Programming Acoustic Modems for Underwater Networking

Andrew Tu, *Student Member, IEEE,* Brian Wilcox *Student Member, IEEE,*, Mark German, Yashar M. Aval, *Member, IEEE,* and Stefano Basagni, *Senior Member, IEEE*

***Abstract*—Underwater acoustic communication and networks have attracted significant attention in recent years, with applications ranging from ocean monitoring to off-shore sensor control, and port surveillance. Experimental data are required to test and develop effective underwater networking protocols before underwater networks can be successfully deployed for real world applications. Unfortunately, there are very few permanent underwater acoustic testbeds currently in operation, making it difficult for full scale tests to be conducted. To meet the demands for experimental data, we are working to deploy a permanent underwater acoustic network at the Northeastern University Marine Science Center in Nahant, MA. At the final stage, the network will consist of at least five SM 975 Teledyne Benthos acoustic smart modems, with one wirelessly connected to the shore through a smart buoy of our design. This paper describes the interface for programming these modems and how we used it to implement a fundamental protocol to be used as performance benchmark for more advanced underwater solutions.**

I. Introduction

With over 70% of Earth surface covered in water, underwater communications and networking are critical technological developments with countless applications in the research and commercial sectors. Wireless underwater networks will enable the remote monitoring and control of sensors and equipment that can be used to collect marine data and analyze their patterns, thus fostering new applications for the sustainable exploitation of this still unknown realm of our planet. Prevailing forms of terrestrial wireless communications, especially radio, are ill adept for long distance underwater transmissions. As such, acoustic communication provides a crucial component towards implementing large scale underwater networks.

Currently, there are very few permanent testbeds for underwater acoustic networks (UANs) in the world, severely limiting experimental results on the design of effective networking protocols. Our end goal is to build a permanent testbed that will enable easier, more efficient testing of protocols for UANs at all layers of the protocol stack. Northeastern University is set to design, develop and deploy a permanent testbed at its Marine Science Center campus in Nahant, MA. The network, called NU MONET for Northeastern University Marine Observatory NETwork, will consist of at least five SM 975 Teledyne Benthos acoustic smart modems with one modem connected to the shore via a radio link to program and control the network and to relay data to the final user. A sketch of the planned NU MONET is depicted in Fig. 1.

A. Tu, B. Wilcox, M. German, Y. M. Aval and S. Basagni are with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA. Corresponding authors e-mail: tu.a@husky.neu.edu.

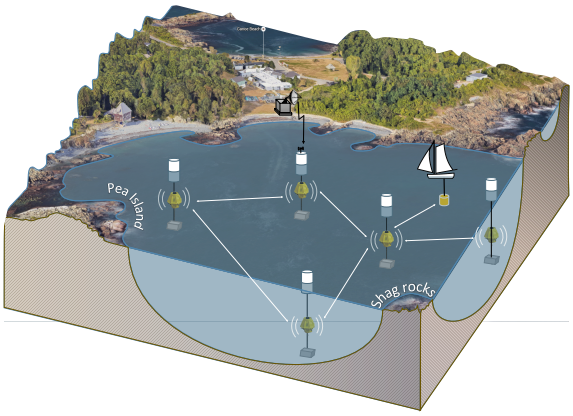

Fig. 1: The NU MONET to be deployed in Nahant, MA. The connections between the modems are underwater acoustic links (shown in white). One modem acts as the network gateway to the terrestrial Internet via a smart buoy (of our design). The connection between the buoy and the shore station is radio (ZigBee[1]-based; black line in the picture).

