World Ocean Simulation System (WOSS): A Simulation Tool for Underwater Networks with Realistic Propagation Modeling

Federico Guerra, Paolo Casari, Michele Zorzi
Department of Information Engineering, University of Padova
Via G. Gradenigo, 6/B, 35131 Padova, Italy
{fguerra,casarip,zorzi}@dei.unipd.it

ABSTRACT

Network simulators are a fundamental tool for the performance evaluation of protocols and applications in complex scenarios, which would be too expensive or infeasible to realize in practice.

With the aim to provide a shared environment for the simulation of underwater networks we have adapted the ns2 network simulator to provide a detailed reproduction of the propagation of sound in water (i.e., by means of ray tracing instead of empirical relations). This has been tied to formerly available simulation frameworks (such as the MIRACLE extensions to ns2) to provide a completely customizable tool, including acoustic propagation, physical layer modeling, and cross-layer specification of networking protocols. In this paper, we describe our tool, and use it for a case study involving the comparison of three MAC protocols for underwater networks over different kinds of physical layers. Our results compare the transmission coordination approach chosen by each protocol, and show when it is better to rely on random access, as opposed to loose or tight coordination.

Categories and Subject Descriptors

C.2.8 [**Communication/Networking and Information Technology**]: Mobile Computing

General Terms

Design, Experimentation

Keywords

Underwater networks, simulation, MAC protocols, propagation modeling, Bellhop, JANUS, periodic traffic, event-driven traffic

1. INTRODUCTION

Underwater networks are foreseen to provide a fundamental tool for supporting a wealth of applications requiring the use of mobile as well as fixed nodes in diverse fields, from environmental monitoring to support in answering distress calls, intrusion detection, and so forth. In light of these developments, there is growing interest in underwater acoustic networking. Underwater networking studies have covered many fields to date, from channel access, to routing and topology control [1, 2, 3]. However, at-sea experimentation of proposed solutions incurs high costs for underwater nodes (especially mobile ones), ships, boats, and sea-trained personnel. This is the main reason why such networks and the related protocols are much more frequently simulated rather than deployed and experimented at sea. While some efforts exist to build cheaper nodes with lower transmission range and scale down experiments to smaller network sizes and areas[1] [4], indeed simulation is a fundamental tool to perform preliminary network evaluations: in particular, it could help pick the best approach out of a number of candidates for actual implementation.

Unfortunately, there is no standard simulation tool currently focused on underwater networks. The most widely adopted approach is to reuse terrestrial wireless network simulators, after changing the wireless propagation model to approximate underwater acoustic propagation. This usually boils down to setting the speed of sound to a constant value of $1.5\,\mathrm{km/s}$, and to implementing empirical attenuation and noise power spectral density formulas, such as those in [5, 6]. This approach is, however, limited, as the cited empirical formulas usually hold as approximations, whose accuracy highly depends on the scenario and on the characteristics of the deployment (i.e., shallow vs. deep water, warm vs. cold seas, flat vs. rough sea bottom, and so forth); moreover, the variation of environmental factors such as the temperature of water and the morphology of the sea bottom may have a non-negligible impact on propagation. For example, consider Figs. 1 and 2, which have been obtained with the channel modeling component of the simulator described in Sec. 3. Fig. 1, represents the attenuation incurred by an acoustic wave transmitted, in August, from the shore of the Pianosa island, located at 42.585°N, 10.1°E. Darker shades of gray represent stronger signal power. The figure shows that the signal reaches the sea bottom bearing high power, and could be thus employed to communicate directly with bottom-mounted sensors deployed off the coast. By way of contrast, Fig. 2 shows the same scenario in February: this time, the average temperature of the water is lower in the upper water layers, changing the way sound is refracted, and causing most of the acoustic power to stay away from the bottom and closer to the surface. In this situation, any bottom mounted sen-

WUWNet'09, November 3, 2009, Berkeley, CA, USA.

[1]Smaller nodes are simpler to handle and manage than larger ones. Small transducers yield more isotropic acoustic emissions and allow to communicate over wider high-frequency bands (which provide high rates at limited distance). In addition, waterproof encasing for small nodes is easier to obtain from very cheap materials.

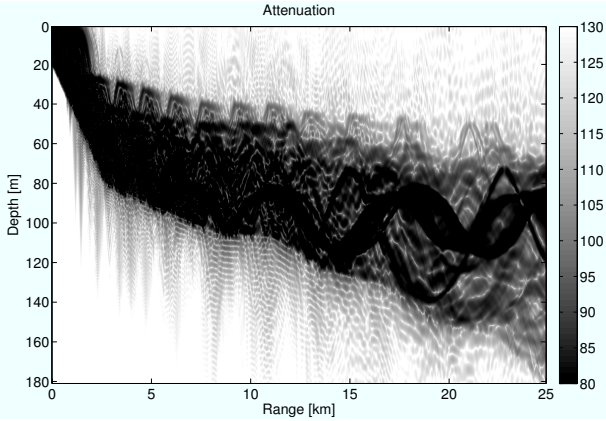

Figure 1: Attenuation incurred by acoustic waves transmitted in August from the shore of Pianosa Island, 42.585°N, 10.1°E. A darker shade of grey represents a stronger signal. Bottom sediments are a mixture of clay and silt; a sharply then smoothly decreasing sea bottom profile can be seen in the lower half of the picture.

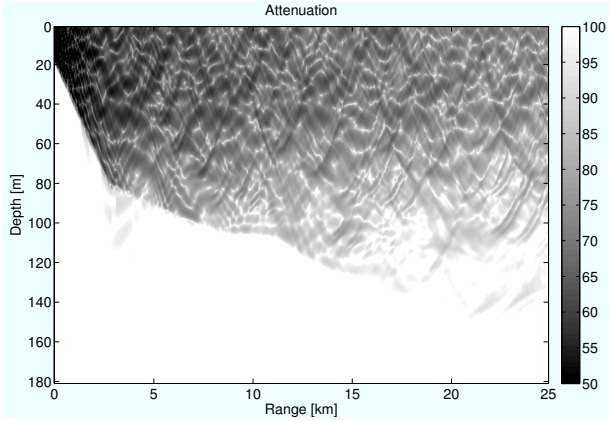

Figure 2: Attenuation incurred by acoustic waves transmitted in February from the shore of Pianosa Island, 42.585°N, 10.1°E. A darker shade of grey represents a stronger signal. The environment is the same as in Fig. 1.

