# A Simulation Framework for Smart Adaptive Long- and Short-range Acoustic Networks

Filippo Campagnaro[‡⋆], Alberto Signori[‡], Roald Otnes[$], Michael Goetz[♯], Dimitri Sotnik[♯], Arwid Komulainen[*], Ivor Nissen[+], Federico Favaro[‡], Federico Guerra[‡], Michele Zorzi[‡⋆§]

[‡] Department of Information Engineering, University of Padova, via Gradenigo 6/B, 35131 Padova, Italy
[$] Norwegian Defence Research Establishment (FFI), 3191 Horten, Norway
[♯] Fraunhofer Institute for Communication, Information Processing and Ergonomics (FKIE), 53343 Wachtberg, Germany
[*] Swedish Defence Research Agency (FOI), Linkoping, Sweden
[+] Bundeswehr Technical Center for Ships and Naval Weapons Maritime Technology and Research, Klausdorfer Weg 2–24, 24118 Kiel, Germany
[§] Consorzio Futuro in Ricerca, via Saragat 1, 44122 Ferrara, Italy
[⋆] Wireless and More srl, Via della Croce Rossa 112, 35129 Padova, Italy
[‡]{signoria,campagn1,zorzi}@dei.unipd.it, fedefava86@gmail.com, federico@guerra-tlc.com
[$]Roald.Otnes@ffi.no, [*]arwid.komulainen@foi.se, [+]ivornissen@bundeswehr.org
[♯]{michael.goetz, dimitri.sotnik}@fkie.fraunhofer.de

***Abstract*—Underwater acoustic networks are characterized by long propagation delays, low data rates and a strong channel variability, that can cause the disruption of acoustic links. For this reason, the underwater network envisioned in the EDA SALSA project adapts its network and physical layer parameters, such as routes to destination, modulation and coding scheme and transmission frequency band, to the observed acoustic channel. Before the actual deployment, the network is analyzed and evaluated via simulations, interfacing the real-world GUWMANET network protocol implementation into the DESERT Underwater simulator, and developing a new time-varying physical layer where channel metrics obtained during a field-test campaign are mapped in the simulator in the form of lookup tables. The results demonstrate how the network performance changes when considering a different channel realization, confirming the importance of considering a time-varying channel when developing communication protocols for underwater acoustic networks.**

## I. Introduction and Related Works

A limiting factor for Underwater Acoustic Networks (UANs) research is the high complexity of their deployment, due to the need for sophisticated facilities, such as well-equipped vessels and specialized crews, to safely maneuver and deploy unmanned vessels, bottom nodes, buoys and all other nodes that compose the underwater assets [1]. In addition, the hardware components of the underwater nodes are usually very expensive, as they are designed to be water- and high pressure-proof, to be resistant to salty water corrosion, and are often equipped with long endurance batteries. Clearly, any problem or failure after the actual deployment will increase the overall cost. Thus, it is crucial to study and test all hardware and software components of the network before the actual deployment to prevent future failures, including the software protocol stack used to coordinate the communication between the nodes. In this pre-deployment phase, extensive simulations of the network should be carried out, not only to tune and optimize the network parameters, but also to avoid possible future network failures and fix software bugs.

When performing network analysis, one of the most complicated and crucial parts is obtaining realistic results from the simulation framework. To reduce the gap between real-world field tests and simulations, the protocol stack employed for the simulations must match the actual protocol stack implementation used for deployment, and the channel model used to simulate the acoustic environments should accurately characterize the area where the network will be deployed. While the former problem can be overcome by reusing the same code of the protocol stack for both experimental evaluation and simulation [2], the latter is more difficult to address since statistical models are themselves approximations. Specifically, standardized and universally accepted models of the acoustic channel do not exist, since the channel characteristics strongly depend on the actual environment of the underwater network deployment [3], making it difficult to obtained generalized models. For this reason, a possible solution to overcome this problem is to move from statistical characterization of the physical channel towards a deterministic solution, either obtaining the channel characterization for the simulator employing a ray-tracer [4] (at the cost of a computational complexity increase), or basing the simulation results on channel metrics collected from real-field experiments [5], [6].

In the past years many simulators have been developed for UANs. SUNSET [7] is a network simulator based on ns2-MIRACLE [8] with the capabilities of reusing the same code for both simulation and experimental evaluation with real modems for data transmission. Recently, the new version of Aqua-Sim, called Next Generation (NG) [9], was developed. Aqua-Sim NG, based on ns3, maintains the same functionalities of the legacy ns2-based Aqua-Sim version, but obtaining an improvement in computational performance. Based on ns2-MIRACLE, the DESERT Underwater framework provides the

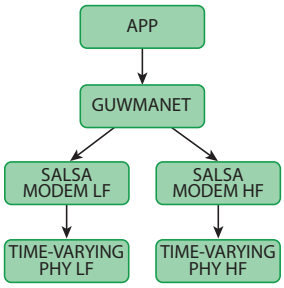



Fig. 1. Example of the protocol stack of the SALSA Framework with the time-varying module for the physical layer.

possibility to simulate and perform experiments with multi-modal underwater acoustic and optical networks [2]. Unet-Stack [10] is an agent-based software package that can be used to simulate underwater networks and then be used for real field experiments, porting the same network architecture into a modem with Unet-Stack support, such as the SUBNERO modem [11].

In this work we present the simulation framework developed for the European Defence Agency (EDA) Smart Adaptive Long-and Short-range Acoustic Networks (SALSA) project [12]. The framework combines the DESERT Underwater Network Simulator [2] and the Gossiping in Underwater Acoustic Mobile Ad-hoc Networks (GUWMANET) protocol [13], and enhances the DESERT simulator with the capabilities of a time-varying physical layer.