This paper concerns programming the Teledyne Benthos modems as a fundamental step towards building the NU MONET. Modem programming largely depends on the API provided with the devices. Teledyne Benthos modems, for instance, come with BenthoNet as the interface for implementing basic networking functionalities (see Section II). Other approaches to underwater network programming go well beyond the modem's relatively limited interface, and concern the use of suites for integrated simulation, emulation, and testbed trials of protocols at all levels of the networking stack. Examples of this approach are provided by tools such as SUNSET [1] and DESERT [2]. The SUNSET (Sapienza University Networking framework for underwater Simulation, Emulation and real-life Testing) framework was developed by a team at the University of Rome "La Sapienza." The package is designed to allow developers to easily conduct simulations, emulations and field tests of underwater communication protocols [1]. DESERT (DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols) is a set of C++ libraries extending the NS-MIRACLE simulation framework to support the development and implementation of UANs [2]. The key idea of both packages is that of using the same code for emulation and simulations of underwater protocol and also, once the physical layer is provided by actual modems, to

run that same code for experimentation in the water. These software packages are designed to be largely platform independent, in that given the appropriate driver, they can be used irrespective of the specific modem. The approach to modem programming described in this paper is of the first kind, i.e., built on the specific interface of the Teledyne Benthos modems. In particular, we develop Matlab-based usecase cases for controlling the modem from an outside device running Matlab (i.e., a laptop or other small-factor computer that can be deployed underwater with the modem itself), and with which we can implement different communication protocols. Our cases range from low-level, driver-like interfaces between Matlab and the hexadecimal-based commands of the modems, to more "autonomous" cases taking care of basic networking primitives, such as sending and receiving data, topology set-up, etc. As a case study, we show how we use these use cases to implement one of the simplest medium access control (MAC) protocols, namely, ALOHA, for which we show results obtained in water that are based on our implementation.

The rest of the paper is organized as follows. Section II describes the SM 975 Teledyne Benthos acoustic smart modem. In Section III we show how to control the modem via Matlab-based programming. In the following Section IV we demonstrate our programming primitives for implementing ALOHA and show some experimental results obtained through our implementation. Finally, Section V concludes the paper.

II. The SM 975 Teledyne Benthos Acoustic Modem

The SM 975 Teledyne Benthos Acoustic Smart Modem is one of the fundamental components of our project. The modem electronics are housed within a vacuum sealed glass sphere, nestled within a two piece polyethylene hardhat. The vacuum seal allows the modem to descend to depths up to 6,700 meters underwater. An omnidirectional transducer extends from the top glass hemisphere and through a cutout in the hardhat. The bottom of the hardhat has an electrolytic dissolving "burn wire" which allows the remote release of the modem from the sea floor.[2] Users can interact and control the modem via a proprietary serial port/power connection near the modem base. The modem contains a 28 Ah battery allowing the modem to run on battery life for up to one year between charges depending on usage. A sketch of the SM 975 is shown in Fig. 2.

The low frequency acoustic modem can emit sounds between 9-14 kHz and has two modes of operation. The first mode uses coherent detection and employs phase shift keying (PSK) modulation to deliver baud rates of up to 15,360 bits/sec, while the second mode is based on incoherent detection and employs frequency shift keying (FSK) modulation, delivering baud rates up to 2,400 bits/sec. Note that while FSK modulation cannot deliver data rates as high as PSK modulation, it has improved reliability as compared to PSK modulation, which makes it a practical solution for more challenging channels.

[2]The modem is positively buoyant and will ascend to the surface when released.

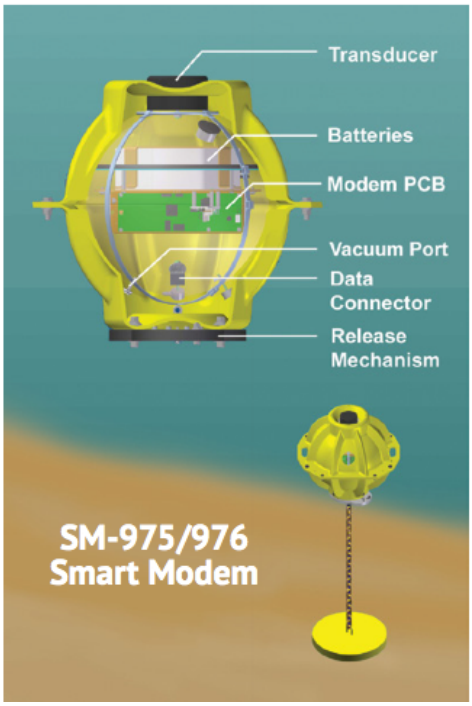

Fig. 2: A cut-away view of the SM 975 Teledyne Benthos Smart Modem.

