An Underwater Network Testbed: Design, Implementation and Measurement *

Zheng Peng, Jun-Hong Cui, Bing Wang
Computer Science & Engineering
University of Connecticut
{zhengpeng, jcui, bing}@engr.uconn.edu

Keenan Ball, Lee Freitag
Applied Ocean Physics & Engineering
Woods Hole Oceanographic Institution
{kball, lfreitag}@whoi.edu

ABSTRACT

This paper presents the design, implementation and measurement of Aqua-Lab, an underwater acoustic sensor network lab testbed. Aqua-Lab consists of a water tank, a set of acoustic communication hardware, and a set of software. One important component of the software is an emulator that we developed to provide user-friendly programming interfaces and emulate realistic network settings. Using Aqua-Lab, we explore basic characteristics of data transmission using Micro-Modems and conduct a set of experiments in both field and lab environments. Our results from the lab testbed are consistent with those from the field experiments, and thus demonstrate that Aqua-Lab can be used to experimentally evaluate algorithms and protocols designed for underwater sensor networks.

Categories and Subject Descriptors

C.2.m [**COMPUTER-COMMUNICATION NETWORKS**]: Miscellaneous

General Terms

Design, Experimentation, Measurement

Keywords

Underwater Sensor Networks, Lab Testbed, Acoustic Communication, Emulator

1. INTRODUCTION

Underwater sensor network is an emerging area. Its wide applications and unique challenges have spurred a big wave of research recently [4, 6, 9, 10]. However, due to the expense of building real underwater network systems and conducting experiments in real underwater systems (such as Seaweb [17] and PLUSNet [18], most research (such as [16], [13], [11] and [8] to name a few) has been using modeling, analysis, and simulation methodologies. The complicated aquatic environments and underwater acoustic communication, however, require a standard platform deployed in real underwater environments, which can be used to test, evaluate, and compare different network algorithms and protocols. The lack of such a platform will slow down further development of the field.

To bridge the gap between real system implementation and modelling/simulation evaluation, in this paper, we describe our work on an underwater acoustic sensor network lab testbed, called **Aqua-Lab**. This testbed is hosted in the UnderWater Sensor Network (UWSN) Lab at the University of Connecticut [1]. It consists of a water tank, a set of communication hardware and software Application Programming Interfaces (APIs). In the testbed, acoustic modems and transducers form the communication hardware. Software APIs encapsulate all the operations of the hardware and provide an abstract layer for users. So users can develop their own applications without knowing the exact mechanisms of the underlying acoustic physical layer. An emulator is also developed for users to emulate different network topologies, propagation delay, and attenuation. Using Aqua-Lab, we explore basic characteristics of data transmission using Micro-Modems and conduct a set of experiments in both field and lab environments. Through experiments we show that Aqua-Lab produces results compatible to those obtained from field tests.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of Aqua-Lab. In Section 3, we describe the design of Aqua-Lab in detail. A case study on a routing protocol for underwater sensor networks is elaborated in Section 4. Experiment results are reported in Section 5. Finally, Section 6 concludes and presents future work.

2. TESTBED OVERVIEW

Aqua-Lab is designed to provide researchers *field test experience in a lab controlled environment*. Hardware resources are carefully chosen to offer a real underwater environment and the freedom to customize environmental parameters with affordable budgets. A water tank is used to represent the underwater environment. Underwater speakers and hydrophones serve as the transducers (transmitters and receivers). Together with Micro-Modems [2] developed by WHOI (Woods Hole Oceanographic Institution) [3], they provide acoustic communication channels that are close to field test setup. With a sound mixer, multiple acoustic signals can be introduced into the channels. Environmental noises and special sound effects can be produced to make the lab testbed even more similar to real environments. All the equipments are connected to a server and are accessible via the Internet. The various devices are shown in Figure 1.

To facilitate experimentation using Aqua-Lab, we propose a programming model that is easy to implement and use. Furthermore,

*This work is supported in part by the NSF CAREER Grant No. 0644190.

WUWNet'07, September 14, 2007, Montréal, Québec, Canada.

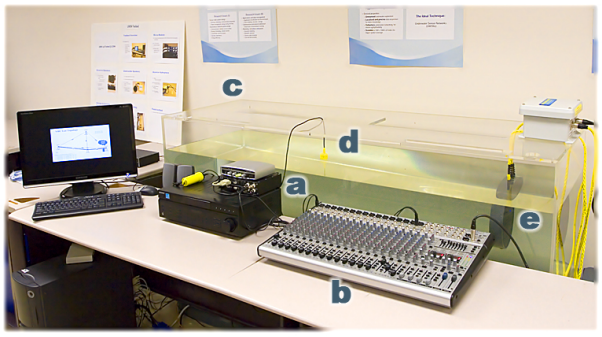

Figure 1: Testbed Overview: a. Micro-Modem, b. Sound Mixer, c. Water Tank, d. Hydrophone, e. Underwater Speaker.

a C library compliant to our model has been built to interface with the Micro-Modems. At the lowest level of the library, we have implemented a set of low-level core functions that interface directly with the modems. These functions provide operations to set up and configure the modems, such as setting the frequency band, setting the baud rate of the serial port, setting data request timeout, putting the modem to sleep-mode, opening a port for communication, closing a port, pinging other modems, reading messages, and writing messages. This level of APIs provide the granularity to perform all sorts of operations with the modem hardware. At a higher level of the library, we have implemented a set of APIs that encapsulate most of the low-level operations and provide users an easy-to-use interface. Furthermore, using our C library, we have implemented an emulator. The purpose of the emulator is to allow users to emulate complex topologies, propagation delay, and signal attenuation using limited lab resources (i.e., a few acoustic modems and a water tank with a limited communication distance).

