Open-source Suites for Underwater Networking: WOSS and DESERT Underwater

Paolo Casari, Cristiano Tapparello, Federico Guerra, Federico Favaro,
Ivano Calabrese, Giovanni Toso, Saiful Azad, Riccardo Masiero, Michele Zorzi

Abstract

The simulation and the experimentation of underwater networks entail many challenges, which for the former are mainly related to the accurate modeling of the channel behavior, while they are typically logistic in nature for the latter. In this paper, we present our experience with WOSS and DESERT Underwater, two open-source suites that address both classes of challenges. The suites build upon and extend the capabilities of ns2 and NS-MIRACLE, two widely known software packages for network simulation. WOSS endows NS-MIRACLE with the capability to generate realistic channel patterns by automatically retrieving and processing the environmental boundary conditions that influence such patterns; DESERT Underwater makes it possible to evolve towards at-sea experiments by reusing the same code written for simulations, thereby minimizing the effort required for network deployment and control. Both suites have been widely tested and used in several projects: some examples are provided in this respect, including an account of some experiments carried out in collaboration with the NATO STO Centre for Maritime Research and Experimentation (CMRE).

Index Terms

Underwater network protocols; channel impulse response; simulation; experimentation; hardware-in-the-loop; ns2; NS-MIRACLE; WOSS; DESERT Underwater.

I. INTRODUCTION

Simulation and real-life experimentation are two key steps in the development of network protocols: the former makes it possible to perform several controlled tests, which may however require an idealized model of one or more system components; the latter is the ultimate proof that a devised solution is effective, or at all feasible.

This is especially true for underwater acoustic networks, where real-world deployments are typically subject to many non-ideal effects, that can hardly be fully reproduced in simulations. In fact, the underwater research community largely conceives real-life experimentation as the only proof that physical-layer (PHY) signal processing algorithms and network protocols actually work in practice. However, while the former can be effectively substituted by the recording and offline processing of channel signatures, no data set exists that provides the channel response over time for several points of a given area, and that correctly reproduces its correlation in time and space, as required to properly run realistic network simulations. In addition to the lack of a widely accepted model for the underwater acoustic channel, the above considerations highlight the need for i) an easy way to simulate underwater networks over realistic channel responses, and ii) an efficient method to move from simulations to real-life experiments.

In this paper, we present two open-source suites that tackle these two challenges and that, together, form a complete and organized solution to approach the study of realistic underwater networking. The first suite, called the World Ocean Simulation System (WOSS) [1], was originally released in 2009 under a BSD 3-clause license, followed by two other releases that introduced several new features. The second suite is named DESERT Underwater, where DESERT stands for "DEsign, Simulate, Emulate and Realize Test-beds." It was initially developed in 2012 as an extension to the ns2/NS-MIRACLE simulator [2], and was released under a BSD license as well [3]. Building specifically on NS-MIRACLE, DESERT Underwater provides many modules that implement medium access control (MAC) protocols, error control schemes, routing

protocols and other solutions that cover the remaining layers of the ISO/OSI protocol stack. In addition, DESERT Underwater implements mobility models to simulate mobile networks, and includes many examples that showcase DESERT's features, in part integrated with WOSS's functionalities. A conspicuous part of DESERT is devoted to facilitating the transition from simulations to sea trials, via a hardware-in-the-loop approach that makes it possible to reuse the same code written for simulations. The user is only required to define an additional set of rules to convert the internal structures of NS-MIRACLE into bit streams and vice-versa. Version 2 of DESERT Underwater has been released in April 2014.

The remainder of this paper is organized as follows. WOSS, its main features (which were available since its first release) and its subsequent improvements will be described in Section II. DESERT Underwater is described in Section III, along with the experiments that showcase its capabilities. Finally, Section IV concludes the paper.

II. The World Ocean Simulation System (WOSS)

The World Ocean Simulation System (WOSS) [1] is a framework aimed at improving underwater network simulations through a more realistic account of acoustic propagation. In addition, WOSS provides a set of routines to facilitate many standard operations, like mobility management, the conversion among different coordinate systems (e.g., Cartesian to spherical), and the maintenance of the data structures required for the simulation of acoustic propagation.

WOSS's main tasks are i) to provide a means to easily query oceanographic databases in order to retrieve the environmental characteristics that simulation software typically requires for the reproduction of underwater acoustic propagation in a given area; ii) to provide functions that process the output of such simulators, in order to facilitate their employment within network simulation software. In this context, external libraries have been created to bind WOSS to the PHY layer of the ns2/NS-MIRACLE simulator [2], although WOSS could be linked to any other software. The current version of WOSS includes all interfaces required to interact with

Bellhop [4], a sound propagation simulator based on ray tracing. In the following, we will describe in more detail how WOSS provides input data to Bellhop and processes its output.

Bellhop requires a number of environmental parameters, including the depth-dependent variation of the speed of sound, or sound speed profile (SSP, which is related to the refraction of sound waves) the profile of the sea bottom, and the acoustic properties of bottom sediments, which influence the pattern and intensity of the bottom reflections. Optionally, Bellhop can also take the profile of the ocean surface as an input, in order to model surface reflections more precisely: WOSS can leverage on this opportunity via functions that generate random sea surface realizations, according to a given surface wave spectrum.[1]

In addition, Bellhop requires the specification of the transmitter-side electro-acoustic transducer's beam pattern, which can be modeled in two ways. In the basic case, the user inputs the angular aperture of the beam emission, i.e., the lowest and highest angle of departure of the beams from the transmitter, with respect to the azimuthal plane: this is equivalent to assuming that the transducer has a flat and unit response over the specified angle span. In the more advanced configuration, the user can input the full shape of the transducer's beam pattern. WOSS supports both modes. Later versions (after v1.2) also allow the user to change the orientation of the beam pattern arbitrarily, in order to simulate the effect of a node pointing its transducer towards a specific direction.

To retrieve the required environmental data, WOSS interfaces the network simulator and Bellhop with oceanographic databases freely available on the Internet (for reference, see also the data flow summarized in Fig. 1). In particular, WOSS reads the location of the nodes (latitude, longitude and depth) from the network simulator and then uses this information to query the databases. For the SSP, WOSS employs the World Ocean Atlas (WOA, http://www.nodc.