The Teledyne modems come pre-loaded with BenthoNet, a software API that enables basic primitives for packet transmission and reception as well as simple networking functionalities, including automatic acknowledgment replies and automatic re-transmission of data packets. We have disabled most of these functionalities as they are not dynamically controllable. This has allowed us to implement channel access protocol with greater control over key parameters.

III. Programming the SM 975

The SM 975 modem is interfaced through a serial port to an external computer, i.e., a laptop or a small form-factor computer such as the BeagleBone Black [3]. Through this interface, we control and program the modem via a series of Matlab functions and scripts. We organize the code for the SM 975 in a series of layers as shown in Fig. 3.

Teledyne Benthos modems interact with the user via messages of four different types: "get," "set," "execute," and "notify." Get is used to read parameters from the modem (e.g., its local address, which is a positive integer in $\{0, 1, 2, \ldots, 63\}$); set is used for changing a parameter (e.g., setting the local address to a different number), and execute asks the modem to perform a certain task (e.g., send a ping). These three commands are always generated by the user. The notify message, on the other hand, is generated by the modem and carries the modem response to the computer (e.g., the response to a get request), or is used to indicate that an event has occurred (e.g., notify the user about the reception of a packet from another modem). These four message types form the entirety of communication modes to and from the modem.

Commands are sent to the modem in a string of hexadecimal bytes through the serial port connection. The available com-

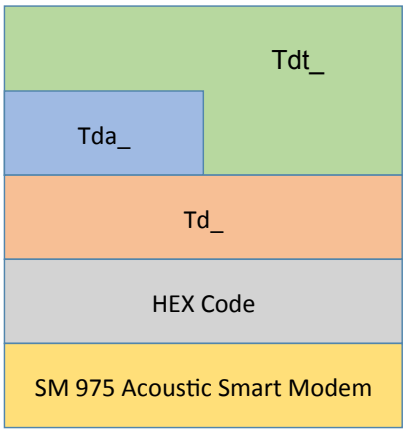

Fig. 3: The Matlab-based NU MONET code architecture for the SM 975. The users work with the "tdt_" layer that uses functions in lower layers to accomplish tasks.

mands are laid out in the proprietary manuals accompanying the SM 975 modems. In order to interface more easily with the modems, we implemented a Matlab base level code to send and receive bytes to/from the modem. This code takes in a command (e.g. "td_get(s,"localAddr", 1)") and converts it to the appropriate hexadecimal byte string.

With the base layer established, we build up a new layer that utilize the "td_", or "Teledyne", functions to perform more automated tasks. Functions in this layer of communication are prefixed by "tda_," where the appended "a" stands for autonomous. For instance, these functions may combine "td_get," "td_notify," and some additional logic for extracting information like the local address from the modem into a usable variable in a protocol code. Desired information returned in the modem response is extracted by the function and stored in memory.

On top of the "td_" and "tda_" layers we define a new layer, termed "tdt_" where the extra "t" is short for "timer." This layer is based on Matlab timers set to periodically trigger the execution of specific functions. Functions defined at the "tdt_" layer are used to implement several use cases for protocol implementation and testing. We describe here five fundamental basic cases. They concern sending a packet, receiving a packet, building the list of a node neighbors,[3] namely, a node "address book," functions for traffic management, and a GUI interface that we built in Matlab for run time control of relevant modem status parameters (e.g., the length of its data queue). Every case utilizes queues to which packets are pushed and pulled. The queues, and a set of global variables acting as "bridges" between case, allow the functions to pass information back and forth. For example, a new data packet generated by the traffic management case is pushed into the global transmit queue, from which the sending case will pop and transmit packets.

