Aqua-Sim Next Generation: An NS-3 Based Underwater Sensor Network Simulator

Robert Martin
Computer Science & Engineering Department
University of Connecticut, Storrs, CT
robert.martin@engr.uconn.edu

Sanguthevar Rajasekaran
Computer Science & Engineering Department
University of Connecticut, Storrs, CT
rajasek@engr.uconn.edu

Zheng Peng*
Computer Science Department
City College of New York, New York, NY
zheng@cs.ccny.cuny.edu

ABSTRACT

Through the use of network simulators we are able to test and examine different combinations of protocols in low cost and controlled environments. To ensure the accuracy of these simulators it is crucial that they consistently expand and enhance their modules to offer extensive support. In this paper, we introduce Aqua-Sim Next Generation, an NS-3 based underwater sensor network simulator. This work is an expansion of Aqua-Sim, transferring to the newer core simulator, NS-3, for enhanced memory management and performance. Additionally, we revitalize legacy supported protocols through better packet handling and header usage. Aqua-Sim Next Generation offers real-world features to close the gap between simulation and real-system tests while providing additional modules for easier user development. With the goal to create a more efficient and functional simulator, we improved Aqua-Sim's architecture. Furthermore, we integrate a specialized underwater information-centric module to assist in simulating innovative techniques in underwater sensor networks. Experimental results show that Aqua-Sim Next Generation runs twice as fast compared with Aqua-Sim while showing large increases in memory management.

KEYWORDS

Underwater Simulator; Underwater Acoustic Network; Protocols; Acoustic Propagation

1 INTRODUCTION

Through the use of network simulators we are able to test and examine different combinations of protocols in low cost and controlled environments. This is especially important for Underwater Sensor Networks (UWSNs) due to their expensive and complex deployment and upkeep costs for real-system testing. Due to the unique research challenges seen in UWSNs [1][5], this requires a standard simulation platform to support testing of different designs, algorithms and protocols. Furthermore, we must ensure that this tool offers a broad range of specialized features to allow for easier and expandable testing.

Aqua-Sim [18] was originally developed on NS-2 [13], as a expandable open-source underwater module. Since its release, Aqua-Sim has continuously expanded its supported protocols on top of NS-2. Like any simulator, NS-2 has its limitations and was superseded by NS-3 [14] in 2008. Aqua-Sim also suffers from many limitations such as poorly arranged architecture, steep user learning curve, restricted real-system module support, and poor memory performance.

To resolve these shortcomings of Aqua-Sim, we propose Aqua-Sim Next Generation (NG), a UWSN simulator based on NS-3. The goal of this core simulator changeover is to improve memory management, through the use of smart pointers, have stronger packet header handling, and overall performance improvements. Consequently, this transition allows us to restructure the architecture of Aqua-Sim and better modularize the simulator's protocol layers. Due to the vast differences between NS-2 and NS-3, this change also requires rewriting Aqua-Sim core and protocol code to meet new API standards.

Due to the harsh conditions of UWSNs it is crucial to simulate components of real-world systems to ensure strong testing results. To accomplish this task we have implemented four main areas of improvements in Aqua-Sim NG. (1) We first offer enhanced channel support which consists of specialized noise generators, multiple channel support, range-based propagation, and trace driven testing. These features allow for localized packet interference as well as varying acoustic propagation ranging between modems. (2) Next we expand physical model support through the implementation of a signal cache, signal-to-interference-plus-noise (SINR) checker and modulation support. This allows for packet decoding at the modem's physical layer dependent on the current underwater channel's condition. (3) Following this, we offer higher protocol layer support which includes synchronization and localization modules, busy terminal model support, and security features. Due to the restricted attenuation of radio frequency in UWSNs, synchronization and localization modules must be included to assist testing of various protocols. Furthermore, due to the high mobility and transmission delays in underwater it is important to be capable of testing specialized security techniques. (4) And the final area of improvement being the integration of adapted information-centric (IC) techniques for UWSNs. Due to the innovative approach of IC architectures, it is highly promising to implement various components and features for future testing. IC's innovative approach opens up new avenues of research in UWSNs, making it a crucial step in expanding underwater simulation.

*Dr. Zheng Peng is the corresponding author.

WUWNET'17, WUWNET'17: International Conference on Underwater Networks & Systems, November 6–8, 2017, Halifax, NS, Canada

ACM ISBN 978-1-4503-5561-2/17/11…$15.00
https://doi.org/10.1145/3148675.3148679

The remainder of this paper is as follow. First we will discuss related simulators and the core differences between NS-2 and NS-3 in Section II. In Section III, we present the redesigned architecture of Aqua-Sim NG. Next, we describe the newly implemented features and added modules in Section IV. Finally, we will present evaluation results in Section V before concluding our work in Section VI.

2 RELATED WORK

Aqua-Sim follows the object-oriented design of NS-2 to implement a complete protocol stack from physical layer to application layer. Aqua-Sim can effectively simulate acoustic signal attenuation and packet collisions in underwater sensor networks. This simulator implements three-dimensional deployment and mobile networks, while supporting a broad range of MAC and Routing protocols.