[1]The surface waves realization is typically time-varying, and the user can specify how often a fresh realization should be drawn.

noaa.gov/OC5/WOA09/pr_woa09.html) database, which collects a wealth of environmental data measured during several experiments conducted all around the world; these data include salinity and temperature samples at typically accepted standard depths (e.g., for sea bottoms with a maximum depths of 100 m, the standard depths are 10, 20, 30, 50, 75 and 100 m) and can be used to compute the SSP through standard equations of state such as the MacKenzie or Del Grosso equations. In the WOA database, the measurements are divided by location and month or season of the year when the measurement was performed: WOSS automatically selects the correct dataset and transfers the related samples to Bellhop. Bathymetry samples are taken by default from the General Bathymetric Chart of the Oceans (GEBCO, www.gebco.net), a public database with a resolution of 30 arc-seconds. To generate a bathymetry profile, WOSS computes the great circle curve that joins the points of coordinates $\mathbf{x}_T = (\theta_T, \phi_T)$ and $\mathbf{x}_R = (\theta_R, \phi_R)$, where θ_T, ϕ_T, θ_R and ϕ_R are the latitude and longitude of the transmitter and the receiver, respectively. Starting from $\mathbf{x}_T$, WOSS samples the curve in steps of equal distance Δs along the great circle, yielding a set of points $\mathbf{x}_i = (\theta_i, \phi_i)$. The value of Δs can be selected by the user. For each point $\mathbf{x}_i$, the bathymetry database is queried to retrieve the respective depth of the sea bottom. If no data is found for the exact coordinates of $\mathbf{x}_i$, the depth belonging to the closest (with respect to the great circle distance metric) geographical coordinates available is returned. Along with the bathymetry, WOSS obtains the geophysical parameters of the bottom sediments from the National Geophysical Data Center's Deck41 dataset (http://www.ngdc.noaa.gov/mgg/geology/deck41.html). In addition to standard databases, WOSS supports custom databases, allowing users to input their own environmental data and thereby improving the accuracy of WOSS when modeling a specific area.

The output of Bellhop is a solution to the propagation equations over a column of water, or a restricted section thereof. When performed over a large set of points throughout the water column, a typical simulation outcome is shown for reference in Fig. 2. The figure reports the acoustic attenuation in dB affecting the transmission of a 25 kHz tone, as a function of the

depth of the receiver and of its planar distance (or range) from the transmitter, which is located on the left side of the picture at a depth of 50 m. Red hues represent a lower attenuation, hence a stronger signal. The shape of the sea bottom is rendered in brown. The environmental parameters employed are representative of Tyrrhenian Sea waters in summer. Fig. 2 shows that a high acoustic signal level is expected in the proximity of the transmitter, but such level decreases for increasing range, and beyond a range of 6 km only a minor portion of the water column is actually insonified.

In fact, the data processed by WOSS is even more detailed than in this figure, as it contains the full set of ray arrivals computed by Bellhop, characterized by their respective complex amplitude and arrival delay. The latter can be used to infer the propagation delay of the whole signal from the transmitter. The standard processing applied by WOSS involves the following steps:

- The power of the arrivals is compared against a reception threshold: those arrivals that fall below the threshold are filtered out: this models a generic preamble detection process at the receiver, which is assumed to have finite sensitivity;
- The arrivals are binned by computing the complex sum of all arrivals within contiguous windows of 50 μs; this models the fact that very close arrivals are practically indistinguishable;
- The arrivals within a maximum delay $\tau_{\max}$ concur to the computation of the useful signal energy, whereas all arrivals beyond $\tau_{\max}$ are assumed to cause self-interference. This models the receiver-side signal processing capability to extract useful energy from a limited portion of the input signal, and helps reproduce the receiver performance limitations due to the channel delay spread.

In WOSS, the noise level is determined through the empirical formulas in [5], a common approach for the simulation of noise affecting underwater communications. User-custom noise realization databases are also supported. We recall that both the noise power calculations and

Bellhop's ray tracing procedure require to specify the frequency of the transmitted signal. In the current version of WOSS, this frequency is computed as the geometric mean of the lower and upper frequency limits of the acoustic spectrum band in use.[2]

A. Advanced WOSS features

The most recent version of WOSS (v1.3.5) supports time-varying environmental parameters. In fact, these parameters, e.g., the temperature of the water at different depths, may change during periods of time comparable to that of a typical networking experiment (e.g., several hours), and these variations should be tracked in order to provide more realistic results. As an example, assume that a user wants to cycle through three different SSPs, spanning a total duration of one day. This scenario is depicted in Fig. 3. At time 00:00, the SSP on the left in the figure is used. This SSP is substituted by the second SSP at time 08:00 and by the third SSP at time 16:00. After a further time interval of 8 hours, the SSP is changed back to the first SSP on the left and the cycle begins again. WOSS can be instructed to vary the SSP over time as described above and, in addition, it allows the user to specify how to compute the transition from the current SSP to the next one.

Another extension of WOSS, presented in the last version, supports the simulation of mobile networks. These models affect the computations performed by WOSS, as a position update may trigger a channel response update.[3] Currently, WOSS natively implements a waypoint mobility

[2]We remark that this approximation is valid only when the system bandwidth can be assumed to be "narrow" with respect to the carrier frequency in use. Computing the impulse response at different frequencies would be possible, but would also represent a significant computational burden, and is left as a future extension.

[3]WOSS knows exactly where a given transmitter and receiver were located when the last channel response was computed. If neither node moved away of that location farther than a tunable distance parameter, WOSS assumes that the channel remained static and does not update the channel response. The default value of the distance parameter that triggers the update is 0; however, the user can fully control the update process by changing this value, and thereby trade off the reproduction accuracy of a space/time-varying channel for the network simulation speed.

model, whereby the user can specify a sequence of locations that a node will traverse, the movement speed between any two such waypoints, and an optional pause time when a waypoint is reached. In addition, WOSS can handle channel simulations in networks where one or more nodes move according to either of the following models, embedded in NS-MIRACLE:

- Gauss-Markov [2], which generates random-waypoint trajectories with a given self-correlation factor between 0 (full randomness) and 1 (linear movement);
- Leader-Follower [6], or group mobility model, whereby one or more nodes (called followers) can be instructed to "stay close" to the trajectory of a given node (called leader) via a tunable attraction factor;
- Linear+Drift, which assigns to the nodes a deterministic speed vector, perturbed by time-varying "noise" vectors that model the impact of currents on the movement of the nodes.

We remark that the management of node mobility and of time-varying environmental parameters (including surface wave realizations) implies that WOSS supports time-varying channel impulse responses. The user is given full control over the events that trigger the computation of a fresh channel impulse response.

As a final option to improve the realism of the performed simulations, WOSS can import performance figures that summarize the physical transmission schemes in use, e.g., derived from offline simulations of a full-fledged transmitter-receiver chain. Such figures are typically provided in the form of tables, and make it possible to translate metrics that NS-MIRACLE can measure (e.g., signal-to-noise ratio, interference power, noise power, and the overlap between interfering and wanted packets) into the packet error probability of a given transmission. An example of this procedure has been used in the "Robust Acoustic Communications in Underwater Networks" (RACUN) project, and is described in [7].