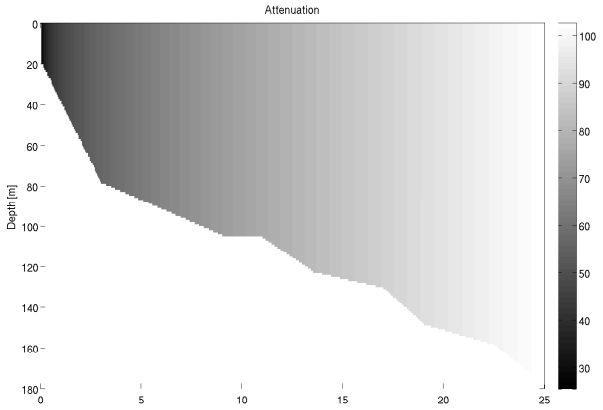

Figure 3: Attenuation incurred by acoustic waves as predicted by empirical formulas in [5]. A darker shade of grey represents a stronger signal. Attenuation has not been calculated below the sea bottom (lower white part of the figure).

sors deployed off the coast may be outside the reach of the transmitter ashore.

The behavior outlined above highlights that an accurate reproduction of the physical layer, including all main propagation effects, is of paramount importance for any simulation. In this paper, we present two contributions. The first is the development of a network simulation tool which explicitly incorporates the propagation details according to the above discussion: this has been possible by integrating the well-known ns2 [7] and MIRACLE [8, 9] simulators with the Bellhop ray tracing tool [10], as discussed in Secs. 2 and 3. While ray-tracing provides accurate emulation of sound propagation, flexible programming at all levels of the protocol stack is made possible by the MIRACLE framework.

The second contribution is a comparative study of MAC solutions for an underwater network, carried out among three different MAC protocols, namely ALOHA [11], Tone-Lohi [1] and DACAP [2]; as these protocols enforce different levels of transmitter-side coordination in the network, our work also provides a discussion of the relationship between the amount of coordination and the final network performance. We present this evaluation in the presence of both periodic and event-driven traffic.

2. PROPAGATION MODELING

From the point of view of network simulation, modeling the propagation of acoustic waves means modeling the statistics of the attenuation incurred by the waves over a link, as well as second-order statistics such as the autocorrelation of link attenuation and the cross-correlation between different links. As there is still no widely agreed upon model for second-order statistics, propagation modeling is usually limited to reproducing the average value of link attenuation. An empirical model for this value is obtained by considering the absorption factor $a(f)$ as expressed by Thorp's formula [5, Chapter V] and the approximated formula $A(d, f) = d^k a(f)^d$, where d is the distance covered by the link, f is the transmit frequency, and k models the geometry of propagation (i.e., cylindrical for $k = 1$ to spherical for $k = 2$). Empirical formulas are also available for the power spectral density of receiver noise, which is expressed as the superposition of different nature- and human-originated factors [5, Chapter VII].

The greatest advantage of this approach is that the cited equations are straightforward to implement and evaluate; its main drawback is that all formulas only hold as an approximation. The actual propagation of acoustic waves largely depends on the physical parameters of water, namely temperature, salinity and local pressure (which in turn depends on depth). These factors influence the local propagation speed, and the way acoustic waves are bent (i.e., refracted) during propagation. A synthetic representation of the factors causing acoustic refraction is provided by the sound speed profile (SSP), i.e., the propagation speed of sound considered as a function of water depth. Different profiles lead to potentially very different propagation, and give rise to such effects as surface sound channels, deep sound channels, convergence zones, shadow zones, and so on (e.g., see [12, pp. 17–33]). Furthermore, the bathymetric profile in the transmission area, the physical transmitter parameters of the electro-acoustic transducer (shape, size, aperture) as well as the type of bottom sediments, also influence propagation. No empirical formula would be able to take all of this into account: besides predicting a larger SNR on average, empirical formulas cannot model such complex phenomena as those listed above; by considering again Figs. 1 and 2, we note that there is a difference of nearly $20\,\mathrm{dB}$ between the attenuation incurred near the bottom at

20 km from the source in August and February. This is enough to break a communication link, yet empirical formulas would not be able to predict it. To make the difference between empirical formulas and ray tracing clearer, Fig. 3 shows the attenuation value computed through equations in [5]. Given the shallow water depth, there is almost no difference between the attenuation incurred at the surface and at the bottom; also, note that Bellhop partially models sound propagation in bottom sediments [10], unlike empirical propagation formulas which are valid for water only. Any network simulation would incur significant differences if run over the empirical instead of the ray tracing model, as we show in Sec. 4.2.

In order to keep into account the previously described factors and corresponding effects, we have considered the use of ray tracing for propagation modeling. A freely available ray tracing tool, Bellhop [10], has been integrated in the simulator for this purpose. The tool takes all parameters cited before as inputs, and uses ray tracing to compute the solution to the propagation equations over a vertical slice of the water column, or over a restricted area within this slice. In the second case, the computation is faster (intuitively, fewer rays converge on a limited area). The solution provided by Bellhop yields either the overall incident acoustic power, or the attenuation incurred by the propagating wave. A second output option offered by Bellhop allows to compute attenuation and time of arrival on a per ray basis, thereby providing a quick means of estimating the power-delay profile of the channel. This second option has not been considered for the moment, and is left as a future work. In other words, we assume for now that the overall received power is the coherent sum of all signals reaching the target through multipath propagation. It should be noted that the greater accuracy[2] provided by the Bellhop ray tracer is limited to the computation of attenuation, i.e., noise is still modeled using the empirical formulas; however, this is a fairly common approach and is deemed to be enough for our requirements. The next section details the features of the simulator employed for our evaluation.