While NS-3 does provide an underwater module, UAN, it is currently limited to four MAC layer protocols and does not support routing layer components. While many of the routing and MAC layers of Aqua-Sim NG could be ported to UAN, the core underwater simulation of this module is lacking in many features, greatly restricting the usage and expandability for users. For example, UAN allows a single channel for each simulation restricting specialized situations such as varying channel conditions or differentiated frequency transmission paths. Furthermore, UAN overlooks other necessary components of UWSNs such as node synchronization and localization support. SUNSET [15] and DESERT [11] are underwater simulator, emulator, and real-life testing frameworks. In SUNSET and DESERT, the authors built their simulator on NS-2 with supporting libraries from NS-2-Miracle [3]. WOSS [7] is a framework focused on simulating acoustic propagation of underwater networks. WOSS also uses supported libraries from NS-2-Miracle as well as Bellhop [12], which is a sound propagation simulator based on ray tracing. UNetStack [2] is a Java/Groovy implemented underwater networking stack. It is important to note that the majority of the aforementioned frameworks are NS-2 based, while many have either transitioned towards library extensions, like NS-2-Miracle, or to NS-3 (UAN-WOSS is under NS-3 code review).

By comparing NS-2 and NS-3 we can start to see some of the main differences between these two core simulators. In NS-2 we see a combination of languages, where C++ is used for the core simulator and oTcl is used for scripting, to reduce recompile time. One of oTcl's shortcomings is with scalability overhead incurred during larger simulation runs. In NS-3 C++ is the single programming language, with some support in Python for visualization. For memory management, NS-2 uses standard C++ management functions, while NS-3 offers object handling and tracking. To properly manage memory allocation, NS-3 uses smart pointers. These smart pointers automatically de-allocate objects, through reference counting, to assist in reducing memory leaks. While smart pointers are not new, due to C++ having proper pointer deallocation tools, they do offer easier usage due to their inherited API design. This allows for maintainable integration from developers and users for more consistent memory management across all protocols. Since NS-2 allocates all defined headers for each packet based on the given protocol, we see excess header assignment in cases of multiple packet types. For NS-3, each packet consists of a single buffer, allowing the developer to manipulate each packet using only necessary headers. Taking everything into account for these two simulators, NS-3 has the potential to outperform NS-2. Another important factor to note is that NS-3 offers an emulation mode, allowing for integration with real networks in future Aqua-Sim NG iterations.

3 AQUA-SIM NG ARCHITECTURE

Due to the vast differences between NS-2 and NS-3 API, Aqua-Sim must be completely revamped to meet these new standards. In some cases this consists of rewriting new functions and complete classes to adhere to NS-3 standards. Since Aqua-Sim has been a collaborative effort among many developers over the past years, this transition also allowed for code clean up and restructuring.

3.1 Core Overview

Recoding the core architecture of Aqua-Sim was necessary when converting our simulator to NS-3. In order to complete this task, we first had to transition all previous NS-2 related events to NS-3 related API standards. This consists of changing how events are scheduled as well as introducing NS-3's smart pointer template. This template ensures proper reference counting allowing for much simpler memory allocation and deleting. Smart pointers offer a crucial step in reducing memory leaks, which is something that plagued Aqua-Sim in the past. Furthermore, we were able to take advantage of NS-3's Timer class, allowing for better control of canceling preset timers once destroyed. Beyond these API implementations, recoding Aqua-Sim NG opportunistically allows us to consolidate redundant classes. One main example is the removal of Node and Sink class, from Aqua-Sim, and instead reduced to a single class, AquaSimNetDevice. By using AquaSimNetDevice (a child class of NS-3's NetDevice) we allow the user or developer to decide how each node should be handled through populating each given device. One main issue that arose during this step of our work was ensuring better restriction among cross-module communication. To overcome this challenge we restructured Aqua-Sim's architecture into well formed logical modules to create visual and coherent modularity among our simulation structure.

In Fig. 1 we see the layout of the new Aqua-Sim Next Generation architecture. By allowing all protocols to interact with AquaSimNetDevice, we allow a customizable central hub for users to adapt their simulation and fully support modularity within each layer. Furthermore, it allows easier integration and development of new protocols and layer specific features. Additionally, Aqua-Sim NG continues to support the previously supported attenuation model originally adopted from [4]. Further analysis of this integrated model can be found in [18].

3.2 Protocol Overview

To adhere to the strong inheritance between base and child classes in NS-3, our first goal was to reassign MAC and Routing layer components. To accomplish this task, we created base classes which primarily control packet flow and tracing components. Child classes can independently handle and create packets uniquely based on their rules and developed goals. In this way, we see a fluid modular layout that is easily expandable and quicker to debug. Furthermore, it allows level specific tracing and logging. To properly integrate packets, we had to uniquely create each header class based on the

