Testing Network Protocols via the DESERT Underwater Framework: the CommsNet'13 Experience

Giovanni Toso*, Ivano Calabrese*, Federico Favaro*, Loris Brolo*, Paolo Casari*‡, Michele Zorzi*‡
*Department of Information Engineering, University of Padova, via Gradenigo 6/B, I-35131 Padova, Italy
‡Consorzio Ferrara Ricerche, via Saragat 1, I-44122 Ferrara, Italy
E-mail: {tosogiov, icalabre, favarofe, brololor, casarip, zorzi}@dei.unipd.it

***Abstract*—In the context of a collaboration with the NATO STO Centre for Maritime Research and Experimentation (CMRE), during the CommsNet'13 campaign we deployed a large set of experiments aimed at measuring some network statistics for three protocols for remote data retrieval in underwater networks, namely Uw-Polling (a controlled access scheme), MSUN (a source routing approach with support for mobility) and U-Fetch (a scheme based on two hierarchical levels of controlled access). The main idea behind the trial was to perform some experiments in order to get a hands-on practical experience with the protocols, evaluate their performance in a systematic way, and observe which specific features of a real world experiment can alter the performance of the protocols, compared to a computer simulation. The results obtained help understand the protocols better and can be used to refine their design and improve their performance. The experiments are run thanks to the flexibility of the DESERT Underwater framework, and managed through a newly designed acoustic remote control framework called RECORDS.**

I. Introduction and Related Work

Network simulators are increasingly accurate and complex. In recent years, several frameworks for Underwater Acoustic Networks (UANs) have been presented [1]–[5]. Despite the level of accuracy of network simulation frameworks, a real world deployment presents a large number of challenges that are difficult to predict and address accurately. In this work, we present some results obtained in the context of the CommsNet'13 campaign, carried out in collaboration with the NATO STO Centre for Maritime Research and Experimentation (CMRE). This campaign gave us the opportunity to test, in a real environment, the following three protocols: Uw-Polling [6] (a controlled access scheme), MSUN [7] (a source routing approach with support for mobility) and U-Fetch [8] (a scheme based on two levels of controlled access).

Thanks to the experience accumulated in previous sea trials and in the specific CommsNet'13 environment, we highlight how the challenges faced during a sea trial can affect the performance of the protocols tested, especially as compared to results obtained by simulation. These results can be effectively utilized to improve the protocols and adapt them to be more robust both in simulation and in real world campaigns.

Our experimentations in the field are made possible thanks to two software packages: the DESERT Underwater framework [1] and the RECORDS framework [9]. These tools, developed by our group and freely available at [10], [11], are designed in order to make the migration from computer simulations to real world trials as seamless as possible. The immediate benefit of this is that the final users have the chance to debug and evaluate the protocols before the trial and, when ready, use the same code in a real experiment, thereby minimizing the migration costs. The three protocols presented have been implemented in the DESERT Underwater framework [1] and have been tested intensively by means of simulations in [8] before the trial. In order to interface the protocols with real hardware, we exploit the capabilities of the DESERT and RECORDS frameworks. We only had to implement, for each of them, a simple layer called *packer module* where we tell the framework how to convert the fields of the protocol headers into a bit stream suitable for transmission, and vice-versa. The schedule of the experiments, the remote reconfiguration of the nodes and the possibility to check the status of the nodes were managed thanks to the features provided by the RECORDS framework.

Similar efforts devoted to frameworks that make it possible to both simulate and experiment underwater network protocols have been presented in the literature. An example can be found in [12] where the authors propose a novel system to remotely control and reconfigure an underwater acoustic sensor network named *Back-Seat Driver*. It uses the SUNSET framework [2] to remotely run network experiments and to check the status of the nodes. The system has been proved to be robust and flexible.

The Underwater Networks Project (UNET) is presented in [4]. The UNET project consists of several related components: the UnetStack framework provides a basic set of agents that allow an underwater network to be deployed; the Unet Simulator, based on UnetStack, allows Unet agents and protocols to be simulated; the Unet-2 Modem is a UnetStack-compatible software-defined modem. Its primary features are flexibility, forward error correction codes, and network protocols to be tested at sea.

Aqua-Net Mate, presented in [3], is an extension to the Aqua-Net framework that gives the possibility to switch seamlessly between simulation mode and experiment mode. The framework introduces also other features like a real-time virtual channel/modem simulator and a newly designed state machine exploited to emulate real acoustic modems. The authors prove

that their solution is effective and that their state machine used in simulation can accurately capture the behavior of real underwater acoustic modems.

The authors of [13] present an acoustic channel emulator to conduct laboratory controlled experiments. A computer, that runs the emulator, is connected to a modem through a serial port and is fed with captured acoustic signals from a modem. The captured signals are processed in order to simulate the effects of a real underwater acoustic channel and the output is finally played by an audio device and recorded by the receiver modem. The presented emulator allows developers to evaluate protocols without deploying them in a real world underwater scenario.

In [14] the authors state that research experimentation in UANs is limited by the high cost of underwater networking experiments, and lack of a single, easily-replicable platform for evaluation. In order to find a compromise, they propose Underwater Platform to Promote Experimental Research (UPPER), a low-cost and flexible underwater platform designed to enable cost-effective and repeatable experimentation. Their solution uses Commercial Off-the-Shelf components to provide a hardware/software solution to interface hydrophones with laptops that act as physical layer. The authors proved that their solution can easily integrate existing and new protocols and thanks to this validate simulation results.