3. THE WORLD OCEAN SIMULATION SYSTEM (WOSS)

The simulation system we specifically set up for this evaluation is based on ns2 [7] and the ns2-MIRACLE extensions [8, 9]. In particular, the latter provides a very flexible interface for protocol coding and for implementing interactions among protocols located at different layers of the ISO/OSI stack, if required. As in [13], acoustic propagation effects have been reproduced by means of the Bellhop ray tracing software [10], as introduced in the previous section. This replaces and improves the previous implementation of attenuation models, which were in accordance to the empirical formulas given in [5, 6], see [14]. We recall that, in order to calculate the solution to the propagation equations between a transmitter and a receiver, Bellhop requires knowledge of the SSP, the bathymetric profile, and the type of bottom sediments (required to model acoustic power losses due to bottom reflections). To provide this data, we have interfaced our simulation framework with databases freely available on the Internet. For the SSP, we employ the World Ocean Database [15], a collection of SSPs measured during a number of experiments all around the world; the measurements are divided by location and day or season of the year when the measurement was performed (recall that sound propagation is affected by water temperature, which in turn undergoes seasonal changes, especially in the superficial layer). As a complement to that, we also have spatially finer data available from the GLINT'08 [16] sea trials (which, however, have not been used for the results shown in this paper). The bathymetric data have been taken from the General Bathymetric Chart of the Oceans [17], a public database offering samples of the depth of the sea bottom with an angular spacing of 30 seconds of arc. Finally, the type of bottom sediments is taken from the National Geophysical Data Center's Deck41 database [18]. To make database interfacing easier, WOSS abstracts from the specific database technology (SQLlite, netcdf, user custom) through a simple Application Program Interface; thanks to this choice, the user only has to write an object implementing the interface, in order to have WOSS make the proper calls to the databases.

The effort of putting together all components required to run Bellhop pays off, in that the user only has to specify at which location in the world the simulated experiments should take place. This is done by setting the wanted latitude and longitude, as well as the size of the network area. The simulator automatically handles the rest. In more detail, assume that some nodes were deployed within the area: the simulator picks their location (i.e., latitude, longitude and depth) and queries the databases for samples of bottom sediments and measured SSPs at each node location (for simplicity, the SSP can also be assumed to be constant, on average, throughout the network area); by linearly interpolating bathymetric data, an approximation of the sea bottom is also provided between each pair of nodes. As a final preparatory step, a Bellhop run with the previous parameters is executed in order to compute the acoustic attenuation between any two nodes. Assuming that the network is static and the average channel features do not change, the computations need be carried out only once.

All Bellhop calculations are performed at a fixed frequency, which is assumed to be representative of the whole transmission band. Specifically, let f_ℓ and f_u be the lower and upper limits of the frequency band: in line with the suggestions in [19] we run Bellhop at the frequency $\sqrt{f_\ell f_u}$. Noise power, accounted for through empirical formulas, is also computed at this frequency: in other words, we model noise as a white process within the signal band.

Having provided a specification of the physical layer, the MIRACLE package handles the remaining part of the simulation, namely the computation of Signal-to-Interference-plus-Noise Ratio (SINR) for all transmissions, the related error rates, and the evolution of the behavior of the nodes based on the simulated MAC and higher-level protocols. The MIRACLE structure has been thought to make the development of protocols and their interconnection easier within the popular event-driven network simulator ns2. It is not within the scope of this paper to describe ns2 or to give many details about MIRACLE (which can be found in [8]). To make the paper more self-contained, however, we wish to recall here that MIRACLE is based on the concepts of `Module`, `Connector`, and `NodeCore`. A `Module` is a piece of code which can contain any protocol (MAC, routing, application...) or new physical layer (PHY) specification. `Module`s are designed in order to be self-contained, easy to interconnect, exchangeable, and reusable. The interconnection of different modules in the same networking stack (e.g., a PHY to a MAC) is the task of specific objects, the `Connector`s. Connectors mark two modules such that information flow is enabled between them; such operations as parameter passing and protocol interactions are thus easier: in particular, connectors are useful in cross-layer protocol design and evaluation. The third main component of MIRACLE, `NodeCore`, actually enables communications among connected modules, manages general-purpose information, and provides other functionalities to all modules. As a further task, `NodeCore` also maintains the geographical informa-

[2]Ray tracing is not advisable for modeling very low-frequency wave propagation; however, we note that within the common operating bands used by available hardware, the approximations inherent in the ray tracing technique are verified to a satisfactory degree of accuracy.

tion of a node. The structure of MIRACLE and its modular design allow any protocol to exchange information with any other protocol it is connected to, without having to resort to ad hoc solutions. The type of information to be passed between modules includes packets, which can be serially processed by different modules as they step through the networking stack. To make any type of information about a packet available to, e.g., statistics extraction or debugging routines, `Connectors` also act as tracers: this makes the implementation of packet tracing automatic and independent of the implementation of the `Modules`. The user only has to choose the type of output to be provided by the tracers, and the related level of verbosity. In case other mobility is required, ns2 provides both an implementation of standard mobility models (such as the random waypoint model) and an interface to employ external mobility traces. MIRACLE extends this through a generic interface to deploy mobility model routines in C++, currently featuring deterministic and Gauss-Markov mobility models.

