# Towards Automated Design of MAC Protocols for Underwater Wireless Networks

Volkan Rodoplu
vrodoplu@ece.ucsb.edu

Amir Aminzadeh Gohari
amirazg@ece.ucsb.edu

Wei Tang
w_tang@umail.ucsb.edu

Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, CA 93106-9560

## ABSTRACT

We present a framework for the automated design of MAC protocols for underwater acoustic wireless networks. We formulate a protocol optimization problem in which the exchange of control packets is explicitly modeled. A protocol optimization program generates the optimal response functions to the reception of control packets. In a single MAC neighborhood, each node is modeled as having a behavioral model of the rest of the nodes in the network, where the node behavior is to be optimized. In this framework, we solve the problem of minimizing the average energy consumption of a network subject to a per-node minimum throughput constraint. Our results display the optimal responses for the scheduling of control and data packets for all of the nodes. This work serves as a starting point for the design of automation tools for MAC protocols in the future.

## Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: Computer-Communication Networks

## General Terms

Design, Performance

## Keywords

wireless underwater protocol, network, automation, MAC

## 1. INTRODUCTION

One of the key methods of improving the performance of underwater wireless acoustic networks is the design of medium access control (MAC) protocols, which call for approaches that are very different from their terrestrial counterparts [1]. In the past three years, many MAC protocols have been proposed for underwater networks, each of which solves a problem under different assumptions on the amount of information available to the nodes. For example, in [2], an optimal MAC scheduling solution is developed under the assumption that the propagation delays for all of the node pairs are available. In [3], the UWAN-MAC protocol was developed under the assumption that the propagation delay information is not available, and that the transmit or receive power is much higher than the sleep power so that it does not pay off to estimate the propagation delay. In contrast, in [4], the propagation delay is estimated, and then utilized in scheduling transmissions. Reference [5] shows that not all propagation delays need to be estimated; knowing the maximum value of the propagation delay throughout the network allows nodes to communicate, albeit at the cost of increased delay. Reference [6] proposes a MAC protocol based on RTS/CTS exchanges, with warnings for deferral upon reception of new RTS packets. Reference [7] points out that the idle power is low for underwater modems, hence, the assumption of sleep schedules may not be justified in the design of MAC protocols for underwater networks.

First, each of the proposed MAC protocols makes different assumptions on the level of side information that is available to each node, especially on the availability of propagation delay information. Second, each of these protocols works in a particular regime of modem specifications: When the idle power is much greater than sleep power, then it pays off to use sleep schedules. Otherwise, it pays off for a node to stay awake both to be able to catch newcomers into the network, and to reduce the end-to-end delay. Third, the type of traffic determines whether a MAC protocol is "high-performance". Underwater sensor networks that collect CTD measurements generate only light traffic, and are highly delay-insensitive. In contrast, an underwater network of cameras requires high data rates, and are highly sensitive to delay if they are used, for example, for surveillance.

In this paper, we undertake a different, novel approach: The automated design of MAC protocols for underwater networks. We show how the process of MAC protocol design can be automated, via the development of appropriate automation tools. Our approach is different from finding the optimal schedules because we explicitly model the exchange of control packets in our optimization: As a result, our tools will be able to decide automatically, whether and when a node should be sending a control packet. The design of automation tools for networking protocol design has the potential to dramatically change the way that we conceive of and design protocols today. One of the key challenges in developing such tools is figuring out how to capture the human intuition that goes into the design process of hand-designed protocols. A second challenge is understanding the proto-


*WuWNet'08,* September 15, 2008, San Francisco, California, USA.

col space of all the protocols, a few instances of which we are familiar based on our design of protocols in the past. However, there are many more protocols that we have not encountered yet, and which will become available to us via these automation tools. A third challenge is defining the notion of protocol optimality, which involves an optimal exchange of control packets. These challenges are undertaken in this paper, and we display a framework and its results that overcome these challenges.

The rest of this paper is organized as follows: In Section 2, we state the basic assumptions and the structure of our solution. In Section 3, we describe how to write a protocol optimization program for energy minimization subject to a throughput constraint, when each node has three modes: sleep, transmit, and listen. In Section 4, we present our optimization results. In Section 5, we interpret our results, and discuss extensions of this framework to more general cases. In Section 6, we present our conclusions and discuss our future plans.

## 2. STRUCTURE AND ASSUMPTIONS

The key idea in this paper is that the protocol designer should aim to write not a particular protocol, but rather capture the communication and resource conditions that constrain the protocol design process, as well as the application-specific constraints, in a protocol optimization program. The optimization program should be as general as possible, to allow the set of all possible protocols within its feasible set. The control packet exchanges, in their most general form, must be modeled so that nodes can choose to generate control packets, as part of the optimization program. Once we specify this "protocol space" via such a program, state-of-the-art solvers can be employed to solve this program. The output of such a solver is a set of optimal *responses* that specify when a node decides to send control information and data in response to receptions from other nodes. Because the explicit exchange of control packets is modeled, these are responses, not just "schedules" that the nodes need to follow scheduled by a global scheduler.