B. WOSS – Summary and Conclusions

WOSS is a powerful tool, that makes it possible to employ realistic underwater channel patterns within network simulators. It was developed as a complement to the popular software ns2 and to its NS-MIRACLE extensions, but also lives as a stand-alone package that can be interfaced to any simulation tool. Along with Bellhop, it generates channel realizations that are strongly tied with the oceanographic parameters of the simulated communications area, and therefore helps achieve better simulation accuracy than, e.g., simple link-budget equations for underwater acoustic links. Moreover, WOSS's support for custom databases makes it possible for users to input their own oceanographic measurements, thereby improving the characterization of a specific network scenario. The support for real transducer beam patterns, for realistic surface wave realizations and for time-varying underwater features improve such accuracy even further. As such, WOSS allows the designer to accurately assess the performance of underwater network protocols, before actually moving to sea trials: without WOSS, the selection of environmental parameters from public or custom oceanographic parameters, the simulation of underwater acoustic propagation, the derivation and processing of channel impulse responses, and their inclusion into underwater network simulators would have to be done by hand, resulting in a much more cumbersome simulation process and requiring detailed knowledge of acoustic propagation phenomena.

The WOSS project was kicked off in 2009, and has been continuously supported ever since. Recently, WOSS has been adapted to the ns3 simulator [8], and is currently undergoing the code review phase required to make it part of standard ns3 releases.[4]

C. Related work

A first effort to include the output of the Bellhop ray tracing software into the network simulator ns2 is found in [9], where Bellhop runs are manually carried out to retrieve the

[4]In the meantime, the source code of WOSS for ns3 can be requested to the author directly [1].

channel impulse response (CIR) associated to the link between each pair of nodes. Such CIR is then maintained constant throughout a network simulation run.

The Underwater Acoustic Networks (UAN) framework released with the ns3 simulator [8] provides PHY, Medium Access Control (MAC), as well as mobility and energy consumption models for Autonomous Underwater Vehicles (AUVs). The propagation model in UAN is based on link budget equations involving spreading loss and absorption loss [5]. A more advanced model, based on Bellhop [9], used to be available as well. However, the user was required to manually enter all needed environmental parameters, which made it impossible to support dynamical environments, where the nodes move or the environmental conditions change over time. In addition, no support to real transducer beam patterns is declared. The UAN PHY model based on Bellhop is currently not maintained in the ns3 release, and must be requested from its authors. Unlike the work in [9] and the PHY model of the ns3 UAN framework, WOSS automatically retrieves environmental features given a user-specified period of the year and the geographical location of the nodes, and updates the power-delay profile of the channel among any two nodes automatically, in the presence of mobility or time-varying environmental conditions. In addition, the empirical model used by UAN [5] is also available in WOSS as a fallback solution.

The work in [10] employs the CIRs provided by Bellhop to test different modulation schemes and receiver-side signal processing techniques, and provides insight into the performance of these techniques in several scenarios.

In [11], the Mime channel simulator is presented, where the data employed to generate CIRs is recorded during sea trials. This approach skips the intermediate step performed by Bellhop (which relates environmental features to the CIR) and has the further advantage of incorporating Doppler spread into the generated CIRs. On the other hand, its main disadvantage is that the simulator can only replicate what has been actually measured at sea.

III. DESERT UNDERWATER: A FRAMEWORK TO DESIGN, SIMULATE AND REALIZE TESTBEDS FOR UNDERWATER NETWORK PROTOCOLS

DESERT Underwater has been presented in [3] as a collection of libraries created to support the design and implementation of underwater network protocols. It has been released with the objective of distributing several protocols for underwater networking, while at the same time speeding up the transition from simulations to sea trials. The latter is achieved by integrating the commands required to communicate with real modems into specific interface modules, thus permitting the reuse of the same protocol code already written for simulations.

DESERT is based on the well established ns2 simulator and on its NS-MIRACLE extensions [2]. In particular, it follows the modular approach of the latter, which has been designed to simulate nodes whose logical architecture is as close as possible to what is typically found on actual devices. To this end, it provides many modules that implement protocols at all layers of the ISO/OSI protocol stack, most of which have also been tested in several sea trials. In addition, in order to simulate real underwater networks with high fidelity, DESERT Underwater comes with mobility models that reproduce realistic mobility patterns, and provides many examples that showcase DESERT's features, in part integrated with the functionalities offered by WOSS.

During the first year of use, DESERT has been successfully tested with several modems (e.g., EvoLogics S2C and White line modems [12], WHOI microModems, Develogic modems) for the real-life implementation of underwater protocols, in both static and mobile networks. Moreover, DESERT has been successfully used not only on desktop PCs and laptops, but also on embedded systems such as the Gumstix, Pandaboard, IGEPv2, NetDCU, RaspberryPi and UDOO platforms, making it possible to realize fully distributed and low-power testbeds. Altogether, these features make DESERT an effective solution for realizing experiments by reusing the same protocol code already written for simulations.

DESERT Underwater has been adopted in several research projects (e.g., see the Related Work

and Acknowledgement pages in the DESERT web site [3]) and received a positive consideration from the underwater research community. However, the size of the source code and initial work required to install ns2 and the different software on which DESERT depends can still be perceived as an obstacle by a first-time user. Given this, we have been working on different aspects of its implementation and we have extended its functionalities, thanks to the knowledge acquired after one year and a half of experience with our framework. In the new release, we overcome the limitations of the previous version and we propose a new set of features that make the software more robust and user-friendly. In particular, a considerable amount of work has been done in order to make the installation of the software and its dependencies automatic, so that it does not require a lot of effort. At the same time, we have efficiently organized the installation procedure in separate modules, so that experienced users can easily extend it to accommodate their specific requirements. Following this approach, we also provide different installer modules, suitable for different hardware architectures (e.g., the Gumstix, RaspberryPi and UDOO platforms cited above). Thus, one of the primary objectives of DESERT Underwater, i.e., using the same code both in simulation and in real underwater networks, has been extended to a broad range of devices. Moreover, we revised the modules of DESERT v1 to accommodate the features of the new release, extended the set of network protocols released with DESERT, and improved the support to acoustic modem architectures.

Finally, the functionalities offered by DESERT have been extended with a remote control framework called RECORDS [13], that provides a set of primitives to remotely control the hardware modems, and thus the network operations.

The improvements and additional features that we implemented have been packed together into DESERT Underwater v2, the next public release of our framework that is available for download at the DESERT Underwater web site [3].

In what follows, we focus on the description of the new control framework for acoustic modems and present two experiments where DESERT proved itself as a viable solution to move

from simulation to real life experimentation.

A. RECORDS: a Remote Control Framework for Underwater Networking Experiments

Since the earliest development of DESERT, while working with deployed underwater networks we experienced the need for a control framework to easily interact with the modems, and to provide a backup communication mechanism in case of ns2 failures. To this end, a first proof-of-concept with limited capabilities (broadcast messages and ns2 start/kill) was implemented and successfully used in two sea trials, first during the test of the SUN protocol in Berlin in 2012 in collaboration with EvoLogics [14], and then during the CommsNet12 trials in La Spezia (Italy), in collaboration with the NATO STO Centre for Maritime Research and Experimentation (CMRE). Given these results, in conjunction with the second major release of DESERT Underwater we decided to re-design it and expand its capabilities, resulting in a new software called RECORDS, which is thoroughly described and evaluated in [13]. In what follows, we briefly summarize the idea behind its implementation.