3. TESTBED DESCRIPTION

In this section, we describe the design of our underwater network testbed, Aqua-Lab. We start from the equipments used in the testbed and then describe the C library that we developed for programming the modem. Afterwards, we describe the emulator that we designed and implemented to emulate realistic underwater environments.

3.1 Hardware Setup

Our testbed contains the following devices:

- **Acoustic modem**: We use Micro-Modem developed by WHOI [2]. It is a low-power acoustic modem which is based on TI DSP TMS320C5416.
- **Underwater speaker**: The underwater speaker we use has a frequency range from $200Hz$ to $32KHz$. At higher frequencies, the sound signal produced by the speaker is directional while it is omni-directional at lower frequencies. This underwater speaker is non-powered.
- **Hydrophone**: A hydrophone works as a microphone in underwater environments. The hydrophone that we choose has power consumption of $800mW$ and requires $7mA$ quiescent current to operate. It supports frequency from $20Hz$ to $100KHz$ and provides a $8.5mm$ mini phone jack as an output interface.
- **Sound mixer**: The sound mixer is used to emulate different underwater environments by introducing special sound effects into the acoustic channel. We choose Behringer SL2442FXPRO Eurodesk 24-Channel Mixer, which produces very pure and clear signals.
- **Aquarium**: An aquarium filled with water is the underwater environment provided in Aqua-Lab. The aquarium is a $2m \times 1m \times 1m$ water tank filled with around two tons of water.
- **Server**: It is a PC used to control the acoustic modems. In our testbed, we connect modems to the serial ports of the server. The server controls the modems by sending commands into the serial ports.

The logical layout of Aqua-Lab with two modems is shown in Figure 2. The distance between the underwater speaker and hydrophone is around 1 meter.

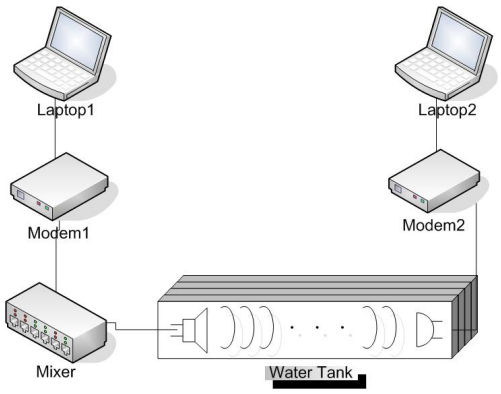

Figure 2: Testbed Setup

3.2 Testbed Library

We have developed two sets of functions. The first set of functions are low-level functions that provide a direct interface to the Micro-Modem for Unix/Linux systems. They allow users to perform low-level hardware operations (including I/O operations) as reading and writing messages through a serial port (RS232). The second set of functions are at a higher level. They are essentially function wrappers that call low-level functions and provide a higher-level interface to make programming easier and faster. These two sets of functions form *low-level* and *high-level* library, respectively, as detailed below.

3.2.1 Low-Level Library

The low-level library defines functions that allow users to easily change hardware settings. Furthermore, it provides functions for I/O operations, including opening/closing a port, configuring a serial port (e.g., setting up the baud rate and setting the frequency band). We describe several representative functions below:

- *modem_send(int fd, void *cyc, void *txa)* and *modem_recv(int fd, void *buf)* use file descriptor *fd* to communicate with the modem. Each call to the function *modem_send* only sends one data unit with size limited by Micro-Modem's MTU (Maximum Transmission Unit). The two parameters *cyc* and *txa* are sent to the modem to initialize and start the data transmission. They are transparent to users and are generated automatically by the library functions.
- *setModemPara(struct nvram *para)* is used to configure modem parameters. It can be used to enable/disable modem diagnostic information, set modem timeout value and working

frequency bank. The structure *nvram* defines all the parameters that are adjustable for Micro-Modem.

3.2.2 High-Level Library

The main objective of the high-level library is to provide a programming model that is similar and as easy to use as the Berkeley socket programming model. With this model, tedious low level programming that is needed for hardware settings can be abstracted so that users can concentrate more on their designs. We now describe several frequently used high-level library functions:

- *int mdmTx(int fd, int src, int dst, char *ascstr):* it generates commands for Micro-Modem to send data. If the data are larger than Micro-Modem's MTU, it will divide data into fragments and then call *modem_send* multiple times to send the data. It also replaces several characters in the data (since these characters have been reserved for modem operation).
- *int mdmRx(int fd, int src, int dst, void *buf):* it reads modem output, assembles the data if they are fragmented, and returns them to the caller. It also recovers special characters.
- *int mdmPing(int fd, int src, int dst):* it tests the connectivity of a pair of Micro-Modems.

