# Development and Testing of an OFDM Physical Layer for the DESERT Simulator

Sara Falleni,[*] Dipon Saha,[†] Israat Haque[†] Tommaso Melodia,[*] Stefano Basagni,[*]
[*]The Institute for the Wireless Internet of Things, Northeastern University, Boston, USA
[†]Faculty of Computer Science, Dalhousie University, Halifax, Canada

***Abstract*—The realization of efficient, robust, and adaptable applications for the emergent Internet of Underwater Things enables the sustainable and effective conservation and exploitation of our oceans and waterways. Recent advances have focused on Orthogonal Frequency-Division Multiplexing (OFDM) physical layers for supporting applications requiring high data rates and swift adaptation to changing underwater conditions. This prompts the need of tools for testing new OFDM-enabled underwater solutions. To this aim, this paper presents the implementation and evaluation of an OFDM-based physical layer module for the popular underwater network simulator DESERT. We aim at modeling the flexibility of the software-defined acoustic SEANet modem by realizing OFDM features that can vary in time, including the number and the selection of subcarriers and their modulation on a per-transmission basis. We demonstrate the usage of the proposed module through the DESERT-based simulation of three simple OFDM-enabled cross-layer MAC protocols in underwater acoustic networks of different sizes. The diverse and detailed set of results are obtained by using our physical layer module simply and swiftly. Our results also confirm the advantages of using the OFDM technology in solutions for underwater networking in challenging environments.**

## I. Introduction

The Internet of Things (IoT) has become an essential building block of the connected smart digital world. The span of such networks crosses both lands and waters. Particularly, *Internet of Underwater Things* (IoUT) networks, especially those using wireless technologies, are now a reality. Increasingly affordable and cost effective devices can now be deployed on the ocean floor, on mobile autonomous vehicles, and on vessels and buoys on the surface, all capable to communicate to each other wirelessly, e.g., through acoustic modems, to report relevant parameters and information to data collectors (*sinks*) at the surface or on land (Fig. 1). This heterogeneous infrastructure enables connecting maritime objects to realize applications for smart marine vehicles, smart shores and oceans, smart ocean dynamics measurements, disaster prediction and prevention, real-time coastal monitoring, and many more [1].

Challenges to the implementation of the IoUT vision, however, abound. They stem from the fact that IoUT devices and networks are intrinsically heterogeneous, have constrained resources, and operate in harsh environments. Further limitations also come from the impossibility to replicate the host of efficient and robust solutions already developed for the terrestrial IoT because the underwater environment severely limits the use of RF communication, prevailing on land. For this reason, wireless underwater networking relies primarily on acoustic communications, which are still considerably limited and rather unreliable: Existing underwater acoustic modems and the networks they enable have slow data rates, limited bandwidth, proprietary protocol stacks, and are not adaptable to environmental dynamics.

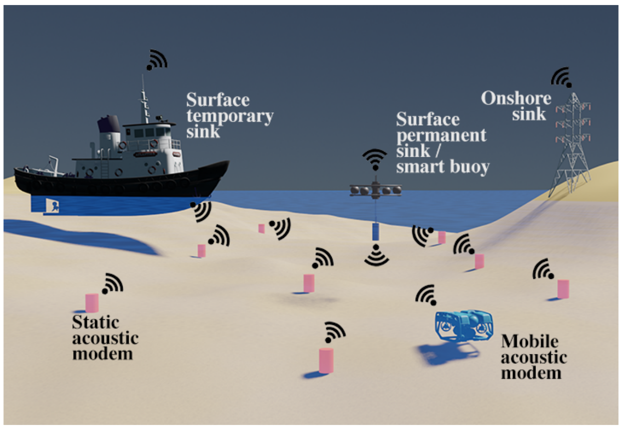


Fig. 1: A sample underwater wireless network.

In order to obviate these limitations, researchers are recently exploring the use of Orthogonal Frequency-Division Multiplexing (OFDM) for increased reliability and adaptiveness in underwater communications. OFDM offers high spectral efficiency, low inter-symbol interference and fading, and low sensitivity to time synchronization errors [2]. For these reasons, OFDM-based physical layers are widely used in technologies for terrestrial wireless networks, such as WiFi and LTE. In the underwater realm, the use of OFDM is being explored in several different ways [3]. The *SEANet Project*, in particular, pursues the realization of an underwater wireless testbed based on a new software-defined device, the SEANet modem, with the flexibility to define, add, update, and swap new components in both hardware and software [4], [5]. Through the use of OFDM, the SEANet modem aims at supporting data rates at least one order of magnitude higher than those of existing commercial platforms over short-range and moderate-range links.

**Contribution**. While acoustic devices like the SEANet modems are being built and tested, there is still no widespread availability of prototypes for evaluating new OFDM-based networking solutions for underwater applications, especially at scale. In fact, to the best of our knowledge, there is no tool, whether hardware or software, that can be used to demonstrate the effectiveness of using OFDM technologies to obtain the robustness and reliability that many applications require.