### A. SALSA Simulation Framework - General Overview

The SALSA simulation framework aims to provide a tool to simulate with high accuracy an underwater network before its actual deployment, testing all software components of the network protocols in order to verify their functionalities and fix all possible bugs. In addition, the simulator allows to test the network in those limit conditions that are not easy to verify in a real experiment, such as a large number of nodes, the presence of many mobile nodes traveling in the area, and the deployment of an intruder or a malicious node jamming the network or performing other Denial Of Service (DoS) attacks. The simulation stack of the dual-frequency acoustic network used in SALSA (depicted in Figure 1), consists of the Generic Underwater Application Language (GUWAL) application layer [13] that is directly connected with the GUWMANET network layer that works at both the network and the Medium Access Control (MAC) layers. The application layer allows the user to schedule the transmission of data and control packets to emulate the behavior of GUWAL. The application can be adapted to different scenarios according to the users' needs, and its description is out of the scope of this paper. GUWMANET is a standalone framework developed in C for sea experiments: its code has been wrapped into the DESERT Framework to simulate the protocol before the sea trial. This makes it possible to exploit the simulation capabilities of DESERT and, at the same time, use in the simulations the same code that will be used for the experimental tests, where GUWMANET runs on top of real modems. GUWMANET can handle multiple frequency bands and modem types (acoustic, optical and radio frequency modem): in the EDA SALSA project this ability is used to decide whether a packet should be transmitted through a Low Frequency (LF) modem using the 4-8 kHz band, or a High Frequency (HF) modem using the 24-32 kHz band, and to receive from both modems simultaneously.

The packets are transmitted between GUWMANET and the modems in the form of telegrams. In addition, each modem also receives commands from GUWMANET in the form of telegrams: these commands are intended to monitor and control the state of the physical layer, asking, for instance, for the instantaneous acoustic noise level, and configuring various parameters, such as the transmission source level and the coding and modulation scheme. The modem layer translates all telegrams received from GUWMANET and intended for the physical layer into DESERT Cross Layer messages that are sent synchronously to the physical layer to perform the required configuration.

In order to reduce the gap between the results obtained from the simulations and the real field experiments, and to simulate channel variability, a time-varying physical layer was developed. This new module employs channel metrics based on the measurements obtained during a field-test campaign, in the form of lookup tables (LUTs). The goal of the time-varying physical layer is to include the high variability of the underwater channel into the simulator, which can facilitate further studies into adaptive protocols. The physical layers can adapt its parameters according to the network layer requests.

The rest of the paper is organized as follows. In Section II we describe in detail the integration between the GUWMANET protocol and the DESERT simulator, and in Section III we present the full description of the time-varying physical layer. In Section IV we describe the simulation scenario and compare the performance of the network simulated with the time-varying physical layer with the same network simulated using the DESERT legacy physical layer. In Section V we finally draw our concluding remarks.

## II. GUWMANET Integration

During the RACUN [6] project (concluded in 2014), GUWMANET was integrated in the DESERT framework in the form of a specific DESERT add-on module, requiring to rewrite the entire protocol from C to C++ using the DESERT APIs. A change of the GUWMANET C code entailed a re-implementation of the DESERT C++ module, doubling the development work and making it difficult to keep the two frameworks aligned. Another disadvantage of this approach is the possible introduction of implementation errors when transferring the C code to DESERT. In addition, a lot of code adjustments were needed to synchronize the real-world implementation with the simulation module. For example, all real-world timers had to be replaced with DESERT C++ timer classes. In addition, the standalone GUWMANET code

is implemented in C and its structure could not run in multiple instances on the same device. For all these reasons, after the RACUN project, FKIE and WTD71 redesigned the complete framework with the simulation porting goal in mind. Indeed, the GUWMANET network protocol implemented in the DESERT framework during RACUN, is very different from the current GUWMANET implementation, as the standalone framework was completely reworked since 2014. First, the GUWAL application layer now includes many new packets to support several new applications, such as data muling and first contact. The new telegrams used in the SALSA project, instead, are extended to enable the interaction with the HF and LF modems, and to support different physical layer modulation and coding schemes (Frequency Repetition Spread Spectrum (FRSS) and JANUS) used in the project. The modulation and coding scheme profile is included in the so-called "modem packet header" together with the address of the node that transmitted that packet: this header is transmitted in the packet preamble in order to simplify the modem reception and let the neighbor nodes estimate the quality of the acoustic link of the transmitter[1].

Given the large number of changes in GUWMANET and the lessons learnt during RACUN, a different approach is used in the SALSA project to keep the core implementation of GUWMANET identical in both the code used for the simulations and the code used in the real world. In this section, the integration of GUWMANET into the DESERT framework is described in detail.

The following list summarizes the main issues addressed to integrate the real world standalone GUWMANET into the DESERT simulator.

- Programming language: the standalone version of GUWMANET is written in C, does not have an object oriented architecture, and uses global data structures to save the protocol state, the transmission queues and the neighbor list: with this design, only a single GUWMANET instance can run at a time. The DESERT simulator, instead, is written in C++. Each DESERT layer is represented by a module with well defined APIs to forward packets to the upper and lower layers and to handle cross-layer messages. This object-oriented architecture makes it possible to have multiple independent instances of each layer (e.g., one instance per simulated node). Hence, a way to run multiple GUWMANET instances is needed, as well as the possibility to interface GUWMANET with the DESERT APIs to talk with the adjacent layers.[2]
- Logging system: in the real-world GUWMANET, each node creates its own log file printing each log entry with the system timestamp. In simulations, all nodes print each log entry into the same file using the simulation time instead of the actual system timestamp. The use of a unique log makes it necessary to additionally print the node addresses in front of each log line.
- Timers: one of the main issues for the GUWMANET integration is the handling of timers. The real-world GUWMANET has its own timer classes that only work with the real system clock. The simulator instead uses its own event-based scheduler and the GUWMANET timers need to be adapted to use the simulation time.

In the remainder of this section, we describe the implementation details performed to solve the aforementioned issues.