3.3 Emulator Design

The emulator creates an abstract layer above the underlying physical network. Users can create multiple virtual nodes in the emulator, map virtual nodes to modems, and define their behaviors in particular events. Note that one modem can be associated with more than one (virtual) nodes. When a node sends out a packet, the packet will not get into the channel immediately. Instead, it will be queued in the emulator. The emulator will decide when to send this packet and which modem it is sent to. Propagation delay can be calculated when location information (i.e., latitudes and longitudes) of the nodes are known. As we shall see, our emulator also allows emulating complex network topologies with only a limited number of modems. The architecture of the emulator is shown in Figure 3.

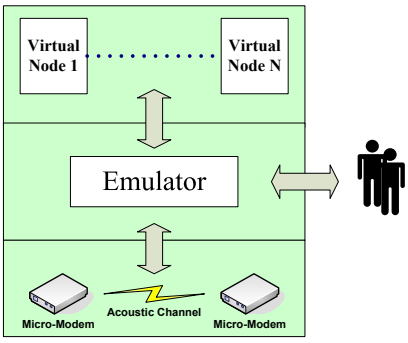

Figure 3: Emulator Architecture

In the following, we describe how to emulate complex topologies, propagation delay, and signal attenuation using our emulator.

3.3.1 Emulating Complex Topologies

With limited acoustic modems available in the testbed, it is very challenging to emulate a complex topology. On the one hand, we need to carefully map the virtual nodes to the modems and map the connections to the physical links. On the other hand, we have to well conserve the properties of the physical network. In principle, we can emulate any complex topology which can be decomposed into a series of sequential small topologies, either the same as the underlying physical network or part of the physical network. We illustrate this principle using a simple example. Assume that we have two modems. Then we can emulate any chain topology (as shown in Figure 4). In the chain topology, the communication between any two neighbors use the physical channel formed by the two modems. Different pairs of neighbors are scheduled to communicate at different times according to the routing algorithm.

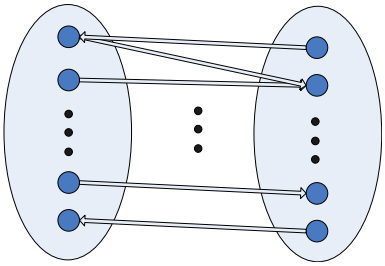

Figure 4: Topology Emulation with Two Modems

3.3.2 Emulating Propagation Delay

In Aqua-Lab, we emulate the propagation delay between a pair of nodes by introducing an additional delay at the sender based on the distance. More specifically, to emulate a distance of d meters, the sender delays the transmission by d/v seconds, where v is the sound propagation speed.

3.3.3 Emulating Attenuation

Besides propagation delay, signal attenuation is another important factor that needs to be emulated in long-distance communication. Since attenuation ultimately affects SNR (signal to noise ratio) at the receiver side, we propose emulating attenuation by adjusting the SNR using the sound mixer. More specifically, we use the sound mixer as an controllable amplifier to adjust the strengths of both the noise and the signal. By adjusting the SNR to the value observed in a field experiment, we emulate the attenuation as that observed in the field experiment.

4. CASE STUDY

As a case study, we implement a routing protocol, VBF (Vector-Based Forwarding) [16], in our testbed. VBF provides robust, scalable and energy efficient routing for underwater sensor networks. It is a position-based routing protocol: nodes close to the "vector" from the source to the destination will forward the packets; other nodes will simply discard the packets. In this way, only a small fraction of the nodes are involved in routing. VBF also adopts a localized and distributed self-adaptation algorithm which allows nodes to weigh the benefit of forwarding packets and thus reduce energy consumption by discarding the low benefit packets. For simplicity, in this paper, we implement a simple version of VBF (without self-adaption) using APIs provided by Aqua-Lab. We next describe our implementation in detail.

Currently, we only have two acoustic modems. Therefore, as described earlier, we create virtual nodes in our emulator to emulate complex topologies. In the emulator, all data first go to the emulator before they are sent via a physical link into the water tank. There is a control module to distribute the messages received from the physical link to different virtual nodes. Each virtual node is represented as a thread and independently runs the VBF protocol

to decide whether it needs to forward a received message or not. More specifically, each packet carries the positions of the sender, the target and the forwarder in three fields. The routing pipe is determined by the vector from the sender to the destination. The radius of the pipe is defined in the *radius* field. After receiving a packet, a forwarder reads the VBF header to obtain the above information. If it is in the routing pipe, it forwards this packet; otherwise drops it.

5. EXPERIMENT RESULTS

In this section, we conduct a set of experiments in both our lab and the field (at Buzzards Bay, MA). Our goal is two fold: (1) obtain a set of measurements on the basic characteristics of data transmission using Micro-Modems; (2) demonstrate that results from our lab testbed are consistent with those from the field experiments, and hence our testbed can be used to experimentally evaluate protocols designed for underwater sensor networks. In the following, we first describe experiment setup and then detail the experimental results.

