acoustic modem, namely, 480bps, which allows us to bypass the in-house MAC mechanism and to use those implemented by CARP and EFlood.[6] All experiments were performed at the lowest transmission power level, resulting in a power consumption for transmissions of 2.8W. Power consumption for reception was 0.5W.

Results are shown in Table 3.

Metric	CARP	EFlood
Packet delivery ratio [%]	96	96
Latency per meter [s/m]	0.038	0.009
Route length [hops]	1.35	1.6
Energy per bit [J/b]	0.03	0.049

(a) Experiment 1.

Metric	CARP	EFlood
Packet delivery ratio [%]	95	87
Latency per meter [s/m]	0.054	0.01
Route length [hops]	1.36	1.4
Energy per bit [J/b]	0.027	0.040

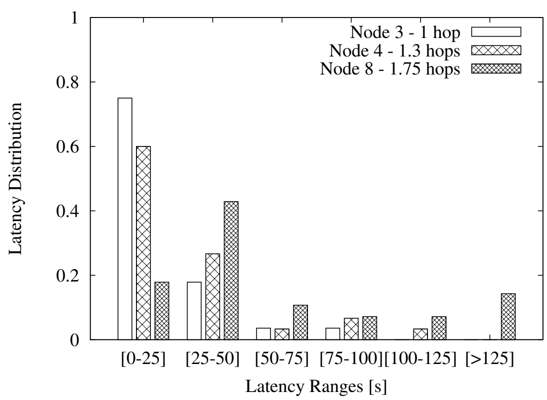
(b) Experiment 2.

Table 3: CommsNet13 experiment results.

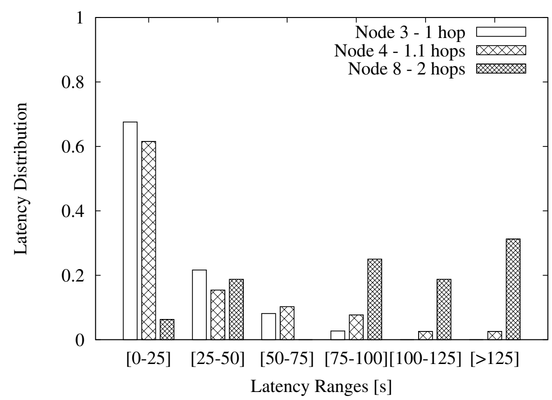
[6] This is the fixed bit rate provided by the modem when transmitting "Instant Messages," i.e., the Evologics modem form for direct communication through which we transmitted the control and data packets of our protocols.

Packet delivery ratio. When the traffic is low, CARP and EFlood show the same PDR. When traffic increases reserving the channel becomes more challenging and both protocols show lower PDR and longer latencies. This is mainly due to the larger number of packet collisions and to the fact that the channel is found busy more often. Moreover, during Experiment 2 the network became temporary disconnected several times during the tests. In particular, we have estimated that the network disconnections prevented node 8 to correctly deliver 48% (29%) of the data packets for CARP (EFlood) at the first transmission attempt. These temporary disconnections are the main reason of the lower PDR obtained by EFlood with respect to that of CARP. EFlood does not implement any notification mechanism to check if a packet has been correctly delivered. All the packets transmitted by node 8 during the disconnection periods are therefore lost. CARP instead, thanks to the use of handshaking and `ACK`s, performs retransmissions, thus increasing the probability of delivering a packet.

End-to-end latency per meter. EFlood incurs a latency per meter lower than that of CARP. In particular,



(a) Experiment 1.



(b) Experiment 2.

Figure 6: CARP: End-to-end latency distribution for traffic sources.

in Experiment 1 the average time it takes to deliver a packet to the sink is 9s for EFlood and 37s for CARP, resulting in a latency per meter of 0.009m/s and 0.038m/s, respectively. Similarly, in Experiment 2 the packet delivery time is an average of 10s (0.01m/s) for EFlood and of 50s (0.054m/s) for CARP. EFlood incurs very low latencies per meter because it does not use control packets for channel access. It just experiences a limited delay due to the random time that a node waits before forwarding a packet. CARP, instead, pays the price of using handshaking and `ACK`s. Figure 6 shows the distribution of the latency of CARP packets from the three source nodes in the two experiments. We observe a higher latency for packets from node 8 in Experiment 2 than in Experiment 1. This is due to the frequent disconnections of node 8. We observe that without considering packets from this node, CARP would show similar latency values in both experiments.