The control framework is entirely written using scripting languages, namely the Tool Command Language (Tcl), Expect and the Bourne Shell, and is released under a BSD 3-clause license. It is composed of several independent modules that interact via TCP sockets, resulting in a portable and stable software, which requires negligible system resources.

The main functionalities offered by the control framework are strongly related to DESERT and remotely allow to:

- Start ns2, with the selection of the script to run and its input parameters;
- Get the status of ns2, with detailed indication about the running instances of ns2 or about the state of a specific run (i.e., running, completed or never started);
- Stop ns2, with the selection of a specific instance or all of them.

Given the broadcast nature of the underwater channel, it is possible to specify the target of the control message, from the complete network to a single node.

In addition to the DESERT-specific tasks, the control framework can also send native commands to the modem either directly or remotely, set several parameters of the nodes (such as the source level, id, system clock, internal framework parameters, enable/disable the error control mechanism), check the status of the nodes (battery level, free space on the storage devices, temperature, CPU consumption, etc.), read local and remote log files, start/stop a random traffic generator for testing purposes (either at boot time or on demand) as well as reset, reboot and turn off the modem.

The control framework can be started directly at boot time and can be configured to be permanent, i.e., to be automatically restarted in case of errors (socket failures, etc.). By combining all these functionalities, it is possible to use the control framework to send text messages between the nodes, thus realizing a chat service between different users: this service can live in parallel to DESERT experiments by sharing the same acoustic channel. Moreover, the messages can be sent to the destination either directly or through multiple hops.

The control framework provides additional functionalities that make it possible to simulate a given network topology by forcing specific packet error rates among selected nodes, and can set independent packet error rate values for each link. These parameters can be set before the simulation and changed in real time. It is also possible to randomly delay the transmission of control packets and to adapt this time interval dynamically.

To deliver remote messages, the framework relies on two algorithms that can be selected by the user. For each message sent, it is possible to choose between a static source routing protocol and a flooding mechanism. Different addressing modes are supported, so that each packet can be sent either in broadcast to all nodes or to a specific subset of destinations. The amount of traffic in the network is limited via several mechanisms: a time-to-live for the flooding scheme, a user-defined path for static routing, and duplicate packet detection and rejection for both of them.

Finally, the control framework offers a module that simulates the behavior of a user. This

module can be employed for several purposes, such as the setting of boot-time modem parameters and the reset of buffers before an experiment is started.

B. Example of Experiment

In this section we provide further insight on the effectiveness of the DESERT Underwater libraries for the field experimentation of underwater protocols. We focus on two experiments carried out during the CommsNet13 trials, organized by CMRE in the gulf of La Spezia, Italy, in September 2013. The first experiment tests a controlled access protocol named Uw-Polling [15] under different data packet generation rates, whereas the second experiment involves MSUN, a multihop forwarding protocol based on source routing [7]. The typical complete deployment topology for this experiment is shown in Fig. 4, and involves bottom-mounted nodes, a gateway buoy, and several autonomous systems.

For the Uw-Polling experiment, we set the gateway buoy to be the sink, and M1–M4 to be sensor nodes, which generate packets at a rate of 0.75, 1, 1.5 or 2 pkt/min/node, depending on the specific experiment. Fig. 5(a), reports the throughput per node (in pkt/min) as a function of the packet generation rate per node. The histogram shows that the throughput is almost equal to the packet generation rate per node for the two lowest traffic values, implying a high packet delivery ratio. However, after achieving its maximum for an intermediate packet generation rate per node, the throughput decreases. The post-processing of the experiment logs reveals that this is due to a number of packets remaining in the queues of the nodes at the end of the experiment, and effectively counted as lost. The cause is that the packets have to wait longer in the queue of the nodes as the packet generation rate increases, as a consequence of the controlled access mechanism. This fact is shown in Fig. 5(b).

We now consider a multihop routing experiment, which was carried out during the night between Sep 15 and Sep 16, 2013. This experiment involved 5 nodes, and lasted for nine hours of uninterrupted operations. Two Folaga AUVs were deployed near the sea bottom at about the

locations denoted by the white marks in Fig. 4, and generated packets at a rate of 1 pkt/min/node. The LOON node M3 was the sink; nodes M1 and M4 acted as relays and did not generate packets. We remark that the Folagas and the sink do not hear one another, hence the packet delivery must happen through one of the two relays, typically the closest one. During the nine hours of the experiment, about 1000 packets were generated, leading to about 3000 transmissions, including relaying and retransmissions. The packet delivery ratio achieved in this experiment was around 0.6, for an average throughput of about 1.24 pkt/min. Before the end of the experiment, the modem control framework was used to query the status of all nodes, which confirmed that the experiment was still running. This confirms the robustness of the DESERT Underwater libraries and of the control framework itself.

C. Summary of DESERT Underwater

DESERT Underwater is a useful research tool to develop, test and analyze real world applications for underwater communications. Based on the well known network simulators ns2 and NS-MIRACLE, it provides a comprehensive set of algorithms and tools to simulate various aspects of a general underwater network. Moreover, thanks to specialized interfaces with the modems, it allows the user to employ the same code both for the simulation and for the field experimentation of an underwater network. Without this structure, it would be necessary to replicate the network protocol code into a specific software package to be run by underwater communications hardware, effectively doubling the development effort and slowing down the capability to perform a sea trial after completing network protocol simulations.

The first release of the DESERT Underwater libraries has been presented in 2012 and, since then, has received positive consideration from the research community and has been used in several research projects. New features and extensions have been developed during the last year and have been made part of DESERT Underwater v2, which has been released recently and can be downloaded from the web site in [3]. Along with DESERT Underwater, we also released the

modem control framework RECORDS [13]. As explained in Sections III-A and III-B, RECORDS is key to several functions, such as checking that the nodes are idle and ready before starting an experiment, checking the quality of the links, and starting a given experiment on all nodes. All these functions can be performed remotely, by transmitting the commands via acoustic communications from a radio-controlled node. Without the framework, these tasks would require a direct cable or radio link to all underwater nodes.

D. Related work

Hardware-in-the-loop experiments reusing code programmed for simulations constitute a well known concept, and modern network simulators such as ns3 [8] were designed to also help their users through this task. In the context of underwater communications, one of the first systems reusing simulation code into real experiments is Aqua-TUNE [16], a testbed partially designed around the code of the University of Connecticut's Aqua-Net system, which also provides a simulation mode via a recent addition called Aqua-Net Mate. In the first implementation of Aqua-TUNE, the network was composed of radio-controlled kayaks, later evolved into buoys. The micro-controller installed in the nodes employs an embedded Linux distribution to run the network protocol code and to control the nodes via an external radio link.