5.1 Experiment Setup

In both lab and field experiments, we use a server to control Micro-Modems. In the lab, Micro-Modems are connected directly to the serial ports of the server. The speed of the serial ports is set to 115.2 Kbps, which is much higher than the speed of the Micro-Modem, and hence it does not constrain the speed of the acoustic transmission.

In the field experiments, a modem is mounted to an anchor so that its location is fixed (e.g., does not move due to ocean current or wind). Each modem is also attached to a buoy for ease of recollection. Each buoy has a GPS receiver to record its location and an antenna to communicate with the devices ashore. The server is located in a building at WHOI. An antenna is installed on the top of the building to communicate with the buoys using radio. A command from the server to the modem first reaches the antenna installed on top of the WHOI building, which then forwards it to the buoy attached to the modem using radio. After receiving the command, the buoy forwards it to the Micro-Modem.

We mainly conduct two types of experiments. The first type of experiments involve two Micro-Modems, as the sender and receiver respectively. The second type of experiments are for testing VBF and involve four Micro-Modems (in the lab, we emulate four Micro-Modems, termed as virtual nodes in Aqua-Lab, using two Micro-Modems). For each type of experiments, we run a testing program on the server. Depending on the purposes of the experiments, a testing program may send messages to the Micro-Modems, receive messages from the Micro-Modems and log the timestamps of various events. When there are concurrent events (e.g., in testing VBF, multiple Micro-Modems may receive the messages simultaneously), we create a thread to control each Micro-Modem respectively.

Our field and lab experiments are conducted from March 5 to April 15, 2007. For ease of deployment, we use the same topology for all the field experiments. This topology contains four nodes, as shown in Figure 5. The distance between a pair of nodes ranges from 303 to 906 meters, marked by the numbers on the edges. The sending signal strength of each modem is fixed to 162 dB.

We next describe the experimental results in detail, starting from the characteristics of Micro-Modems. We then describe our measurements on delay, raw sending rate and throughput. Afterwards, we describe our experimental results on running VBF. At the end, we present our preliminary study on noise and packet-loss measurements, and attenuation emulation.

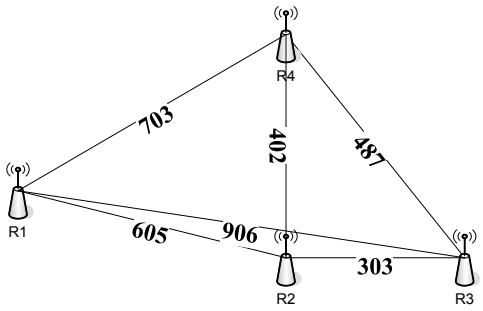

Figure 5: The topology used in field experiments at Buzzards Bay, MA. The numbers on the edges represent distances (in meters).

5.2 Characteristics of Micro-Modems

We measure two basic characteristics of Micro-Modems. The first is on the time required to send a data packet. The second is the effect of the payload size on transmission time. Both measurements are conducted in the lab.

5.2.1 Micro-Modem Working Time Breakdown

When using Micro-Modems, sending a data packet from a sender to a receiver contains three steps, as shown in Figure 6. In the first step, the sending modem initiates a CI (Cycle-Init) message to the receiver. This message contains a list of parameters including the source ID, receiver ID, packet type, etc. It synchronizes the sender and the receiver. After receiving this message, the receiver enters the receiving mode. In the second step, the sending modem transmits a data packet to the receiving modem[1]. In the last step, the receiving modem returns an ACK to the sending modem to indicate that the data packet has been received correctly. This step is optional.

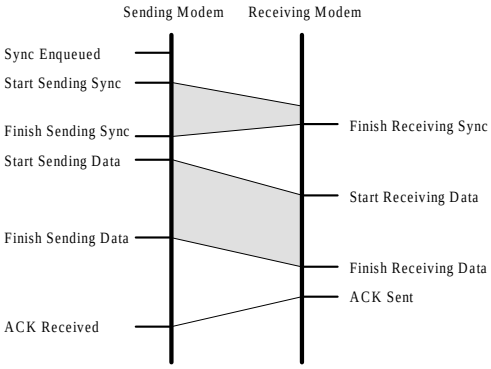

Figure 6: Micro-Modem Workflow

We conduct an experiment in which the sending modem sends 200 data packets, each with a payload of 32 bytes (the maximum size allowed by a Micro-Modem). For each data packet, we record the times when the CI message is sent out and when the sender finishes sending the CI. The difference of these two values is the time required for synchronization (both timestamps are based on

[1]More advanced version of Micro-Modems allow multiple data packets following a CI message. However, our lab modems do not support this feature.