Energy per bit. CARP consumes less energy per delivered data bit than EFlood. In particular, the energy per bit of EFlood is 63% (48%) higher than that of CARP in Experiment 1 (Experiment 2). This is because of the flooding nature of EFlood where packets are frequently duplicated in the network. This increases interference, retransmissions, and redundant delivery, all of which require more energy. CARP, instead, is a single-path protocol that selects relays based on link quality and hop count, which limits data retransmissions. Moreover, CARP reduces energy per bit thanks to: 1) The use of trains of packets that limits the number of handshakes, and 2) the use of an handshaking mechanism enhanced to handle unidirectional links and to decrease retransmissions of data packets in case of `ACK` losses. We observe that the energy per bit of Experiment 2 is lower than that of Experiment 1 for both CARP and EFlood (Table 3). This is once again due to the temporary disconnections of node 8. During these times, nodes do not receive and forwards packets from this node. As a consequence, the energy per bit is lower. This is more pronounced for EFlood than for CARP. When using EFlood 29% of the packet generated by node 8 are not forwarded by any of the nodes in the network. In the case of CARP, when node 8 is disconnected it sends a large number of `PING` packets in the (inane) attempt of finding a relay. We notice, however, that during the disconnections node 8 accumulates the packets it generates. When the link comes back up and a relay is finally found, trains of data packets are transmitted back to back, which is a source of energy saving. In other word, the energy spent for a larger number of `PING` transmissions is offset by the savings of longer trains of packets. (The number of trains with more than one packet in Experiment 2 was observed to be 50% higher than that of Experiment 1. Furthermore, the average number of packets per train in the second experiment is 40% higher than that in the first one.)

5. Conclusions

In this paper we presented CARP, an efficient new routing protocol for UWSNs. CARP is designed according to the cross-layer design paradigm, in that it efficiently uses short control messages to perform joint channel access and relay selection. In particular, data forwarding exploits link quality information, leading to the selection of nodes that exhibit a history of successful transmissions towards the sink. CARP also implements simple mechanisms to effectively avoid loops, route around connectivity voids and shadow zones.

It also takes advantage of modem power control, if available, for selecting robust links. The effectiveness of CARP in delivering packets in reasonable time and with low energy demands is shown by simulations in networks with desirable size and through experiments at sea on real multi-hop UWSNs. Different packet sizes and ratios between transmission and propagation delays have been considered in our performance evaluation. Our results show that CARP robust mechanism for relay selection doubles the packet delivery ratio of two other routing protocols, namely, FBR and EFlood. CARP also outperforms FBR and EFlood in terms of energy consumption, and delivers packets significantly faster than FBR.

Acknowledgments

This work was supported in part by the EU FP 7 ICT project SUNRISE "Sensing, monitoring and actuating on the UNderwater world through a federated Research InfraStructure Extending the Future Internet." We gratefully acknowledge the NATO STO CMRE for their invaluable feedback and for providing expertise and logistics during experimentation.

References

[1] N. Meratnia, P. J. M. Havinga, P. Casari, C. Petrioli, K. Grythe, T. Husoy, M. Zorzi, CLAM—Collaborative embedded networks for submarine surveillance: An overview, in: Proceedings of IEEE/OES OCEANS 2011, Santander, Spain, 2011, pp. 1–4.

[2] J. Heidemann, M. Stojanovic, M. Zorzi, Underwater sensor networks: Applications, advances and challenges, Philosophical Transactions of the Royal Society A 370 (2012) 158–175.

[3] R. Jurdak, A. G. Ruzzelli, G. M. P. O'Hare, Design considerations for deploying underwater sensor networks, in: Proceedings of SensorComm 2007, Valencia, Spain, 2007, pp. 227–232.

[4] J. M. Jornet, M. Stojanovic, M. Zorzi, Focused beam routing protocol for underwater acoustic networks, in: Proceedings of ACM WUWNet 2008, San Francisco, CA, 2008, pp. 75–82.

[5] C. Petrioli, R. Petroccia, D. Spaccini, SUNSET version 2.0: Enhanced framework for simulation, emulation and real-life testing of underwater wireless sensor networks, in: Proceedings of ACM WUWNet 2013, Kaohsiung, Taiwan, 2013, pp. 1–8.

[6] M. B. Porter, The BELLHOP manual and user's guide: Preliminary draft, heat, Light, and Sound Research, Inc. (2011).

[7] P. Casari, M. Zorzi, Protocol design issues in underwater acoustic networks, Elsevier Computer Communications 34 (17) (2011) 2013–2025.

[8] T. Melodia, H. Khulandjian, L.-C. Kuo, E. Demirors, Advances in underwater acoustic networking, in: S. Basagni, M. Conti, S. Giordano, I. Stojmenovic (Eds.), Mobile Ad Hoc Networking: Cutting Edge Directions, John Wiley & Sons, Inc., Hoboken, NJ, 2013, Ch. 23, pp. 804–852.

[9] A. Wahid, K. Dongkyun, Analyzing routing protocols for underwater wireless sensor networks, International Journal of Communication Networks and Information Security 2 (3) (2010) 253–261.

[10] M. Ayaz, I. Baig, A. Abdullah, I. Faye, Review: A survey on routing techniques in underwater wireless sensor networks, Journal of Network and Computer Applications 34 (2011) 1908–1927.