Scheduling solutions assume that the nodes have access to the same, global control information. In contrast, a protocol is a process that resolves control information asymmetries between the nodes. The aim of protocol optimization is to specify the response that every node should generate given its knowledge state, to each possible input from the other nodes, such that a global network objective function is extremized. One of the ways to solve the protocol optimization program, is to attribute to each node within the optimization program, reasoning capabilities by which it can predict the responses of the other nodes to its behavior. Hence, every node in the optimization program is modeled as having, within itself, a behavioral model of the other nodes in the network, as shown in Fig. 1. (The figure assumes that there is a single MAC neighborhood with 3 nodes.) Throughout the paper, we assume that node $k$'s behavioral model of node $q$ is a correct model; that is, it matches $q$'s behavior exactly. Note that the behaviors themselves are optimization variables.

Throughout this paper, we assume that there is a single MAC neighborhood with $N$ nodes, each of which has an acoustic modem. The underwater acoustic medium has a propagation delay that is equal to the distance between two nodes divided by the speed of sound underwater. Each

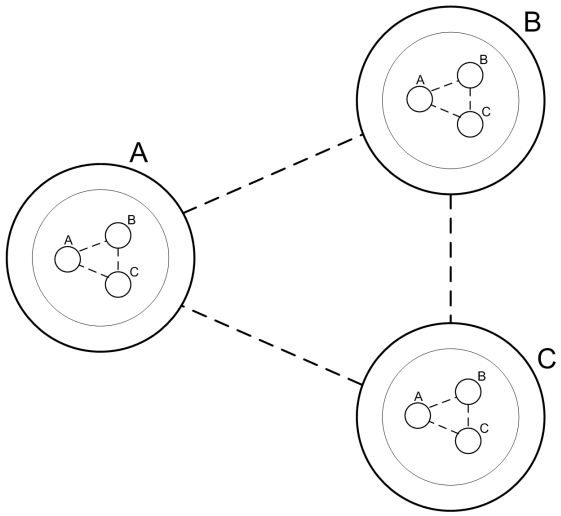


**Figure 1: Each node has, within it, a model of its "world".**

node has the capability to transmit, listen, and sleep, and can choose to do these autonomously. Further, it knows that the other $N-1$ nodes exist, and that they share the same MAC neighborhood; however, it does not know their transmit, listen, sleep schedules in the beginning. By exchanging control information packets with other nodes, a node may gain more information about the other nodes.

For modeling and simulation purposes, we have to represent the continuous time axis of this asynchronous channel in discrete time. To this end, we choose a time slot duration that is small enough that the error between this slot-synchronous model and the real asynchronous continuous-time model falls below a certain threshold. In this slot-synchronous model (which is, for example, used in any MATLAB simulation), even though the slots are globally aligned, none of the nodes initially know the origin of the discrete-time axis of any other node. Due to this model, from this point on, we shall refer to "time slots" in this global manner. Hence, in our propagation delay model, when a node $a$ transmits in a slot, that transmission lands at node $b$'s receiver, exactly $\tau_{ab}$ slots later, where $\tau_{ab}$ is the propagation delay from $a$ to $b$. In this paper, we model only the propagation delay through the channel; we do not model physical layer noise, and the data link layer issues. We plan to incorporate these into our future models.

In any slot, each node is in exactly one of the following modes: *sleep, transmit, and listen.* These are the *intrinsic* modes of a node; that is, they are without any regard to the node's interaction with the outside world. In a time slot, if a node $a$ listens, and there is exactly one transmission received from another node $b$, then the node *receives.* If node $a$ listens, and no transmission is received in that slot, then node $a$ is *idle* in that slot. If node $a$ listens, and at least one transmission lands at node $a$'s receiver in that slot, then node $a$ is *used*; this may correspond to an active reception or it may be a collision in which case it cannot decode either message; however, it still spends the receive power. These three modes of a node *receive, idle, used* are *extrinsic* modes; that is, they arise from the interaction of a node's intrinsic mode with the outside world.

In order to illustrate our framework, our aim in this paper

will be to minimize the total energy consumption of an underwater acoustic network subject to a minimum throughput constraint per node. To this end, we first state our energy consumption model: For simplicity, the energy specification in each mode is the same for every node: $E^{SP}$, $E^{TX}$, $E^{RX}$, $E^{ID}$ are the sleep, transmit, (active) receive, and the idle consumption energies per slot. These energies are found by multiplying the corresponding powers from the modem specs, by the length of the common time slot used in the simulations. We assume that the energy when a node is in the "used" mode is equal to that of the active receive mode[1].

Since we would like to model what control information to include, as part of our optimization program, it is important to think about the most general form of control information that is relevant in a MAC protocol optimization. In its most general form, a node may transmit to other nodes, control information that includes (1) its entire past, and (2) its entire plan for its predictable future. By the "entire past" of a node, we mean the entire set of receptions of CONTROL and DATA information of a node, from the other nodes, including, the time at which this information was received according to this node's clock. The predictability arises from the coherence time during which the environmental conditions remain the same. In order to model the coherence time of the channel, we use a time horizon of $M$ discrete time slots over which a node will optimize its decisions. In addition, the control information packet in which it sends its entire future will be within this $M$-window. Our finite-size $M$-window can also reflect an additional delay deadline by which the packets need to be sent out.