In order to minimize the development costs, Evologics GmbH [15] provides a real-time emulator that aims to optimize underwater network protocol development by removing, in the early stages, the need to purchase actual modems. The proposed tool can execute the same code used by the modem without any modifications, emulates all features of the modem's data-link protocol layer, and includes a simulator of the physical protocol layer. The proposed solution offers a time-saving solution that minimizes development costs for upper layer network protocols and simplifies integration of acoustic modems into underwater infrastructure.

An emulation system that makes it possible to test network protocols for acoustic networks on real hardware without requiring actual sea trials has also been presented in [16].

The rest of the paper is organized as follows: Section II presents the DESERT framework, the three protocols tested during the sea trial, and the RECORDS framework used to manage the experiments and control the modems acoustically. Section III describes the scenario and the equipment used during the sea trial, and presents some of the real-world experiments conducted at CommsNet'13. Section IV concludes the paper.

II. Framework

A. DESERT Underwater Framework

The DESERT Underwater framework is a set of libraries based on the well known *ns2-nsMiracle* network simulator. DESERT was created with the goal to support the design and implementation of underwater network protocols both in simulation and in real-world trials. It provides several protocols spanning all layers of the ISO/OSI stack as well as libraries that make it possible for the final user to interface the protocols with real hardware in a straightforward way. This enables the reuse of the same code for both simulations and real field experiments. The first release of DESERT Underwater dates back to 2012, and was presented in [1]. A second version, released in 2014, is introduced in [17]. The change-log of the second version is wide, and the major updates include: a better interface to the modems, improved and completely automated installation procedures, better compatibility with embedded systems, and a more detailed documentation. The three protocols tested in this sea trial (summarized in Section II-B), have been implemented as libraries within the DESERT framework. For each protocol, we additionally implemented a so-called *packer module*, i.e., a special library that contains all instructions to compress and convert the header of a specific protocol into a bit stream, and vice-versa. All packers are interfaced to the *Adaptation Layer* (AL) module of DESERT [17], which manages such conversions and fragments/de-fragments the packets when needed. When all the packers of a specific stack are combined together, in the transmission phase the AL is able to serialize the headers, compress them and create a packet suitable for the modem; conversely, during the reception phase the AL is able to decode and recreate the original packet.

B. Protocols

In this section, we shortly summarize the protocols tested during the CommsNet'13 campaign. The interested reader is referred to the cited papers (and the references therein) for additional details.

Uw-Polling is a Medium-Access-Control (MAC) protocol that uses the polling paradigm to retrieve data packets from a network of nodes usually placed on the seafloor [6]. The retrieval is typically administered by the sink, which is also the only mobile node in the network. Uw-Polling works in three phases: i) neighbor discovery; ii) retrieval of a summary of available data from the discovered nodes; iii) sequential polling of the nodes according to a given priority criterion. In general, Uw-Polling privileges more recent data, hence the nodes upload their packets to the sink by managing their data queues in a Last-In-First-Out fashion.

The **Multi-sink Source routing protocol for Underwater Networks** (MSUN) [7] is a routing protocol partly inspired to Dynamic Source Routing (DSR), to which it adds several new features that improve the performance of source routing in underwater scenarios, especially in the presence of multiple sinks. Like DSR, MSUN is a reactive source routing protocol, in that the source nodes choose a complete route toward the sink for each of their packets, and embed a full specification of this route in the packet header. In turn, this avoids a hop-by-hop relay election process. MSUN operates by first tracing routes to the desired destination via a path request–path reply mechanism, and then by using the best route among all discovered ones. In these experiments, the shortest path metric was used to single out the routes. Unlike Uw-Polling, MSUN serves its data queue according to a First-In-First-Out (FIFO) policy. For error control, MSUN controls per-packet Stop-and-

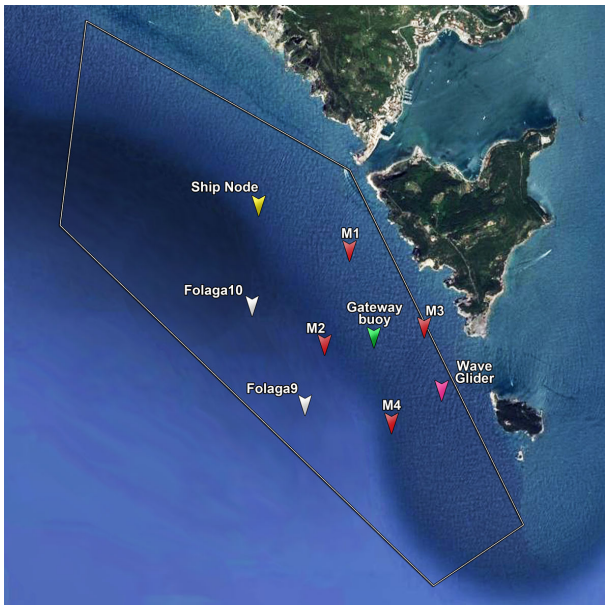


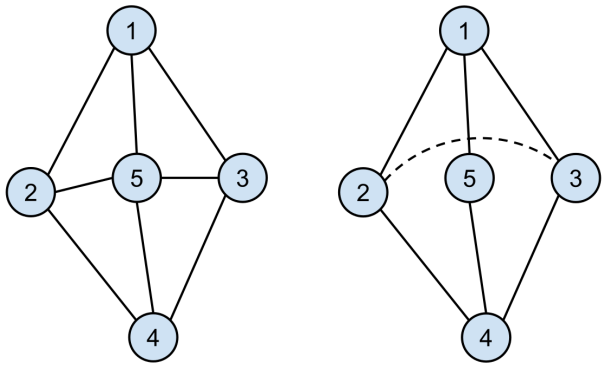
Fig. 1. A comprehensive snapshot of the network deployment during the CommsNet'13 sea trials in La Spezia, Italy. The external line delimits the operational area. The network includes bottom-mounted nodes, a ship node, an acoustic/radio gateway buoy, two autonomous underwater vehicles and one autonomous surface system. (Modified from a Google Maps screenshot.)

