# Reconfigurable Acoustic Modem for Underwater Sensor Networks

Ethem Mutlu Sözer
MIT Sea Grant College Program
292 Main Street
E38-308
+1 (617) 253 6916
emsozer@mit.edu

Milica Stojanovic
MIT Sea Grant College Program
292 Main Street
E38-308
+1 (617) 253 7136
millitsa@mit.edu

## ABSTRACT
There is a growing interest for underwater sensor networks where long term monitoring of water masses around the world for scientific, environmental, commercial, and military reasons is desired. In this paper we will present the concept of a highly flexible acoustic modem called the Reconfigurable Modem (rModem) that can be used for rapid testing and development of such networks.

## Categories and Subject Descriptors
B.m [**Hardware**]: Miscellaneous

## General Terms
Experimentation

## Keywords
underwater, acoustic, network, experiment, rapid prototyping, rmodem.

## 1. INTRODUCTION
Underwater sensors are used for monitoring of interesting underwater environments. These sensors can be paired with acoustic modems configured in a network setting. Such sensor networks, which may also be connected to the Internet through a gateway, can provide near real-time access to the data collected by the nodes.

The challenging communications media created by the underwater acoustic channel force the researchers to redesign conventional networking algorithms. Specifically, the underwater environment presents an unreliable channel, due to extended multipath and high Doppler shift, together with long propagation delay, caused by the comparably low speed of sound (1500 m/s) [1]. The networking algorithms should be able to provide reliable and high throughput connections through this unreliable, long delay channel.


*WUWNet'06,* September 25, 2006, Los Angeles, California, USA.


Unlike the well known radio channels, the underwater acoustic channels do not have widely accepted models. Modified radio channel models are used in simulations for preliminary evaluation of physical and network layer algorithms. However, a new algorithm can only be fully tested through experimental studies where communication signals are transmitted through a real underwater channel. While off-line processing of the recorded signals is suitable for evaluation of point-to-point communication algorithms, multi point networking experiments require real-time processing.

Off-the-shelf modems can be employed for networking experiments. However, these modems usually present a black-box type physical layer where the user can choose from a number of configurations depending on the bit rate and/or reliability needs. Currently, there are a number of off-the-shelf modems that provide M-ary Frequency Shift Keying (MFSK) with channel coding, M-ary phase shift keying with equalization, and spread spectrum techniques. The physical layer implementations of these modems are closely tied to the hardware platform and cannot be modified. Also, some networking algorithms may require information and services from the physical layer that are not readily available, such as reliability information and power control. Proprietary software changes would be needed to extract such information or even hardware modifications may be needed to provide accurate, high resolution power control.

Most of the off-the-shelf modems are intended to be used as point-to-point links, and therefore lack resources to implement networking algorithms. Network layer algorithms can be implemented on a host computer that communicates to the modem through a serial connection. However, these algorithms have to utilize the packet structure dictated by the underlying physical layer. As shown in [2], the packet size may be an important factor in achieving optimum throughput.

Reconfigurable modem (rModem) is designed to simplify the experimental studies of new underwater acoustic sensor network algorithms on all layers with the possibility of cross layer optimization. rModem provides a unified simulation and rapid prototyping environment by exploiting Simulink tools. Simulation code developed by researchers during the development phase can be converted with little effort into real-time code that runs on a DSP board. The rModem hardware provides high processing gain and large memory, therefore relaxing the coding requirements and minimizing engineering work.

The rest of the paper is organized as follows: In sections 2 and 3, we introduce the hardware and software structure of the rModem. We also present a graphical user interface designed to aid data collection during the experimental studies. We conclude with a description of the first experimental trial of rModem and future directions.