The SUNSET framework [17] was developed starting in 2010, and first released in May 2012. Broadly speaking, SUNSET and DESERT Underwater have one similar functional objective, i.e., to help users operate the transition from simulation to experimentation with real hardware. However, the two frameworks are different in several respects. For example, DESERT Underwater comes with a wide set of protocols and network modules, some ready for experimentation, some meant mainly for simulations; comparatively fewer protocols were released with SUNSET, which mainly focused on the support for emulation and experimentation. In any event, both DESERT and SUNSET are interoperable with NS-MIRACLE, as both stem from it. SUNSET implements a more efficient real-time scheduler which reduces the RAM and CPU usage with respect to

the one provided with ns2, and proves useful on devices with limited processing capabilities; on the other hand, this requires the general NS-MIRACLE user to explicitly call SUNSET's real-time scheduler when interfacing the code to underwater modems; DESERT hinges on ns2's real-time scheduler, which may be less efficient but requires no change to the scheduler calls in the protocol code, when moving from simulation to field experimentation. Both DESERT and SUNSET rely on NS-MIRACLE's native cross-layer messages for exchanging information among the layers of the network protocol stack. Starting from version 2 (June 2013), SUNSET provides support for publish-subscribe interactions. DESERT v1's functions for converting NS-MIRACLE packet structures into actual bit streams and vice-versa are embedded within the modem interface, and are thus less modular than SUNSET's Packet Converters. DESERT v2's version of the same functions are inspired to the Packet Converters, and are complemented by a completely new Adaptation Layer module [7], which automatically fragments and re-assembles the packets whenever the PHY Service Data Unit (PSDU) is shorter than the packet length. Finally, our distributed modem control framework RECORDS has been explicitly designed as a stand-alone module released along with DESERT, in order to create a simple, lightweight module that does not hinge on ns2. Its SUNSET counterpart called the back-seat driver, instead, runs a SUNSET instance based on ns2 [18] and employs the same protocols implemented in SUNSET to deliver remote commands. The back-seat driver has not been publicly released yet.

A more recent implementation of an underwater acoustic testbed has been performed by the University of Buffalo [19], with system flexibility and modularity in mind. The testbed is reconfigurable from the network stack down to the transmit waveform, and the authors plan to make it accessible to the underwater community at large. The testbed also supports laboratory experiments via a channel emulator employing Matlab processing to reproduce the behavior of an acoustic channel.

In Oct. 2013, the National University of Singapore's Applied Research Lab and the SubNero company released v1.1.1 of UNetStack [20], a Java/Groovy implementation of an underwater

networking stack based on the agent-oriented programming (AOP) paradigm provided by the open-source fjåge framework. Roughly speaking, UNetStack's agents are equivalent to network layers, with the difference that agents are not forced into any hierarchy. A UNetStack simulator makes it possible to test the developed software and agents. UnetStack-compatible software-defined modems embed a version of this simulator, so that the simulated software can be used directly on the modems. No specific compatibility with other commercial modems is mentioned: in this respect, DESERT v2 takes a different approach, as it can be automatically cross-compiled for use on the embedded computers typically integrated within underwater modems, or externally connected to them. Another difference between UNetStack and DESERT is that the former allows the user to program and test different modulation and coding schemes at the physical layer, whereas the latter focuses on the network protocol stack.

IV. Concluding remarks

In this paper, we presented two open-source frameworks, WOSS and DESERT Underwater, that respectively address the need to bring realistic channel impulse responses into underwater network simulators, and the need for an efficient transition from simulations to underwater networking experiments. Both frameworks are based on the widely known ns2 and NS-MIRACLE simulation engines, which are extended i) to provide a realistic characterization of the acoustic channel via WOSS, and ii) to increase NS-MIRACLE's protocol count with many underwater network protocols via DESERT Underwater, which also supports the porting of protocol simulation code into real experiments. A framework that remotely controls the execution of the experiments via acoustic commands complements DESERT Underwater in the field.

Our software has been widely tested in collaboration with major institutions and modem manufacturers in the field, making up a complete solution for underwater simulations, laboratory activities and at-sea experiments.

ACKNOWLEDGMENTS

This work was supported in part by the European Commission under the 7th Framework Programme (Grant Agreement 258359 – CLAM) and by the Italian Institute of Technology within the Project SEED framework (NAUTILUS project).

The authors would like to thank Loris Brolo for developing U-Fetch, a cross-layer MAC/routing protocol which is part of DESERT Underwater v2.

Some features of DESERT Underwater and WOSS have been added thanks to several discussions with the partners of the RACUN project (especially Roald Otnes, Paul van Walree and Michael Goetz) and with the partners of the CLAM project (especially Arne Lie, Roberto Petroccia and Daniele Spaccini). The MSUN protocol was developed in the context of the multinational EDA RACUN project and experimented during CommsNet13 with the permission of the RACUN consortium.

Special thanks go to the NATO STO CMRE, La Spezia, Italy, for the organization of the CommsNet12 and CommsNet13 trials and for their invitation to participate.

REFERENCES

[1] F. Guerra, P. Casari, and M. Zorzi, "World Ocean Simulation System (WOSS): a simulation tool for underwater networks with realistic propagation modeling," in *Proc. of ACM WUWNet 2009*, Berkeley, CA, Nov. 2009, WOSS is available online at the URL http://telecom.dei.unipd.it/ns/woss/.

[2] N. Baldo, M. Miozzo, F. Guerra, M. Rossi, and M. Zorzi, "Miracle: the multi-interface cross-layer extension of ns2," *EURASIP J. of Wireless Commun. and Networking*, 2010. [Online]. Available: http://www.hindawi.com/journals/wcn/aip.761792.html

[3] 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. of IEEE/OES OCEANS*, Yeosu, Korea, May 2012, DESERT Underwater is available at the URL http://nautilus.dei.unipd.it/desert-underwater.

[4] M. Porter *et al.*, "Bellhop code," Last time accessed: May 2014. [Online]. Available: http://oalib.hlsresearch.com/Rays/index.html

[5] R. Urick, *Principles of Underwater Sound*. New York: McGraw-Hill, 1983.

[6] M. Rossi, L. Badia, N. Bui, and M. Zorzi, "On group mobility patterns and their exploitation to logically aggregate terminals in wireless networks," in *IEEE VTC Fall*, Dallas, TX, US, September 2005.

[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] ns3 Network Simulator, http://www.nsnam.org/, Last time accessed: May 2014.

[9] N. Parrish, L. Tracy, S. Roy, P. Arabshahi, and W. Fox, "System design considerations for undersea networks: link and multiple access protocols," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 9, pp. 1720–1730, Dec. 2008.

[10] L. M. Wolff, E. Szczepanski, and S. Badri-Höher, "Acoustic underwater channel and network simulator," in *Proc. MTS/IEEE Oceans*, Yeosu, South Korea, May 2012.