In the rest of this section we describe these cases and structures in details.

The data production and transmission case utilizes the transmission queue to pass and store packets. The transmission queue is a series of parallel queues with each queue specific to a modem in the address book. Having several parallel queues specific to a modems affords more versatility than one large queue for all data packets. By separating the packets by modem, the sending function can randomly select a non-empty queue to pull packets from and send. In the event that connection to a modem is lost (thus resulting in delays and dropped messages), the sending function can continue to operate normally via another queue without getting stuck waiting on the retransmission of the lost packet. Furthermore, if a modem is dropped from the network, any packets generated for that modem can be deleted more easily, without having to sift among many packets with different destinations.

We have functions for generating data packets (for instance, for testing purposes) which also include pertinent time stamps and diagnostic information. A data generating function is associated with its own timer. This function can be customized to implement different traffic generation patterns, such as Constant Bit Rate traffic, or random traffic generated according to a distribution relevant to a specific application. When a data packet is produced for transmission and its destination node is chosen, it is inserted into the transmit queue for that selected destination.

When the tx_timer function is called, the lengths of all the transmission queues are evaluated. If the sum of these lengths is greater than 0, (i.e., there are messages ready to be sent), the function generates random numbers between 1 and the current size of the address book to be used as indexes. The queue lengths of the randomly produced indexes are checked until an index with queue length greater than 0 is found. The function extracts the oldest message to be sent from the queue at the given index (i.e., the first message in that queue) and sends it, moving the message after transmission to a buffer queue and shifting all messages in the transmission queue forward. When an acknowledgment (ACK) is received from the recipient, the packet is deleted from the buffer. If an ACK is not received within a designated timeout, the modem will attempt to retransmit the message from the buffer a defined number of times, which is protocol dependent.

All notifications that a modem receives or generates sit in the serial buffer pending processing by the notification timer function "tdt_notify." When "tdt_notify" is called, the function immediately checks the length of the serial buffer to see if a new packet has been generated. If the length is greater than 0, (i.e., bytes for a notification packet have been generated by the modem and are sitting in the buffer), tdt_notify reads the available bytes into a packet. A signature is generated and assigned to the packet as it is processed. The signature, comprised of key elements of the packet, is used to efficiently discriminate and act upon packets depending on the contained data. The information received is interpreted by "tdt_notify" upon which it is either pushed to the appropriate queue or passed along to another function for further processing. The data stored in queues will be pulled by the corresponding timer function. This process can also work in reverse mode, in which "tdt_notify" reads from the queues of the other functions, and forwards the information to the modem.

[3] Two nodes are said to be neighbors if they can communicate directly to each other, i.e., when there is a physical channel between them that they can use to exchange information.

Each modem has an address book containing the addresses, distances, and other pertinent information of every modem within transmission range. New modems can be added to an address book in one of three ways. The first is through a "ping" message broadcast to all modems within transmission range. Any modems who hear the ping respond with their own address and distance at a random time (up to time t_{ping}). After the timeout period, the ping originator produces a notification containing the addresses and distances of all the modems who responded. The function "Tdt_notify" identifies this notification based on its signature, and passes the packet into another function, "tdt_AddToAddrBook," with a flag indicating a ping response. The flags "Ping," "justAddr," or "Range" specify how the information is being presented to the function so that it can be properly extracted. In particular, "Ping" may contain multiple addresses and distances, "justAddr" only contains a single integer pertaining to the extracted address, and "Range" contains both address and distance updates. The other ways addresses can be added to the address book involve overhearing modems talking to one another. When a message is transmitted from a modem, the modem also sends several system messages with information such as timestamps, range updates, and acoustic information. Any modem that overhears these messages generates notifications indicating their receipt, but ignores the actual data packets unless they are addressed specifically to them. Through signature identification, "tdt_notify" can pass information from certain packets to "tdt_AddToAddrBook" tagged with either "justAddr" or "Range," depending on the nature of the receipt. When packet information is passed to "tdt_AddToAddrBook," the address is first checked against the current entries in the address book. If an address already exists in the address book, the function simply updates the modems distance (if applicable). If the address is new, all pertinent information (address, distance, time last heard from, etc.) will be added to the address book.