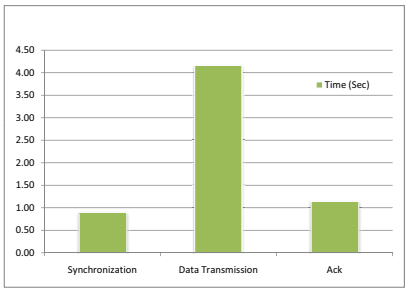

Figure 7: Time Breakdown

the system time of the server). Similarly, we record the times when a data packet is sent out and when it is received at the receiver to calculate the duration for data transmission. Last, we record the times when the receiver returns an ACK to the sender and when the sender receives the ACK. The difference is the duration for acknowledgment. Figure 7 plots the time required for synchronization, data transmission and acknowledgment. The confidence intervals are very tight and hence omitted. We observe that, on average data transmission takes around 4 seconds; ACK transmission takes 1 second; while synchronization takes less than 1 second.

5.2.2 *Effect of Payload Size*

A Micro-Modem allows a maximum payload of 32 bytes in a data packet. We now explore the effect of payload on the sync-transmission time. The size of the payload is set to 10, 20 or 30 bytes. For each payload size, we send 30 packets to obtain the delay (from when the sender sends out the CI message to when the receiver finishes receiving the data packet) for each packet, as shown in Figure 8. As shown in the figure, the delays are very close for the different payload sizes. We later discovered the reason is that, for a payload size less than 32 bytes, a Micro-Modem pads the payload to be 32 bytes. We use a payload of 32 bytes for a data packet in the rest of the paper.

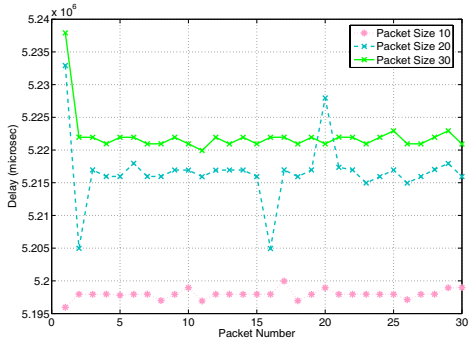

Figure 8: Sync-Transmission Duration vs Payload Size

5.3 Delay, Sending Rate and Throughput

We now report our measurements between a sender-receiver pair in the field and the lab. Our focuses are on (1) the delay in sending a data packet; (2) the raw sending rate that is achieved by Micro-Modems (i.e., the number of bits transmitted by a Micro-Modem per second); and (3) the actual throughput at the application-level (i.e., the amount of payload transmitted by a Micro-Modem per second).

5.3.1 *Delay Measurement*

In the field experiments, we use four Micro-Modems placed as in Figure 5. These modems are referred to as $R_1, \ldots, R_4$. A sequence of 200 packets are sent from R_i to R_j, $i, j = 1, \ldots, 4$ and $j > i$. The distance between R_i and R_j varies from 300 to 900 m. At the sender, two consecutive packets are spaced by 15 seconds. This spacing is to avoid interference of consecutive packets and to avoid overwhelming the receiver. We observe that the delay (from when the sender sends out the CI message to when the receiver finishes receiving the data packet) in the field experiments increases linearly with the distance (see Figure 9). This indicates that the propagation delay (the only component that depends on the distance in the delay) is a linear function of the distance. In other words, the propagation speed of acoustic signals in water is approximately a constant. When fitting a linear line to the measurement, we obtain that the propagation speed is roughly 1500 meter per second, which is consistent with that reported in [14]. In our lab experiments, we emulate the delay for a distance of d meters as $d/1500$ second. As shown in Figure 9, the delays from the lab are approximately 0.5 second lower than those in the field experiments. This difference is due to the extra processing time in the field experiments (including processing time at the antennas and at the buoys).

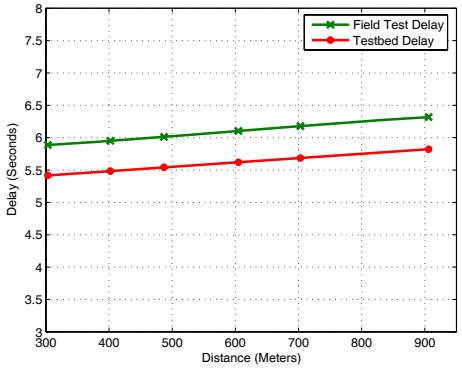

Figure 9: Delay with Different Distances

5.3.2 *Raw Sending Rate*

We also obtain the raw sending rate of the Micro-Modems. A Micro-Modem codes packets (by introducing redundancy) to combat data corruption in the harsh underwater environment. More specifically, it codes the CI message (of 4 bytes) into 80 bits and codes the 32-byte payload to 621 bits. Suppose the distance between a sender-receiver pair is d. Let t be the sync-transmission time for a packet between this sender-receiver pair. Assume that the sound speed is 1500 meters per second in water. We obtain the raw sending rate as $(80+621)/(t-2d/1500-0.21)$, where 0.21 is a constant processing time [2]. Figure 10 plots the raw sending rate from the field experiments, where $d = 300$ m. We observe that the raw sending rate is in the range of 156.5 to 157.5 bps.