With this paper, we contribute to the field of *simulation-based* testing of new OFDM-based solutions for underwater wireless networked systems. In particular, we present the definition, development, and demonstration of use of an *OFDM-based physical layer module* for the popular underwater simulator *DESERT* (for DEsign, Simulate, Emulate and Realize Test-beds) [6], [7], which we chose for its widespread use and availability. DESERT consists of a set of C++ libraries that extends the functionalities of ns2-MIRACLE [8] to include modules explicitly built for modeling underwater solutions at all layers of the protocol stack. DESERT is open source, consistently maintained and supported, and comes with easy to use interfaces to add and test new modules.

We model the OFDM implementation of the SEANet modem by extending the physical and interference modules of DESERT. Particularly, our new module, called *uwofdmphy*, models the modem flexibility by implementing OFDM features that can vary in time, including the number of subcarriers used and the modulation within each subcarrier. This allows different selections of subcarriers on a per-transmission basis.

We showcase the use of the module by investigating the performance of three simple OFDM-based MAC protocols in underwater acoustic networks of different sizes. The three protocols, all based on the well known ALOHA medium access control scheme [9], use the module uwofdmphy differently, mimicking a single carrier MAC, a multi-carrier MAC on a fixed subset of subcarriers, and a MAC that uses two randomly chosen subcarriers each time a data packet needs to be (re)transmitted. Results investigating packet delivery ratio and end-to-end latency are obtained by the three protocols using our physical layer module simply and swiftly, according to their different definitions. These simulation-based results demonstrate the potential of OFDM-based underwater solutions, opening the door to a more efficient and intelligent Internet of Underwater Things.

**Paper organization.** The rest of the paper is organized as follows. Section II describes the core of our work, namely the DESERT implementation of an OFDM-based physical layer. In Section III we demonstrate its usage by simulating the functions of simple ALOHA-based MAC protocols implemented in DESERT. We conclude the paper in Section IV.

## II. DESERT and the uwofdmphy Module

In this section, we introduce the proposed OFDM-based physical layer module *uwofdmphy* for DESERT. We first describe the general structure of the DESERT framework and state how a physical layer can be modeled in it. Then, we provide the details of *uwofdmphy* and the required modifications to the existing DESERT physical layer and interference modules to support the proposed OFDM module.

### *A. The DESERT architecture*

*DESERT Underwater* is a complete set of public C++ libraries that extend the ns2-MIRACLE simulator [8] to support the design and implementation of underwater network protocols [6]. Its most recent version is 3.1 [7]. A sketch of its general architecture is shown in Fig 2.

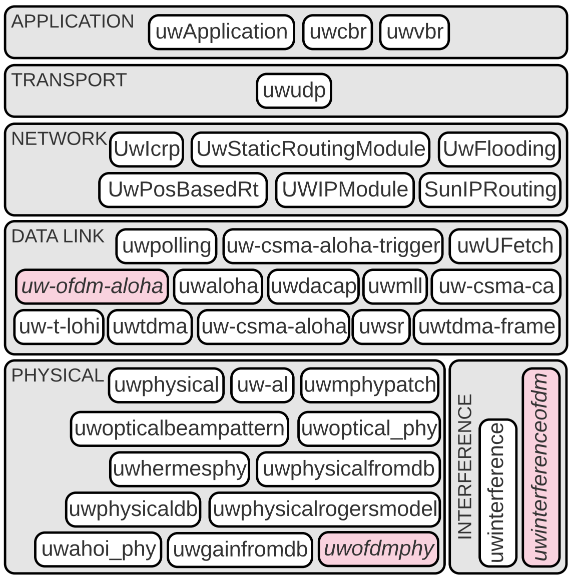


Fig. 2: DESERT: Overall architecture.

Following the traditional Internet reference model [9], the DESERT protocol stack consists of Application, Transport, Network, Data Link, and Physical layers. The Application Layer mostly concerns data generation, including modules for producing constant bit rate and variable bit rate traffic, and a module that allows using DESERT with data from TCP and UDP streams. The Transport Layer offers flow multiplexing and demultiplexing over UDP, whereas the Network Layer allows the implementation of routing protocols. DESERT offers templates for different routing mechanisms: Static, dynamic, and flooding. Moreover, it assigns IPv4-compliant addresses to the nodes. The Data Link Layer provides various Medium Access Control protocols such as ALOHA, ALOHA-CS, DACAP, a MAC based on polling, T-Lohi, and ARQ. It also provides the module *uwmll* to perform ARP functions, namely, to translate an IP address to a corresponding MAC address. The Physical Layer implements modules for acoustic and optical underwater communication, modules to interface DESERT with real devices (e.g., modems), and a dummy module that enables physical layer-less simulations (*uwmphypatch*). An Interference Layer complements the Physical Layer to model wireless interference of concurrent communications. The interference module computes the interference happening at receiver nodes. Each of the modules is implemented using

C++ classes that can be extended to refine their behavior or to create new ones.

The DESERT physical layers for acoustic communications are shown in Fig. 3, which also indicates the C++ class hierarchy.

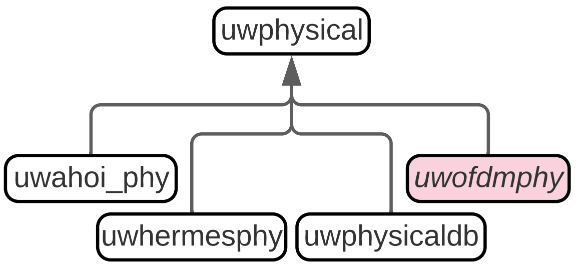


Fig. 3: The DESERT phy layers for acoustic communications.