We developed a GUI in Matlab to display the contents of the address books of a node, the lengths of its queues and other status information. The GUI is updated every .5 seconds through a timer. Two curves are depicted that graph the length of the tx_queue with respect to time and the end-to-end delay for each received data packet. Additionally, the GUI offers several buttons and sliders to input and change modem settings mid run. This includes a slider to set how often packets are generated, toggles for enabling data generation and sending pings, and buttons to clear the TX queue and send packets.

IV. Case Study: ALOHA

We demonstrate the use of our primitives for programming the SM 975 by implementing a basic MAC protocol for channel access, namely, ALOHA.

The ALOHA protocol is one of the earliest and most basic random access protocols proposed for wireless broadcast channels. In the original version of this protocol, packets are sent in their entirety immediately upon generation, indiscriminately and without limiting factors [4]. ACKs are sent back to the sending nodes to confirm the successful delivery of a packet. In the event of packet collision, the sending nodes will immediately attempt to retransmit their packets with probability p, opting to remain silent with probability of $1-p$. The randomness in packet retransmission decreases the probability of repeated collisions. Nodes will attempt to retransmit a total of *retransmission_attempts* times before discarding the packet.

Fig. 4 shows the set of actions performed within each of the SM 975 modems for generating a data packet and transmitting it according to our Matlab implementation of ALOHA.

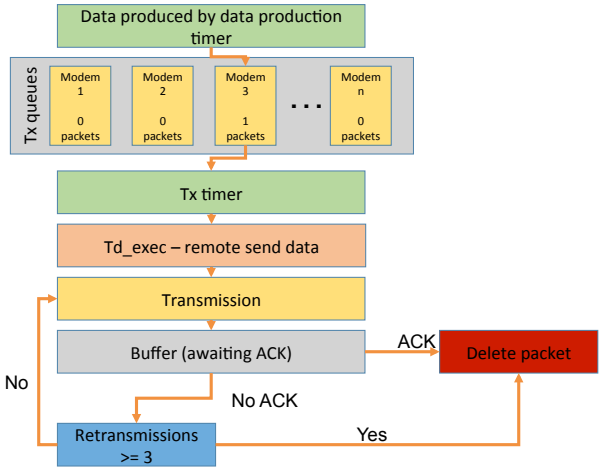

Fig. 4: The path of a data packet from production to deletion after transmission in our implementation of ALOHA.

The data production timer generates a packet and places it in one of the transmission queues (tx_queue) of a selected destination, which is any one of the other $n-1$ modems. In our experimentation below data packets are generated according to a Poisson distribution, while the data packet destination is any of the other modems, selected randomly and uniformly. All transmissions and retransmissions are coordinated through the tx_timer. When the tx_timer is called, the generated packet is transmitted by passing it into a "td_exec" remote send data command, which converts it into the appropriate hex string for the modem. After transmission, the packet is moved from the tx_queue into a buffer queue until an ACK is received from the recipient. If an ACK is not received within a designated timeout, it is assumed that the packet did not reach its intended destination and the modem will attempt to retransmit the packet. The modem will then attempt to resend the packet with probability p. The probability of a successful transmission is derived in [5] as $p_{success} = p(1-p)^{2(n-1)}$, where n is the number of nodes in the address book. The probability of a successful transmission is maximized by setting the (re)sending probability to $p = \frac{1}{2n-1}$. The packet is then transferred to the end of the buffer queue while it awaits an ACK. With probability $1-p$, the modem will not attempt to retransmit and will remain silent for the duration of a packet before attempting to retransmit again. This process repeats until the message is transmitted successfully and the sender receives an ACK from the recipient, at which point the packet is deleted from the sender memory. The packet can also be dropped and deleted from memory if it fails to be successfully transmitted within a predefined number of attempts (set to three in our experimental evaluation and in Fig. 4).