### A. Interface and data structures

The first step to integrate the GUWMANET C implementation into DESERT is the development of a C++ wrapper class. This class is needed to interact with the higher and lower layers of DESERT and hence implements the APIs used in the simulator to receive, send up and send down the simulated packets. To this aim, we created a function that wraps each of the telegrams that carry a data packet into a DESERT packet before forwarding it. This function is declared as "extern C" to allow a C program to call this C++ function. In addition, also the DESERT command API is implemented inside the C++ wrapper: this function is used to configure the protocol parameters with a simulation script, written in TCL. The GUWMANET logic itself is kept in the original C files, which are untouched during the whole integration. Therefore, the only operation required to synchronize the simulation and the real-world implementation is copying the files from one to the other.

A GUWMANET wrapper object is created for each node: this wrapper enables the possibility to use an independent GUWMANET instance in each of the nodes, by storing all data structures and timers of a node separately. If a packet is received, the receive function of the wrapper of that node is called by the simulator. The function extracts the telegram from the DESERT packet and calls the original receive function of GUWMANET. The original GUWMANET uses a single global data structure that stores the state of the node, such as the timers and the transmission queues. The simulator, instead, runs multiple instances of the GUWMANET layers, and needs to use a different data structure for each node, that is not possible with the current C implementation. To solve this issue, the binary data containing the information stored in the data structure of each node is stored in a database, and when a packet is received or an event occurs at a certain node, the receive function loads from this database the information of the receiving node into the GUWMANET global data structure. This is possible as long as the simulator event scheduler processes the events in sequence and not in parallel. The use of pointers to load and maintain the stored data structures updated allows us to avoid copying big data structures and hence reduce the simulation time.

[1]The creation of the packet preamble and the link quality estimation operations performed by the physical layer are considered out of the scope of this paper and will not be discussed further.

[2]In fact in the real-world system GUWMANET is interfaced with the modems and the GUWAL application layer via serial connections and TCP sockets, thus it requires a specific interface to interact with the DESERT modules.

### B. Logging

The GUWMANET logging system had to be extended in order to enable all nodes to write their logging information in the same log file. Indeed, in the real-world GUWMANET, each node creates its own log file printing each log entry with the system timestamp. In simulations, instead, all nodes log into the same file storing the logging information with the simulated time. To accomplish this task, the logging information is extended by adding the node address in front of each log line. Therefore, the log library was adapted accordingly, removing the creation of a log file for each node: all logging information of all nodes is printed to standard output, together with the layer name, the node address and the simulation time. This library was written in C++ as it needs access to C++ classes to retrieve the current simulation time from the scheduler. The log print function itself was declared to be "extern C" and has the same interface as the original log library of the standalone implementation. The node name was included in the global data structure to let the log function know which GUWMANET instance issued the logging operation. This function is called every time a node receives a packet or when a timer expires.

### C. Timer

The real-time timers used in the real-world GUWMANET to schedule events were interfaced with the DESERT simulation time. To perform this task, in the SALSA simulation framework the timers are stored in a sorted list where the next expiring timer is always the one in front of the list. With this design it was possible to implement this interface using only one instance of a single DESERT C++ timer class in each node, while the handling of the timer list was kept identical to the real-world GUWMANET. The DESERT timer was placed inside the C++ GUWMANET interface class. After an event (e.g., the reception of a packet or the expiration of a timeout) is completely processed, the DESERT timer is rescheduled to the next item of the timer list. When the DESERT timer expires, the content of the node involved in the event is restored in the GUWMANET global variables, and the DESERT expire routine for that event is executed.

## III. Physical Layer

Many physical layer parameters can be adapted according to the GUWMANET indications. The physical layer can change modulation scheme, source level, data rate, and other parameters. The physical layer modulation schemes considered in SALSA are FRSS and JANUS.

FRSS [14] is a coherent modulation scheme that is known to perform well under a wide variety of underwater acoustic channel conditions [14]–[16]. FRSS comes with a choice of four different data rates, labeled FRSS1 (fastest, least robust) through FRSS4 (slowest, most robust). This flexibility will accomodate studies into an adaptive choice of the data rate.

JANUS [17] is a non-coherent modulation scheme standardized by NATO. Among other things, it is suitable for first contact applications between different modems.

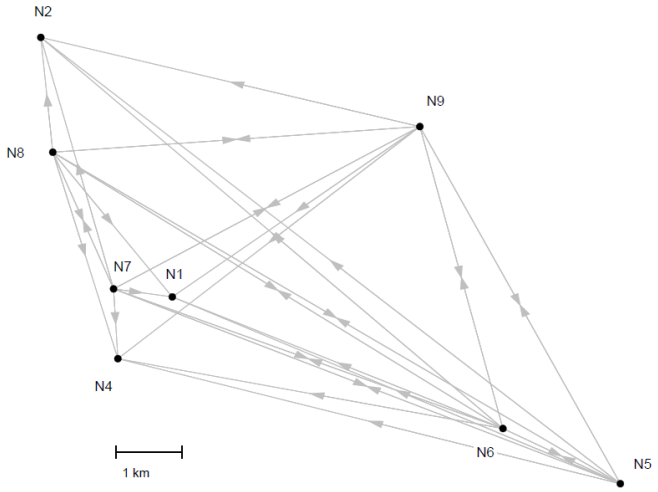



Fig. 2. Topology used to collect the data to be exploited in the time-varying physical layer. Figure from [15].

A different modulation does not only change the bit rate and the packet duration, but also the probability of correct reception. Given that the packet errors are mapped in the simulator in the form of LUTs, a different LUT is selected depending on the modulation used, as explained in Section III-A.

Finally, the physical layer provides the packet delay spread and the input and output Signal to Noise Ratio (SNR) of each received packet and, upon request, can notify GUWMANET of the perceived noise level. These parameters are first sent to the modem layer using the DESERT CrossLayer messages: the modem then converts these messages in GUWMANET telegrams and sends them to the network layer.