For the following discussion, it is useful to summarize the way SINR is computed in MIRACLE. Assume that a packet of length $L\,\mathrm{Bytes}$ is transmitted in a time T_u with power P_u; say that this packet incurs attenuation a_u as it travels toward the receiver (attenuation values are provided by interfaces to the physical layer code, e.g., Bellhop). During the time T_u the packet is received, interfering transmissions may disturb the reception. Assuming that the interference power due to a single transmission $i = 1, 2, \ldots$ is constant over its duration T_i, the power of this transmission can be modeled as $I_i(t) = P_i a_i \mathbb{1}(t_i, t_i + T_i)$, where t_i is the transmission epoch of packet i, and $\mathbb{1}(t_i, t_i + T_i)$ is equal to 1 if $t_i \leq t \leq t_i + T_i$, and 0 elsewhere. At this point, MIRACLE computes the SINR by considering the average interference over the duration T_u of the wanted transmission:

$$SINR(u) = \frac{P_u a_u}{N + \frac{1}{T_u}\int_t^{t+T_u} \sum_i I_i(\tau)\mathrm{d}\tau}\,, \qquad (1)$$

where N is the noise power in the band of the signal. More details on the MIRACLE PHY-level models and assumptions can be found in [8]. After the calculation of $SINR(u)$, the probability of error of the transmission is easily derived through standard formulas [20]. As an example, for Binary Phase Shift Keying (BPSK) the probability of error is $\frac{1}{2}\mathrm{erfc}\sqrt{SINR(u)}$, where $\mathrm{erfc}(\cdot)$ is the Gaussian complementary error function, and for incoherent Binary Frequency Shift Keying (BFSK) it is $\frac{1}{2}\exp(-\frac{1}{2}SINR(u))$. In all error equations, the SNR is scaled by a factor $\xi = 4\,\mathrm{dB}$ in order to model receiver inefficiencies (e.g., due to transducers and amplifiers). For example, this choice scales the probability of error of BPSK from $3 \cdot 10^{-5}$ to $2 \cdot 10^{-3}$ for an SINR of $10\,\mathrm{dB}$. Modeling Frequency-Hopping (FH)-BFSK requires to explicitly account for the hopping pattern employed by each node, and for the corresponding interference caused to the transmission of a bit in each sub-band. After this, FH-BFSK can be treated as the superposition of as many BFSK systems as the number of frequency bins in the hopping pattern. By calling p the bit error rate computed by MIRACLE, the packet error rate is finally obtained by assuming independent bit errors, i.e., through the formula $1-(1-p)^{8L}$, where L is the packet length in Bytes.

4. CASE STUDY

For the following evaluation, we assume that 10 nodes are arranged in a 5×2 grid, with nearest neighbors $1\,\mathrm{km}$ apart, close to the Pianosa island, a protected site off the north-western coast of Italy. The geographical coordinates are 49.25°N 10.125°E, which, along with the choice of a summer SSP, specify the set of environmental parameters considered in the following evaluation.

We assume that the nodes are equipped with standard modem hardware, such as the Teledyne-Benthos modem [21] or the WHOI micromodem [22], which enable transmissions using either Frequency Hopping Binary Frequency Shift Keying (FH-BFSK) or coherent Binary Phase Shift Keying (BPSK) in different bands. While we mainly refer to uncoded BPSK in our results, we are also interested in the interference-resilience of FH-BFSK; this is obtained by having different nodes employ different frequency hopping patterns, thereby reducing the probability that two transmissions collide, and are therefore lost. For these reasons, the FH-BFSK modulation has also been chosen for the JANUS protocol, a standard for unsolicited beacon-like communications [23, 24] currently under development. Its main drawback, however, is the very low transmit bit rate, on the order of only tens of bits per second, which limits both its applications and the higher-level protocols that can operate on top of it [23]. In our implementation of FH-BFSK within WOSS, we assume that the raw bit rate is $160\,\mathrm{bps}$ and that a convolutional code of rate $1/2$ is used to encode data, reducing the effective bit rate available to data to $80\,\mathrm{bps}$. The whole signal is carried in the 9–14 kHz band. These parameters are in line with those used in [23]. We compare this to an uncoded BPSK modulation (also supported by most standard modem hardware). Thanks to coherent detection, BPSK is potentially much faster than FH-BFSK: we account for this by setting the bit rate to $4800\,\mathrm{bps}$ and a higher carrier frequency of $24\,\mathrm{kHz}$.

We assume that all data generated by the nodes must be reported back to a sink, located at the center of the network area. In order to comply with the low FH-BFSK transmit rate, we set the packet size L to $50\,\mathrm{Bytes}$; however this might prove too inefficient for BPSK, for which we instead set a packet size of $600\,\mathrm{Bytes}$. In any event, signaling packets (i.e., RTSs, CTSs, Warnings and ACKs) are assumed to be $4\,\mathrm{Bytes}$ long. We consider two different packet generation patterns: namely, either according to a Poisson process of parameter λ packets per second per node or in an event-driven fashion, which will be presented later in Sec. 4.3.

We chose three MAC protocols (ALOHA [11], T-Lohi [1] and DACAP [2]), which are representative of a different amount of coordination among nodes (specifically, none, light, and strong signaling). A larger amount of coordination generally requires greater signaling, thus greater overhead: we specifically address whether this also yields greater benefits. Note that the protocols above do not require any time synchronization among nodes. For each protocol, we considered both an ACK and a no ACK version: in the former case, unless differently specified, we set the maximum number of retransmissions of any packet to 5. Preliminary results on these protocols have been presented in [25], while a related study on the effects of seasonal parameters as well as of the number of nodes is provided in [26].

4.1 Simulated protocols

ALOHA [11] is the first random contention-based channel access protocol, and lets any node immediately send data as soon as this data is generated. In multiuser networks, this implies that collisions may take place if two users attempt to access the channel at the same time. Standard contention resolution techniques are to be applied in this case, e.g., instantaneous channel sensing, for which we opted in the implementation presented here; in other words, the terminals sense the channel before transmissions and stay silent so long as they perceive any energy; after that, they immediately transmit. In addition, we employ a standard MAC-level backoff policy: after a transmission error, the sender waits for a random amount of time between 0 and twice the maximum propagation delay before rescheduling a new attempt; the value of the maximum backoff

time is doubled up to 5 times upon consecutive errors, and reset to the minimum upon a success.

ALOHA, in general, is expected to offer poor throughput performance and to be very prone to congestion. Nevertheless, it can be a feasible option in many underwater networks [3], if the traffic patterns are light enough and the long propagation delays alter the probability that collisions take place (unlike in radio networks, where two simultaneous transmissions in the same area always collide). We recall that Slotted ALOHA offers better performance with respect to simple ALOHA: however, we do not consider this version in this paper, as we do not assume any synchronization among nodes.