In the general problem set-up, there is no pre-determined leader, or any ordering of the nodes. In such a set-up, each node is allowed to initiate its own "thread" to send control messages to the other nodes. Depending on when the threads are received at the nodes, the threads can be terminated, and a single thread can be kept alive at any node, at any one time. Hence, the optimal protocol solves two problems simultaneously: A form of the leader election problem, where a leader can be elected with probability $(1-\epsilon)$, where $\epsilon > 0$ can be chosen to be as small as desired and fixed in the beginning[2], and the minimum energy problem, in which part of the energy is used in exchanging control messages that order the nodes and the other part on actual data transmissions. Such a formulation is certainly possible; however, it does not provide the simplest setting in which to expose the main ideas. As a simplification in this paper, we shall lose optimality by imposing a pre-determined node order, in which the nodes will be allowed to send control messages. This will constitute a single thread $A \rightarrow B \rightarrow C \rightarrow D...$, where $A$ is the first one to send the first control message; after $B$ receives this control message, it may initiate a control message to $C$, and so on. Hence, we focus on a single thread, initiated by $A$, and with no competing threads. This is for expositional simplicity; however, it also shows how this automated design paradigm can be used in practice. Fairly general protocol structures can be imposed for simplicity at an abstract level: These sacrifice protocol optimality, however, still allow optimization over a fairly large set of protocols, typically much larger than what is available via hand-design of protocols.

[1]Our model can be easily modified to take these as different if needed. We also do not model the "transition energy" to switch between different modes; however, this can easily be incorporated into our optimization model.

[2]The leader election problem can be solved only in this probabilistic sense in an asynchronous system.

# 3. PROTOCOL OPTIMIZATION

Our main aim in this section, will be to set up a protocol optimization program that minimizes the average energy consumption of a node, subject to a per-node throughput constraint $\theta_{min}$, in a single MAC neighborhood with $N$ nodes. All of the nodes use acoustic links for transmission, and the propagation delay from Node $a$ to Node $b$, denoted by $\tau_{ab}$, does not vary with time within the $M$ time slots. In this paper, we formulate this optimization program, assuming that each node knows the propagation delay between every pair of nodes, but that it does not know the waveforms used by the other nodes. The propagation delay estimation and this program can in fact be jointly formulated, and we discuss this joint formulation in Section 5.

We let $\aleph$ be the entire set of nodes. By our final assumption in the last section, Nodes are labeled as 1 through $N$, according to this order, where 1 is the leader that will initiate the control packet thread, and 1 through $N$ is the pre-determined sequence of nodes in this thread.

We now fix a node $k$ in the node set, and go inside the scope of Node $k$, and define the variables that Node $k$ uses in its optimization program. All of the node names, such as $q$, which appear below, are representations inside the scope of the optimization program of Node $k$.

Let $z_q^{\mu}[i]$ be a Boolean variable that is 1 if and only if Node $q$ is in intrinsic mode $\mu \in \{\text{SP, TX, LI}\}$ in time slot $i$, where $SP$ stands for "sleep", $TX$ stands for "transmit", and $LI$ stands for "listen". Hence, $z_q^{SP}[i] = 1$ if and only if Node $q$ sleeps in slot $i$; $z_q^{TX}[i] = 1$ if and only if Node $q$ transmits in slot $i$; $z_q^{LI}[i] = 1$ if and only if Node $q$ listens in slot $i$. We let $z_q^{US}[i] = 1$ if and only if Node $q$ is "used" in that slot; that is, at least one transmission lands at Node $q$'s receiver at time $i$.

We use global time slot indices $i : 1 \leq i \leq M$, where $M$ is the time horizon for our optimization program. In a real application, the choice of $M$ is determined by the delay constraint on data, as well as the channel coherence time.

We let the Boolean variable $z_q^{ID}[i] = 1$ if and only if Node $q$ listens at time $i$, and there is no transmission that lands at $q$'s receiver at time $i$. The Boolean variable $(z^{RX})_q^l[i] = 1$ if and only if Node $q$ successfully receives a transmission at time $i$, from Node $l$. The auxiliary Boolean variable $y_q^l[i] = 1$ if and only if Node $q$ is "free to receive" at time $i$ from Node $l$; that is, if Node $l$ decides to transmit to Node $q$ such that its transmission to Node $q$ lands at Node $q$'s receiver at time $i$, there is no other transmission that lands at that time at node $q$, and Node $q$ does not transmit in that slot, either. Further, we let $y_q^{\emptyset}[i] = 1$ if and only if no node's transmission lands at node $q$'s receiver at the $i$th time slot. We let $z_q^{RX}[i] = 1$ if and only if Node $q$ correctly receives from some other node at time $i$.

We let the Boolean variable $\tilde{\mathbf{x}}^q[i] = 1$ if and only if, in Node $k$'s model of the world, Node $q$ sends a control packet in time slot $i$. We let the Boolean variable $S_q^l[i] = 1$ if and only if, in Node $k$'s model of the world, a transmission of a control packet from Node $l$ to Node $q$ has been successfully received by Node $q$ *some time before time* $i$. This variable denotes "success", namely that in Node $k$'s model of the

world, Node $l$ has been able to "get its control packet through to Node $q$" by time $i$.