### A. Time Varying Channel

The time-varying physical layer employs a database to store the link quality metrics that are used for the simulations. To store, access and manage the entries of the database, the NetCDF set of libraries are employed [18]. The database contains the link quality metric for each of the links between the nodes depicted in Figure 2. Every time in the simulation when there is a transmission from node A to node B, the two nodes are associated to the nodes in the database closest to A's and B's positions, based on their geographical position. To obtain a time-varying link, the quality metrics are stored in the database for periodic time intervals. Then, the value for the quality metric associated to A and B is chosen based on the actual simulation time: the time value in the database closest to the simulation time is selected to be used in the simulation (with the database time wrapping around if the simulation time extends beyond the maximum database time). Eventually, the value of the link quality metric between the two nodes is employed to compute whether or not a packet transmitted by node A is received correctly by node B. The time-varying physical layer supports simulations with dual-band connectivity, as the channel quality metrics have been

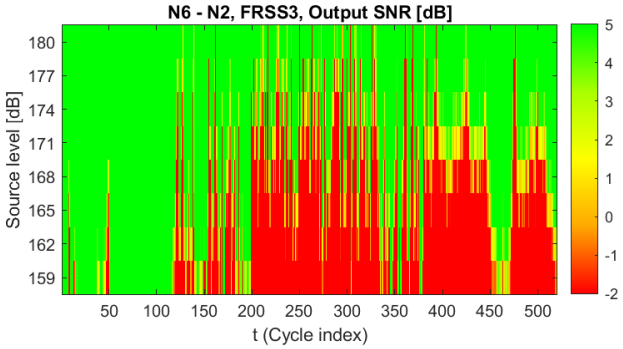



Fig. 3. Example of LUT content for one combination of TX node, RX node, and Technology, for the LF band. Cycle interval is 180 s.

obtained for both an LF band and an HF band. Based on the band employed in the simulation, the link quality search in the database will be performed in the proper set of channel metrics.

Figure 2 shows the nodes' deployment and location used to perform the time varying channel data acquisition campaign in Oslofjorden [15]. Note that drawbacks of the method include the sparse spatial sampling used for nearest-neighbor interpolation, and the influence of node depth which is not modelled.

### B. LUT details

The LUT data is first stored in Matlab .mat files, with a format building on what is used in the recently published ASUNA framework ("a shared underwater network emulation data set") [5], before being converted to netCDF database files which are imported in an ns2/DESERT module. This gives the possibility to later reuse our efforts to integrate ASUNA with the ns2/DESERT network framework.

Compared to the format described in [5, Sec. III-D], we have added the extra dimension of varying source level (with the resulting SNR reduction emulated in replay by adding scaled time-adjacent noise recordings). In the nomenclature of [5] we set $N = 9$ as the maximum number of nodes in the topology (index 3 was not used for any stationary node, and is here used for the moving node), $P = 5$ as the number of "Technologies" (FRSS1-4, and JANUS), $T$ as the number of cycles in the run, and $L$ as the number of emulated source levels. We prepared one .mat file for each frequency band, with the following contents:

- A `TopMat` matrix of size $T \times N \times N \times P \times L$, where each entry `TopMat(t,i,j,p,l)` contains the link quality, in terms of output SNR, for the link between nodes `i` and `j` through physical layer Technology `p` at cycle `t` and source level index `l`.
- A `TopMat_binary` matrix of size $T \times N \times N \times P \times L$, where each entry `TopMat_binary(t,i,j,p,l)` contains the link quality, in terms of success (no bit errors) or failure (one or more bit errors), for the link between nodes `i` and `j` through physical layer Technology `p` at cycle `t` and source level index `l`.

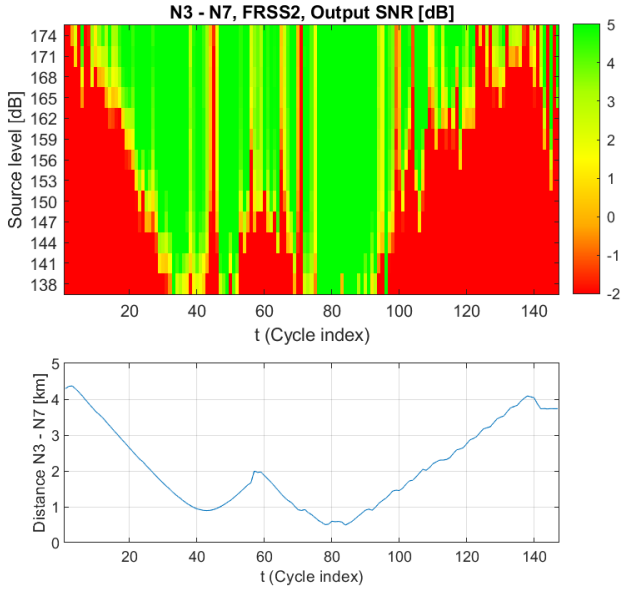



Fig. 4. Example of moving node LUT content for one combination of TX node, RX node, and Technology, for the HF band. Lower panel shows distance vs time. Cycle interval is 92 s.

- A `LocMat` matrix of size $T \times N \times 3$, where the three entries `LocMat(t,i,1:3)` represent the two UTM coordinates and the depth of node `i`, respectively, at cycle `t`.
- A `TechMat_TX` matrix of size $T \times N \times P$, where each of the $k = 1, \ldots, P$ entries `TechMat_TX(t,i,1:P)` is 1 if node `i` is able to transmit Technology $k$ at cycle `t`, and 0 otherwise.
- A `TechMat_RX` matrix of size $T \times N \times P$, where each of the $k = 1, \ldots, P$ entries `TechMat_RX(t,i,1:P)` is 1 if node `i` is able to receive Technology $k$ at cycle `t`, and 0 otherwise.
- An `AdjMat` matrix of size $T \times N \times N \times L$, where each entry `AdjMat(t,i,j,l)` is 1 if nodes `i` and `j` are linked by any Technology (as indicated in `TopMat_binary`) at time `t` and source level index `l`, and 0 otherwise.
- A scalar `delta_T` which holds the number of seconds between each cycle.
- An `SL` vector of length $L$ which holds the source level in dB re $\mu$Pa$^2$m$^2$ corresponding to each source level index `l`.
- A `Technologies` cell array of length $P$, holding the name of each Technology.