The module uwphysical is the main physical layer module for acoustic communications in DESERT. It extends the UnderwaterMPhyBpsk of ns2-MIRACLE. The module uwphysical handles the transmission of packets coming from the upper layers and the reception of packets coming from the channel. Following the model introduced by Stojanovic [10], it computes the received signal power and the noise power. Also, using the interference module, it calculates the interference power, later used to compute the Packet Error Rate (PER). Given the PER, it uses a random variable to choose if the packet should be discarded. For PER computation, uwphysical supports single carrier acoustic underwater networking with different modulation schemes: BPSK, BFSK, 8PSK, 16PSK, and 32PSK.

DESERT currently features the following modules that extend uwphysical.

- uwahoi_phy is used to simulate the behavior of the AHOI modem, modeling the modem performance with and without interference [11].
- uwhermesphy models the behavior of the Hermes underwater acoustic modems, where the success of a packet is given by a lookup table that has distance vs. PER, and the interference is always destructive.
- uwgainfromdb is used to compute the attenuation of the signal reading from a file. The PER is computed as in uwphysical.
- uwphysicaldb builds hash tables of transmitter-receiver communication that concern SNR vs. PER and SINR vs. PER.

(Classes at this layer include also drivers to interface with real modems; not shown in the figure.)

Similar to the other physical layer modules of DESERT, our new *uwofdmphy* module extends the uwphysical. The following sections describe the extension in details.

### *B. Building the uwofdmphy Module*

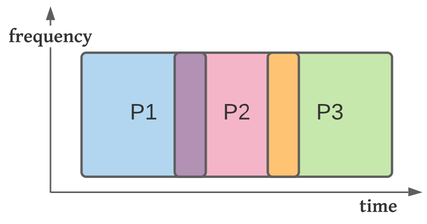
***uwofdmphy*** is intended to model an OFDM-like physical layer [2]. We are in particular interested in modeling the OFDM implementation of the software-defined acoustic SEANet modem, a high-data rate, high-bandwidth acoustic modem that includes OFDM as one of its physical layers [4]. Given the hardware-dependent center frequency and bandwidth, the SEANet modem divides the bandwidth in a number of $2^k$, $k \geq 0$, subcarriers and it is capable to decide the subcarriers to use to send or receive a packet. The chosen subcarriers do not need to be contiguous, and each can be modulated differently. To model this capability of the SEANet modem in DESERT we need to modify/extend the following. 1) We modify the physical layer packet header *mphy_pktheader.h* to keep track of which subcarriers are used to transmit a packet and which modulation is used for each subcarrier. 2) We extend the DESERT interference module *uwinterference*, to model the kind of interference that happens in OFDM systems.

In the next three sections, we describe the details of these modifications and we then provide the structure of our *uwofdmphy* module, indicating how we extend the DESERT uwphysical to implement it.

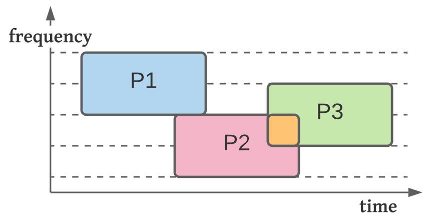
*1) Physical layer header for OFDM:* ns2-MIRACLE (and DESERT) defines the structure of a packet at the physical layer in the header file *mphy_pktheader.h*. In the following discussion, each subcarrier is identified by a number between 0 and $2^k - 1$. To support a multi carrier implementation we add four fields that define the following: (i) The number of subcarrier of the OFDM system: integer variable `carrierNum`$= 2^k$; (ii) the width of each subcarrier: integer variable `carrierSize`; (iii) if each subcarrier is used to transmit the packet or not: binary vector variable `carrier`, of size `carrierNum`, where `carrier`$[i]$ is 1 if subcarrier $i$ is used, 0 otherwise, and (iv) the type of modulation used for each subcarrier: char vector variable `carMod`, where `carMod`$[i]$ can be any of the modulation currently supported by DESERT. These additions to the physical packet header are important not only for physical layer-related computations, such as that of the Signal to Interference plus Noise Ratio (SINR) and Packet Error Rate (PER), but also to allow protocols at all layers of the networking stack to perform cross-layer operations, if needed (Section III-B).

*2) OFDM-specific interference module:* DESERT uses an interference model that is the extension to underwater communications of the interference model of ns2-MIRACLE [8]. In particular, the SINR of packets whose reception overlaps in time at the receiver is used to compute packet errors and to determine whether a packet should be received or discarded. While in single carrier systems two packets overlapping in time always interfere, in multi carrier systems this is not always the case. The single vs. multi carrier interference model is illustrated in Fig. 4.

Fig. 4a shows three packets P1, P2 and P3 transmitted on the same frequency and arriving at the same receiver in such a way that the reception of P1 partially overlaps with that of P2, whereas P2 partially overlaps with that of P3. In this case, all three packets are potentially discarded by the receiver. Fig. 4b illustrates the same scenario in a multi



(a) Interfering packets in a single carrier physical layer.



(b) Interfering packets in a multi carrier physical layer.