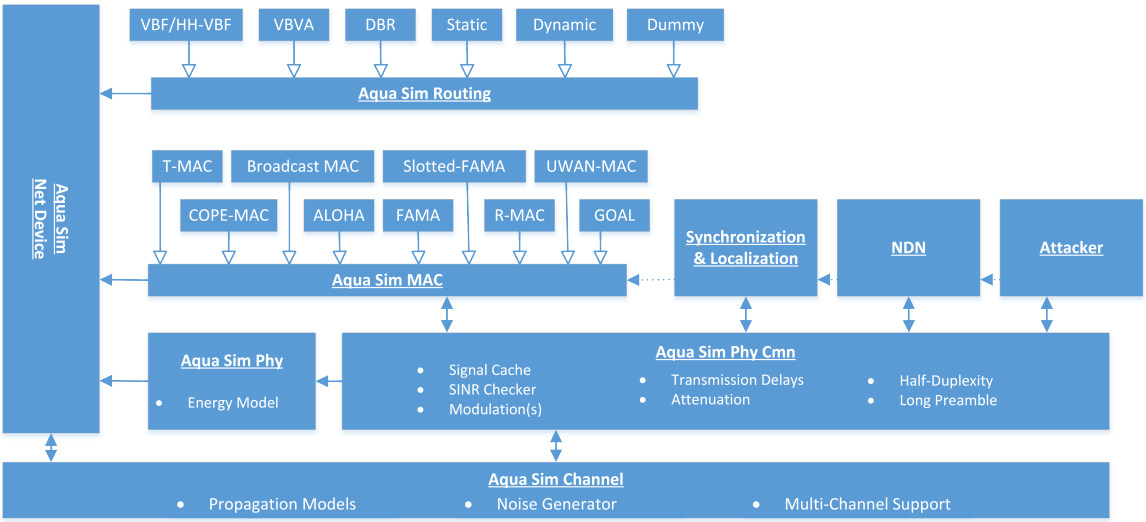

Figure 1: Aqua-Sim Next Generation architecture.

single buffer component of NS-3. This required construction of new header classes and editing header usage within each protocol. More importantly, this required proper buffer management among layer passing in order to ensure correct reading of header bytes. For example, if a packet is encoded with two headers, before sending, we must ensure that on a receiving node these two headers are decoded in the proper single buffer order. Many issues emerged during this phase of Aqua-Sim NG due to interpreting the different protocols and coding styles previously created on Aqua-Sim. This lead to many questionable areas of Aqua-Sim's protocol creation that had to be restructured and debugged thoroughly. Furthermore, we had to ensure constrained header usage among simulation layers to restrict against illegal buffer access. While these header buffer accesses may not lead to simulator crashes, since they technically may not be trying to access out-of-bounds memory, they are accessing headers in illegal order causing unexpected values and false simulation results.

Beyond meeting the API requirements, other details were needed in transitioning Aqua-Sim to NS-3. One basic modification was rewording variable and function names to meet coding styles of NS-3. While trivial sounding in nature, this required replacement without breaking the predefined code.

Also important to note that many new protocols are being introduced in this iteration of Aqua-Sim. Much of these protocols were anticipated for Aqua-Sim 2.0 release, but due to the transition of Aqua-Sim to NS-3, this work was placed on hold and instead transfered to Aqua-Sim NG. This ensures that Aqua-Sim NG will have a more expansive protocol pool for user testing and development.

To complete this switch to NS-3 we also had to adapt the front end of the simulator. This entailed restructuring previous oTcl scripts to function properly in C++. These scripts typically are used to create the simulation scenario, assign all protocols in use, and drive the simulator. Additionally, multiple new test scripts had to be created, from scratch, to ensure proper functionality of all simulation layers. To reach this goal we also had to create helper scripts. These scripts are used to assist in fully populating each component of the simulation and applying protocol settings. An example of this can be seen within AquaSimHelper, in which a user can create and populate a new AquaSimNetDevice with all provided protocol layers by simply calling the Create function.

4 NEW FEATURES & MODULES

4.1 Channel Support

Due to the diverse nature of UWSNs it is important to properly support channel features. Furthermore, conditions can consistently change among a given channel consequently effecting acoustic transmissions. With increasing popularity among UWSNs, the need to simulate these type of varying channel scenarios increases. To accommodate this demand, we can break our new channel features into four main areas: noise generator, multiple channel model support, range-based propagation model, and trace driven support.

Noise generators are an essential part in acoustic communication due to how devastating external noise can be for properly receiving transmissions [8]. Applications such as marine life, weather, boating, and malicious nodes can take part in introducing external noise into an UWSN. Three different generators are introduced to help replicate real-system noise. The first being a basic noise generator. In this class we use Urick's equation for acoustic noise [17]. This model uses water turbulence, wind, shipping noise, and water temperature to determine current noise in decibels. Next we incorporate random noise. This generator allows a user to set the noise bounds, independent of environmental factors, which introduce time-varying noise variation. Lastly, we implement a periodic noise generator, to create noise for a given length after a given period of time. This is meant to allow fluctuating channel noise similar to marine life or different types of network attacks. For example, a malicious node could try jamming the acoustic channel by periodically creating noise to block all other transmissions from sending or decoding messages.

To encourage simulation that mimics underwater features in Aqua-Sim NG we have expanded the channel model to support multiple channels. Since modems and networks can span over multiple kilometers, a channel's conditions may vary dependent on locality. Therefore, it is beneficial to allow multiple channels within a single simulation. This expansion will allow users to use different propagation models and noise generators with varying values and assigned devices. Importantly, this encourages the use of mobile nodes in a more adaptable environment.

In the past, Aqua-Sim has supported two versions of propagation transmission range. The first is based on sender's transmission power compared with the receivers power level threshold to determine if a given packet can reach the receiver. While the second channel propagation range directly sets the modem's maximum transmission range. The downfall in both of these propagation models is that we are assuming a static acoustic transmission speed of 1500 m/s. To ensure better accuracy in Aqua-Sim NG, we implemented a varying propagation model. This allows transmission range to be uncertain until time of modem transmission, dependent on current channel conditions to properly implement sound speed profiling. To accomplish this we incorporated Mackenzie's equation for acoustic speed [9]. By using modem depth, water salinity, and temperature this equation gives us current ocean acoustic speed in m/s. Additionally, we added Urick's signal-to-noise-ratio (SNR) model [17]. This can be summarized as the following:

$$SNR = TSL - TL - NL \tag{1}$$

where TSL is the transmitting power level of the source in decibels, TL is the transmission loss between the sender and receiver, and NL is the total noise of the channel.