Fig. 3 shows an example extract of LUT content for FRSS. The JANUS receiver does not provide output SNR, hence we let the entries of `TopMat` for this Technology be +5 dB when there is successful reception, and –5 dB otherwise.

The DESERT interface into the lookup tables is a function call on the form getPER(Position* TxPos, Position* RxPos, double SourceLevel, signal_type_e sig_type, uint num_bits). As the lookup tables only cover a small set of node positions, the implementation will find the nearest available

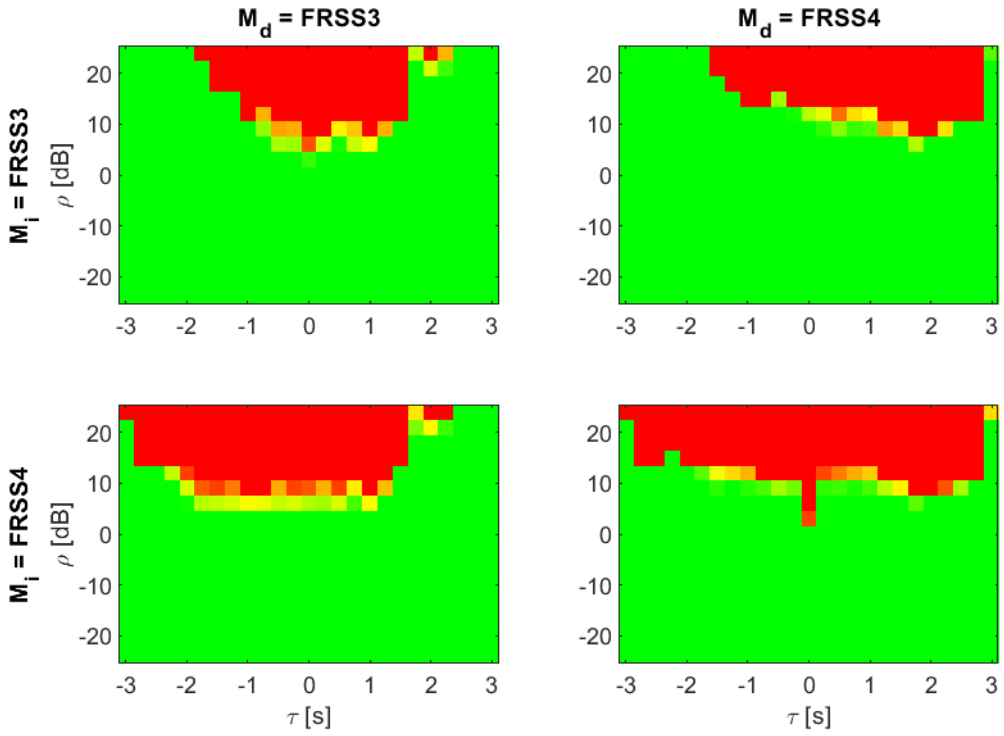



Fig. 5. Example of collision LUT content for one combination of TX node (N8), RX node (N1), and interfering node (N9). Only two desired ($M_d$) and interfering ($M_i$) Technologies are shown here. Color map is the same as in Figs. 3-4.

node position to each requested position (nearest neighbor approximation). If the smallest distance found is above a configurable threshold, a Packet Error Probability (PEP) of 1.0 is returned. In addition to the parameters provided in the interfaces, the current time "NOW" of the simulator is used for the stationary node LUTs, to access the correct time index in the database. Nearest neighbor approximation is also used for the source level, selecting the highest or lowest available value if the desired value is outside bounds. Finally, the output SNR values in the LUTs are, in combination with the number of bits, converted to PEP using empirical expressions (outside the scope of this paper) found through Additive White Gaussian Noise (AWGN) simulations of the error-correcting code used with FRSS.

### C. LUTs for mobile nodes

There are also LUTs for two mobile nodes that were present during part of the sea trial where the data was collected, transmitting to the stationary nodes in the LF and HF band, respectively. These are stored in a similar format as the data for stationary nodes. Fig. 4 shows an example of LUT content for a moving node. As this data is a function of time and space, the time variation is ignored and instead the LUT time when the actual moving node position was closest to the requested moving node position is used.

### D. Collision LUTs

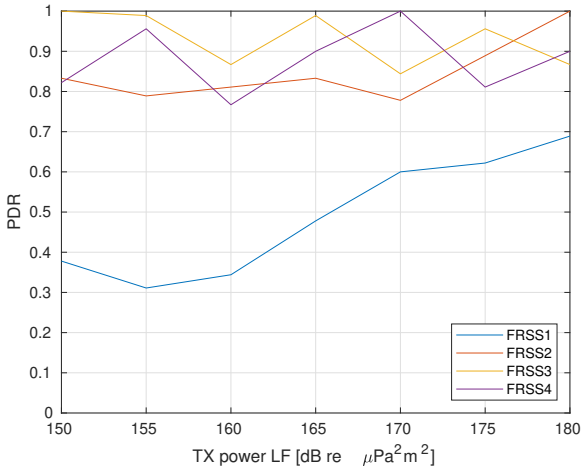
LUTs describing the effect of collisions under the channel conditions where also generated, by time-shifting and adding received signals so they overlap. These LUTs have dimensions $N \times N \times N \times P \times P \times R \times T$, where each entry `CollMat(Kd,Ki,Kr,Md,Mi,r,t)` contains the link quality, in terms of output SNR, for the link between nodes `Kd` and `Kr` through physical layer Technology `Md` when interfered by node `Ki` through physical layer technology `Mi`, at $\rho$ index `r` and $\tau$ index `t`. Here, $\rho$ represents one of $R$ different relative amplitudes between desired and interfering signal, and $\tau$ represents one of $T$ different time shifts between the signals. Fig. 5 shows an example of collision LUT content.