We use our implementation to conduct underwater experiments with three SM 975 acoustic smart modems running ALOHA. We deployed the modem in an indoor rectangular tank filled with fresh water positioned in a line (Fig. 5).

Fig. 5: Three TB SM 975 acoustic modems. In this lab setting the devices are controlled by laptops outside the tank.

We investigate the metrics of *packet delivery ratio*, i.e., the percentage of packet successfully delivered to their intended destination, as well as the end-to-end delay incurred by packets from the moment they are created to when they are delivered. The results of the performance evaluation of ALOHA are shown in Fig. 6.

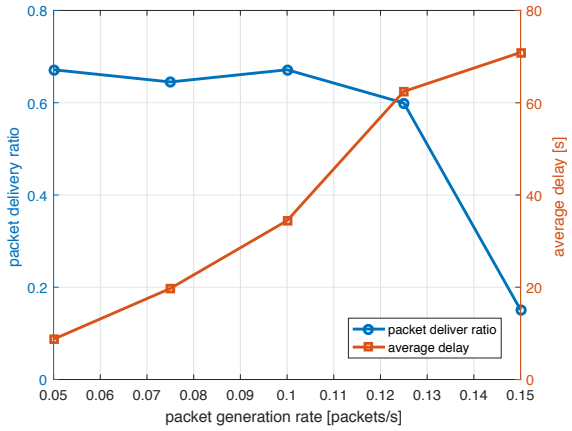

Fig. 6: Packet delivery ratio and end-to-end delay for the ALOHA MAC protocol in the setting of Fig. 5.

We observe that as the rate of packet generation increases, the packet delivery ratio decreases while the end-to-end delay greatly increases. As for its terrestrial counterpart, the simplistic nature of ALOHA used under water leads to very low channel utilization.[4] This is the consequence of not having an effective collision detection/avoidance mechanisms, like those used by CSMA/CD and /CA-based protocols. Curiously, the higher propagation delay of the underwater channel helps improving channel utilization. In fact, two senders at different distances from the same receiver and transmitting a packet at the same time are less likely to incur collision at the receiver than if the networks was terrestrial (radio), where the speed of light will instead lead to certain collision. Unfortunately, we did not benefit from this effect as our current testbed setting restricts us to very similar distances between modems. However, future experiments that run in open water (i.e. during final deployment in the ocean) may benefit from this effect due to the more variable distances between modems.

[4] It is notoriously known that the maximum channel utilization of un-slotted ALOHA in radio networks with saturated traffic is about 18% [6].

V. Conclusions and Future Works

We used Matlab-based programming to develop a series of use cases for controlling and programing Teledyne Benthos SM 975 Acoustic Smart Modems. We demonstrate how to use these cases by implementing a basic channel-access networking protocol, ALOHA, and running underwater experiments. Eventually, we plan to case these basic use cases for developing more complex modern protocols designed specifically for use in UANs. We have started implementing one such protocol (TARS) and hope to begin testing soon. The TARS (Traffic Adaptive Receiver Synchronized) protocol is a more complex protocol designed specifically for underwater networking that will be implemented via the cases outlined in this paper. Once we have established a stable base of protocols, we hope to expand our small scale underwater testing to the ocean, and collect more realistic results for further analysis. In addition to programming the modems, we are also working on other portions of the NU MONET in preparation for a full deployment of five modems at the Northeastern Marine Science Center campus in Nahant, MA, over the Summer of 2016. This includes the design and construction of a "smart buoy" that will serve as the wireless link to the shore and provide power to the modems.