To help enable consistent, real-world testing, we added a channel feature named trace driven support. This component is meant to allow a user to input real-world data specialized for the channel model. Trace driven support creates a static baseline for multiple protocol testings, by keeping channel conditions consistent between tests. More importantly, this model allows channel conditions of varying values to be easily input and simulated without a large overhead on the user or Aqua-Sim NG.

4.2 Physical Support

To ensure a more realistic simulation experience the physical layer support functionality had to be expanded for Aqua-Sim NG. To accomplish this, we restructured how packets are handled by the channel and physical layer. Dependent on the current channel's propagation model, nodes will receive transmitted packets after a given transmission delay based on acoustic attenuation. It is important to note that not all of these transmissions will be decodable by the receiving node. Factors such as external noise, packet collisions, or busy modem's state may cause packet errors. To simulate if a received message can be decoded on the physical layer, we implemented two main components: signal cache and SINR checker. Signal cache is meant to keep track of all incoming packets for centralized decodability analysis. To accomplish this we implement a linked list of packet smart pointers which can be properly handled before passing back to the physical layer and sequential upper layers. The SINR checker is used to detect if a packet is usable based on a preset decodable threshold.

With the expansion of UWSN, many testbeds add new modems and devices over time when resources and new technologies become available. Therefore, it only makes sense for different types of nodes to be present in a single network. Additionally, some newer devices may have the capability to switch between different modulations to increase bandwidth or energy efficiency. To help simulate these new testing avenues, we implemented the ability to have one or multiple modulations on a given physical layer. This allows users to switch between different modulations dependent on the current testing constrains. Furthermore, it helps minimize the gap between simulation and real-world devices.

To help distinguish among different non-MAC layer packet types, such as synchronization packets from regular packets, we introduced a demux among the physical layer. This demux allows users to add new modules which are independent of MAC or routing layers and can expand individually. For example, this allows for creation of localization packets on a given node without having to specialize a MAC protocol in properly handling these separate packet types.

4.3 Upper Layer Support

The upper layer support consists of a much more diverse grouping of improvements seen in Aqua-Sim NG. Many of these enhancements are based on community questions, comments and internal suggests for underwater simulation. To summarize, we will discuss synchronization and localization support, busy terminal model, and security integration.

Due to the restricted usage of electromagnetic waves in UWSNs, standard Global Positioning System and other localization schemes are ineffective in this environment. Instead we must use the properties of acoustic attenuation or anchor nodes to assist in localizing our network. Additionally, network location is an ongoing real-world challenge due to the high mobility of UWSNs. To assist in simulating this field on Aqua-Sim NG, we implemented a base class for localization which consists of components such as angle of arrival, Euclidean distance for two and three dimensional space, location error, and localization list management. Beyond the base class, we also implemented a range-based localization scheme based off Zhou *et al.* proposed work [20]. In this protocol, a node determines its localization with a given confidence rating after overhearing a certain amount of neighboring node's estimated location. Similar to localization, synchronization also suffers from lack of fast, reliable, and abundant transmissions. Instead, UWSNs must restrict excess message passing due to restricted modem and channel resources. To help simulate synchronization, we have added a specialized class for handling clock skew, synchronization packets, and clock readjustments.

Another potential issue with Aqua-Sim is the busy terminal problem [21]. This issue relates to the half-duplex and high transmission delays of UWSNs. More importantly it is targeted at MAC protocols which typically assume a packet transmission can systematically interrupt another nodes reception leading to improved channel utilization. In wireless sensor networks this is typically not a problem due to the available fast transmission rates. This logic in UWSN is incorrect due to underwater devices incapability of interrupting during receiving or transmitting. This is especially true when a

modem is overhearing a packet destined for another modem and, therefore, must be labeled as busy. To overcome this problem, we have implemented a busy terminal queue for all outgoing packets. This queue is located at the base class of the MAC layer and works directly with the physical layer to check and manage modem states when trying to transit packets. In the case that the modem is in the busy, sending, or receiving state, a packet can not be transmitted and instead will enter the queue. Once the current event is finished, the modem will transition to an idle state allowing for all remaining packets within the queue to be handled accordingly.

With the drastic growth of the UWSN community, comes the necessity to protect against various types of attacks. While it may be impossible to protect against all real-system attacks, it is critical to help improve testing in this field through simulation efforts. To complete this task we created an attack module consisting of the following classes: base class, Denial of Service (DoS), sinkhole, selective forwarding, and sybil attack classes. While channel jamming or malicious noise generation is a security threat, we saw it as fitting to group this under a noise generator feature and, therefore, within the channel model. Each of these classes replicate a base form of a given attack model and can be easily assigned by declaring the attacker flag on the net device. Through this assignment, a user can target specific nodes to behave like malicious modems and test accordingly. Similar to the MAC and Routing layer, the attack base class handles packet management and pointer deallocation. Each child class models different types of malicious scenarios while allowing for expandability if a developer chooses to simulate layer specific attacker attributes. For example, this could allow for a malicious node to integrate critical depth parameters making it the most optimized node for retransmission in a depth-based routing scenario. This attacker module also provides attributes such as creating DoS packets, adjusting packet drop frequency, and location spoofing.