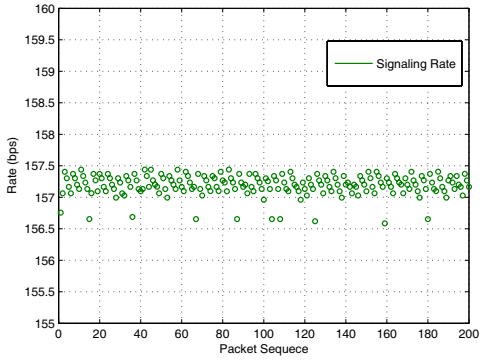

Figure 10: Raw Sending Rate

5.3.3 Throughput Measurement

We now calculate the amount of useful data (i.e., payload in data packet) transmitted by a Micro-Modem per unit of time, referred to as *application-level throughput*. In particular, we implement a stop-and-wait protocol for reliable data transfer. In this protocol, the sender sends a data packet, and then waits for the ACK corresponding to this packet. If no ACK comes back after 15 seconds, the sender retransmits the packet. This simple protocol provides a baseline on application-level throughput using Micro-Modems.

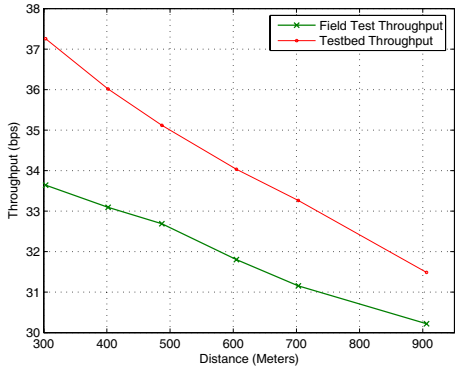

Figure 11: Throughput Measurement

For each pair of modems, we send 200 packets and obtain the application-level throughput as follows. Let n be the number of packets that are transmitted successfully. Let T be the amount of time required to transmit these packets (including synchronization, data transmission and acknowledgement). Then the application-level throughput is $256n/T$ bps because the payload of each data packet is 32 bytes (i.e., 256 bits). Since the above calculation excludes corrupted and lost packets (because they may be caused on the radio link between the server and the buoy in the field experiments), it provides the application-level throughput under ideal conditions. Figure 11 plots the throughput thus obtained in both the field and lab experiments. We observe that, for each distance, the throughput in the lab is approximately 10% higher than that in the field. This discrepancy is also due to the additional processing time (at the antennas and at the buoys) in the field experiments.

5.4 Measurements on VBF Protocol

We now report experimental results when running VBF protocol in the field and the lab. The topology is shown in Figure 5. The sender is modem R_1 and the sink is modem R_3. The position of each modem is known (from the GPS information recorded at the buoys). The pipe radius of VBF is set to 350 m. According to VBF protocol, R_4 discards the messages from R_1 while R_2 serves as a relay to forward them to the sink R_3. We then obtain the end-to-end delay for each packet from R_1 to R_3. One complication in our field experiments is that R_3 can actually receive the messages from R_1 directly since the transmission range of a Micro-Modem could be up to 4 kilometers [7]. However, separating R_1 and R_3 by 4 kilometers will make the deployment too difficult. Therefore, in our experiments, R_3 simply discards messages from R_1 and only records the receiving time of packets forwarded by the relay R_2.

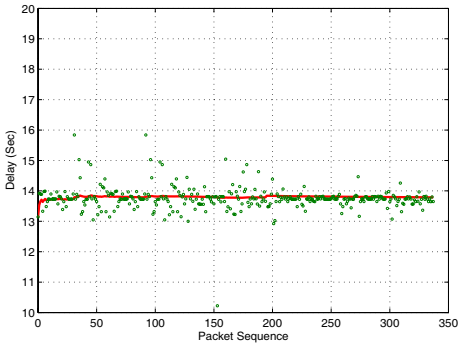

Figure 12: VBF Delay in Buzzards Bay

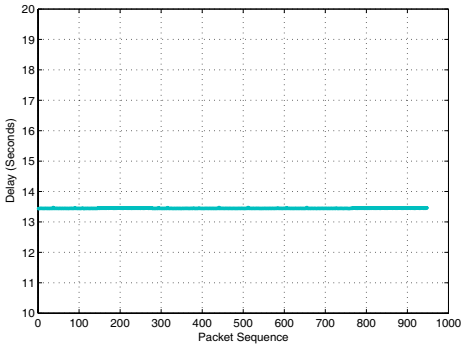

Figure 13: VBF Delay in Testbed

In the field experiment, R_1 sends 500 packets while 338 packets are successfully received by R_3 (through relay R_2). At the sender, two adjacent packets are separated by 15 seconds for the same reasons as mentioned earlier. Figures 12 and 13 plot the end-to-end delay in the field and lab experiments respectively. The average end-to-end delays in the field and lab experiments are 13.8 and 13.4 seconds respectively, both higher than the sum of the delays on the two hops (i.e., from R_1 to R_2 and from R_2 to R_3). This is due to additional processing time at relay R_2.

5.5 Signal and Noise Strength

In our field experiments, we observe packet losses and corruption occasionally. Our testbed can emulate packet losses and corruption by using the sound mixer to adjust the signal and noise strengths. We next present preliminary results on the impact of signal and noise strengths on packet losses. All the readings of signal and noise strengths below are the amplification ratio of the sound mixer which is controllable. Since the signal strength from the Micro-Modem is a constant, the amplification ratio can provide a reasonable way to determine the effect of signal strength.