In Node $k$'s model, before the first control packet gets through to $q$, Node $k$ models $q$'s initial waveform as resulting from a probability distribution $\mathbf{P}$. This probability distribution itself is an optimization variable. In this paper, we shall optimize over only the set of memoryless distributions, thereby losing optimality. In our future work, we plan to include distributions with memory.

We let the Boolean variable $W_q^{\mu}[i] = 1$ if and only if, in Node $k$'s model of the world, Node $q$ is in intrinsic mode $\mu$ at time $i$, *before* the success $S_q^{q-1}$ has occurred. Note that due to our imposition of a single thread, the only success that matters is $S_q^{q-1}$, namely the successful reception of a control packet from the preceding node in the sequence. (For Node 1, the success is taken as already having occurred.) Intuitively, $W_q^{\mu}$'s are the random Boolean waveforms, one per intrinsic mode $\mu$, by which Node $k$ models Node $q$ before Node $q-1$ gets a control packet through to $q$ in Node $k$'s model of the world.

We say that Node $l$ "has access" to Node $m$'s waveform (one per intrinsic mode) if a chain of control packets has reached from Node $l$ to Node $m$ (possibly via other nodes or directly). When a control packet reaches a node, the node becomes aware of the planned future schedules of the nodes that are "superior" to that node; that is, the set of nodes that are up the chain from that node. Based on this information, the node can adjust its own waveform (one per intrinsic mode). Now, because the nodes up the chain from this node are modeling in advance, their obtaining access to this node after a chain of control packets gets through, they should model this node's waveform as random before the control packet success, and deterministically afterwards. We let the Boolean variable $x_q^{\mu}[i] = 1$ if and only if, in Node $k$'s model of the world, Node $q$ is in intrinsic mode $\mu$ at time $i$, *after* the success $S_q^{q-1}$ has occurred.

We let $R_k$ denote the "superior set" of Node $k$; namely, $R_k \stackrel{\text{def}}{=} \{1, 2, ..., k-1\}$, according to our labeling convention. We let $F_k$ denote the "inferior set" of Node $k$; namely, $F_k \stackrel{\text{def}}{=} \{k+1, k+2, ..., N\}$. Finally, we let $H_k$ denote the union of the inferior set of Node $k$ and itself, namely $H_k \stackrel{\text{def}}{=} \{k, k+1, k+2, ...N\}$.

The control packet structure of node is as follows: The "entire past" of a node is the content of the set of control packets that it has heard from other nodes: In this case, through node $k-1$, node $k$ has heard all of the contents of the control packets in its superior set $R_k$, which contain the future planned schedules of those nodes. In its own control packet, Node $k$ will send this information, as well as its own future planned schedule, which we discuss next.

Node $k$'s model of the world is as follows: It takes the waveforms of its superior set, that are communicated to it via control packets, as given. We let $(\mathbf{x}_q^{\mu})^*, q \in R_k$ denote the optimally determined waveform of Node $q$ (one per intrinsic mode), which is communicated to $k$, which $k$ thus takes as given. Node $k$'s aim is to optimize its own waveforms $x_k^{\mu}$, from the current time $I$ at which it runs its optimization program, until $M$. However, it also models that after the chain of successes occurs at the nodes in the inferior set of $k$, the waveform of the node at which the success occurred becomes a deterministic waveform because Node $k$ knows that each node $q \in F_k$ will determine its own waveform optimally while taking $R_q$'s waveforms as given, and taking $F_q$'s waveforms as optimization variables. Hence, foreseeing future successes in advance, the superior nodes utilize this knowledge in choosing their own waveforms optimally in the first place.

We assume in this paper, that all of the control packets are broadcast (rather than unicast or multicast). This makes sense in our context because through the control packets, we aim to disseminate as much side information as possible to all of the nodes who can hear it.

Recall that $I \stackrel{\text{def}}{=} \min_{i:1 \le i \le M}\{i \mid \text{Node } k \text{ has received a control packet from Node } k-1 \text{ before time } i\}$. We let $\{-u\}$ denote the set of all of the nodes in the node set, except the node $u$. We define the maximum propagation delay in the network $\tau_{max} \triangleq \max_{(r,s)\in\aleph\times\aleph, r\neq s} \tau_{rs}$[3]. We let $(W_r^{\mu})^{[I,M]}$ denote the random waveform of node $r$ in the time interval $[I, M]$, one waveform for each intrinsic mode $\mu$. We let $\left(\bigtimes (W_r^{\mu})^{[I,M]}\right)_{r\in F_k}$ denote the vector of such random variables, over $k$'s inferior set $F_k$. Furthermore, $(w_r^{\mu})^{[I,M]}$ is a particular value of this vector. We let $\mathcal{W}_r^{\mu}$ denote the (Boolean) alphabet in which the waveform for the intrinsic mode $\mu$ of node $r$ takes it values. We let $\{\mathbf{X}_q^{\mu}\}_{\mu\in\{TX,SP,LI\}}$ define the collection of $\mathbf{x}_q^{\mu}$'s over the set of intrinsic modes.