4.4 Information-Centric Integration

UWSNs are a vast and compounding source of new ideas and innovative techniques. To adhere to this ideology we must ensure that Aqua-Sim NG has the capability to adapt to changes in the field of underwater technologies. More directly, we focus on a paradigm shift for handling content within our networking architecture, through the use of IC networking. Compared with standard IP networking, IC focuses on content to manage user queries, improve efficiency, and overall network load balancing. Furthermore, we see in [19] that shifting away from host-to-host connections provides innovative techniques and avenues of new research. UWSNs are not immune to this and, therefore, must broaden their conceptual thinking on what a typical architecture may look like. More importantly, adapting components from IC could open the doors to new solutions or improvements to the many ongoing problems seen in underwater.

To narrow our IC module on Aqua-Sim NG, we choose to adapt components from Named Data Networking (NDN) [6]. We choose this architecture due to its mobility handling, name hierarchy, and integrated routing strategies. To fully support NDN, we introduce a separate section in Aqua-Sim NG consisting of four main components: Content Storage (CS), Forwarding Information Base (FIB), Pending Interest Table (PIT), and quality of life features. Each of these components is meant to mimic standard NDN architecture while adapting to meet resource restricted and limited node networks often seen in UWSNs. Furthermore, we continue our goal of modularity and expandability by breaking down each NDN addition for easier user development and analyze.

Our implemented NDN module acts as a standalone component that can work in parallel with the implemented MAC and routing layers. Through the use of a packet demux at the physical layer, we are able to distinguish between NDN packets and standard packet types. Therefore this newly integrated module can behave as an additional feature for improving underwater communication instead of completely revamping our UWSN architecture. Additionally, these newly usable features can provide beneficial avenues of research and protocol improvements. The CS offers in-network caching for potential improvement in transmission robustness and decreased retrieval time of cached data. Through the use of PIT and FIB, our simulator can advance in unique routing strategies that are information-centric. Due to the parallelism of the NDN module, new UWSN related NDN protocols can evolve freely while advantageously using Aqua-Sim NG's physical and channel layers. It is important to note, that while NS-3 provides a dedicated NDN module, it does not adhere to Aqua-Sim NG core components. Therefore, we thought it necessary to create our own module specialized for underwater while allowing for expandability and properly aligned with UWSN's limitations.

5 PERFORMANCE EVALUATION

To fully evaluate the improvements made to Aqua-Sim NG, we compare simulation memory performance and scalability. It is important to note that testing for this work was completed using code from the GitHub repository at [10]. Testing was completed using NS-3 version 3.24 as well as the most recent pushed commits of Aqua-Sim NG 1.0 (at the time of this publication). Simulation results were collected on a Intel Core i7 computer with 16 GiB of memory running Ubuntu 16.04 OS. To collect the following results we used Valgrind [16] and Linux's time system call. For Valgrind we used the memory leakage tool as well as Massif for heap profiling. Our results are compared with Aqua-Sim (version 1.0) and SUNSET. For the different simulations we choose to use four different sets of protocols due to their simplicity and support by all simulators. These tested protocols are as follows: ALOHA with static routing, Broadcast MAC with static routing, FAMA with static routing, and VBF with Broadcast MAC. Unless stated otherwise testing is completed with ten nodes and one sink in a string topology. Since our focus is on simulation performance we assume a fully connected network with all nodes' location being static.

We first look at memory leakage among Aqua-Sim. Since NS-2 does not directly provide a means to memory reference counting, previous versions of Aqua-Sim have always had issues with memory leaks. As seen in Fig. 2, we see millions of memory leaked bytes independent of the simulation run time. While VBF shows a larger amount of leaks, we still see a large amount of leaks with Broadcast MAC and FAMA. Furthermore, due to VBF's expanded code size and complexity an increased leakage amount can be expected, while the undermining issue seen in the other protocols leads us to believe

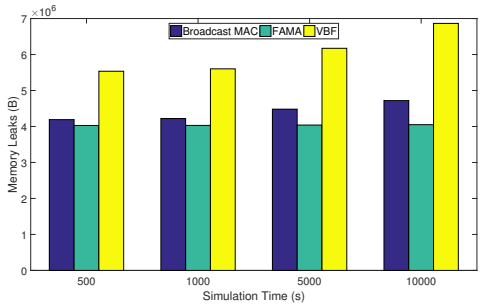

Figure 2: Aqua-Sim (version 1.0) memory leak tests.

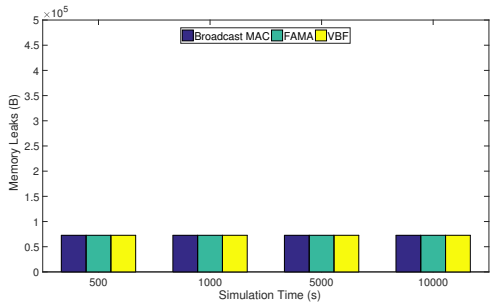

Figure 3: Aqua-Sim NG memory leak tests.

there are many performance issues with Aqua-Sim. In contrast, we see Aqua-Sim NG's memory leaks in Fig. 3 using the same simulated protocols. Aqua-Sim NG shows to have about 72,000 bytes of leaks for all provided tests. This value is due to the phenomenon between the C++ standard library and Valgrind, and therefore is negligible. Additionally, this large decrease in leaks is due to easier pointer disposal methods, using NS-3's smart pointers, and overall NS-3's improved core functionality. To further break down this memory leak comparison, we examine lost allocated memory and non released memory recorded at time of simulation termination. Fig. 4 shows the results of all three simulators using the ALOHA protocol set. Based on these results we believe that while Aqua-Sim does suffer from performance decrease with increased simulation time, the main leakage issue is related to simulator pointers and core modules. While for SUNSET, we speculate that based on the steep increase in memory leakage due to increased simulation time that the issue is more strongly correlated to memory allocation mismanagement.