Wait retransmissions in a cross-layer fashion, and leverages on failed transmission counts to measure the health of a given path. Optionally, MSUN can work in multicast or broadcast mode, using a form of restricted flooding in both cases.

The **U-Fetch** protocol has been developed with the aim to strike a balance between the two routing approaches offered by Uw-Polling and MSUN [8]. U-Fetch divides the network into three hierarchical levels: the Sensor Nodes (SNs), which are located on the sea-floor and monitor the area surrounding their position, are at the lowest level of the hierarchy; the information retrieved by the SNs is then conveyed to the Head Nodes (HNs), the only nodes allowed to communicate directly with the sink; finally, the mobile sink is at the highest level of the hierarchy, and is the final destination of all packets. Two distinct communication patterns are set up by U-Fetch to administer SN–HN as well as HN–sink communications. Both patterns are based on controlled access, and except for some details [8] are basically analogous to the pattern employed by the sink in Uw-Polling.

C. RECORDS Framework

For the successful completion of sea trials involving autonomous and remote underwater acoustic nodes, it is important to monitor the status of the network in real time. In an environment where cabled or radio links to the modems are not always available, it is necessary to find other ways to interact with the nodes, in order to instruct them about the actions to be carried out. To satisfy these requirements, we developed a robust and versatile framework called RECORDS [9]. RECORDS is an open source framework that makes it possible to remotely monitor and control a heterogeneous network of underwater



(a) Uw-Polling. (b) MSUN and U-Fetch.

Fig. 2. Logic topology for the baseline experiments.

acoustic nodes running side-by-side to the protocol under testing. RECORDS exploits acoustic communication links to deliver control messages, and thus avoids the need to deploy cabled or wireless connections to control the nodes. RECORDS is composed by several modules that have been written entirely using scripting languages. This choice improves the portability of the modules and makes it possible to develop them directly on the embedded systems. The communication among the modules is obtained via loopback TCP sockets. The modules available in RECORDS offer: a full network stack with basic multiple access interference mitigation, two network protocols, an application layer that acts as an abstraction layer for the modem; a watchdog daemon; a profiler module to evaluate the impact of the framework on the system resources; a module to remotely manage the experiments with DESERT, as well as further utilities [9]. RECORDS has been successfully tested in several testbed and real-world experiments and has been proven that be a stable, lightweight and robust solution for the control of underwater networks [9].

III. Experiments

A. Scenario and Equipment

The CommsNet'13 trial was conducted in collaboration with the NATO STO Centre for Maritime Research and Experimentation (CMRE) in the period September 9-22, 2013. The operational area was offshore the Palmaria and Tino Islands, in northern Tyrrhenian sea, Italy. An aerial view of the area is reported in Fig. 1. CMRE made several systems available to perform the CommsNet trial, including the Littoral Ocean Observatory Network (LOON), a gateway buoy, two eFolaga Autonomous Underwater Vehicles (AUVs), four MANTA systems, a WaveGlider and the NATO Research Vessel Alliance. In particular, the LOON is a set of 4 acoustic nodes deployed on the sea bottom in a diamond shape. These nodes are fixed and cabled to shore. In Fig. 1 they are colored in red and labeled *M1* to *M4*. The gateway buoy is anchored to the sea bottom in a central position with respect to the LOON nodes, and includes a WiFi antenna for easier system access. It is also equipped with its own acoustic modem (in green in Fig. 1). The eFolaga AUVs (manufactured by Graal Tech [18]) were added to the network as static nodes in the positions marked

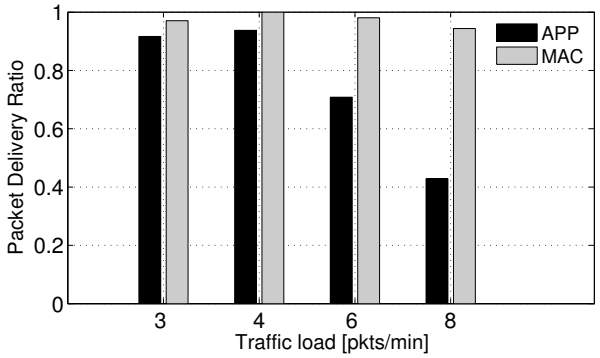


Fig. 3. Application-layer PDR for Uw-Polling as a function of the packet generation rate per node.

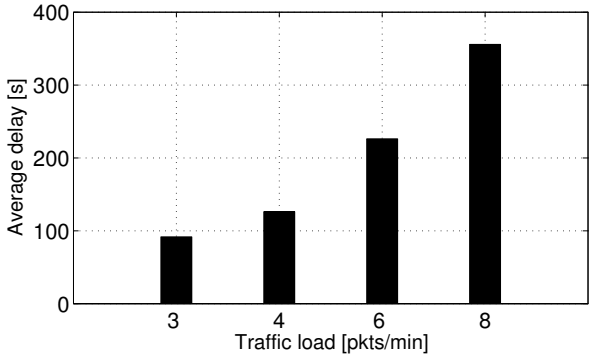


Fig. 4. Average delivery delay for Uw-Polling as a function of the packet generation rate per node.