## 2. rModem Hardware

The current implementation of the rModem hardware includes a main board with a daughter card interface (Figure 1). The main board has following features:

- Four configurable input and output channels suitable for multiple-input-multiple-output (MIMO) algorithms,
- Analog anti-aliasing filter with 1 kHz to 100 kHz bandwidth for wide range of operation,
- 240 kHz 16-bit analog-to-digital and digital-to-analog converters,
- DSP controlled, amplitude sensor aided automatic gain control (AGC) with up to 18dB gain,
- High resolution output volume control with 256 linear (or eight 3dB) steps for power control applications,
- Altera Cyclone II FPGA with 35k gates for IF processing,
- Texas Instruments TMS320 C6713 floating point digital signal processor (DSP) at 300 MHz with maximum 1800 MFLOPS,
- 32 Mbyte SDRAM for program and memory storage,
- 32 Mbyte FLASH RAM for persistant program and data storage,
- 56 pin daughter card port for expansion.

The FPGA IF processing enables the users to operate at any carrier frequency and bandwidth within the 1 kHz to 100 kHz band by only setting carrier coefficients, filter coefficients, and interpolation/decimation rates. The high processing power floating point DSP minimizes the time required to convert the simulation software into real-time application by eliminating the need to hand optimize the code.

Multiple input and output channels combined with an FPGA and a high end DSP increases the power requirement of the rModem hardware as compared to the off-the-shelf modems. Since rModem is intended to be used a research tool rather than a commercial product with long deployment life, high processing power is tolerated over long battery life.

The daughter card interface was utilized to add a power amplifier and preamplifier board to the rModem. This daughter card based design enables the user to operate rModem with various transducers. The current rModem has a daughter card that can drive a Teledyne AT-408 omnidirectional transducer that operates in the 9-14 kHz band.

Since one of the deployment platforms for the rModem will be autonomous underwater vehicles (AUVs), special attention was paid to the size of the system. The main board dimensions are 3" x 7". The peripheral boards stack on the DSP board. The final size of the system depends on the number of peripherals.

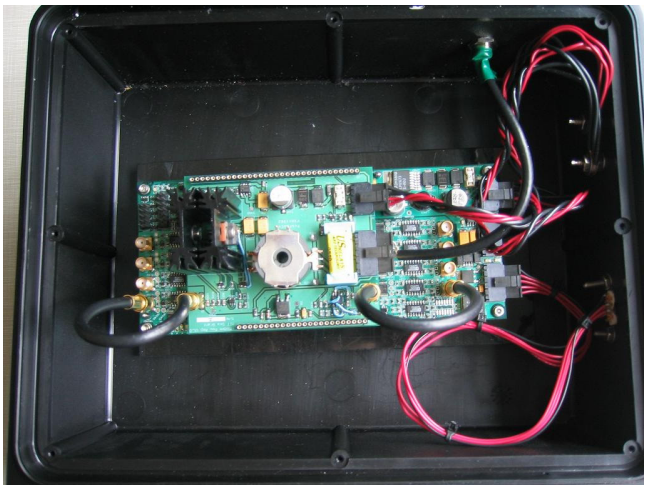

**Figure 1. Modem hardware has a main board and a daughter card for power amplifier and preamplifier.**

## 3. rModem Software

The rModem software is based on the Simulink environment and utilizes the Real-Time Workshop toolbox from MathWorks. The Simulink model of a typical rModem configuration is given in Figure 2. The low level functionality of rModem, such as the hardware drivers for analog parts and serial port, is implemented through target board blocks (e.g. UART Rcv and DAC block). The physical layer (layer 1), data link control layer (layer 2), and network layer (layer 3) are implemented with custom Simulink blocks.

We defined a communication packet as a collection of a preamble, dead time, training symbols, and data symbols. Packets are divided into frames. The frame size can be selected by the user. The physical layer sends one frame worth of data to the analog-to-digital converter at each clock tick. At the receiving side, the rModem processes one frame of symbols at each clock tick.

In a simulation environment, the entire communication packet can be processed in one simulation time instance regardless of its size. Also, one can make sure that the transmitter and receiver are synchronized in time. However, in a real-time system, communication signals have to be processed in frames rather then packets due to the limited memory of DSP systems and stringent real-time constraints.