[11] P. Xie, J. H. Cui, L. Lao, VBF: Vector-based forwarding protocol for underwater sensor networks, in: Proceedings of Networking, Coimbra, Portugal, 2006, pp. 178–181.

[12] N. Nicolaou, A. See, X. Peng, J.-H. Cui, D. Maggiorini, Improving the robustness of address-based routing for underwater sensor networks, in: Proceedings of MTS/IEEE OCEANS 2007, Aberdeen, Scotland, 2007, pp. 1–6.

[13] 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.

[14] D. Shin, D. Hwang, D. Kim, DFR: An efficient directional flooding-based routing protocol in underwater sensor networks, Wireless Communications and Mobile Computing 12 (17) (2012) 1517–1527.

[15] B. Jalving, Depth accuracy in seabed mapping with underwater vehicles, in: Proceedings of MTS/IEEE OCEANS 1999, Seattle, WA, 1999, pp. 973–978.

[16] H. Yan, Z. J. Shi, J.-H. Cui, DBR: Depth-based routing for underwater sensor networks, in: Proceedings of IFIP-TC6 NETWORKING 2008, Singapore, 2008, pp. 72–86.

[17] U. Lee, P. Wang, Y. Noh, L. Vieira, M. Gerla, J.-H. Cui, Pressure routing for underwater sensor networks, in: Proceedings of IEEE INFOCOM 2010, San Diego, CA, 2010, pp. 1–9.

[18] Y. Noh, U. Lee, P. Wang, B. S. C. Choi, M. Gerla, VAPR: Void-aware pressure routing for underwater sensor networks, IEEE Transactions on Mobile Computing 12 (5) (2013) 895–908.

[19] K. Zeng, W. Lou, J. Yang, D. R. Brown, On geographic collaborative forwarding in wireless ad hoc and sensor networks, in: Proceedings of WASA 2007, Washington, DC, 2007, pp. 11–18.

[20] L.-C. Kuo, T. Melodia, Tier-based underwater acoustic routing for applications with reliability and delay constraints, in: Proceedings of the IEEE WIMAN 2011, Maui, HI, 2011, pp. 1–6.

[21] D. Johnson, D. A. Maltz, Dynamic source routing in ad hoc wireless networks, in: T. Imielinski, H. F. Korth (Eds.), Mobile Computing, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996, Ch. 5, pp. 153–181.

[22] G. Toso, R. Masiero, P. Casari, O. Kebkal, M. Komar, M. Zorzi, Field experiments for dynamic source routing: S2C EvoLogics modems run the SUN protocol using the DESERT underwater libraries, in: Proceedings of MTS/IEEE OCEANS 2012, 2012, pp. 1–10.

[23] X. Lin, N. Shroff, R. Srikant, A tutorial on cross-layer optimization in wireless networks, IEEE Journal on Selected Areas in Communications 24 (8) (2006) 1452–1463.

[24] D. Pompili, T. Melodia, I. F. Akyildiz, Routing algorithms for delay-insensitive and delay-sensitive applications in underwater sensor networks, in: Proceedings of ACM MobiCom 2006, Los Angeles, CA, 2006, pp. 298–309.

[25] D. Pompili, I. F. Akyildiz, A multimedia cross-layer protocol for underwater acoustic sensor networks, IEEE Transactions on Wireless Communications 9 (9).

[26] M. Stojanovic, On the relationship between capacity and distance in an underwater acoustic communication channel, SIGMOBILE Mobile Computing and Communication Review 11 (4) (2007) 34–43.

[27] The VINT Project, The ns Manual, http://www.isi.edu/nsnam/ns/, 2002.

[28] 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.

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

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

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

[32] L. Gu, J. A. Stankovic, Radio-triggered wake-up for wireless sensor networks, Real-Time Systems 29 (2–3) (2005) 157–182.

[33] G. Lu, D. De, M. Xu, W.-Z. Song, B. Shirazi, A wake-on sensor network, in: Proceedings of ACM SenSys 2009, Berkeley, CA, 2009, pp. 341–342.

[34] Teledyne Benthos, Teledyne Benthos undersea systems and equipment.
URL `http://www.benthos.com/index.php/product\_dashboard/acoustic\_modems`

[35] Evologics, Evologics S2C acoustic modems, http://www.evologics.de/.

[36] Kongsberg maritime, Instruction manual cNode transponder.
URL `http://www.km.kongsberg.com/ks/web/nokbg0397.nsf/AllWeb/4ADD212486A1B94EC125780000355234/`

[37] A. Annunziatellis, S. Graziani, S. Lombardi, C. Petrioli, Petroccia, CO2Net: A marine monitoring system for CO_2 leakage detection, in: Proceedings of IEEE OCEANS 2012, Yeosu, Korea, 2012, pp. 1–7.

[38] LSTS, University of Porto, Manta communications gateway, http://lsts.fe.up.pt/vehicles/manta.

[39] Graaltech, Folaga.
URL `http://www.graaltech.it/en/index.php`