Fig. 4: Single carrier vs. multi carrier packet interference.

carrier setting, where the transmission of packet P1 shares no subcarrier with packet P2 and one subcarrier with packet P3, whereas packet P2 shares one subcarrier with that of packet P3. In this case packet P1, with no subcarrier in common with packet P2 and no overlap in time with packet P3, is received correctly. Packets P2 and P3, however, may be discarded as their reception partially overlaps on the shared frequency.

To model this OFDM-like behavior we ported the DESERT interference model at the subcarrier level. To this purpose, we extend the *uwinterference* module of DESERT, and use it as a parent class for our *uwinterferenceofdm* module. We add the double vector `carrier_power` to the DESERT class *ListNodes*, which is a class that keeps track of the interference power at each node. The added vector keeps track of the interference power on each subcarrier at a given node.

*3) The uwofdmphy physical layer module:* The main concept behind the *uwofdmphy* module is that each of the subcarriers used to transmit a packet is handled independently. To implement this concept, the *uwofdmphy* module extends the uwphysical module of DESERT with a new variable and redefines functions to account for subcarrier independence.

The new variable that the *UwOFDMPhy* class adds to the UnderwaterPhysical class of uwphysical is a list named `pktqueue_`. The list contains the packets that are currently being received by the node and whose subcarriers do not overlap. Whether these packets will be correctly received or not depends also on the SINR (*uwinterferenceofdm* module). The new functions are:

• `getOFDMNoisePower()`. This function uses the original getNoisePower function of the UnderwaterMPhyBpsk class of ns2-MIRACLE to compute the noise power that the node would receive on the full bandwidth. The obtained value is then scaled to the amount of bandwidth used to transmit the packet. The function returns a double value.

• `getOFDMPER()`. The original function getPER of uwphysical uses the signal-to-noise or the signal-to-noise-plus-interference ratio and the modulation to compute the Bit Error Rate (BER) and uses the number of bits to compute the PER. Our `getOFDMPER` computes the BER for each used subcarrier. It then computes the average BER over all used subcarriers and uses the size of the packet to calculate the PER, which is returned (a double).

• `freqOverlap()`. This Boolean function checks if an incoming packet overlaps in frequency, namely, has common subcarriers with packets that are currently being received. It returns true in case the packets are overlapping, false otherwise.

We also redefine the following functions of uwphysical.

• `startRx()`. The `startRx` function of uwphysical discards a packet if another packet is already being received. In the *UwOFDMPhy* class implementation, the function The `startRx` has been redefined so that a packet is discarded if another packet is already being received over overlapping subcarriers. In order to evaluate the possible overlapping it uses the new `freqOverlap` function. If the new packet is not overlapping with a packet being received, The `startRx` saves the new packet in the `pktqueue_` list.

• `endRx()`. This function is called whenever the reception of a packet if completed. In uwphysical, this checks whether the packet has been discarded (by `startRx()`) or not. In the negative, it triggers the computation of relevant parameters such as SINR, BER, etc. The version implemented in *uwofdmphy* takes care also of checking whether the packet is in the `pktqueue_` list and, if so, it removes it.

• `getTxDuration()`. The new `getTxDuration` uses the percentage of bandwidth used over the available bandwidth and the modulation of each carrier to compute the effective bit rate for each packet.

## III. Sample Usage and Testing of uwofdmphy

In this section, we show how the new *uwofdmphy* module can be used in DESERT to simulate OFDM-enabled underwater networking protocols. As our purpose is that of showcasing the use of the new module, we investigate a simple networking scenario, comprised of networks of variable size where we compare the performance of simple ALOHA-based MAC protocols for different amounts of traffic.

We start by describing the simulation scenarios; we then introduce the MAC protocols that use *uwofdmphy*; we finally illustrate the results of the comparative performance evaluation of the selected protocols.

### A. Underwater Scenarios

We consider a single-hop network where $N$ nodes, $N \in \{5, 20, 35, 50\}$, generate and transmit packets to a *collector node*, called the *sink*. The nodes are statically deployed, randomly and uniformly, in a cubic area of $70 \times 70 \times 70$ cubic meters whose upper face coincides with the sea surface. The sink is positioned at the center of the upper face of the deployment area, some 10 m below the surface. The nodes and the sink are equipped with an acoustic device modeled after the SEANet modem [4], allowing them to exchange packets. As the nominal SEANet modem transmission range is around 100 m, all nodes can transmit and receive from the sink (star-shaped networks).

Data packets are generated in time according to a Poisson distribution with $\lambda = [0.1, 0.5, 1]$ to model low, medium, and high traffic scenarios, respectively. Each node implements the protocol stack depicted in Fig. 5.

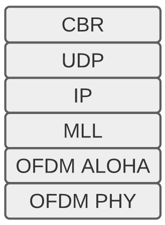


Fig. 5: The network protocol stack used by each node.