[11] R. Otnes, P. van Walree, and T. Jenserud, "Validation of direct and stochastic replay using the Mime acoustic communication channel simulator," in *Proc. UComms*, Sestri Levante, Italy, Sep. 2012.

[12] EvoLogics GmbH, "Underwater SC2R acoustic modem series," Last time accessed: May 2014. [Online]. Available: http://www.evologics.de/en/products/acoustics/index.html

[13] 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.

[14] G. Toso, R. Masiero, P. Casari, O. Kebkal, M. Komar, and M. Zorzi, "Field experiments for dynamic source routing: S2C evologics modems run the SUN protocol using the DESERT Underwater libraries," in *Proc. MTS/IEEE Oceans*, Hampton Roads, VA, Oct. 2012.

[15] 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, Korea, May 2012.

[16] Z. Peng, S. Le, M. Zuba, H. Mo, Y. Zhu, L. Pu, J. Liu, and J. Cui, "Aqua-TUNE: A testbed for underwater networks," in *Proc. IEEE/OES Oceans*, Santander, Spain, Jun. 2011.

[17] 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.

[18] R. Petroccia and D. Spaccini, "Implementing a back-seat driver to remotely control the experiments in an underwater acoustic sensor network," in *Proc. MTS/IEEE OCEANS*, Bergen, Norway, Jun. 2013.

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

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

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

Paolo Casari [SM'13] (casarip@dei.unipd.it) received a Ph.D. in information engineering in 2008 from the University of Padova, where he is currently a postdoctoral research fellow. He has been actively researching cross-layer design for MIMO ad hoc networks and wireless sensor networks (WSNs). After a stay at the Massachusetts Institute of Technology in 2007, he started working on underwater acoustic networks, which is now his main research interest. He is currently involved in several projects related to underwater networking, and was technical manager of the Italian WISE-WAI and NAUTILUS projects. He also collaborates with Consorzio Ferrara Ricerche (CFR) as a research fellow. He has been part of the TPC of several international conferences, and has been guest editor for the Hindawi Journal of Electronics and Computer Engineering Special Issue on Underwater Communications and Networking.

Cristiano Tapparello [M'12] (cristiano.tapparello@gmail.com) received the M.Sc. degree (with honors) in Computer Engineering and the Ph.D. degree in Information Engineering from the University of Padova, Italy, in 2008 and 2012, respectively. In 2011, he visited the Center for Wireless Communication and Signal Processing Research (CWCSPR) at the New Jersey Institute of Technology (NJIT), Newark, NJ, where he performed research on the design of networking protocols for energy-harvesting wireless networks. From Jan. 2012 to Oct. 2013 he has been a Postdoctoral Researcher at the Department of Information Engineering at the University of Padova. He is currently a Postdoctoral Research Associate in the Wireless Communications and Networking Group (WCNG) in the Department of Electrical and Computer Engineering at the University of Rochester, NY. His research interests include stochastic modeling and optimization of wireless systems, energy scavenging solutions for wireless sensor networks, and the design and implementation of mobile cloud computing systems and practical algorithms for constrained embedded systems.

Federico Guerra (federico@guerra-tlc.com) received the Laurea Specialistica degree (M.E.)

in Telecommunications Engineering in 2008 from the University of Padova, Italy. Shortly thereafter, he joined the research team of Consorzio Ferrara Ricerche (CFR) under the direction of Professor M. Zorzi. During this time, he was involved in the design of software for the simulation and performance evaluation of underwater acoustic networks. He is one of the developers of the MIRACLE libraries, a well known extension that brind cross-layer capabilities multi-interface simulation capabilities into NS2. He is the developer and maintainer of the World Ocean Simulation System (WOSS). He was involved in the CLAM and the NAUTILUS projects and in collaborations with the NATO CMRE related to MAC analysis and protocol design. In March 2011 he joined u-blox, a leading fabless semiconductor provider of embedded positioning and wireless communication solutions, as Software Engineer. Since May 2011 he has also collaborated with CFR as a consultant on underwater acoustic network simulation topics.

Federico Favaro (favarofe@dei.unipd.it) received both the bachelor and master degree in telecommunications engineering in 2008 and 2011, respectively. Shortly thereafter, he joined the Department of Information Engineering (DEI) of the University of Padova as a research engineer, and worked on several aspects related to the simulation and experimentation of underwater networking protocols. His main research interests encompass the implementation and testing of Medium Access Control protocols and software interfaces between simulation software and actual modems, which he performed and employed in the context of several projects and experimental trials at sea.

Ivano Calabrese (icalabre@dei.unipd.it) received the Bachelor and Masters Degree in Telecommunication Engineering from the University of Padova, Italy, respectively in 2009 and 2012. Since 2012 he works as a research engineer and software developer for Consorzio Ferrara Ricerche, Italy. In 2014 January he started a collaboration with Patavina Technologies s.r.l. (Italy) within the AllSeen Alliance consortium (LINUX FOUNDATION collaborative Project). He

collaborated actively in several research projects such as RACUN (European Defence Agency), NAUTILUS (IIT Project SEED program) and took part in several experimental campaigns on underwater communication sponsored by the NATO CMRE (La Spezia, Italy).

Giovanni Toso (tosogiov@dei.unipd.it) received a Bachelor and Masters Degree in Computing Engineering from the University of Padova, Italy, respectively in 2009 and 2011. In 2012, he worked as a research engineer and software developer for Consorzio Ferrara Ricerche, Italy. In 2013, he joined the School of Information Engineering of the University of Padova, where he is currently a Ph.D. student. His research interests cover the design, analysis, evaluation and real-world experimentation of protocols for underwater acoustic networks. He collaborated actively in several projects related to underwater networks, such as CLAM (EU FP7), RACUN and the Italian NAUTILUS project.

Saiful Azad (sazadm684@gmail.com) received his B.Sc. in Computer and Information Technology at IUT, Bangladesh, his M.Sc. in Computer and Information Engineering at IIUM, Malaysia, and his PhD in Information Engineering from the University of Padova, Italy, in 2013. After the completion of his PhD, he joined the Department of Computer Science at the American International University–Bangladesh (AIUB) as a faculty member. His work on underwater acoustic networks started during his PhD program and is still his main research focus. His interests also include the design and implementation of communication protocols for different network architectures, QoS issues, network security, and simulation software design.

Riccardo Masiero (masieror@dei.unipd.it) received both his Bachelor degree in Information Engineering and his Masters degree in Telecommunication Engineering from the University of Padova (Italy) in 2005 and 2007, respectively. In April 2011, he completed the PhD program in Information Engineering, also at the University of Padova. During the PhD, his research activity

has been focused on distributed techniques for data collection in Wireless Sensor Networks (WSNs). In 2010, he carried out a six-month research activity at INRIA, Sophia Antipolis (France) as a visiting PhD student within the MAESTRO team. During that period he focused his activity on distributed optimization techniques for Delay Tolerant Networks (DTNs). As a post-doc in Padova since 2011, his research activity focused on underwater networking, in the context of the the NAUTILUS and RACUN projects. He is one of the project leaders and developer of the DESERT Underwater libraries.