By using Valgrind's Massif tool we are able to analyze heap usage of each simulator over recorded timestamps. Through the misuse of heap, or larger peak heap size, we see slower searching and storing of data. While heap profiling alone may not be the root of simulation performance, it does shine light on potential inefficiencies related to protocol and core simulation data structure management. In Fig. 5 we analyze Broadcast MAC, FAMA, and VBF for Aqua-Sim and Aqua-Sim NG. Here we see that for Aqua-Sim NG all three protocol sets peak around half of a MiB for heap size. While Aqua-Sim quickly increases its heap size, and peaks at roughly 5.5 MiB for VBF, the worst case in this example. This denotes the large overhead

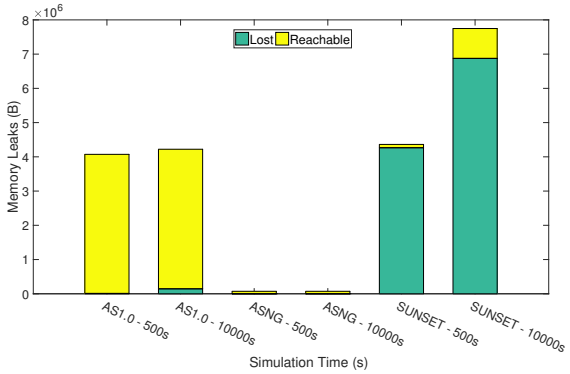

Figure 4: Elaborated memory leak tests for Aqua-Sim, Aqua-Sim NG, and SUNSET. Lost represents allocated memory lost during simulation while reachable shows memory still accessible at termination.

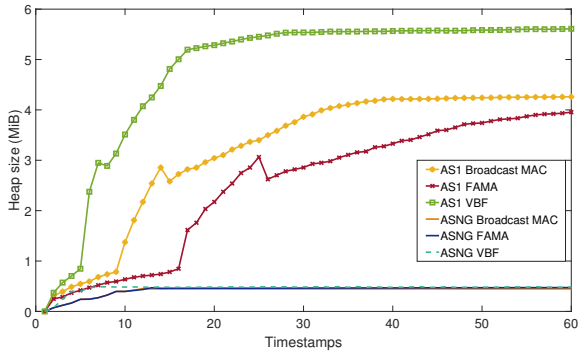

Figure 5: Heap performance for Aqua-Sim and Aqua-Sim NG of various protocols.

in data handling that can be expected when running past iterations of Aqua-Sim. Fig. 6 shows heap performance for all three simulators using the ALOHA protocol set. We see both SUNSET and Aqua-Sim steadily increase heap size overtime. This is most likely due to poor core simulator and protocol data structure management. Additionally, this shows the large memory performance gained from transitioning Aqua-Sim to NS-3.

To help depict Aqua-Sim NG's improvements we also must look at simulation execution time from start to finish. To accomplish this we use Broadcast MAC, due to its simplistic complexity, and vary the amount of nodes used in our string topology. The results of these program run time tests can be seen in Fig. 7. In this figure, we see the real time given from Linux's time command using the -p option for further precision. Additionally, we recorded test run times ranging from 500 to 5000 seconds in increments of 100. Simulated nodes range from 10 to 200 to supplement strain on the given test. In the figure this ranging of nodes is denoted by each *n* line. For both Aqua-Sim and Aqua-Sim NG we see a similar real time for 10 nodes within the network. While it becomes quickly clear that with 50 or

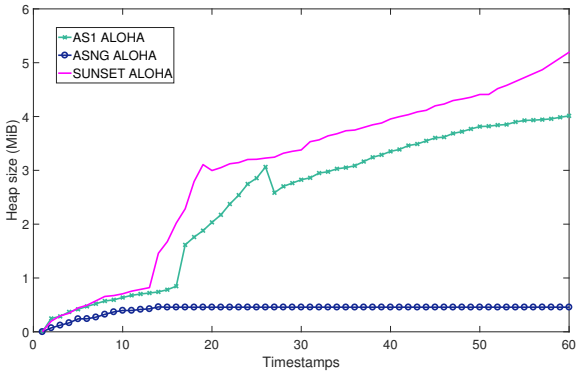

Figure 6: Heap performance comparing Aqua-Sim, Aqua-Sim NG and SUNSET using the ALOHA protocol.

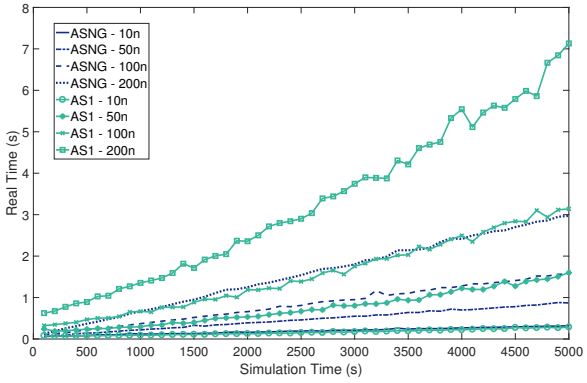

Figure 7: Program run time of Aqua-Sim and Aqua-Sim NG using Broadcast MAC.