Specifically, at the Application Layer, we use the DESERT uwcbr module to generate packets, with the option to scatter packets in time according to a Poisson Distribution. At the Transport Layer, a node uses UDP (uwudp). The module IP defines the Network Layer as Internet-based (UWIPModule). To this purpose, it converts ns2 nsaddr_t and uint8_t addresses to addresses of the form $x.x.x.x$. The Data Link Layer creates and maintains the ARP tables to map IP addresses to MAC addresses (uwmll). Its MAC sublayer (OFDM ALOHA in the stack) implements the three ALOHA-based protocols described below through the newly defined *uw-ofdm-aloha* module that we added to DESERT. Finally, the Physical Layer is implemented by using our *uwofdmphy* module (Section II). The module models communication through acoustic waves that propagate at $1500\,\mathrm{m/s}$ sent according to the OFDM technique.

Consistent with the SEANet modem capabilities, each node uses a $125\,\mathrm{kHz}$ center frequency and a $125\,\mathrm{kHz}$ bandwidth. In our experiments, we assume that the bandwidth is divided into 8 subcarriers that can be used independently, numbered 0 through 7. Nodes choose subcarriers on a per transmission basis. Communications from the sink to the nodes (e.g., acknowledgments) use the entire bandwidth. Packets are 1536 B long, corresponding to the standard packet size that the SEANet modem is currently able to send and receive. When using the full bandwidth, nodes are able to communicate at 64000 bps. The modulation used in each subcarrier is BPSK. Packets awaiting transmissions are queued in the node RAM. As the MicroZed processor of the SEANet modem has ample memory (1 GB of RAM), we consider the size of the queue "infinite." (And indeed in all our experiments we never observed packet lost for buffer overflow.)

We assume the presence of an interfering device external to our network (e.g., a sonar or other acoustic transmitter) that communicates over the frequency range corresponding to subcarriers 3 and 4. We model interference from this device through the probability $p$ that a data packet transmitted on either of those two subcarriers arrives at the sink corrupted. In our experiments we set $p = 0.1$.

The performance of our network is evaluated by investigating the following two metrics.

- *Packet Delivery Ratio* (PDR), defined as the percentage of packets that are successfully received by the sink.
- *End-to-end latency*, defined as the average time to deliver a packet to the sink successfully.

Each simulation run lasts 10000 s. Results are obtained by averaging the outcome of a number of simulations that is enough to achieve 95% confidence and 5% precision.

### B. ALOHA-based Underwater MAC Protocols

The following three simple MAC protocols are based on the well-known basic ALOHA protocol [9]. All three protocols use OFDM at the Physical Layer, taking advantage of it, or not, in different ways. Particularly, through the modified packet header, the MAC protocols indicate to the physical layer how many and which subcarriers are to be used to transmit the packet, and which modulation should be used in which subcarrier (in our case, the latter is always BPSK).

*1) S-ALOHA:* The protocol S-ALOHA, for Simple ALOHA, is the implementation of ALOHA "as is." When the protocol receives a packet, it transmits it using the OFDM physical layer indicating that all 8 subcarriers should be used, as in a single carrier system. Packets received correctly are acknowledged by 10 B packets. A packet is retransmitted if an acknowledgment is not received within 5 seconds. We use the DESERT implementation of ALOHA to compute the backoff time before retransmission. Based on this implementation, no node awaits more than $40\,\mathrm{s}$ to retransmit a packet. After three retransmissions a packet is discarded.

*2) IA-ALOHA:* The protocol IA-ALOHA, for *Interference Aware* ALOHA, assumes that a node is capable to determine whether devices external to the network use some of the frequencies. In our experiments we assume that nodes know that subcarriers 3 and 4 are used by the external device, and avoid using them, instructing the OFDM physical layer to use the other frequencies. Clearly, using the remaining 6 frequencies is done at a lower data rate (48000bps). Other ALOHA-related parameters and timers are set as for S-ALOHA.

*3) RF-ALOHA:* The protocol RF-ALOHA, or Random Frequency ALOHA, selects two out of the 8 available frequencies randomly to transmit a packet. This approach is similar to that described by Shen and Li for terrestrial networks [12]. (We added a backoff time before retransmitting a packet.) While this solution clearly reduces the likelihood of interference, it transmits packets at reduced data rates (16000bps). ALOHA-related parameters and timers are set as for S-ALOHA.