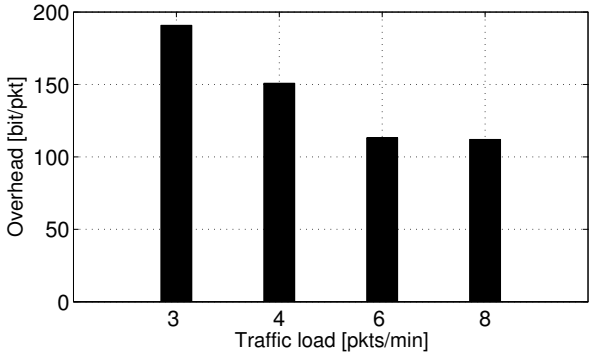


Fig. 5. Overhead for Uw-Polling as a function of the packet generation rate per node.

in white in Fig. 1. The four MANTA [19] gateways worked as replacements for the eFolagas, and could be deployed on a boat in the same positions. The NATO Research Vessel Alliance [20] was also used as a static node with a MANTA system.

An autonomous floating surface vehicle named WaveGlider (manufactured by Liquid Robotics [21]) was also occasionally available. In Fig. 1, this system is colored in pink and is labeled as *Wave Glider*.

Throughout the sea trial, each modem was labeled with a unique acoustic ID: IDs 1 to 4 for the nodes of the LOON, ID 5 for the Gateway buoy, IDs 7 and 8 for the two eFolagas, ID 6 or 13 for the WaveGlider depending on the payload, ID 9 for the MANTA that was used to control the ship node.

Since not all the nodes were available for the entire sea trial, we decided to set up a plan in order to carry out network experiments with a stable subset of nodes. The rationale that led to this choice was to test our three network protocols in a scenario with the same configuration. Specifically, we decided to use the four LOON nodes and the gateway buoy, which featured the longest up time and were easiest to control. Accordingly, we planned a set of experiments where we set the total traffic generated by the nodes to 3, 4, 6 and 8 packets per minute (pkt/min). Each protocol introduced in Section II-B was tested under all packet generation rates for a total of 12 experiments, each lasting 25 minutes. In the next paragraph, we will describe the baseline experiments in more detail for each protocol.

B. Baseline Experiments

1) Uw-Polling: In the baseline scenario, for the Uw-Polling protocol, the gateway buoy is configured to be the sink, whereas the four LOON nodes are configured as sensors. A graphical sketch of the network is reported in Fig. 2(a). Fig. 3 shows the Application (APP) layer and Medium-Access-Control (MAC) layer Packet-Delivery-Ratio (PDR) as a function of the packet generation rate in pkt/min. We observe that the PDR for the two lowest values of the traffic load is only slightly lower than 1: hence, Uw-Polling proves to be very reliable in these cases. Notably, the PDR at 3 pkt/min is slightly lower than at 4 pkt/min due to less favorable channel conditions during the 3 pkt/min experiment. Taking a closer look at the MAC-layer PDR, however, we can note that, with Uw-Polling, the nodes can reliably deliver the packets to the sink once they are polled, thanks to the channel reservation mechanism. As the traffic generation rate increases, however, a large number of packets remain in the nodes' queues, which in turn leads to a decrease of the APP-layer PDR. From this result, we can conclude that the fine tuning of the protocol timings can help, where shorter (although less safe) guard times are applied between subsequent transmissions of signaling and data packets. Other than the configuration of the timers, a more detailed inspection of the type of packets lost shows that some control packets required by the protocol in order to set up the polling cycle are lost. This makes some nodes skip their polling turn. Protecting these packets with some form of redundancy is desirable and is the objective of our future work.

A measure of the impact of skipping polling turns can be seen in Fig. 4, which shows the average end-to-end delay as a function of the packet generation rate per node. We note that, for the highest traffic values, the average end-to-end delay is particularly high, exceeding 350 s. In fact, due to the loss of signaling packets, some nodes had the chance to participate in the polling cycle only 6 or 7 times out of the 15 polling cycles initiated by the sink. This originates larger delays, as the generated packets have to be stored in the node queue in the meantime. Furthermore, for increasing traffic values, the probability that a node can empty its queue (by transmitting all packets therein) every time it is polled is lower, because the maximum number of packets per polling cycle is limited to 5 in order to increase the fairness of the protocol. Finally, Fig. 5

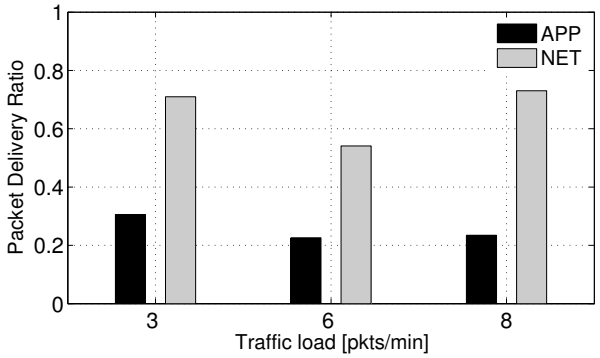


Fig. 6. PDR for MSUN as a function of the network packet generation rate.