Given all of the above definitions, the optimization program, run by node $k$ at time $I$, is as follows:

**Protocol Optimization: Energy Minimization**

$\mathbf{P} \triangleq P_{\left(\bigtimes (W_r^{\mu})^{[I,M]}\right)_{r\in F_k}}(\{(w_r^{\mu})^{[I,M]}\})$

$\mathbf{a} \triangleq \{\tilde{\mathbf{x}}^q, \{\mathbf{X}_q^{\mu}\}_{\mu\in\{TX,SP,LI\}}\}_{q\in H_k}$

$$\min_{\mathbf{a}} \sum_{\{(w_r^{\mu})^{[I,M]}\}\in\left(\bigtimes (\mathcal{W}_r^{\mu})^{[I,M]}\right)_{r\in F_k}} \mathbf{P} \sum_{i=I}^{M+\tau_{max}} \sum_{l\in\aleph} \left(E^{TX} z_l^{TX}[i] + E^{RX} z_l^{US}[i] + E^{ID} z_l^{ID}[i] + E^{SP} z_l^{SP}[i]\right) \quad (1)$$

subject to:

1. $\forall\, q \in H_k, \forall\, i : I \le i \le M,$

$$z_q^{SP}[i] + z_q^{TX}[i] + z_q^{LI}[i] = 1$$

2. $\forall\, q \in H_k, \forall\, i : I \le i \le M,$

$$z_q^{ID}[i] = z_q^{LI}[i]\; y_q^{\emptyset}[i]$$

3. $\forall\, q \in H_k, \forall\, i : I \le i \le M, \forall\, l \in \{-q\}$

$$\left(z^{RX}\right)_q^l[i] = z_l^{TX}[i - \tau_{lq}]\; y_q^l[i]$$

4. $\forall\, q \in H_k, \forall\, i : I \le i \le M,$

$$z_q^{US}[i] = z_q^{LI}[i] \bigvee_{l\in\{-q\}} z_l^{TX}[i - \tau_{lq}]$$

[3]The exact value of $\tau_{max}$ is not assumed to be known by any node. However, a node is required to know at least the order magnitude of $\tau_{max}$ within a single MAC neighborhood, so that it can choose a large enough one that will suffice. A node can estimate this order of magnitude based on the range of its own transmission. Each node's own estimate of $\tau_{max}$ can be substituted above.

5. $\forall\ q \in H_k, \forall\ i : I \leq i \leq M,$

$$z_q^{RX}[i] = \bigvee_{l \in \{-q\}} (z^{RX})_q^l[i]$$

6. $z_q^{\mu}[i] =$

$$\begin{cases} (x_q^{\mu})^*[i] & q \in R_k, 1 \leq i \leq M \\ x_q^{\mu}[i] & q = k, I \leq i \leq M \\ \left(w_q^{\mu}[i](S_q^{q-1}[i])'\right) \vee \left(x_q^{\mu}[i] S_q^{q-1}[i]\right) & q \in F_k, I \leq i \leq M \end{cases}$$

7. $\forall\ q \in F_k,$

- $S_q^{q-1}[I] = 0$
- $\forall\ i : I+1 \leq i \leq M,$

$$S_q^{q-1}[i] = \bigvee_{j=(I+\tau_{q-1,q})}^{[(i-1)+\tau_{q-1,q}]} \tilde{x}^{q-1}[j-\tau_{q-1,q}]\ y_q^{q-1}[j]$$

8. $\forall\ q \in \aleph, \forall\ v \in \{-q\},$

$y_q^v[i] =$

$$\begin{cases} 0 & I \leq i \leq \tau_{vq} \\ \bigwedge\limits_{\substack{m \in \sigma_{\{-v\}} \\ i-\tau_{mq} \geq 1}} z_q^{LI}[i](z_m^{TX}[i-\tau_{mq}])' & I_{max} \leq i \leq M + \tau_{max} \end{cases}$$

where $I_{max} = \max\{\tau_{vq}+1, I\}$

9. - $\forall\ q \in R_k : \tilde{x}^q[i] = (\tilde{x}^q)^*[i], 1 \leq i \leq M$
   - $\forall\ q \in H_k : (\tilde{x}^q[i] = 1) \Rightarrow (z_q^{TX}[i] = 1), I \leq i \leq M$

10. $\forall\ l \in H_k,$

$$\frac{1}{(M-I+1)} \sum_{i=I}^{M+\tau_{max}} \left[ z_l^{RX}[i] - \left( \bigvee_{m \in \{-l\}} \tilde{x}^m[i-\tau_{ml}] y_l^m[i] \right) \right] \geq \beta\ \theta_{min}$$

$0 < \theta_{min} \leq 1,\ 0 < \beta \leq 1$

11. $\forall\ l \in H_k, \forall\ i : I \leq i \leq M,$

$$(S_N^{N-1}[i] = 1, S_N^{N-1}[i-1] = 0) \Rightarrow \left( \frac{1}{(M-i+1)} \sum_{j=i}^{M+\tau_{max}} \left[ z_l^{RX}[i] - \left( \bigvee_{m \in \{-l\}} \tilde{x}^m[i-\tau_{ml}] y_l^m[i] \right) \right] \geq \theta_{min} \right)$$