Acknowledgments

This research was supported in part by the NSF MRI grant CNS 1428567. Tu, Wilcox and German were supported by REU grants associated to this MRI award. Andrew Tu and Stefano Basagni were supported by NSF through a GENI SAVI travel grant. The work was also performed under the sponsorship of the EU FP 7 ICT project SUNRISE "Sensing, monitoring and actuating on the UNderwater world through a federated Research InfraStructure Extending the Future Internet."

Biographies

Andrew Tu is a rising second year undergraduate student at Northeastern University majoring in computer engineering and computer science. He is a student member of the IEEE and a member of Toastmasters International. Tu has been an NSF REU supported student on the NU MONET project since early October 2015. During the Summer of 2016, Tu was sponsored through a GENI SAVI travel grant to work on the NU MONET project in Rome, Italy.

Brian Wilcox is a senior in Electrical and Computer engineering co-oping at MIT Lincoln Laboratory. He is also an IEEE student member and the Vice-President of Northeastern's Eta Kappa Nu chapter.

Mark German is entering his third year in the Computer Engineering program at Northeastern, and currently in a six month co-op at Schlumberger. He received an award for Outstanding Student Research in Engineering and Technology at Northeastern's RISE 2016 event.

Yashar M. Aval graduated from the University of Tehran, Tehran, Iran, in 2002 and received the M.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2005, and his Ph.D. degree in electrical engineering at Northeastern University, Boston, MA, USA, in 2015. Currently, he is working as Associate research engineer at Northeastern university. His research interests include digital communications, OFDM, underwater acoustic communications, and underwater acoustic networks.

Stefano Basagni is associate professor at the Department of Electrical and Computer Engineering at Northeastern University, in Boston, MA. He holds a Ph.D. in electrical engineering from the University of Texas at Dallas (December 2001) and a Ph.D. in computer science from the University of Milano, Italy (May 1998). Dr. Basagni's research interests concern research and implementation aspects of mobile networks and wireless communications systems, radio and acoustic sensor networking, definition and performance evaluation of network protocols and theoretical and practical aspects of distributed algorithms. Dr. Basagni has published over six dozens of refereed technical papers and book chapters that are highly cited (his h-index is currently 32, with over 8500 citations to his works). He is also co-editor of three books. Dr. Basagni serves as a member of the editorial board and of the technical program committee of ACM and IEEE journals and international conferences. He is a senior member of the ACM (including the ACM SIGMOBILE), a senior member of the IEEE (Computer and Communication societies), a member of ASEE (American Society for Engineering Education) and of CUR (Council on Undergraduate Research).

REFERENCES

[1] 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 *Proceedings of ACM WUWNet 2013*, Kaohsiung, Taiwan, November 11–13 2013, pp. 1–8.

[2] R. Masiero, S. Azad, F. Favaro, M. Petrani, G. Toso, F. Guerra, P. Casari, and M. Zorzi, "DESERT underwater: an NS-miracle-based framework to DEsign, simulate, emulate and realize test-beds for underwater network protocols," in *Proceedings of IEEE OCEANS 2012*, Yeosu, Korea, May, 21–24 2012, pp. 1–10.

[3] What is beaglebone black? [Online]. Available: https://beagleboard.org/black

[4] N. Abramson, "The ALOHA System—another alternative for computer communications," in *Proceedings of AFIPS 1970*, 1970, pp. 281–285.

[5] J. F. Kurose and K. W. Ross, *Computer Networking—A Top-Down Approach Featuring the Internet*, 6th ed. Addison Wesley, 2013.

[6] L. G. Roberts, "ALOHA packet system with and without slots and capture," Stanford Research Institute, Advanced Research Projects Agency, Network Information Center, Stanford, CA, Tech. Rep. ASS note 8, June 1972.