As the collision LUTs already have many dimensions, time variation is not modeled. Instead, a single selected time cycle with good input SNR is used for each link.

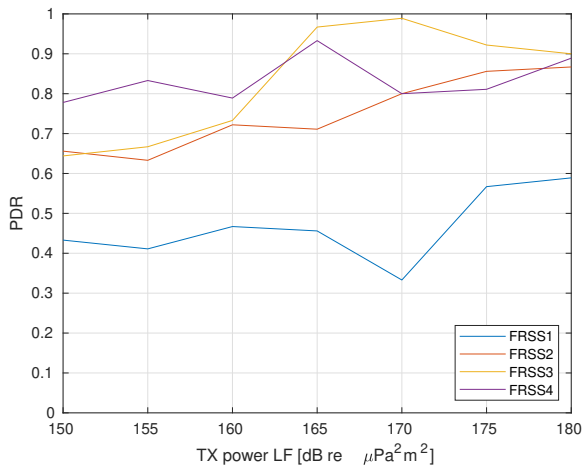
We have implemented a collision model in DESERT which keeps track of a list of interfering packets received at a given node. For each received packet, it is first checked if the regular LUT causes the packet to be received in error. If not, it is checked for each of the interfering packets received within an overlapping time interval whether that collision causes the packet to be received in error.

## IV. Simulation Scenario and Results

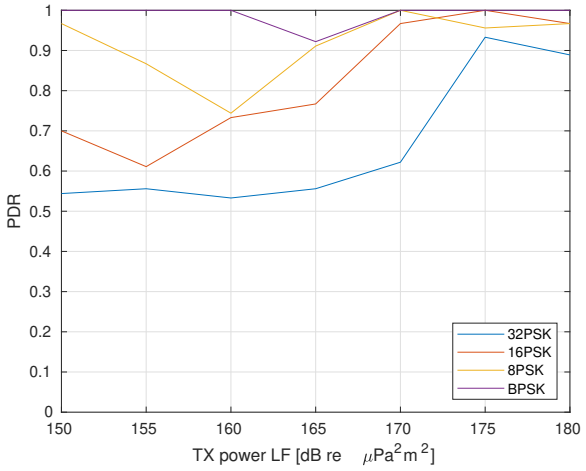
A simple simulation scenario is presented to illustrate the proposed framework.



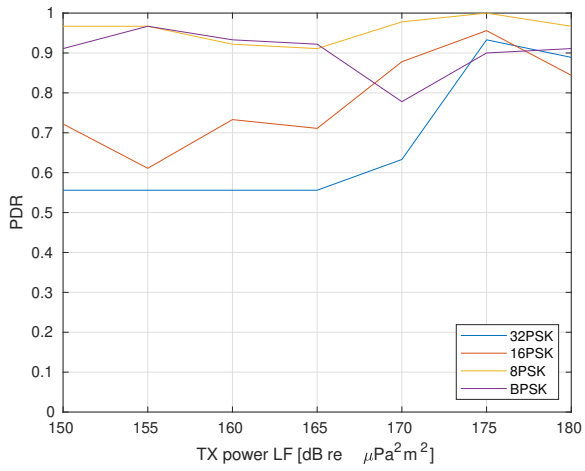

(a) LUT, time offset 0 s.




(b) LUT, time offset 40000 s.




(c) No LUT, time offset 0 s.




(d) No LUT, time offset 40000 s.

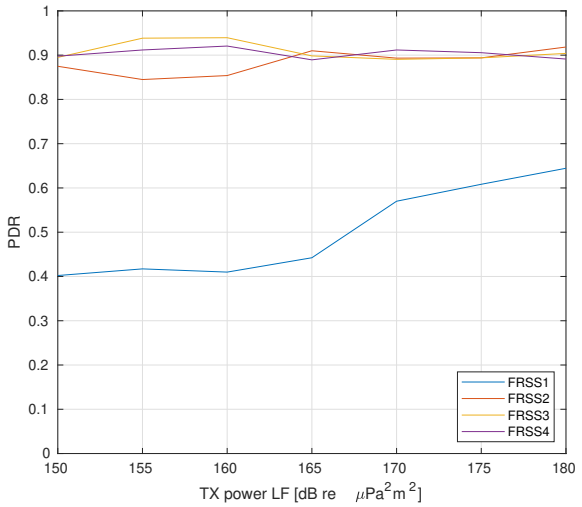
Fig. 6. Overall packet delivery ratio versus LF transmit power of a single simulation run. Fig. 6a and 6b are based on simulations using the LUTs and Figures 6c and 6d using Urick's model.

### *A. Simulation Scenario*

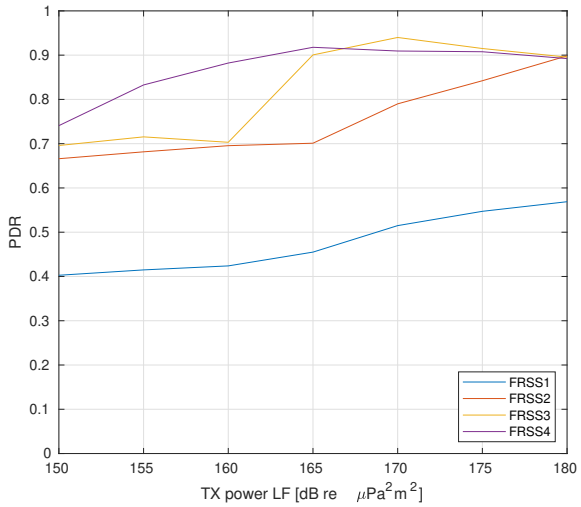
The proposed framework and the effects of the time-varying physical layer on network performance is illustrated with a simple network simulation scenario based on the topology shown in Fig. 2. In addition to the static nodes, two AUVs are included which move between the endpoints of the two runs that was performed to gather data for the LUTs for moving nodes.