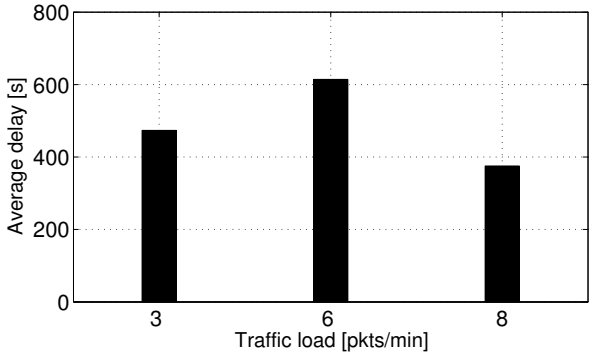


Fig. 7. Average delivery delay for MSUN as a function of the network packet generation rate.

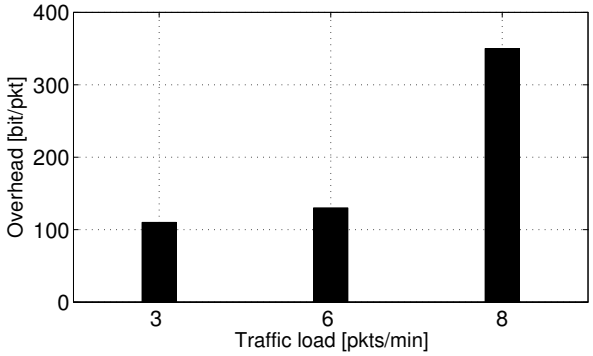


Fig. 8. Overhead for MSUN as a function of the network packet generation rate.

shows the overhead of the protocol as a function of the traffic generation rate. For Uw-Polling, and all the others protocols considered in this work, the Overhead metric is calculated as the total number of bits transmitted for control packets (taking into account that packets of different sizes may be used), divided by the number of unique data packets correctly delivered to the sink. As expected, for low packet generation rates, the overhead reaches almost 200 bits/pkt, decreasing to 150 bits/pkt for higher traffic, even though the PDR is lower. This means that Uw-Polling can effectively manage the inherent overhead due to the handshake and polling process, mainly thanks to the transmission of multiple data packets back-to-back in each polling turn.

It is worth remarking that several practical aspects have to be considered when testing Uw-Polling in real scenarios. For instance, we have to account for the real hardware times as we cannot pass all the data packets to the modem too quickly in order to not overload the hardware buffers. This would bring the modem to discard some packets and this would have an impact on the performance. Furthermore, we have to set a guard time in order to cope with underwater channel delays, which are usually higher in real experiments than in simulated scenarios. Hence, we have to find out a good tradeoff between good performance and sufficiently large timers. Usually, the timings of the protocol set for simulation are too tight, because in a simulated controlled scenario we can get rid of the hardware processing time and of the impairments of the channel. These tight timers in a real experiment bring the timers to expire before all the nodes have sent all the packets. This has a heavy impact on the performance, especially on the PDR. Hence, we have to take into account some guard times to make the protocol more resistant to the variable conditions of the real world experiment. This will have a direct impact on the end-to-end delay and on the throughput, but will let the protocol deliver efficiently almost all the packets generated.

2) MSUN: In this section we present a detailed analysis of the performance of MSUN in 3 out of the 4 experiments composing the baseline experiment plan. The results are available for packet generation rates equal to 3, 6 and 8 pkt/min in the network. The results for the scenario with 4 pkt/min are not shown, because the experiment was not completed successfully (namely, the PC that was connected to the gateway buoy disconnected, causing the control over the sink to terminate and the experiment to stop forcedly). The topology of the network in this case is slightly different than the one in the Uw-Polling experiments, and is depicted in Fig. 2(b). Due to the limited distance among the LOON nodes (IDs 1 to 4) and the destination node (ID 5), after some tests performed via the RECORDS framework, we observed that all sources would have been able to deliver data packets directly to the destination. In order to test the routing capabilities of MSUN, we had to blacklist some of the links via software primitives, thereby forcing multihop paths from each source to the sink. Specifically, for the following experiments we set 2 source nodes (IDs 2 and 3) and blacklisted the direct links $2 \leftrightarrow 5$ and $3 \leftrightarrow 5$. The link $2 \leftrightarrow 3$ was initially blacklisted, and activated after 12 min in each experiment.

In Fig. 6 we show the APP-layer and NET-layer PDR. We observe that the PDR at the APP layer is roughly one third of the value at the NET layer for all values of the traffic load. The reason for the low values of the APP PDR is multifold. From the log analysis we observed that the number of *Path Search* packets sent is generally low:[1] in the three experiments, the number of these kinds of packets sent for each data packet generated are 0.14, 0.12 and 0.32, respectively for a traffic load value of 3, 6 and 8 pkt/min. These values are comparatively

[1]The *Path Search* packets are generated by the MSUN protocol to establish the routes toward the destinations. More details on this process can be found in [7].

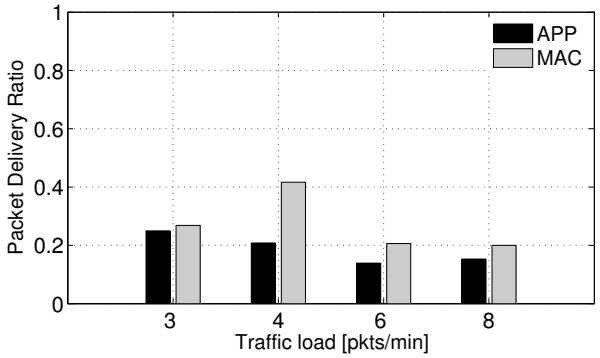


Fig. 9. PDR for U-Fetch as a function of the packet generation rate per node.