$0 < \theta_{min} \leq 1$

12. $\forall\ q \in F_k, \forall\ i : I \leq i \leq M,$

$$S_q^{q-1}[i] = 0 \implies \tilde{x}^q[i] = 0$$

13. $\forall\ q \in \aleph, \forall\ i : M < i \leq M + \tau_{max},$

$$z_q^{TX}[i] = 0$$

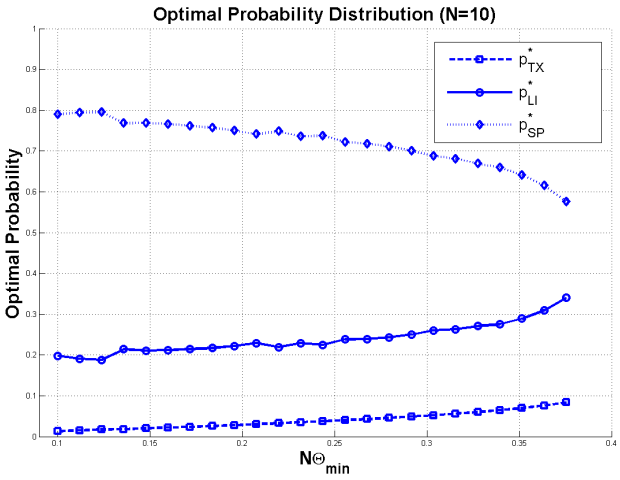


**Figure 2: The optimal values of the parameters of the $\mathbf{P}^*$ distribution as a function of the required aggregate throughput.**

The objective is to minimize the total energy consumption of all the nodes, in Node $k$'s model of the world, from the current time $I$, to time $M$, averaged over the $w_r^{\mu}$ waveforms (one per intrinsic mode) utilized by the nodes in this time interval, according to the joint probability mass function $\mathbf{P}$. Above, 1) states that a node can be only one of its intrinsic modes; 2), 4), and 5) state the definitions of the *idle, used* and *receive* extrinsic modes, respectively; 6) states the "evolution equations": For every node in the superior set $R_k$ of Node $k$, the waveforms, which have been communicated in the control packets, are taken as given. For itself, Node $k$'s waveform $\mathbf{x}_q^{\mu}$ is an optimization variable. The nodes in the inferior set of Node $k$ are modeled, by $k$, as transmitting with initial random waveforms in the beginning, until the first success to that node occurs. This successful transmission of the control packet brings to the node, the waveforms that are used by the nodes in its superior set. 7) states the definition of "success"; namely, success occurs, in $k$'s model, when the control information from node $q-1$ gets through to $q$. The second part of 9) states that if a control information packet is scheduled in slot $i$, then the node is required to be transmitting in that slot; 12) states that a node is allowed to schedule a control packet (in our restricted scheme) only when a control packet from a node in its superior set has already gotten through to it; and 13) states a simulation-type constraint in Node $k$'s model, which does not allow any node to schedule transmissions in the last $\tau_{max}$ slots. This is required to address the boundary condition, namely that the window size $M$ is finite.

We now discuss 10) and 11) above in more detail. The parameter $\beta$ is a margin that is close to 1, and $\theta_{min}$, as mentioned, is the minimum average[4] target throughput per node. 11) states that if, in $k$'s model of the world, the success to the last node $N$ occurs in the remaining time slots, then, $k$ will aim to schedule its own waveform (one per intrinsic

[4]The term $1/(M-I+1)$ rather than $1/(M+\tau_{max}-I+1)$ is correctly used as the normalization, in order not to penalize the nodes for the unavoidable edge effect that occurs due to $\tau_{max}$. By 13), the nodes are allowed to send in only the slots up to time $M$.

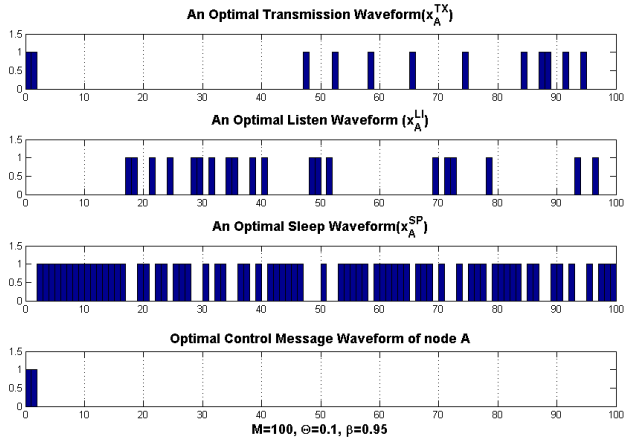


**Figure 3: The optimal response functions computed for the WHOI Modem specifications.**

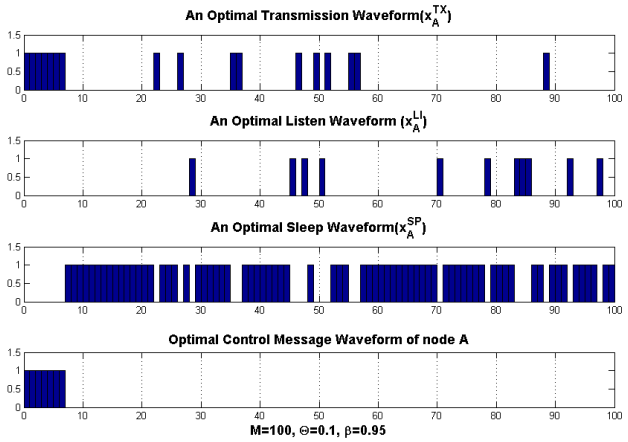


**Figure 4: The optimal response functions computed for an underwater modem that has the Cisco Air Modem energy specifications.**