Tone-Lohi (T-Lohi) [1] is a reservation-based MAC protocol. It works by having nodes detect and count the number of neighbors simultaneously contending for channel access during a preliminary reservation phase; a light contention for channel access is then administered among these nodes based on a traffic-adaptive backoff algorithm, where the backoff length depends on the number of contending nodes in the same neighborhood. Many protocol operations, including the detection of contenders, are driven by wakeup tones. These tones allow nodes to stay asleep for most of the time, providing substantial energy savings during the reservation phase. On the other hand, the use of tones requires a specific detector, which permanently operates in a low power listening mode.

The reservation procedure should make collision-free channel access more likely, and works as follows. Any nodes seeking channel access must first send a reservation tone, and then wait to detect possible contenders. If the node does not hear any other tone, it wins the contention and immediately transmits its data. Otherwise, if other tones are heard, the node starts a contention with the corresponding transmitters, whereby nodes back off before sending another tone, and the first to send a tone and not hear any other tone for a silence time of fixed length wins and transmits data. In the aggressive version of T-Lohi (aT-Lohi), the silence time is equal to the maximum propagation delay; in the conservative version (cT-Lohi) it is twice as long (for more details, please refer to [1]). In the following, we will consider only aT-Lohi, augmented with the same MAC-level backoff policy described for ALOHA. A comparison including cT-Lohi is available in [26].

The **Distance-Aware Collision Avoidance Protocol** (DACAP) [2] is a non-synchronized data access scheme following the well known Request-To-Send (RTS) / Clear-To-Send (CTS) handshake. Depending on the presence of simultaneous handshakes nearby and on the relative distances of the nodes, two relevant scenarios may arise: i) the receiver, after sending the CTS, overhears an RTS, meaning that a future data packet transmitted by a neighbor will threaten the pending reception; in this case the receiver sends a short warning packet to its transmitter; ii) if any node overhears a packet meant for another neighbor, or receives a warning from a receiving party, it defers the data transmission [2]. The length of the idle period is chosen so as to make strong interference unlikely.

The handshaking pattern described above (which is indeed heavier than the method employed in T-Lohi) is not resolutive, as in some cases the warning packets arrive too late, or the nodes become aware of potential collisions too late to defer data transmissions; however, the degree of protection allowed by the handshake and the adaptive message timing is fairly high. We observe that a minimum handshake length must be set for all the nodes. In a network where most links are as long as the transmission range, this minimum length must be as high as twice the maximum propagation delay; in a deployment such as ours, where all links are shorter, it can be reduced. Again, we consider two versions of the protocol, namely with and without ACKs. In the first case, the protocol requires slightly different timings with respect to the second case, in order to accommodate the ACK message [2]. We apply to DACAP the same MAC-level backoff described for ALOHA and T-Lohi.

4.2 Results for Poisson traffic

This first part of our performance evaluation aims at determining which channel access protocol works best under random generation of traffic according to a Poisson process. We recall that the three protocols are representative of different types of handshakes, namely no handshaking (ALOHA), light handshaking (T-Lohi) and heavy handshaking (DACAP).

We start by comparing the performance of all protocols using a BPSK modulation and $L = 600\,\mathrm{Bytes}$, and specifically address the differences due to the use of empirical attenuation formulas, as opposed to the attenuation predicted by Bellhop. Figs. 4 through 7 depict throughput, success ratio, overhead and application-level success ratio for all protocols. For the moment, we consider only the no-ACK version for simplicity.[3] As the sink is the only receiver in the network, we define normalized throughput as the ratio of the number of bits that correctly reach the sink per second, divided by the modulation bit rate (i.e., $4800\,\mathrm{bps}$ in this case), as this is the maximum rate at which information can reach the sink. Success ratio is defined as the ratio of the number of correctly received data packets to the total number of data packets sent. The overhead is the fraction of sent bits that are dedicated to signaling packets, i.e., ACKs (for all protocols), RTSs, CTSs, and Warnings (DACAP), as well as tones (T-Lohi). For ACK versions, this does not include retransmissions of erroneous packets. Finally, application-level success ratio is required in order to understand the level of network performance perceived by upper network layers. In fact, as the network is saturated by incoming traffic, the transmission queue of the nodes may become full, so that the nodes begin to lose generated packets. The application success ratio is then computed over all generated packets, instead of just the packets that are transmitted. The figures show two sets of curves: solid ones are related to the realistic PHY-level model (Bellhop), whereas dashed ones represent the use of empirical attenuation formulas.

The first piece of information conveyed by these results is that empirical formulas yield better protocol performance, mainly due to a lower attenuation. This leads to a larger SNR, which deviates from the values obtained with Bellhop by 5 to even $20\,\mathrm{dB}$. The resulting transmission performance is therefore better, which is reflected by the MAC-level success ratio in Fig. 5. In turn, more packets are delivered to the sink, leaving more time to organize handshakes and subsequent transmissions for newly generated datagrams. Both throughput (Fig. 4) and application-level success ratio (Fig. 7) benefit from this situation; furthermore, the lower number of transmissions required to correctly receive a packet yields a lower overhead, as seen from Fig. 6. These observations apply to ALOHA more apparently than to the other protocols: ALOHA's throughput keeps growing linearly with increasing traffic up to $\lambda = 200\,\mathrm{bps}$ per node when using empirical formulas, as opposed to $50\,\mathrm{bps}$ when employing Bellhop. We also note that ALOHA's success ratio keeps above 0.8 for all considered traffic values when using empirical formulas, against a decrease to 0.45 when using Bellhop. Furthermore, the use of empirical formulas would have predicted a worse performance for aT-Lohi than ALOHA, whereas this is rather not the case with the more realistic Bellhop-computed attenuation (see discussion below).

From this first set of results, we also see that DACAP's performance is not substantially impacted by the difference between the attenuation models: this can be expected, given that DACAP's per-

[3] Similar results can be obtained for the ACK versions as well.