5.5.1 Environmental Noises

Environmental noise is one of the most important factors that affect network performance. Noise measurements from real environments are helpful for understanding the characteristics of underwater channels, which can further help improving the design of acoustic modems.

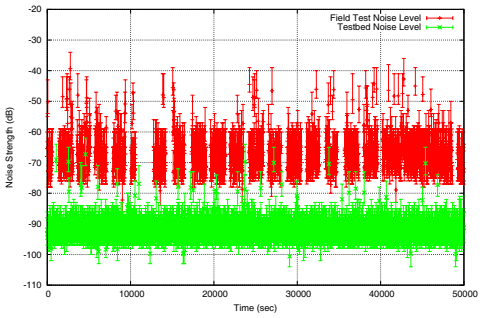

Figure 14: Comparison of Noise Levels

During our field experiments, we collect the environmental noise level every 5 minutes. In every measurement, 14 readings are collected in a duration of one second. For the sake of comparison, we also collect the noise levels in our lab in the same manner. The results from both the field and lab are plotted in Figure 14. We observe that the accumulative average noise level in the field (at Buzzards Bay, MA) is around $-55dB$, while it is $-85dB$ in our lab. Furthermore, the noise level in the field exhibits significant variance due to random underwater events (e.g., an approaching ship or tide). Since noises can be controlled using a sound mixer in our testbed, we can introduce additional noises to simulate real environments.

5.5.2 Packet Loss

We first examine the effect of signal strength on packet loss. In this experiment, 100 packets are sent from one modem to another one. We adjust the signal strength by changing the volume of the underwater speaker from our sound mixer. As shown in Figure 15, the loss rate decreases when increasing the signal strength.

We now examine the effect of noise on packet loss by introducing white noise into the channel using the sound mixer. As shown in Figure 16, approximately, the loss rate increases linearly when the background noise increases exponentially.

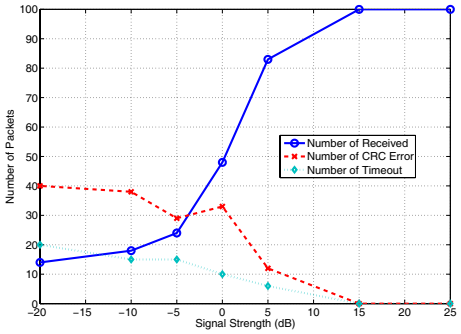

Figure 15: Packet Loss with Varying Signal Strength

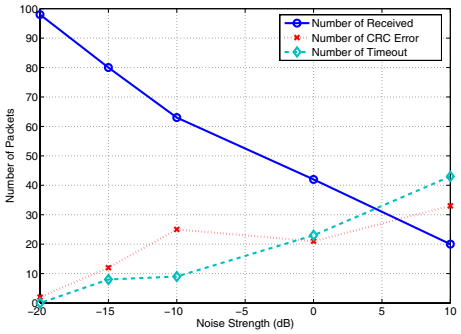

Figure 16: Packet Loss with Varying Noise Level

5.6 Emulating Attenuation

We observe losses and errors in both our field test and lab test. Since losses and errors are heavily affected by attenuation in wireless communication, it is important to emulate attenuation in Aqua-Lab. We next demonstrate that attenuation can be effectively emulated in Aqua-Lab.

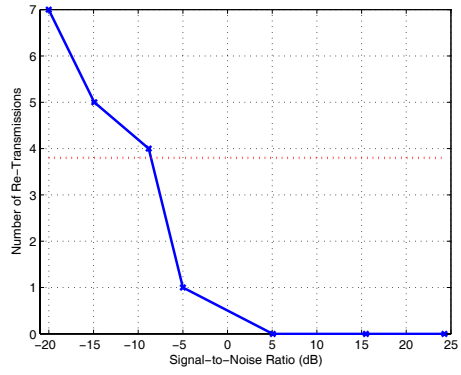

Figure 17: Number of Retransmissions vs SNR

We set up a sender and receiver pair, and the sender sends 100 data packets to the receiver. A packet is retransmitted until success-

fully delivered to the receiver. Therefore, the number of retransmissions equals to the number of losses and errors. Since attenuation directly affects the SNR at the receiver, and hence the loss and error rate, we use the number of retransmissions to measure the of amount of attenuation. We set the distance between the sender and receiver as 487 meters, which is one distance we used in the field test. Figure 17 plots the number of retransmissions in our testbed[2]. In the field test, the average number of retransmissions is 3.8, which is denoted as a dotted line in Figure 17. From the figure, we infer that the SNR in our field test under the same distance (i.e., 487 meters) may be roughly between $-10dB$ to $-5dB$.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented Aqua-Lab, a configurable lab testbed for underwater acoustic sensor networks. Aqua-Lab contains a real physical environment, a set of communication hardware, a programming library, and an emulator. The emulator allows users to emulate complex topologies, propagation delay and attenuation (with limited resources available in the testbed). We have conducted experiments to characterize data transmission using Micro-Modems in the testbed. Furthermore, by comparing results from the lab and the field environments, we demonstrate that our lab testbed can produce results compatible to those from the field experiments. Thus, Aqua-Lab can be used to experimentally evaluate algorithms and protocols designed for underwater sensor networks. Aqua-Lab will be open for researchers to explore many system issues in real underwater environments.