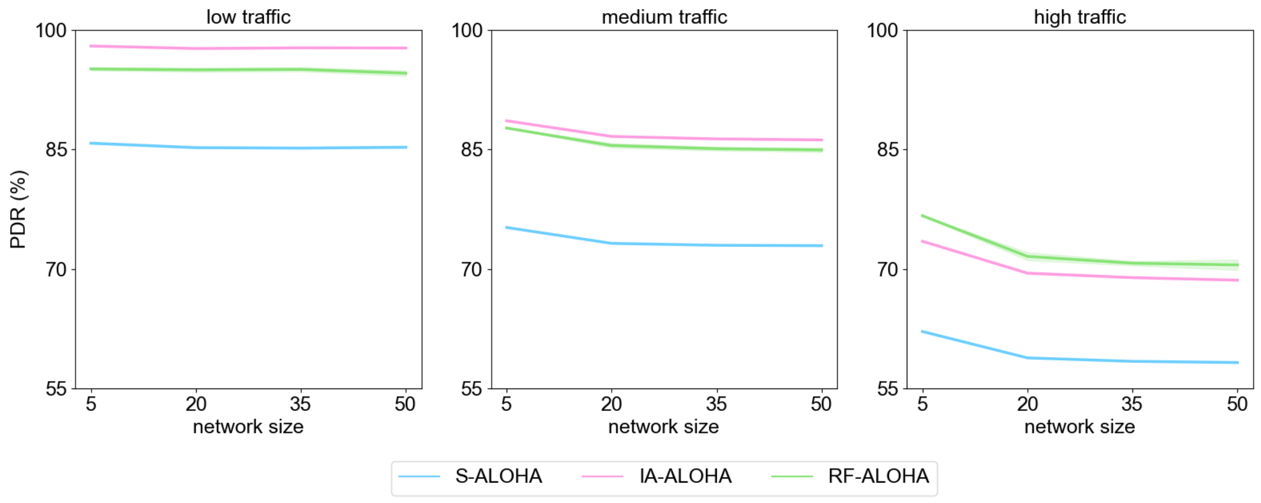


Fig. 6: Packet Delivery Ratio.

## C. *Performance Results*

Results of our simulations are shown in Fig. 6 and Fig. 7.

*1) Packet delivery ratio:* Results concerning the PDR are shown in Fig. 6 for the three considered classes of traffic.

We notice that overall, the PDR is not overly sensitive to the size of the network, because of the star-shaped nature of our networks centered at the sink.

Independently of traffic, S-ALOHA always achieves the worst performance, as packets are lost because of the interference from the external device on subcarriers 3 and 4 and interference (*collision* of data packets) at the sink. At low traffic, S-ALOHA achieves PDR values of 85.80%, 85.24%, 85.15%, and 85.29% in networks with 5, 20, 35, and 50 nodes, respectively. IA-ALOHA, which always avoids subcarriers 3 and 4, obtains the best performance, achieving a 98% PDR for 5 nodes and 97.67%, 97.75% and 97.72% for networks with 20, 35, and 50 nodes, respectively, offering a 14% improvement over S-ALOHA. RF-ALOHA obtains PDRs of 95.14%, 94.95%, 95.14%, and 94.50% in networks with 5, 20, 35, and 50 nodes, respectively, with a 10% improvement over S-ALOHA. Its performance is slightly inferior to that of IA-ALOHA because nodes may use subcarriers 3 and 4.

Results at medium traffic show similar trends, albeit the values of the PDR are smaller because of an increased number of collisions. Particularly, S-ALOHA obtains an average PDR of 75.28% for 5 nodes, 73.12% for 20 nodes, 72.98% for 35 nodes, and 72.91% for 50 nodes. IA-ALOHA obtains the best performance: The PDRs obtained in networks with 5, 20, 35 and 50 nodes are 88.69%, 86.60%, 86.32%, and 86.2%, respectively, outperforming S-ALOHA by 21%. RF-ALOHA obtains PDR values of 87.71%, 85.49%, 85.08%, 84.94% for networks of 5, 20, 35, and 50 nodes, respectively, with a 17% improvement over S-ALOHA.

Finally, at high traffic, S-ALOHA offers the worst performance averaging 62.15% for 5 nodes, and 58.85%, 58.36%, and 58.33% in networks with 20, 35, and 50 nodes, respectively. In this case, RF-ALOHA performs better than IA-ALOHA. Its PDR in networks with 5, 20, 35, and 50 nodes averages at 76.96%, 71.59%, 71%, and 70.78%, respectively, offering a 24% improvement over S-ALOHA. The PDRs values obtained by IA-ALOHA for 5, 20, 35, 50 nodes are 73.66%, 69.49%, 68.89%, 68.60%, respectively, offering an 18% improvement over S-ALOHA.

The reason why with increasing traffic the PDR performance of RF-ALOHA improves with respect to that of IA-ALOHA, outperforming it at high traffic, has to do with the fact that the number of packets colliding at the sink is higher when 6 subcarrier are used (IA-ALOHA) instead of two, chosen at random (RF-ALOHA). This observation is confirmed by looking at the number of collisions and retransmissions. While the number of collisions is similar for the two protocols at high traffic, the number of successful retransmissions is higher for RF-ALOHA because each retransmission happens on newly randomly selected subcarriers, leading to a higher PDR.

*2) End-to-end latency:* Results concerning the end-to-end latency at low, medium, and high traffic are shown in Fig. 7.

The figure depicts a breakdown of the latency in its three main components: Queuing delay (average time spent by a packet awaiting transmission—bottom of the bars, in a lighter color), transmission delay (pushing out the packet—middle section of the bar, in medium hue color), and propagation delay (the average time it takes to a packet to travel from the source to the sink—top of the bar, in a darker color).

We observe that the transmission delay component of the end-to-end latency is influenced by the different amounts of bandwidth that each protocol uses to transmit packets. S-ALOHA uses the whole available bandwidth hence being able