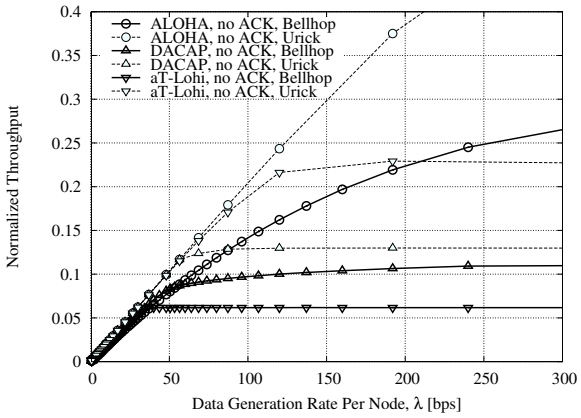

Figure 4: Throughput as a function of traffic for all protocols, BPSK, $L = 600\,\mathrm{Bytes}$, using Bellhop and formulas in [5].

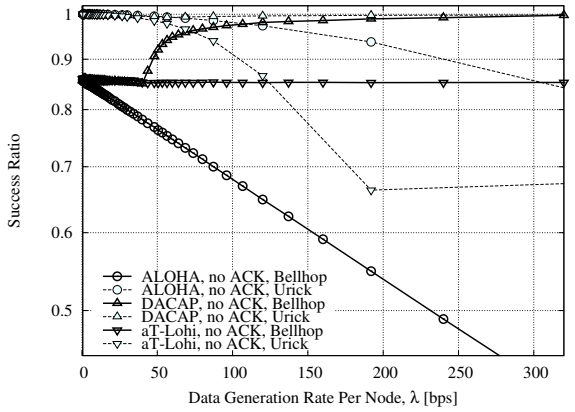

Figure 5: Success ratio as a function of traffic for all protocols, BPSK, $L = 600\,\mathrm{Bytes}$, using Bellhop and formulas in [5].

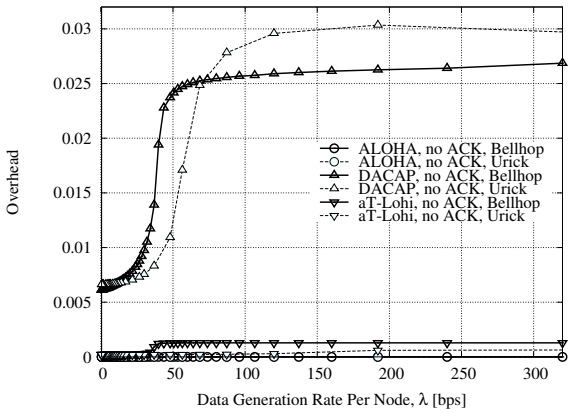

Figure 6: Protocol overhead as a function of traffic for all protocols, BPSK, $L = 600\,\mathrm{Bytes}$, using Bellhop and formulas in [5].

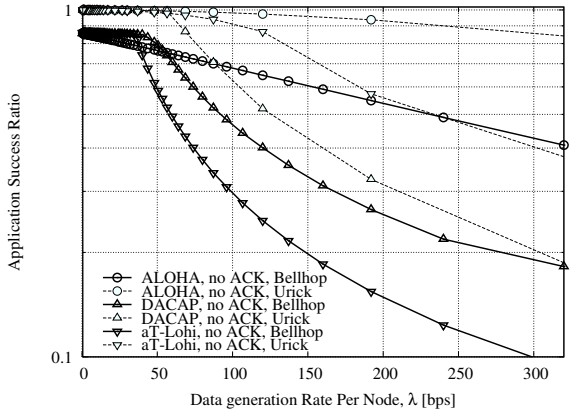

Figure 7: Application-level success ratio as a function of traffic for all protocols, BPSK, $L = 600\,\mathrm{Bytes}$, using Bellhop and formulas in [5].

formance is limited by the handshake times (including the round-trip times, packet transmission deferral and time allotted to Warnings). In fact, DACAP's throughput reaches a maximum of about 0.13 in both cases (though for different traffic values). Consistently, the success ratio is stable at a value very close to 1 thanks to DACAP's heavy collision avoidance signaling, and this is true for both empirical and Bellhop attenuation.

Similar observations hold for aT-Lohi as well, with the exception of success ratio, which is lower with empirical formulas unlike DACAP's and ALOHA's. In fact, in the case of aT-Lohi, the handshake for channel access has strict timing that increases the chances of collisions and interference. With Bellhop, larger attenuation prevents nodes at opposite sides of the network from hearing each other's tones, so that collisions are more likely (recall that receivers do not clear transmissions in cT-Lohi). With empirical formulas, lower attenuation relieves this until the offered traffic exceeds $\lambda = 120\,\mathrm{bps}$ per node, after which transmissions are more frequent: lower attenuation yields now the drawback of a larger interference, thus a lower SINR. It is worth noting at this point that the success ratio curves for most protocols, and in particular for DACAP with Bellhop, tend to increase after a first decrease phase. The reason is MAC-level backoff, which forces nodes to longer periods of silence, decreasing the level of interference and the chance of collision. DACAP especially benefits from this effect, which adds to the periods of silence required by handshakes: Bellhop, with its larger attenuation and thus lower transmission SINR, tends to amplify this effect, explaining the sharp increase in success ratio (solid line with upward triangles in Fig. 5). However, the increase in success ratio is not primarily due to the fact that more packets get through (the throughput increase is actually very limited), but rather to the fact that fewer packets are transmitted. This can also be seen from the application success ratio (Fig. 7) which keeps decreasing with increasing traffic, since saturated nodes will discard any new packets.

While the previous results are focused on the comparison of different propagation models, they indeed allow to draw some first conclusions about the relationship between network performance and handshake overhead: namely, low overhead protocols tend to achieve a larger throughput, at the expense of a lower success ratio. Robust handshake-based protocols such as DACAP can instead leverage on their better organization of transmissions in order to achieve a very high success ratio at the price of lower throughput.