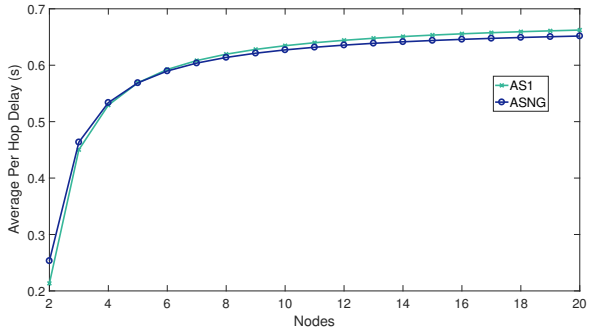

Figure 8: Average per hop packet delay of Aqua-Sim and Aqua-Sim NG using VBF and Broadcast MAC protocols.

more nodes, Aqua-Sim takes twice as long in real time, than Aqua-Sim NG. For direct comparison reasons, we only used Broadcast MAC protocol set, but it is important to note that with the use of more complex protocols this program run time performance will most likely only improve for Aqua-Sim NG.

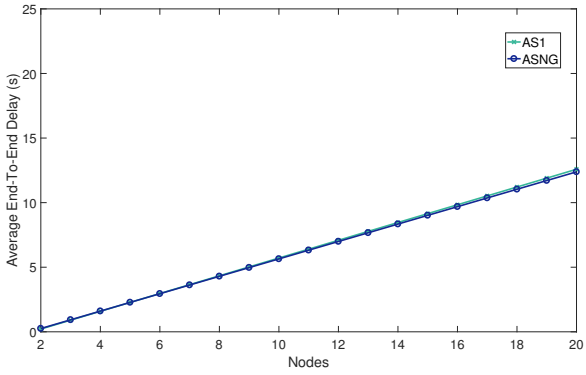

Figure 9: End-to-end packet delay of Aqua-Sim and Aqua-Sim NG using VBF and Broadcast MAC protocols.

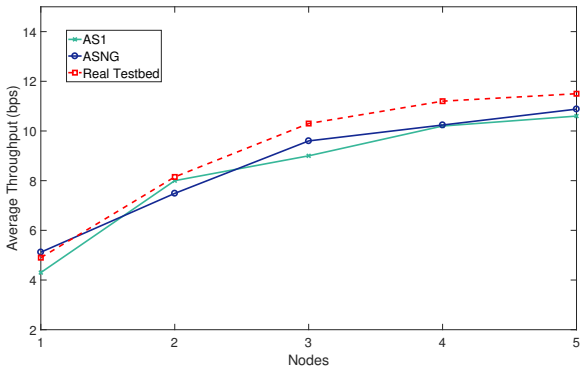

Figure 10: Average network throughput using ALOHA protocol with varying amount of traffic generating nodes. Input traffic rate set to 0.02 packets per second for each sender.

Beyond performance results of Aqua-Sim NG, we must also review the accuracy of our simulation. To accomplish this we first compare packet delay times on Aqua-Sim and Aqua-Sim NG using VBF and Broadcast MAC protocols. Since many of these protocols were initially created and tested on Aqua-Sim 1.0, this gives us a benchmark for reliability and accuracy. Our topology was set to a string node formation with each simulation lasting for 400 seconds. Additional parameters include vector width of 40 meters, packet size of 400 bits, modem baud rate of 10 kbps and traffic generation set to 0.25 packet per second. Fig. 8 shows results for average per hop delay and Fig. 9 depicts average packet end-to-end delay results. For both graphs we see a gradual increase in delay as the amount of nodes are increased. This is due to propagation, transmission, and protocol processing delays on both simulators. More importantly, we see almost identical simulation results in each graph with slight variation in Fig. 8 most likely due to each simulator's random variable delay.

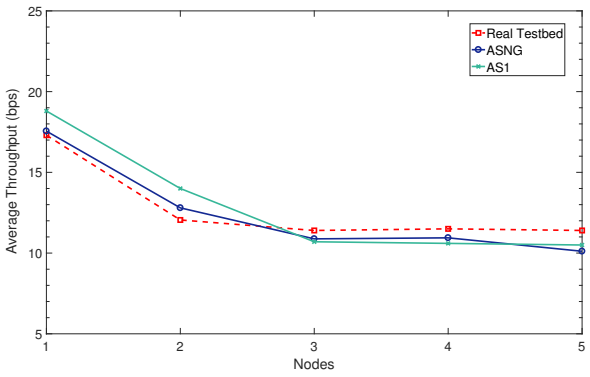

Figure 11: Average network throughput using ALOHA protocol with varying amount of traffic generating nodes. Total input traffic rate set to 0.1 packets per second.

Secondly, we review accuracy of Aqua-Sim NG compared with results depicted in Aqua-Sim's original work using ALOHA protocol to show average network throughput. In this evaluation we look at previously collected Aqua-Sim and real testbed results, as seen in [18], with a pentagon shaped network topology consisting of a single sink in the center. Additional parameters include modem baud rate of 80 bps, packet size of 256 bits, and max retransmissions set to 3. For each test we adjust the amount of nodes inputing traffic into the network. For Fig. 10 every node generates traffic at a rate of 0.02 packets per second, whereas Fig. 11 sets the total input traffic at 0.1 packets per second. Therefore, dependent on the amount of data senders n, the input traffic of each node is $\frac{0.1}{n}$. For both of these figures we see relatively similar results from Aqua-Sim NG compared with real testbed and Aqua-Sim results. Additionally, we see slight variations in Aqua-Sim NG which are most likely due to the simulator's random variables used in ALOHA protocol.