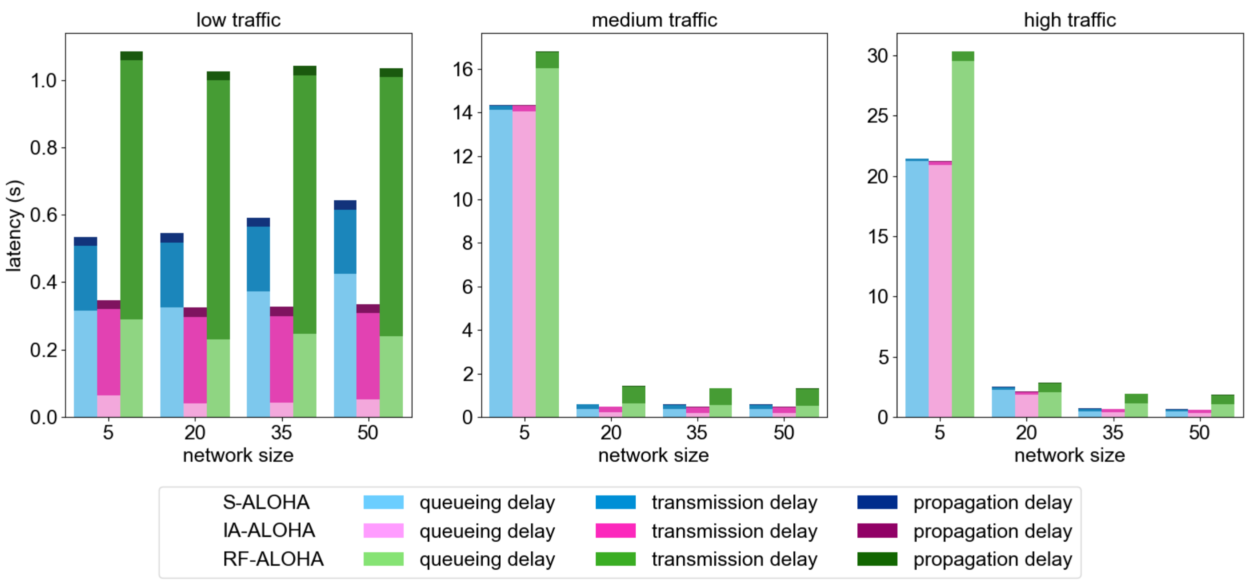


Fig. 7: End-to-end latency.

to achieve 64000 bps. In this case, the transmission delay is at least 0.192 s for a packet long 1536 B. IA-ALOHA, using 6/8 of the available bandwidth, can achieve bit rates not to exceed 48000 bps, and needs at least 0.256 s to push out a packet. Finally, as RF-ALOHA uses 1/4 of the available bandwidth, it obtains a 16000 bps bit rate, needing 0.768 s to transmit a packet. Table I provides a summary of these values, where TDelay indicates the transmission delay for packets that are 1536 B long and #Carriers indicated the number of subcarriers used by each protocol.

TABLE I: Achievable bit rates and transmission delays.

| | Bit rate (bps) | TDelay (s) | #Carriers |
|---|---|---|---|
| S-ALOHA | 64000 | 0.192 | 8 |
| IA-ALOHA | 48000 | 0.256 | 6 |
| RF-ALOHA | 16000 | 0.768 | 2 |

Latency is dominated by transmission (which is constant) and queuing delays, the latter becoming dominant as traffic increases. Propagation delays are reasonably low given the small extent of the network deployment area. In fact, they visibly affect the performance of the three protocols only at low traffic.

In general, we notice that increasing traffic implies higher latency for all three protocols. As expected, this depends on the higher interference that a higher number of packets incur as well as overwhelming queuing delays.

We observe that irrespective of network size and traffic, on average IA-ALOHA is the faster protocol. This is because it uses a relatively large bandwidth (6 subcarriers out of 8), which allows a higher bit rate and incurs a noticeably lower number of retransmissions.

We also notice that RF-ALOHA is always slower independently of network size and traffic. This has to do with the fact that it only uses two subcarriers to transmit packets.

We finally note that at medium and high traffic, the value of the end-to-end latency in networks with 5 nodes is remarkably higher than in bigger networks. This is because each node has to send a number of packets (on average, a fifth of the total amount offered to the network) that is considerably larger than the number of packets per node in bigger networks ($N \geq 20$). If any of the packets generated early in the simulation incurs high delays (multiple retransmissions each after large backoff times) then the following packets suffer increasingly long queuing delays. We observe that several simulation runs in scenarios with 5 nodes suffered this case.

When the traffic is low S-ALOHA achieves an average latency of 0.53 s for 5 nodes, 0.54 s for 20 nodes, 0.59 s for 35 nodes, and 0.64 s for 50 nodes. Independently of network size, IA-ALOHA is the faster protocol. Packets need on average of 0.35 s, 0.32 s, 0.33 s, and 0.34 s when the network size is 5, 20, 35, and 50 nodes, respectively, improving over S-ALOHA by 40%. RF-ALOHA, instead, has the longest end-to-end latency, achieving an average of 1.08 s, 1.03 s, 1.04 s, and 1.04 s in networks with 5, 20, 35, and 50 nodes, respectively, suffering, on average, an 80% increase in time over S-ALOHA.

When the traffic is medium, S-ALOHA achieves an average of 14.34 s for 5 nodes, 0.60 s for 20 nodes, 0.58 s for 35 nodes, and 0.58 s for 50 nodes. Irrespective of network size IA-ALOHA achieves faster average end-to-end latency. Packets need, on average, 14.32 s, 0.49 s, 0.47 s, and 0.46 s when the network is respectively composed of 5, 20, 35, and 50 nodes, offering a negligible improvement for networks with 5 nodes and a 20% improvement for networks with 20, 35, and 50 nodes. RF-ALOHA, instead, has longer end-to-end times, achieving an average of 16.81 s, 1.43 s, 1.34 s, and 1.32 s for networks with 5, 20, 35, and 50 nodes. It suffers a 20% increase in time in networks with 5 nodes and a 130% increase for networks with 20, 35 and 50 nodes over S-ALOHA.