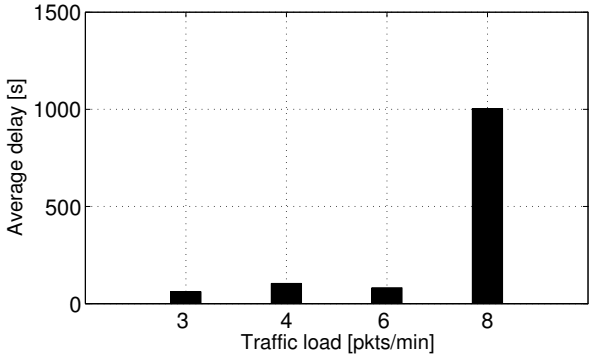


Fig. 10. Average delivery delay for U-Fetch as a function of the packet generation rate per node.

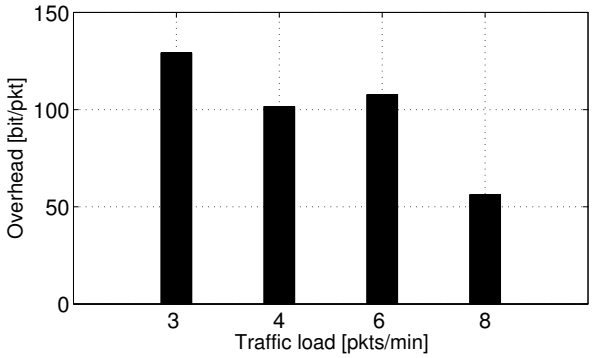


Fig. 11. Overhead for U-Fetch as a function of the packet generation rate per node.

low, which in turn indicates that the path search procedure is less effective than expected against difficulties such as packet collisions and channel-induced packet losses. If we consider the ratio between the number of retransmissions for each unique data packet sent, we obtain values between 3 and 4 (respectively 3.3, 3.3 and 3.9), a clear indicator that the protocol must retransmit several times the same data packet in order to deliver it to the next hop. Once the path is created, the protocol is in fact able to exploit it and deliver packets, but in order to do it, the node has to retransmit the packet up to 4 times, which slows down the queue servicing process. At the end of the experiments, this is reflected by a large number of packets undelivered because they are in the queues of the nodes.

In Fig. 7, we report the average end-to-end delay as a function of the traffic load in the network. The results reflect the trend of the PDR: higher PDR values correspond to lower delays. For higher PDR values, the average number of packets in the buffer of the nodes is lower, which means lower service time and lower average delays. For a global traffic load of 3 pkt/min the average delay is 480 s, and grows up to 616 s in the case of 6 pkt/min whereas it decreases to 375 s in the last experiment considered with 8 pkt/min. It is worth noting that in the experiment with 6 pkt/min, compared to the one with 8 pkt/min, the average delay decreases by 40%, and the NET-layer PDR increases by 35%. Conversely, if we consider the case of the experiment with 6 pkt/min to the one with 4 pkt/min we can see that the average delay decreases by 23% and the NET PDR increases by 30%. These results stress the relationship between these two metrics.

In Fig. 8, we finally show the overhead of the protocol as a function of the traffic generation rate. As expected, for low traffic (hence fewer generated data packets), the overhead reaches roughly 100 bits/pkt, increasing up to 350 bits/pkt for higher traffic, even though the PDR is lower. If we relate these results to the previous ones, we can state that for networks with higher loads, even if the MSUN protocol spends more control bits, the PDR does not increase, in fact it decreases. This is caused by the cross-interference between data packets and control traffic, and means that MSUN fails to convert the extra route establishment and acknowledgement effort into an effective traffic to the sink.

To sum up these results, we observe that the protocol was able to complete the experiments allowing the nodes to exchange a reasonable amount of traffic. Thanks to this analysis, we can draw some conclusions as to how we could improve the performance of the protocols in future sea trials. More specifically, we can state that the protocol was able to effectively establish paths joining the nodes and the sink, but probably these path should be refreshed more often in order to exploit new links. A considerable number of packets was still in the buffer of MSUN at the end of the experiments. If we consider that the protocol was able to create paths among the nodes, that the data generation rate was not critical for the network and that the number of data packets transmitted was considerably high compared to the number of data packets received, a matter of further investigation is the interplay between MSUN and the channel access mechanism, currently delegated to a simple Aloha-based protocol. A possible improvement is to embed medium access capabilities in MSUN and, by doing this, to extend the protocol and make it a cross-layer scheme with MAC functionalities.

3) U-Fetch: For the U-Fetch protocol experiments, we employed nodes 2 and 3 as sensor nodes (SNs), nodes 1 and 4 as head nodes (HNs) and the gateway buoy (with ID 5) as the sink. The topology of the network is depicted in Fig. 2(b).

Fig. 9 reports the APP-layer PDR as a function of the traffic load in pkt/min. For all the experiments the APP-layer PDR is generally low. The reason is that most of the data packets generated by the APP layer of the SNs remain in the queue

of the nodes and, due to the unreliable channel conditions, are never transmitted through the path to the HNs and from there to the sink. This prevents U-Fetch's handshake from being completed successfully. In addition, some data packets received from the SNs by the HNs remained into the queues of the HNs and were never delivered to the sink. This happens because the HNs and the sink are typically unable to establish a connection, due to the lack of acoustic connectivity between the HNs and the sink. In particular, many signaling packets used to establish a connection between a HN and the sink never arrived at their destination: hence, no connection could be established and the data packets could not be transmitted, ultimately decreasing the PDR.