A low traffic load is generated by having one new packet sent every 100 s, alternating the source node between all bottom nodes and AUVs. For the bottom node positions that did not have transmit capability during the at-sea data collection [15], nearest neighbor approximation is applied as discussed earlier. The destination node is always the node denoted N7 and each node transmits 10 packets. The packet size is 20 bytes.

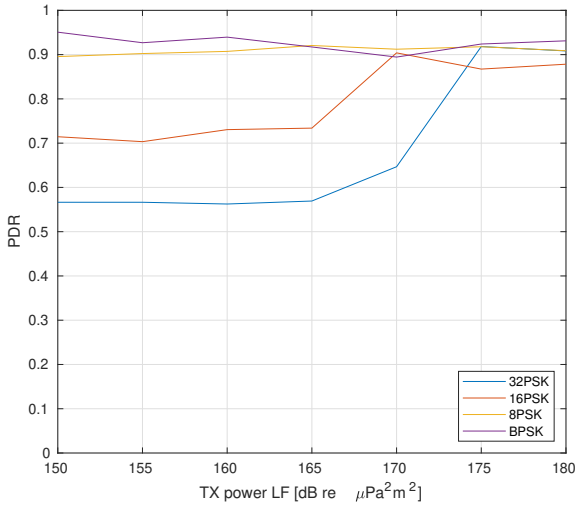
All four FRSS modes are evaluated: during one simulation run the FRSS mode is fixed and the same mode is used for both the LF and HF modems. As a comparison, the legacy physical layer based on Urick's propagation model already included in DESERT is used. In order to make the comparison somewhat fair, the bit rate for the legacy model is adjusted to be equal to the corresponding effective bit rate of each FRSS profile, specifically, 429 bps, 227.6 bps, 130.4 bps and 72.6 bps. The legacy model computes the signal SNR as presented in [19], and computes the BER based on the modulation as described in [20]. A practical spreading of 1.7 is set, and a shipping activity of 0.5 and wind speed of 5 m/s are considered. The interference model selected is MEANPOWER, that spreads the interference uniformly over the entire packet duration, considering the errors uniformly distributed. In order to capture the reduced robustness aspect of increasing the bit rate, four PSK configurations have been selected, namely 32PSK, 16PSK, 8PSK and BPSK, because,



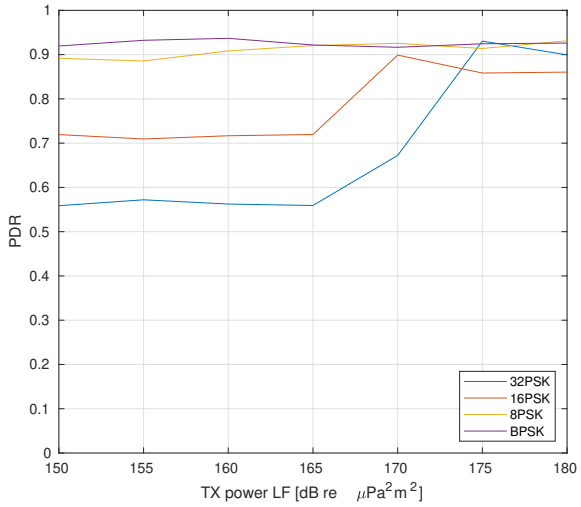

(a) LUT, time offset 0 s.




(b) LUT, time offset 40000 s.




(c) No LUT, time offset 0 s.




(d) No LUT, time offset 40000 s.

Fig. 7. Overall packet delivery ratio versus LF transmit power averaged over 20 simulation runs. Figures 7a and 7b are based on simulations using the LUTs and Fig. 7c and 7d using Urick's model.

together with BFSK, these are the available modulations in the DESERT legacy model. For both physical layer models the transmit power of the LF modem is varied between 150 and 180 dB re $\mu$Pa$^2$m$^2$.

Two different time offsets for the LUTs are evaluated, one starting at the beginning of the LUT and one with a time offset of 40,000 seconds. For the non time-varying setups the time offset simply leads to a different seed value for the random number generators.

### *B. Results*

The results from the network simulations are shown in Fig. 6 which shows the average packet delivery ratio (PDR) of the overall network for different LF transmit power levels for different bit rates at the physical layer. Comparing the two different time offsets for the LUTs (Fig. 6a and 6b) the network performance is generally better at all FRSS rates in the first part of the LUT (time offset 0) compared to the later part of the LUT due to the channel experienced by the nodes. The results show that the optimal FRSS profile from a network perspective is not easily determined, though the highest rate performs arguably worst due to noise and, mostly, reverberation [15]. The link distances in the simulated scenario may be too small to differentiate significantly between FRSS2–4.

The fact that different FRSS profiles are optimal at different time offsets and output power levels indicates that adaptive approaches could benefit the network performance. The packet

size most likely also affects the choice of FRSS profile and the rather small packets in this simulation negatively affect the higher rate profiles as a larger proportion of the transmission time consists of overhead.

For the legacy physical layer model the overall PDR is generally higher. 32PSK performs poorly for a transmission power of less than 170 dB re $\mu$Pa$^2$m$^2$, whereas the PDR increases up to more than 90% when the transmission power is more than 175 dB re $\mu$Pa$^2$m$^2$. Similarly, for 16PSK the PDR becomes higher than 85% when the transmission power is more than 167 dB re $\mu$Pa$^2$m$^2$, while for the other modulation schemes the PDR is already higher than 80% also for a transmission power of 150 dB re $\mu$Pa$^2$m$^2$.