The basic building blocks of a communication system, as we used them in the simulation environment, are simple algorithms that generate an output based on the input. To utilize these same blocks in a real-time system, some intelligence must be introduced into the system. Most of the blocks in the real-time system are used to control the data flow through the modem based on this framed structure. With the real-time modem control blocks in place, a researcher can insert the block that was designed for simple simulations into the appropriate section of the real-time system and obtain a prototype to test the performance of the new algorithm in real environments. This design strategy separates the real-time programming

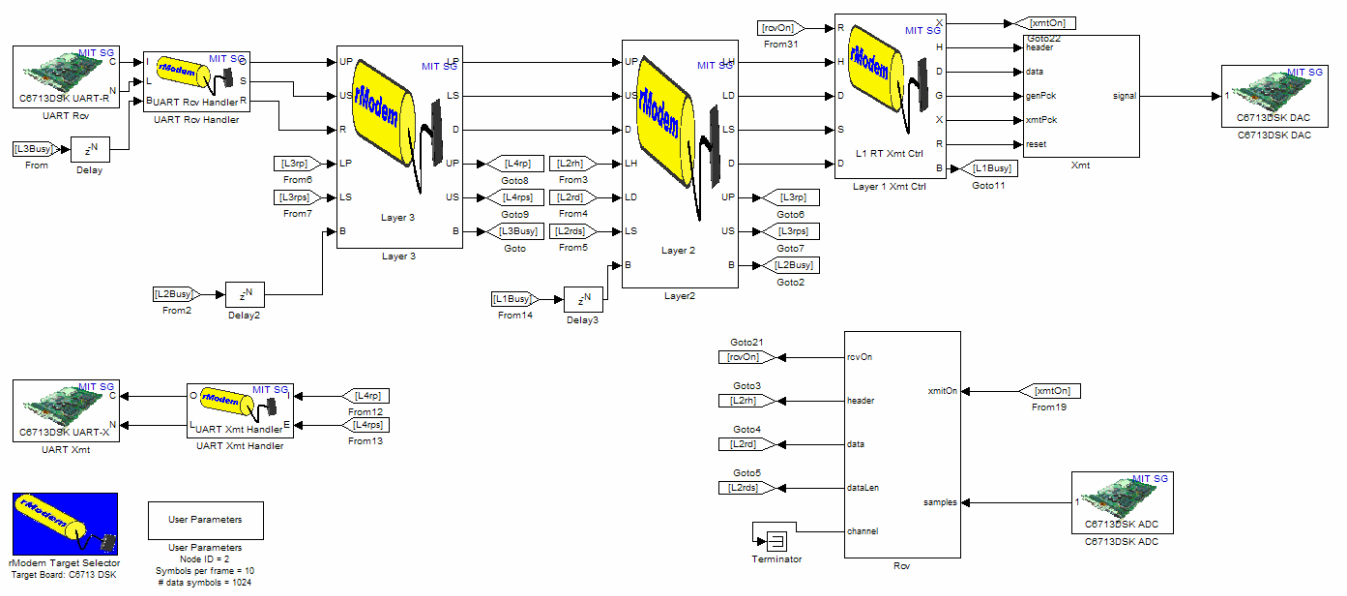


**Figure 2 The software block diagram of rModem as defined by the Simulink model**

requirements from the algorithm implementation, therefore making it possible to rapidly prototype new algorithms.

Some complex algorithms may require more time to be run by the DSP then the frame duration allows. If such an algorithm is implemented within the rModem, the real-time constraint cannot be met. This requirement can be relaxed by introducing processing delay during reception, as in the case of pseudo-real-time mode. In this mode, the rModem collects the received samples in memory instead of processing them immediately. The collected samples are then passed to a lower priority thread that can be interrupted by high level processes. When the receiver algorithm is run to completion, the results are passed to the upper layer, e.g. layer 2 [3].