Michele Zorzi [F'07] (zorzi@dei.unipd.it) received his Laurea and Ph.D. degrees in electrical engineering from the University of Padova, Italy, in 1990 and 1994, respectively. During academic year 1992-1993, he was on leave at the University of California, San Diego (UCSD). After being affiliated with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy, the Center for Wireless Communications at UCSD, and the University of Ferrara, in November 2003 he joined the faculty of the Information Engineering Department of the University of Padova, where he is a professor. His present research interests include performance evaluation in mobile communications systems, random access in mobile radio networks, ad hoc and sensor networks, energy constrained communications protocols, and underwater communications and networking. He was Editor-in-Chief of IEEE Wireless Communications from 2003 to 2005, Editor-in-Chief of IEEE Transactions on Communications from 2008 to 2011, and Guest Editor for several Special Issues in IEEE Personal Communications, IEEE Wireless Communications, IEEE Network, and IEEE JSAC. He served as a Member-at-Large of the Board of Governors of the IEEE Communications Society from 2009 to 2011, and is currently its Director of Education.

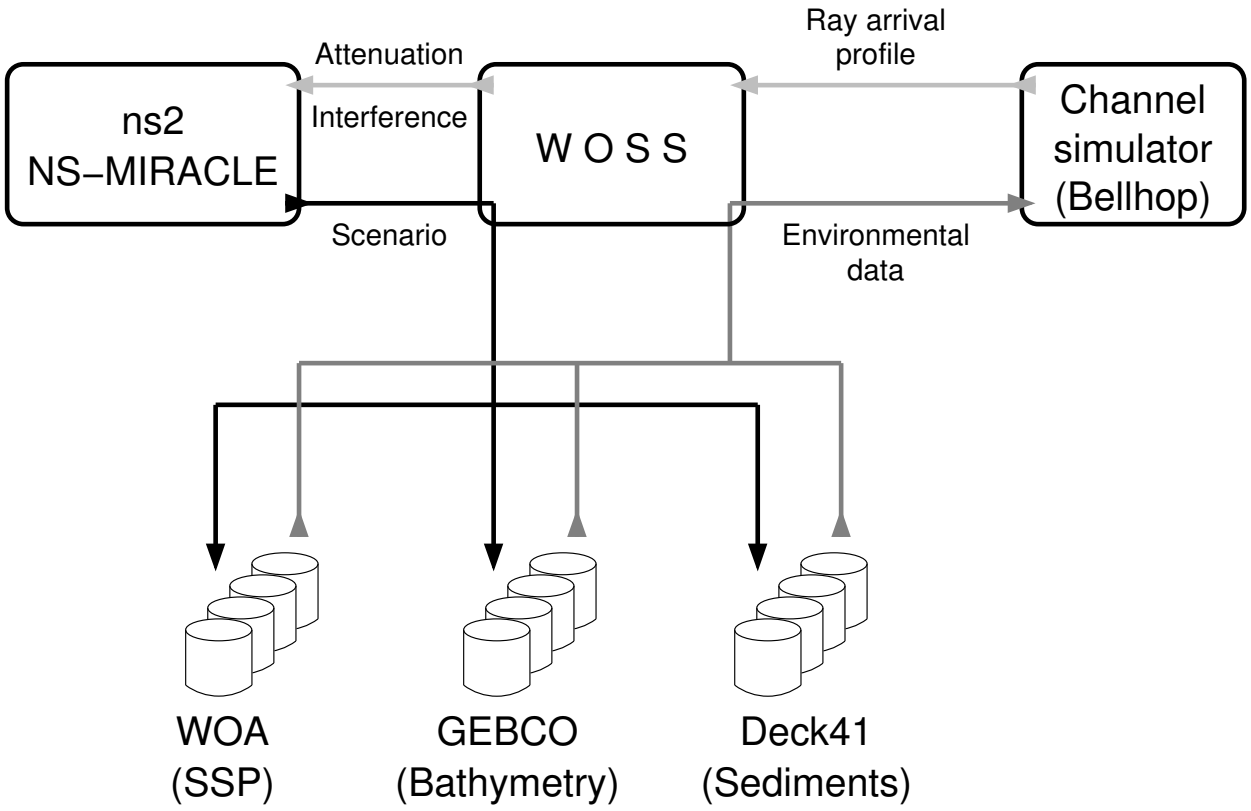


Figure 1. Schematic representation of the data flow between ns2/NS-MIRACLE, WOSS, the oceanographic databases and Bellhop: the network simulator provides scenario information (e.g., the geographical position of the nodes, the season of the year, etc.) to WOSS, which employs it to query oceanographic databases for environmental data. Such data is passed to the channel simulator (in this case, Bellhop), which returns a simulated channel response to WOSS. After some post-processing, WOSS provides information on channel attenuation, self-interference and multiple-access interference to the network simulator.