mode) such that fairness is achieved exactly for all nodes in $H_k$. (This may not be feasible if $\theta_{min}$ becomes too large.) Constraint 10) states a laxer constraint, namely that overall, whether the success to the last node occurs or not, Node $k$, in its model of the world, will aim to schedule its own waveform such that fairness is achieved for all nodes in $H_k$ within a margin $\beta$. We aim for a laxer constraint because overall, we account for the case where the entire thread might not complete.

Finally, we note that in this formulation every DATA packet is broadcast (rather than unicast or multicast) in this single MAC neighborhood. This will be the case in underwater sensor networks, when a sensor node's DATA can be received and cached by each of its neighbors, to create replicas to allow for flexibility in the routing protocol design. Unicast or multicast DATA constraints can also be easily added to the above program; however, for simplicity, we do not treat this in this paper.

# 4. RESULTS

In this section, we present examples of the results of running our protocol optimization program. The methodology for solving the protocol optimization program can be described as follows: First, all of the Boolean equations are converted into linear inequalities that involve integer variables. For example, $a \vee b = 1$ is converted into $a + b \geq 1$. By writing down the rules for the conversion of all of the basic operations, and defining new variables for every such instance, the entire program is converted into a form that is a Binary Integer Linear Program (BILP), for every fixed $\mathbf{P}$. There are state-of-the-art engines for solving BILPs, of which MiniSAT+ is probably the most well-known. For each fixed $\mathbf{P}$, the complexity of the resulting BILP is $\mathcal{O}(NM)$. (Note that the equality constraints are easily eliminable via substitution.) Since, in our case, this program needs to be run by a chain of $N$ nodes, the total computational complexity is $\mathcal{O}(N^2M)$, for a fixed $\mathbf{P}$ vector.

The optimal $\mathbf{P}$ distribution is determined by solving a program similar to that of Node 1. The only difference is that $x_1^\mu = w_1^\mu$; that is, the initial waveform of Node 1 is also taken as random; however, the remainder of the constraints are the same. Optimizing over the entire set of $\mathbf{P}$ distributions is computationally intensive. We have decided to incur a loss in optimality, by searching over only the memoryless, time-invariant $\mathbf{P}$ distributions. (Here, time-invariant means that the distribution parameters do not vary over time.)

In our set-up, we use the parameters of the WHOI modem, namely, the ratio of the transmit, receive, idle and sleep energies are 10000:3000:80:0. Fig. 2 shows the $\mathbf{P}^*$ distribution for $N = 10$. The optimal values $p^*_{SP}$, $p^*_{TX}$, $p^*_{LI}$ (two independent, one dependent) of the distribution parameters are shown as a function of $N\theta_{min}$. As the aggregate throughput increases, the value of the optimal sleep parameter is reduced, and that of the optimal transmit parameter increases. Hence, the protocol optimization tells us explicitly, the optimal distribution that every node should use in selecting its initial, random transmit-listen-sleep waveforms.

Fig. 3 presents the optimal response function of Node 1 (labeled as Node $A$ in the plot) for the $N = 3$ case, when the modem energy specifications are chosen to be those of the WHOI modem. The pair-wise propagation delays are taken to be $\tau_{12} = 5$, $\tau_{23} = 5$, $\tau_{13} = 8$. We present this to compare it with Fig. 4, where the optimal response function of Node 1 (again labeled as Node $A$ in the plot) is computed for the Cisco Air Modem, which has the transmit, receive, idle, sleep power ratio of 2240:1350:1350:75. Even though the Cisco Air Modem would not work underwater, we make this comparison to show how the generated response functions would differ dramatically for an underwater modem with the same energy specifications as the Cisco Air Modem. For the WHOI modem, we see that $A$ schedules only a single control packet in the first time slot. In contrast, for the underwater modem with the Cisco Air Modem specs, $A$ schedules four successive control packets starting in the first slot. The reason is that for the WHOI modem, because the transmit power is much greater than the receive and idle powers, it schedules as few control packets as possible, and instead listens to the channel. In contrast, the underwater modem with the Cisco Air Modem specs, attempts to maximize the probability of getting through to the other so that it can sleep as much as possible later, rather than listen. The reason is that its transmit power is comparable to its idle power, which is much higher than its sleep power.