Fig. 10 show the average end-to-end delay express in seconds as a function of the traffic load. The average end-to-end delay for the U-Fetch protocol is computed as the time needed for a data packet to be delivered to a sink node from the time when the packet was created by a SN. For a low traffic load the delay is on the order of 100 s. The delay increases considerably (up to 1000 s) for a packet generation rate of 8 pkt/min. In this case, some data packets generated by the SNs are delivered to the sink through the HNs. These packets must wait for the connection between SN and some HN to be established, and after that they must walk up the queue of the SNs and be transmitted successfully to the sink. In turn, this increases the average delivery delay. The U-Fetch experiments carried out during CommsNet'13 can be considered somehow suboptimal. The results are lower than those obtained with the DESERT simulator in the laboratory. In the trial, we often observed that, due to bad channel conditions, the HNs were unable to deliver packets to the sink node, including those required to establish connections. The CommsNet'13 trials made it possible to identify potential weak spots in the handshake procedure of U-Fetch: this will greatly help improve the effectiveness of such procedures. The most important future step in the development of U-Fetch is the implementation of a robust mechanism to rotate the role of HN among the nodes: having such a mechanism in place is expected to improve the performance of the protocol considerably, by avoiding that packets get stuck in head nodes or sensor nodes due to imperfect or unstable network connectivity.

Fig. 11 reports the overhead in bit/pkt as a function of the traffic load. This parameter is correlated to the throughput. Remembering that the highest throughput is achieved for the highest data generation rate (2 pkt/min), in correspondence of it, more packets get through to the sink, and the average overhead per packet decreases.

C. Additional Experiments

In addition to the baseline experiment plan described in Section III-B, during the sea trials we had the opportunity to carry out some extra experiments to test the behavior of the protocols in some specific configuration. In this section we present two experiments that aim to highlight special difficulties that the protocols had to face. The results presented concern an experiment for Uw-Polling and one for MSUN. For each of them we provide the list of the nodes involved, some of the metrics obtained and a brief conclusion.

TABLE I
EXTRA EXPERIMENTS METRICS.

	Uw-Polling	MSUN
APP PDR	0.62	0.98
MAC/NET PDR	0.87	0.98
Throughput (pkt/min)	2.6	2.35
Overhead (bit/pkt)	163	251.42

1) Uw-Polling: The experiment with Uw-Polling involves the four nodes of LOON, the gateway buoy, one eFolaga and the ship node. The buoy acts as sink, while all the others nodes generate data packets. For this specific experiment, the overall network traffic is set to 4 pkt/min. The main difference between this experiment and the ones presented in the baseline plan concern the total number of nodes involved in the transmission that grows from 5 up to 7. The results of Uw-Polling are summarized in Table I.

As we can see from the PDR (both at APP-Layer and MAC-Layer), most of the packets transmitted by the nodes are correctly delivered to the sink. However, as in the case of the baseline experiments, the APP-layer PDR is lower than the MAC-Layer one. This is because some packets remain in the nodes' queues without the possibility of being transmitted to the sink. The overhead is almost equal to the overheads calculated in the case of the baseline experiments. The delay, instead, is slightly higher, as more nodes than in the baseline experiments participated in the polling cycle. As a consequence, each node had to wait longer for its turn. However, the throughput experienced is good thanks to the fact that, once a node reaches its turn, it can transmit the packets in a burst, and enjoys a contention-free channel. Also in this case, as already mentioned in the baseline experiments, a fine-tuning of Uw-Polling's internal timings may achieve better delivery delays, possibly at the price of a less robust protocol.

2) MSUN: The experiment with MSUN involves all the nine nodes in Fig. 1 except the waveglider that was not available. The experiment was carried out on Sep 14th, at 11:24 pm with an overall network traffic equals to 3 pkt/min. All the nodes were set to generate traffic except the node of the LOON with ID 2 that was set as sink. This experiment is different from the ones proposed in the baseline analysis because in this specific case we tested the network with no links blacklisted via software. The nodes could therefore exploit the shortest paths to the sink without any software limitation. Thanks to the reasonably low network traffic load and to the network topology (all sources were roughly at the same distance from the sink), the APP and NET PDR are very close to one and almost all the packets sent are correctly received, with a very low number of repetitions. The numerical results are summarized in Table I. This simple experiment shows that, in a network that offers a direct link from the sources to the sink, MSUN can exploit them, and deliver packets directly to the destination effectively with a low overhead. From the log files we discovered also a drawback of this approach: in this experiment all nodes sent all data packets directly to the sink, without exploiting

multihop routes. This result is due to the relatively small distance between all the sources and the sink and to the routing metric of MSUN that minimizes the number of hops from the source to the destination. In such conditions a routing protocol is effectively not needed. In case of packets with errors, MSUN can offer an ARQ mechanism, but the performance observed is more related to the status of the channel and to the behavior of the data link protocols (CSMA based) rather than to the routing protocol. This consideration must be taken into account before the deployment and, whenever needed, solved with some software-level blacklisting of some network links. In any event, it should be kept in mind that this will not reproduce the conditions of a multihop underwater network, but rather harsher conditions, with a higher level of interference among concurrent transmissions.

IV. Conclusions

In this paper we presented the CommsNet'13 sea trial campaign, aimed at testing three protocols for remote data retrieval, namely Uw-Polling, MSUN and U-Fetch, in a real world scenario, and under different network configurations. We presented the protocols in a baseline topology and evaluated, for each of them, the impact of a real environment on their behavior compared to expected results according to a previously published set of simulations. We summarized the experiments by providing concepts to consider when a user wants to migrate from a network simulator to a real environment. The results prove also that the proposed solutions are stable and that it is possible to use them in the real world, but also that some additional effort is required to optimize their behavior, especially in the presence of unfavorable channel conditions. The experiment was possible thanks to the features offered by DESERT Underwater and RECORDS, two frameworks developed to facilitate as much as possible, and in a seamless way, the migration from network simulators to real world trials.