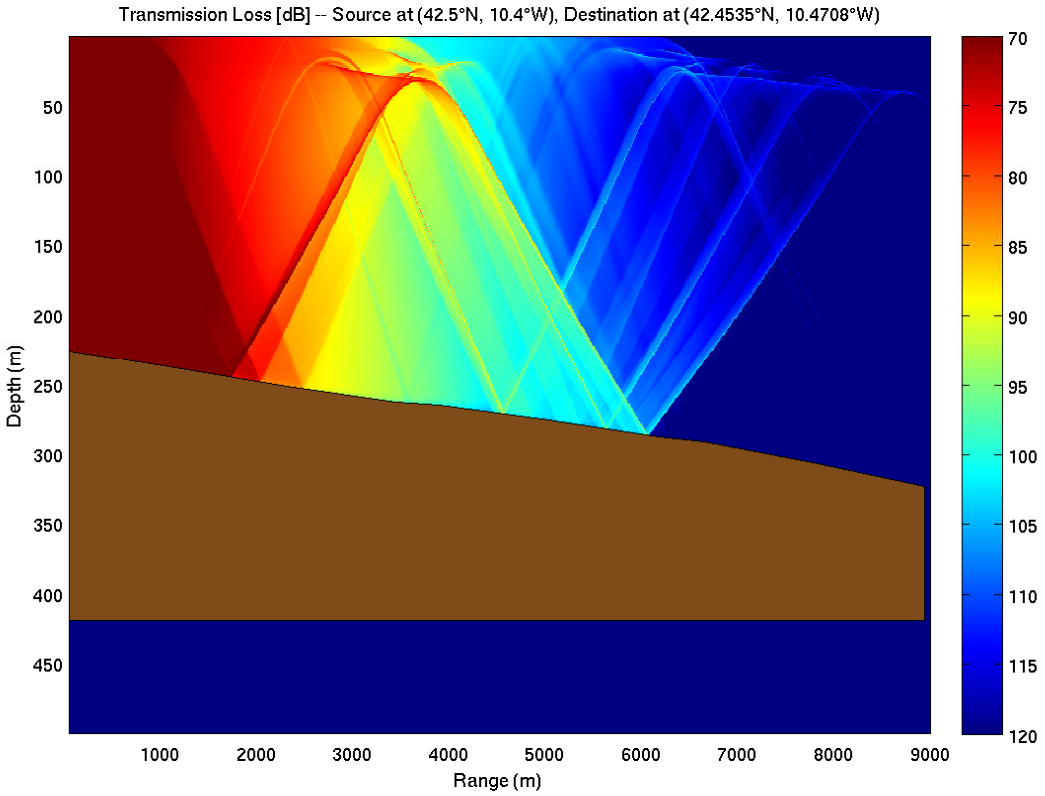


Figure 2. Example of attenuation computed by Bellhop. In this particular case, WOSS replicated Bellhop's "incoherent" option, which derives the attenuation from the sum of the powers of all complex arrivals. The simulation environment reproduces summer conditions in north Tyrrhenian waters, west of Italy. The frequency of the transmitted signal is 25 kHz. (Best viewed in color.)

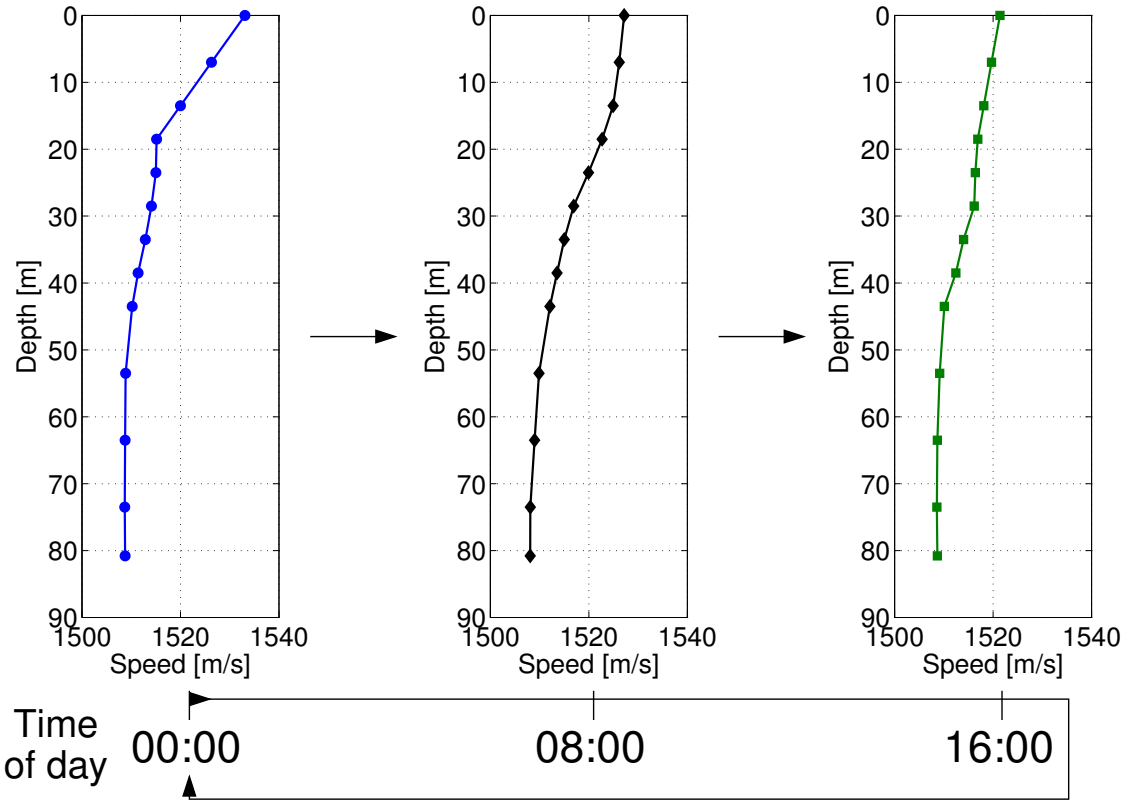
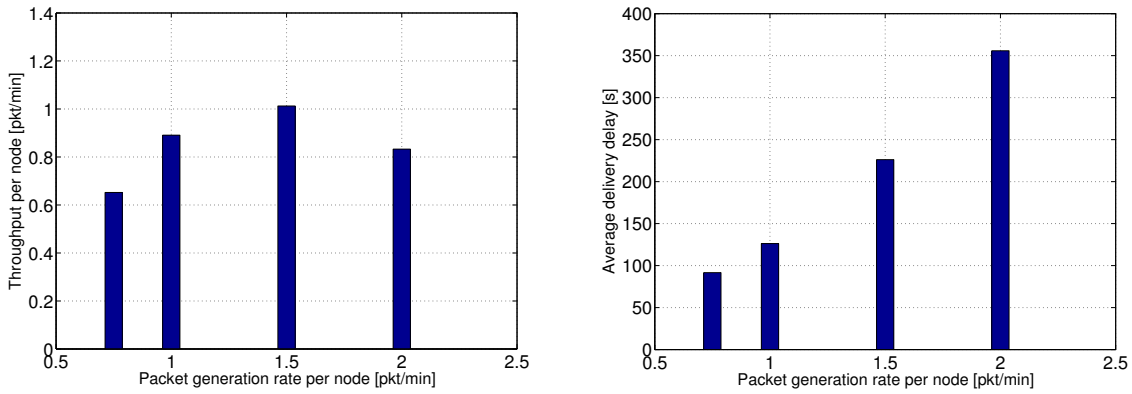


Figure 3. Example of SSP evolution over time in WOSS. In this case, the user set three different SSPs, each to be applied for a total simulated period of 8 hours. At the end of the last period, WOSS automatically wraps around and re-applies the first SSP, i.e., the same set of SSPs is repeated for each simulated day.

Figure 4. Typical topology of a complete network deployment during the CommsNet13 campaign carried out in La Spezia, Italy, in September 2013. The yellow pins denote the operational area. Nodes M1–M4 are bottom mounted nodes forming the so-called Littoral Ocean Observatory Network (LOON). The gateway buoy is a moored node mounting an acoustic modem and accessible via a radio interface. The wave glider is an autonomous system which harvests wave energy for propulsion. Folagas are GraalTech's low-cost, torpedo-shaped AUVs. Mantas are University of Porto's portable acoustic/radio gateway nodes. The ship node is in fact also a Manta, connected to an underwater modem hanging off the RV Alliance. (Best viewed in color.)



(a) (b)

Figure 5. Throughput (a) and packet delivery delay (b) for a set of experiments involving the Uw-Polling protocol [21]. Each value of the packet generation rate corresponds to a different experimental run. The experiments involved the LOON nodes M1–M4 as packet generators, and the gateway buoy as the sink (see Fig. 4).