We now focus on a joint comparison of the throughput of all protocols (both ACK and no ACK versions) using only Bellhop. In Fig. 8 we consider the BPSK modulation used for the previous set of results, whereas in Fig. 9 we employ the lower rate FH-BFSK-based PHY level described in Sec. 4. Adding ACK versions in Fig. 8 confirms the claim that one further piece of overhead does

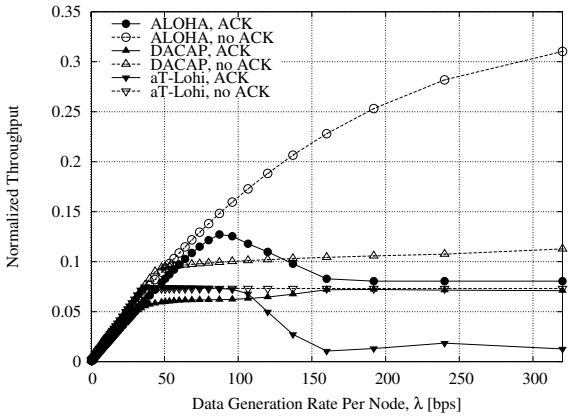

Figure 8: Throughput as a function of traffic for all protocols, BPSK, $L = 600\,\mathrm{Bytes}$.

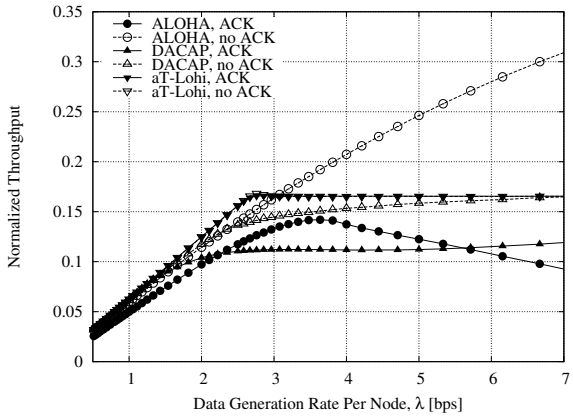

Figure 9: Throughput as a function of traffic for all protocols, FH-BFSK, $L = 50\,\mathrm{Bytes}$.

not necessarily yield better throughput, but rather decreases it in this case: the reason is the even longer waiting time between transmissions; moreover, ACKs may collide with data packets or other ACKs, and if the ACK is lost the transmitter resends the packet, thus decreasing the efficiency of the scheme. We remark that this is not a claim against ACKs: they do increase the success ratio (not shown here due to lack of space) for most protocol configurations, but in this case the throughput benefit is not as attractive. The normalized throughput of the protocols with FH-BFSK is slightly higher with respect to BPSK, which reflects the better resilience to collisions of the non-coherent modulation scheme. We observe improvements for all protocols, especially DACAP and the ACK-based protocols in general. Thanks to FH-BFSK, there is a lower chance that signaling packets collide among themselves or with data, thereby limiting losses due to interference. In other words, throughput is low mainly because of idle waiting periods. The drawback of FH-BFSK is highlighted, however, by the values of traffic at which that throughput is obtained, which are one order of magnitude lower than with BPSK, due to the significantly smaller data rate available.

4.3 Results for event-driven traffic

This subsection aims at comparing ALOHA, DACAP and T-Lohi under event-driven traffic. To simulate events, we assume that a moving object traverses the network, triggering packet generation by nodes that detect the object nearby. Such packets are generated at the rate of 1 packet every 10 seconds whenever the object is within the detection range of a node, whereas no packets are generated when the object is out of this range. For simplicity, we assumed that the event detection range of a node is fixed and equal to $1.5\,\mathrm{km}$, which adequately covers the area using the same 10-node topology considered before. Note that this is different from the communication range, which is instead unvaried and large enough to have all nodes be within reach of the sink. Due to lack of space, we will focus only on the use of FH-BFSK modulation, with packet size $L = 50\,\mathrm{Bytes}$.

Unlike in the previous evaluation, throughput is not of primary importance here: hence, we focus on the time of arrival of the first (correct) packet triggered by the event detection, regardless of the corresponding ACK (a measure of the readiness of a protocol), and the time of arrival of the last packet (a measure of the ability of a protocol to handle bursty traffic effectively). Both ACK and no ACK versions of the protocols are considered; ACK versions are fully reliable (i.e., packets are retransmitted until correct detection is confirmed). For the no ACK versions, we also measure the packet error rate. These results are reported in Table 1.

Protocol	First arrival [s]	Last arrival [s]	Error rate
ALOHA (no ACK)	1.15	29.4	0.44
DACAP (no ACK)	2.07	168	0.091
aT-Lohi (no ACK)	1.48	121	0.003
ALOHA (ACK)	1.15	113.1	—
DACAP (ACK)	2.07	251	—
aT-Lohi (ACK)	1.48	133	—

Table 1: Protocol performance in the event-driven traffic scenario using FH-BFSK, $L = 50\,\mathrm{Bytes}$.

We observe first that ALOHA does not require any preliminary signaling, therefore its first arrival time is the lowest, both with and without ACKs. Let us focus on the last arrival time now: if ACKs are used, ALOHA requires $113\,\mathrm{s}$ to complete all transmissions, whereas the other handshake-based protocols require longer times. These first observations are expected, but we observe that our analysis provides a quantitative evaluation of the timings and error rates. A second observation is that, while no ACK ALOHA bears an unacceptable error rate, the no ACK versions of the other protocols benefit from handshakes to achieve a lower error rate at the price of a higher completion time (last arrival). In particular, aT-Lohi's errors are on the order of 0.3%, which corresponds to a completion time of $121\,\mathrm{s}$, close to that of ACK ALOHA. DACAP yields instead a 9% error rate, and takes longer to complete. This is in line with throughput results (see Fig. 9), and can be explained with the larger chance of collision among packets (both signaling and data) caused by bursty traffic.

We highlight that the best choice for bursty traffic (ALOHA with ACKs) is not the best one under Poisson traffic (see Fig. 4). Therefore, if we had to implement one protocol to handle both scenarios, both DACAP and T-Lohi may represent acceptable choices, especially if the application running on top of the protocols can withstand error rates on the order of 10%. Alternatively, we may modify ALOHA such that packets requiring ACKs (e.g., those generated by specific events) are flagged, making it possible for the good throughput of no ACK ALOHA to coexist with the good performance of ALOHA with ACKs under event-driven traffic. We also note that other environmental effects should be considered here: e.g., multipath may generate multiple tone arrivals, distorting the way T-Lohi counts contenders. Both lines of research are left for future work.