Our initial experiences with Aqua-Lab are very encouraging. At the same time, we also realize that there exist several limitations at this stage. First, the sound mixer can only be controlled in the lab, which means that remote users cannot adjust SNR. We plan to to use MATLAB to do noise playback and control the noise strength using MATLAB scripts. This makes it possible for remote users to dynamically change the noise strength via the Internet and emulate different channel conditions for different links. Another limitation is that we cannot emulate concurrent transmissions using two modems. But with more modems available in our water tank, we can easily extend our testbed to support concurrent transmissions/collisions.

In short, we believe Aqua-Lab is a platform that is more flexible and affordable than a field testbed. Furthermore, since it contains real acoustic communication channels, it provides an environment closer to reality than that by simulators. Last, Aqua-Lab can serve as a useful developing environment for real systems: the codes working in Aqua-Lab will be functioning in field tests (since we use the same acoustic modems as those in real systems).

We would like to pursue our future work in the following directions: 1) Implement and test other protocols, such as MAC and reliable transfer protocols, aiming to complete the protocol stack and provide a basic set of protocols for users; 2) Explore other acoustic modems, such as r-modem [12], s-modem [15], and mooring-modem [5], and examine the reconfigurability of Aqua-Lab; 3) Introduce sensing component and investigate integrated sensor node design using Aqua-Lab.

7. REFERENCES

[1] Underwater Sensor Network (UWSN) Lab, University of Connecticut, http://uwsn.engr.uconn.edu.

[2] Micro-Modem, http://acomms.whoi.edu/micromodem.

[3] Woods Hole Oceanographic Institution (WHOI), http://www.whoi.edu.

[4] I. F. Akyildiz, D. Pompili, and T. Melodia. Underwater acoustic sensor networks: Research challenges. *Ad Hoc Networks (Elsevier)*, 3(3):257–279, March 2005.

[5] B. Benson, G. Chang, D. Manov, B. Graham, and R. Kastner. Design of a Low-cost Acoustic Modem for Moored Oceanographic Applications. In *Proc. ACM International Workshop on UnderWater Networks (WUWNet)*, 2006.

[6] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou. Challenges: Building scalable mobile underwater wireless sensor networks for aquatic applications. *IEEE Network, Special Issue on Wireless Sensor Networking*, 20(3):12–18, 2006.

[7] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball. The WHOI Micro-Modem: An Acoustic Communcations and Navigation System for Multiple Platforms. In *Proc. of IEEE Oceans Conference, Washington DC*, 2005.

[8] Z. Guo, B. Wang, and J.-H. Cui. Efficient Error Recovery with Network Coding in Underwater Sensor Networks. In *Proc. IFIP Networking, Atlanta, GA, USA*, 2007.

[9] J. Heidemann, Y. Li, A. Syed, J. Wills, and W. Ye. Research challenges and applications for underwater sensor networking. In *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2006.

[10] J. Partan, J. Kurose, and B. N. Levine. A Survey of Practical Issues in Underwater Networks. In *Proc. of ACM International Workshop on UnderWater Networks (WUWNet)*, pages 17–24, September 2006.

[11] D. Pompili, T. Melodia, and I. F. Akyildiz. Routing Algorithms for Delay-insensitive and Delay-sensitive Applications in Underwater Sensor Networks. In *Proc. of ACM Conference on Mobile Computing and Networking (MobiCom)*, 2006.

[12] E. Sozer and M. Stojanovic. Reconfigurable Acoustic Modem for Underwater Sensor Networks. In *Proc. ACM International Workshop on UnderWater Networks (WUWNet)*, 2006.

[13] A. Syed and J. Heidemann. Time Synchronization for High Latency Acoustic Networks. In *Proc. IEEE Infocom, Barcelona, Spain*, 2006.

[14] R. J. Urick. *Principles of Underwater Sound*. McGraw-Hill, 1983.

[15] J. Wills, W. Ye, and J. Heidemann. Low-power Acoustic Modem for Dense Underwater Sensor Networks. In *Proc. ACM International Workshop on UnderWater Networks (WUWNet)*, 2006.

[16] P. Xie, J.-H. Cui, and L. Li. VBF: Vector-Based Forwarding Protocol for Underwater Sensor Networks. In *Proc. of IFIP Networking, Coimbra, Portugal*, 2006.

[17] J. Rice and et al. Evolution of Seaweb Underwater Acoustic Networking. In *Oceans Conf., Providence, RI, pp. 2007-2017*, 2000.

[18] M. S. Stewart and J. Pavlos. A means to networked persistent undersea surveillance. In *Technology Symposium*, 2006.

[2]Here the SNR is measured at the receiver before the demodulation module. Therefore, it might be be lower than the final SNR after demodulation. We argue that our major objective here is to show that we can emulate attenuation. Thus the specific SNR values are not of critical concern.