While the PDR difference of the two plots with the LUTs is mainly caused by changes in the channel, the PDR difference of the two plots without LUTs is mainly caused by sporadic collisions due to a different seed in the randomization of the packet transmissions. The difference between the two plots is mitigated averaging upon multiple simulation runs, as proven by Fig. 7c and 7d, while with the LUTs the plots are still quite different as they experience a different acoustic channel (Fig. 7a and 7b).

## V. Conclusion

In this paper we described the SALSA simulation framework, where nodes equipped with LF and HF modems able to switch modulation according to the network layer indications are interconnected with each other. The network layer is based on the GUWMANET protocol, that forwards the packets generated by the GUWAL application layer with a redesign of the gossiping algorithm to address the aspects of underwater acoustic mobile ad hoc networks. The simulator consist of three main components, namely: the DESERT Underwater network simulator, the integration of GUWMANET into the DESERT simulator, and a time varying physical layer implemented using LUTs based on the tracks obtained during an extended field-test campaign. The latter also provides GUWMANET information about the acoustic noise, the input and output SNR, the delay spread, and other metrics, to allow the network layer to decide which physical layer configuration should be used to transmit a certain packet to a certain destination. In fact the physical layer can use different modulation and coding schemes, as well as different transmission powers. Simulation results present how the network performance changes when considering different channel realizations, something that could not be accomplished using the legacy DESERT physical layer model.

This work is a guideline, so that scientists can insert their real-world network protocols into the DESERT Framework and test them against this benchmark in a realistic scenario with a channel based on real measurements. This would enable a fair performance comparison and will help the underwater network community evaluate new protocols with a common benchmark.

## Acknowledgment

This work was partially funded by the EDA SALSA project, with partners from the Netherlands, Germany, Sweden, Finland, and Norway. Credits are due to the numerous people who contributed to the sea trial where the data was collected. Paul van Walree is thanked in particular for the sea trial design and data analysis which the lookup tables are built on.

## References

[1] J. Heidemann, W. Ye, J. Willis, A. Syed, and Y. Li, "Research challenges and applications for underwater sensor networking," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, 2006.

[2] F. Campagnaro, R. Francescon, F. Guerra, F. Favaro, P. Casari, R. Diamant, and M. Zorzi, "The DESERT underwater framework v2: Improved capabilities and extension tools," in *Proc. UComms*, Lerici, Italy, Sep. 2016.

[3] M. Stojanovic and J. Preisig, "Underwater acoustic communication channels: Propagation models and statistical characterization," *IEEE Communications Magazine*, vol. 47, no. 1, pp. 84–89, Jan. 2009.

[4] 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.

[5] P. Casari, F. Campagnaro, E. Dubrovinskaya, R. Francescon, A. Dagan, S. Dahan, M. Zorzi, and R. Diamant, "ASUNA: A Topology Data Set for Underwater Network Emulation," *IEEE J. Oceanic Engineering*, vol. 46, no. 1, pp. 307–318, Mar. 2021.

[6] 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.

[7] 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.

[8] N. Baldo, M. Miozzo, F. Guerra, M. Rossi, and M. Zorzi, "MIRACLE: The Multi-Interface Cross-Layer Extension of ns2," *EURASIP Journal on Wireless Communications and Networking*, Jan. 2010. [Online]. Available: http://www.hindawi.com/journals/wcn/2010/761792/cta/

[9] R. Martin, S. Rajasekaranand, and Z. Peng, "Aqua-Sim Next Generation: A NS-3 Based Simulator for Underwater Sensor Networks," in *Proc. ACM WUWNet*, Halifax, Canada, Nov. 2017.

[10] M. Chitre, R. Bhatnagar, and W. S. Soh, "UnetStack: An agent-based software stack and simulator for underwater networks," in *Proc. MTS/IEEE Oceans*, St. John's, NL, Canada, Sep. 2014.

[11] "Subnero M25M Series Modems," last time accessed: Dec. 2020. [Online]. Available: https://subnero.com/products/modem.html

[12] H. Dol, "EDA-SALSA: Towards smart adaptive underwater acoustic networking," in *Proc. MTS/IEEE Oceans*, Marseille, France, 2019.

[13] M. Goetz and I. Nissen, "GUWMANET - multicast routing in underwater acoustic networks," in *Proc. MCC*, Warsaw, Poland, Oct. 2012.

[14] R. Otnes, P. A. van Walree, H. Buen, and H. Song, "Underwater acoustic network simulation with lookup tables from physical-layer replay," *IEEE Journal Oceanic Engineering*, vol. 40, no. 4, pp. 822–840, Oct. 2015.

[15] P. van Walree and M. Colin, "In-situ performance prediction of a coherent acoustic modem in a reverberant environment," *IEEE Journal Oceanic Engineering*, 2021, accepted.

[16] P. van Walree, D. Tollefsen, and V. Forsmo, "Under-ice acoustic communication in the Nansen basin," in *Proc. UComms*, online, August 2021.

[17] R. Petroccia, J. Alves, and G. Zappa, "Janus-based services for operationally relevant underwater applications," *IEEE Journal Oceanic Engineering*, vol. 42, no. 5, pp. 994–1006, Oct. 2017.

[18] "Network common data form (NetCDF)," last time accessed: Dec. 2020. [Online]. Available: https://www.unidata.ucar.edu/software/netcdf

[19] M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," *ACM Mobile Comput. and Commun. Review*, vol. 11, no. 4, pp. 34–43, Oct. 2007.

[20] N. Benvenuto and M. Zorzi, *Principles of Communications Networks and Systems*, 1st ed. Wiley, 2011.