Protocol	Poisson traffic	Event-driven traffic
ALOHA (no ACK)	**Good**: best throughput, though low success ratio	**Bad**: shortest timing, but worst error rate
DACAP (no ACK)	**Average**: limited throughput, high success ratio	**Average**: acceptable error rate and completion time
aT-Lohi (no ACK)	**Average**: fair throughput, high success ratio, low overhead	**Good**: very low error rate, takes slightly longer than ALOHA
ALOHA (ACK)	**Bad**: fair throughput but lowest success ratio	**Good**: shortest completion time with full reliability
DACAP (ACK)	**Average**: low throughput but good success ratio	**Bad**: longest completion time
aT-Lohi (ACK)	**Bad**: lowest throughput not balanced by success ratio	**Average**: Completion time not as good as ALOHA's
Best protocol(s)	**ALOHA-no ACK** (for throughput), others (for different metrics)	**ALOHA-ACK** (best timing), **T-Lohi-no ACK** (very close timings at 0.3% error rate)

Table 2: Summary evaluation of protocol performance under Poisson and event driven traffic.

5. CONCLUSIONS

In this paper, we have described a simulator for underwater networks which incorporates a ray tracing tool for a more realistic reproduction of underwater propagation. The simulator is integrated with free world databases for environmental parameters, allowing the user to easily specify the operational area of the network as well as the desired time of the year to be simulated, as both affect the outcome of the simulation. As a case study, we have then employed the simulator to compare ALOHA, aT-Lohi and DACAP, three protocols for underwater networks bearing different levels of coordination among nodes. We have considered both periodic and event-driven traffic, showing that the best protocol in one case is not necessarily the best in the other case. The summary of our conclusions is reported in Table 2.

Acknowledgment

This work has been supported in part by the NATO Undersea Research Center under contracts no. 40800700 (ref. NURC-010-08) and 40900654.

The authors would like to thank those people at NURC who have contributed to the genesis and improvement of the ideas behind this work: Kim McCoy and Giovanni Zappa, for inviting the authors at the workshops where the JANUS protocol was being defined (see [24]), for the many ensuing discussions and useful tips, as well as for the chance to work with the JANUS group as NURC visiting researchers; Alessandro Berni, Diego Merani and Robert Been, for being first points of contact within NURC.

6. REFERENCES

[1] A. Syed *et al.*, "Comparison and Evaluation of the T-Lohi MAC for Underwater Acoustic Sensor Networks," *IEEE J. Select. Areas Commun.*, vol. 26, pp. 1731–1743, Dec. 2008.

[2] B. Peleato and M. Stojanovic, "Distance aware collision avoidance protocol for ad hoc underwater acoustic sensor networks," *IEEE Commun. Lett.*, vol. 11, no. 12, pp. 1025–1027, Dec. 2007.

[3] M. Zorzi, P. Casari, N. Baldo, and A. F. Harris III, "Energy-efficient routing schemes for underwater acoustic networks," *IEEE J. Select. Areas Commun.*, vol. 26, no. 9, pp. 1754–1766, Dec. 2008.

[4] C. Schurgers *et al.*, "Underwater networking projects." [Online]. Available: http://circuit.ucsd.edu/~curts/wisl

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

[6] 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.

[7] ns2 Network Simulator, http://www.isi.edu/nsnam/ns/.

[8] N. Baldo *et al.*, "NS2-MIRACLE: a modular framework for multi-technology and cross-layer support in network simulator 2," in *Proc. of NSTools*, Nantes, France, 2007.

[9] "Ns2-miracle source code download page." [Online]. Available: http://telecom.dei.unipd.it/pages/read/58/

[10] M. Porter *et al.*, "Bellhop code." [Online]. Available: http://oalib.hlsresearch.com/Rays/index.html

[11] L. G. Roberts, "ALOHA packet system with and without slots and capture," *ACM SigComm Computer Communication Review*, vol. 5, no. 2, pp. 28–42, 1975.

[12] F. B. Jensen *et al.*, *Computational Ocean Acoustics*, 2nd ed. New York: Springer-Verlag, 1984, 2nd printing 2000.

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

[14] "Model for underwater channel in ns2," 2008. [Online]. Available: http://telecom.dei.unipd.it/download/

[15] "World ocean atlas." [Online]. Available: www.nodc.noaa.gov/OC5/WOA05/pr_woa05.html

[16] H. Schmidt *et al.*, "GOATS 2008: autonomous, adaptive multistatic acoustic sensing," Massachusetts Institute of Technology, Tech. Rep., 2008. [Online]. Available: www.onr.navy.mil/sci_tech/32/reports/docs/08/oaschmi3.pdf

[17] "General bathymetric chart of the oceans." [Online]. Available: www.gebco.net

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

[19] P. C. Etter, *Underwater acoustic modeling and simulation*, 3rd ed. Spon Press, Taylor & Francis group, 2003.

[20] J. G. Proakis, *Digital Communications*, 3rd ed. McGraw-Hill, 1995.

[21] "Teledyne benthos undersea systems and equipment," www.benthos.com.

[22] L. Freitag *et al.*, "The WHOI Micro-Modem: An Acoustic Communications and Navigation System for Multiple Platforms," http://www.whoi.edu, 2005.

[23] K. McCoy, "JANUS: from primitive signal to orthodox networks," in *Proc. of IACM UAM*, Nafplion, Greece, Jun. 2009.

[24] "JANUS workshop proceedings." [Online]. Available: http://nrcsp.zftp.com/users/janus-tmp

[25] F. Guerra, P. Casari, and M. Zorzi, "MAC protocols for monitoring and event detection in underwater networks employing a FH-BFSK physical layer," in *Proc. of IACM UAM*, Nafplion, Greece, Jun. 2009.

[26] ——, "A performance comparison of MAC protocols for underwater networks using a realistic channel simulator," in *Proc. of MTS/IEEE Oceans*, Biloxi, MS, Oct. 2009.