When the traffic is high, S-ALOHA achieves an average of 21.44 s for 5 nodes, 2.51 s for 20 nodes, 0.70 s for 35 nodes, and 0.66 s for 50 nodes. Independently of network size, IA-ALOHA achieves faster average end-to-end latency. Packets need, on average, 21.29 s, 2.11 s, 0.67 s, and 0.63 s when the network is made up of 5, 20, 35, and 50 nodes, offering negligible improvement over S-ALOHA at 5 nodes, a 15% improvement at 10 nodes and a 5% improvement for other network sizes. As in previous cases, RF-ALOHA has longer end-to-end times, achieving an average of 30.27 s, 2.83 s, 1.95 s, and 1.85 s for networks with 5, 20, 35, and 50 nodes, respectively, and suffering a 40% increase in time at 5 nodes, a 12% increase in time at 20 nodes and a 180% increase in time at 35 and 50 nodes over S-ALOHA.

## IV. Conclusions

The paper contributes to the field of simulation-based testing of new OFDM-based solutions for underwater wireless networked systems. Particularly, we create a new module to model an OFDM physical layer for the popular underwater network simulator DESERT. The module, which extends the generic DESERT physical layer, aims at modeling the flexibility of a software-defined acoustic modem—the SEANet modem—by implementing OFDM features that can vary in time, including the number of subcarriers and the modulation within each subcarrier. This allows different selections of subcarriers on a per-transmission basis. We showcase the module in use through results that can be obtained by using it for simulating sample OFDM-based MAC protocols in underwater acoustic networks of different sizes. The diverse and detailed set of results are obtained by different protocols using our physical layer module simply and swiftly, according to their different definitions. Our results also confirm the advantages of using OFDM technology in solutions for underwater networking, and therefore the need of tools for testing them.

## Acknowledgment

Sara Falleni, Stefano Basagni and Tommaso Melodia were supported in part by grant NSF CNS 1726512 "MRI: SEANet: Development of a Software-Defined Networking Testbed for the Internet of Underwater Things." Israat Haque and Dipon Saha were partially supported by an Ocean Frontier Institute Seed fund from Dalhousie University.

## References

[1] M. Jahanbakht, W. Xiang, L. Hanzo, and M. Rahimi Azghadi, "Internet of Underwater Things and big marine data analytics—A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 904–956, January 20 2021.

[2] H. Rohling, Ed., *OFDM: Concepts for Future Communication Systems*, ser. Signals and Communication Technology. Berlin Heidelberg, Germany: Springer-Verlag, March 23 2011.

[3] S. Zhou and Z. Wang, Eds., *OFDM for Underwater Acoustic Communications*. Hoboken, NJ: John Wiley & Sons, Ltd, March 28 2014.

[4] E. Demirors, J. Shi, A. Duong, N. Dave, R. Guida, B. Herrera, F. Pop, G. Chen, C. Cassella, S. Tadayon, M. Rinaldi, S. Basagni, M. Stojanovic, and T. Melodia, "The SEANet project: Toward a programmable internet of underwater things," in *Proceedings of IEEE UComms 2018*, Lerici, Italy, August 28–30 2018, pp. 1–5.

[5] S. Falleni, D. Unal, A. Neerman, K. Enhos, E. Demirors, S. Basagni, and T. Melodia, "Design, development, and testing of a smart buoy for underwater testbeds in shallow waters," in *Proceedings of IEEE/MTS Global OCEANS 2020*, Singapore–U.S. Gulf Coast, October 5–14 2020, pp. 1–7.

[6] 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 *Proceedings of UComms 2016*, Lerici, Italy, August 30–September 1 2016, pp. 1–5.

[7] "DESERT Underwater." [Online]. Available: http://desert-underwater.dei.unipd.it

[8] N. Baldo, F. Maguolo, M. Miozzo, M. Rossi, and M. Zorzi, "ns2-MIRACLE: A modular framework for multi-technology and cross-layer support in network simulator 2," in *Proceedings of ValueTools 2007*, Nantes, France, October 22–26 2007, pp. 1–8.

[9] J. F. Kurose and K. W. Ross, Eds., *Computer Networking: A Top-Down Approach*, 8th ed. Hoboken, NJ: Pearson, 2021.

[10] M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," *ACM SIGMOBILE Computing and Communication Review*, vol. 11, no. 4, pp. 34–43, October 2007.

[11] F. Campagnaro, F. Steinmetz, A. Signori, D. Zordan, B.-C. Renner, and M. Zorzi, "Data collection in shallow fresh water scenarios with low cost underwater acoustic modems," in *Proceedings of UACE 2019*, Hersonissos, Crete, Greece, June 30–July 5 2019, pp. 281–288.

[12] D. Shen and V. Li, "Stabilized multi-channel ALOHA for wireless OFDM networks," in *Proceedings of Globecom 2002*, vol. 1, Taipei, Taiwan, November 17–21 2002, pp. 701–705.