6 CONCLUSION

We have presented Aqua-Sim Next Generation (NG), the newest iteration of Aqua-Sim, with enhanced features and modules to better represent underwater sensor networks. Aqua-Sim NG's architecture is revamped and transitioned to NS-3 core simulator. Aqua-Sim NG offers legacy protocol support alongside additional implemented protocols using better packet handling and header usage. For more advanced channel support we introduce various noise generators, multiple channel model support, transmission range uncertainty, and trace driven testing capability. The physical layer of Aqua-Sim NG is expanded to offer better signal cache handling, modulation support, and easier packet decodability checking. Additionally, our new simulator offers synchronization and localization support, busy terminal model integration, and security features. We also implemented a new module adapted from Named Data Networking, to help integrate innovative information-centric techniques in underwater simulations. Evaluation results of Aqua-Sim NG show great improvements in memory performance and simulator real time scalability.

REFERENCES

[1] Ian F. Akyildiz, Dario Pompili, and Tommaso Melodia. 2005. Underwater Acoustic Sensor Networks: Research Challenges. *Ad Hoc Networks (ELSEVIER)* 3 (2005), 257–279.

[2] The NUS ARL and SubNero. 2016. Unet - The Underwater Networks Project. (2016). http://www.unetstack.net

[3] Nicola Baldo, Federico Maguolo, Marco Miozzo, Michele Rossi, and Michele Zorzi. 2007. Ns2-MIRACLE: A Modular Framework for Multi-technology and Cross-layer Support in Network Simulator 2. In *Proceedings of ValueTools '07*. Article 16, 8 pages.

[4] L. Berkhovskikh and Y. Lysanov. 1982. *Fundamentals of Ocean Acoustics*. New York, Springer.

[5] Jun-Hong Cui, Jiejun Kong, M. Gerla, and Shengli Zhou. 2006. The challenges of building mobile underwater wireless networks for aquatic applications. *IEEE Network* 20, 3 (May 2006), 12–18.

[6] Lixia Zhang et al. 2010. *Named Data Networking (NDN) Project*. Tech. Rep. NDN-0001.

[7] Federico Guerra, Paolo Casari, and Michele Zorzi. 2009. World Ocean Simulation System (WOSS): A Simulation Tool for Underwater Networks with Realistic Propagation Modeling. In *Proceedings of ACM WUWNet '09*. Article 4, 8 pages.

[8] Y. Luo, L. Pu, M. Zuba, Z. Peng, and J. H. Cui. 2014. Challenges and Opportunities of Underwater Cognitive Acoustic Networks. *IEEE Transactions on Emerging Topics in Computing* 2, 2 (June 2014), 198–211.

[9] K.V. Mackenzie. 1981. Nine-term equation for sound speed in the oceans. *J. Acoust. Soc. Am.* (1981).

[10] Robert Martin. 2017. Aqua-Sim Next Generation. (2017). https://github.com/rmartin5/aqua-sim-ng

[11] R. Masiero, S. Azad, F. Favaro, M. Petrani, G. Toso, F. Guerra, P. Casari, and M. Zorzi. 2012. DESERT Underwater: An NS-Miracle-based framework to design, simulate, emulate and realize test-beds for underwater network protocols. In *2012 Oceans - Yeosu*. 1–10.

[12] M. Porter *et al.* 2014. BELLHOP gaussian beam/finite element beam code. (2014). http://oalib.hlsresearch.com/Rays/

[13] NS-2 2011. (2011). http://www.isi.edu/nsnam/ns/

[14] NS-3 2017. (2017). https://www.nsnam.org/

[15] Chiara Petrioli, Roberto Petroccia, and Daniele Spaccini. 2013. SUNSET Version 2.0: Enhanced Framework for Simulation, Emulation and Real-life Testing of Underwater Wireless Sensor Networks. In *Proceedings of ACM WUWNet '13*. 43:1–43:8.

[16] The Valgrind Developers. 2017. Valgrind. (2017). https://www.valgrind.org/

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

[18] P. Xie, Z. Zhou, Z. Peng, H. Yan, T. Hu, J. H. Cui, Z. Shi, Y. Fei, and S. Zhou. 2009. Aqua-Sim: An NS-2 based simulator for underwater sensor networks. In *OCEANS 2009*. 1–7.

[19] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos. 2014. A Survey of Information-Centric Networking Research. *IEEE Communications Surveys Tutorials* 16, 2 (2014), 1024–1049.

[20] Z. Zhou, Z. Peng, J. H. Cui, Z. Shi, and A. Bagtzoglou. 2011. Scalable Localization with Mobility Prediction for Underwater Sensor Networks. *IEEE Transactions on Mobile Computing* 10, 3 (March 2011), 335–348.

[21] Yibo Zhu, Jun-Hong Cui, Zheng Peng, and Zhong Zhou. 2014. Busy Terminal Problem and Implications for MAC Protocols in Underwater Acoustic Networks. In *Proceedings of ACM WUWNet '14*. Article 1, 8 pages.