Current rModem software supports layers from physical layer (layer 1) to network layer (layer3). These layers are connected to each other through a predefined interface. Any C-callable function that supports this interface can be easily converted into an rModem block and employed in the real-time model. It is also possible to include more layers in the rModem model. However, the memory requirements of higher layers should be taken into consideration.

## 4. Graphical User Interface

The rModem provides a JAVA based graphical user interface (GUI). This interface can be used to control the rModem hardware, send and receive packets, and log events and data.

The GUI defines a packet structure with the following fields: node ID, recipient ID, type, data size, user parameter 1, user parameter 2, and data. The user can set the size, i.e. number of bits reserved for each field. The GUI then checks if the total size of the packet conforms to the size expected by the rModem.

So far, we have defined three types of packets: text, file, and jpeg. The text type packets are intended to create an instant messaging type environment, where short text messages are exchanged between nodes. The file type packet is utilized to send large amount of binary data that needs to be saved at the destination. The jpeg type is a special case of file type where the image is automatically displayed on the screen.

The GUI can also be used to automate experiments through scripting. A script file is a text file that contains a delay time and a command on each line. The delay time determines the time the GUI waits before executing the corresponding command. If the modem is not ready to accept a new command when the delay timer expires, the GUI waits until the rModem becomes available again. Through scripting, a user can prepare various experimental scenarios and execute them multiple times without the need for constant human interaction.

The GUI can log all the communication between the host computer and the rModem. All the transmitted and received commands, including data packets, are logged in a text file with a time stamp. The data is stored in separate files to ensure readability of the log file. The log files can also be used as script files, therefore providing a way of recreating the experiments.

## 5. Experimental Work

The first experimental test of rModem was conducted in Woods Hole, MA, in June 2006. Four nodes were prepared to test a cooperative communications scheme [4]. The communication packets that contain a preamble (used for acquisition), training, and data, originated from the source node. Two intermediate nodes were configured as repeaters that acquire this packet utilizing the preamble, and store the training and data portions. The stored samples were retransmitted after appending a new preamble and a second training signal. One of the repeaters was required to time reverse the received samples. The destination node received both retransmissions to process them jointly.

The experiment was conducted with the source node and the destination node connected to a player/recorder device, while the repeater node was an rModem outfitted with a transducer. We modified the structure given in Figure 2 to implement the repeater functionality. We removed layer 3 and layer 2. We converted the UART Rcv Handler block to a command handler since no data was transmitted or received. We combined the Rcv and Xmt blocks to store, process, and forward the received signals according to the algorithm defined in [4].

The rModem successfully received the packets from the source node and stored passband training and data samples of the received packet. The stored samples were then time reversed and retransmitted with the addition of the new preamble and training sequence. The time reversed signals were recorded at the destination node. They are currently being processed off line. During this experiment, we did not test demodulation and detection in the rModem platform, which will be the subject of the next test.

## 6. Conclusions and Future Work

We demonstrated that the rModem hardware and real-time components of the software are operational. We will continue the development of rModem physical and network layer blocks with the ultimate goal of implementing an experimental underwater network.

**Acknowledgement**

This material is based upon work supported by the National Science Foundation under Grant No. 0520075.

## 7. REFERENCES

[1] Stojanovic, M. Recent advances in high-speed underwater acoustic communications. In IEEE J. Oceanic Eng., Vol. 21, pp. 125-136, April 1996.

[2] Stojanovic, M. Optimization of a data link protocol for an underwater acoustic channel. In *Proceedings of the IEEE Oceans'05 Europe Conference* (Brest, France, June 2005).

[3] Sozer, E.M. and Stojanovic, M. Simulation and rapid prototyping environment for AUV networks. In *Proceedings. of UUST'05* (Durham, NH, August 2005).

[4] Vajapeyam, M., Mitra, U., Preisig J. and Stojanovic,M. Distributed TR-STBC schemes for cooperative underwater acoustic communications. In *Proceedings of Oceans'06 Asia Pacific*. (Singapore, May 2006)