Acknowledgments

This paper is dedicated to the memory of our friend and colleague Giovanni Toso who prematurely passed away on August 10, 2014.

We gratefully acknowledge the collaboration, support and assistance of Dr. Cristiano Tapparello for the execution of the experiments.

The authors would like to thank the NATO STO CMRE for the possibility to participate in the CommsNet'13 trials, and extend their gratitude to the CMRE personnel, as well as to the commanders and crew of the NRV Alliance for their collaboration.

The MSUN protocol is foreground information for the multinational four-year EDA project "Robust Acoustic Communications in Underwater Networks" (RACUN), EDA Project Arrangement No. B 0386 ESM1 GC.

References

[1] R. Masiero, S. Azad, F. Favaro, M. Petrani, G. Toso, F. Guerra, P. Casari, and M. Zorzi, "DESERT Underwater: an NS-Miracle-based framework to DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols," in *Proc. IEEE/OES OCEANS*, Yeosu, Republic of Korea, May 2012.

[2] C. Petrioli, R. Petroccia, and D. Spaccini, "SUNSET version 2.0: Enhanced framework for simulation, emulation and real-life testing of underwater wireless sensor networks," in *Proc. ACM WUWNet*, Kaohsiung, Taiwan, Nov. 2013.

[3] Y. Zhu, S. Le, L. Pu, X. Lu, Z. Peng, J.-H. Cui, and M. Zuba, "Aqua-Net Mate: A real-time virtual channel/modem simulator for Aqua-Net," in *Proc. MTS/IEEE Oceans*, Bergen, Norway, Jun. 2013.

[4] The NUS ARL and SubNero, "UNET–The Underwater NETworks project," Last time accessed: May 2014. [Online]. Available: http://www.unetstack.net

[5] M. Zuba, Z. Jiang, T. Yang, Y. Su, and J. Cui, "An advanced channel framework for improved underwater acoustic network simulations," in *Proc. IEEE/MTS OCEANS*, San Diego, CA, Sep. 2013.

[6] F. Favaro, P. Casari, F. Guerra, and M. Zorzi, "Data upload from a static underwater network to an AUV: Polling or random access?" in *Proc. MTS/IEEE Oceans*, Yeosu, Republic of Korea, May 2012.

[7] C. Tapparello, P. Casari, G. Toso, I. Calabrese, R. Otnes, P. van Walree, M. Goetz, I. Nissen, and M. Zorzi, "Performance evaluation of forwarding protocols for the RACUN network," in *Proc. ACM WUWNet*, Kaohsiung, Taiwan, Nov. 2013.

[8] F. Favaro, L. Brolo, G. Toso, P. Casari, and M. Zorzi, "A study on remote data retrieval strategies in underwater acoustic networks," in *Proc. MTS/IEEE Oceans*, San Diego, CA, Sep. 2013.

[9] G. Toso, I. Calabrese, P. Casari, and M. Zorzi, "RECORDS: a remote control framework for underwater networks," in *Proc. IEEE/IFIP Med-Hoc-Net*, Piran, Slovenia, Jun. 2014.

[10] "DESERT Underwater," Last time accessed: June 2014. [Online]. Available: https://github.com/uwsignet/DESERT_Underwater

[11] "RECORDS source code," Last time accessed: June 2014. [Online]. Available: https://github.com/uwsignet/records

[12] R. Petroccia and D. Spaccini, "A back-seat driver for remote control of experiments in underwater acoustic sensor networks," in *Proc. MTS/IEEE OCEANS*, Bergen, Norway, Jun. 2013.

[13] H. Kulhandjian, L. Kuo, T. Melodia, D. A. Pados, and D. Green, "Towards experimental evaluation of software-defined underwater networked systems," in *Proc. IEEE UComms*, Sestri Levante, Italy, Sep. 2012.

[14] N. Ahmed, W. bin Abbas, and A. A. Syed, "A low-cost and flexible underwater platform to promote experiments in UWSN research," in *Proc. ACM WUWNet*, Los Angeles, CA, Nov. 2012.

[15] "Evologics," Last time accessed: February 2014. [Online]. Available: http://www.evologics.de/

[16] R. Odugoudar, I. Nissen, and C. Albrechts, "Ad-hoc network emulation framework for underwater communication applications," in *Proc. ACM WUWNet*, Woods Hole, MA, Sep. 2010, poster presentation.

[17] P. Casari, C. Tapparello, I. Calabrese, F. Favaro, G. Toso, S. Azad, R. Masiero, and M. Zorzi, "Open-source suites for the underwater networking community: WOSS and DESERT Underwater," *IEEE Network S.I. Open Source for Networking: Development and Experimentation*, Sep. 2014.

[18] "GRAALtech," Last time accessed: June 2014. [Online]. Available: http://www.graaltech.it/it/index.php

[19] "Underwater Systems and Technology Laboratory," Last time accessed: June 2014. [Online]. Available: http://whale.fe.up.pt/

[20] "NRV Alliance," Last time accessed: June 2014. [Online]. Available: http://www.cmre.nato.int/research/research-vessels/nrv-alliance

[21] "Liquid Robotics," Last time accessed: June 2014. [Online]. Available: http://liquidr.com/