Our main point is that we are able to generate the optimal

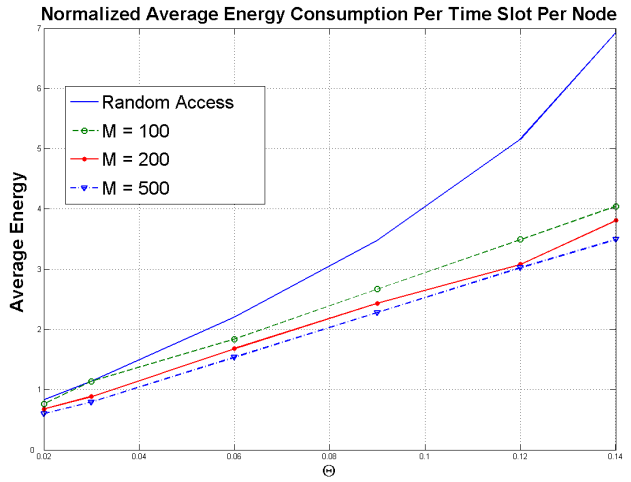


**Figure 5: The average energy consumption versus the minimum required throughput per node, parameterized by $M$, compared with random access.**

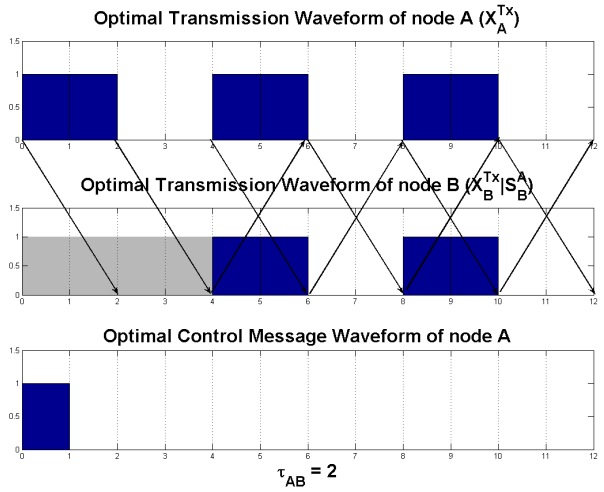


**Figure 6: The "butterfly" effect, discovered to be the optimal transmission solution, when throughput is the objective function.**

response functions quickly when the modem specification changes. Many protocols that are hand-designed have to be thrown away, and MAC protocols have to be re-designed completely when the modem specifications change. However, here, we show that we can quickly find the new optimal protocol, as a set of response functions, when the modem changes. This reconfigurability, even though performed offline, is expected to dramatically decrease the protocol design cycles in the future.

Fig. 5 displays the average energy consumption per node per time slot as a function of $\theta_{min}$ for the WHOI modem specifications, parameterized by $M$. As the required throughput increases, we see that the performance of protocol optimization increases with respect to the pure random access scheme; that is, the gap between them widens. Further, we see that increasing $M$ does not result in much improvement beyond 200; hence $M$ need not be large to obtain close-to-optimal energy consumption.

Finally, even though we did not present the formulation for throughput maximization in this paper, we have carried out the formulation to maximize the throughput of an $N$-node network with delays. This formulation can be obtained by replacing the objective function in the energy minimization program with the throughput maximization function. For the $N = 2$ case, we know that the optimal transmission schedule after the control packet gets through from Node $A$ to Node $B$, is the "butterfly" transmission where each node transmits at intervals of $\tau_{AB}$ in order to fill each other's time axes perfectly. Indeed, our optimization protocol engine produces this butterfly transmission, as shown in Fig. 6, for $\tau_{AB} = \tau_{BA} = 2$ time slots. The figure shows the optimal solution, where, as it turns out, Node $A$ schedules its control packet in the first slot. The second figure shows $B$'s optimal response function, conditioned on the success from $A$, namely $S_B^A$, in which case, $B$ starts following the butterfly transmission scheme.

## 5. DISCUSSION

The result of our protocol optimization program is a set of optimal waveforms. Because we restricted our protocol structure at the top level, to a single thread of control packets that began at Node 1, each node in this case has a single response function that determines how it responds to the successful reception of this control packet. (When multiple threads are involved, there will be more than one response function.)

These optimal response functions are thus computed offline via our protocol optimization program, and are downloaded to the nodes at deployment time. The optimal response function shows how to respond to control packets, when the node is in a particular "role". If the node's role is the leader (Node 1), then it looks up exactly which $\tilde{\mathbf{x}}$, and $\mathbf{x}$ to use from the downloaded look-up table. In the role of Node 2, the second one down the chain, the node looks up with what waveform to respond to the control packet reception. In essence, this is a protocol, albeit a simple one. A key issue is how the nodes would know how to respond to different network conditions. For example, as the values of the parameter $\theta_{min}$ changes, a different response will be necessary. Our idea for future work is to "extract a protocol" as a relationship between the values of these network parameters, and the optimal response functions. We will require that this mapping be simple, perhaps at the cost of some performance loss. However, the main drive is to automate this process rather than hand-design.

We now discuss how two key mechanisms, which we could not address within the scope of this paper, can be added to the protocol optimization framework. First, our protocol optimization program does not have any acknowledgement (ACK) mechanism. With an ACK mechanism, better results than what we presented might be obtained because a node $k$ need not schedule its control information again after it hears that it has been received by $k + 1$. Building such a mechanism into the program involves defining "successes" for the reception of acknowledgements from the other nodes. Note that by [8], there is no mechanism that can guarantee consensus in an environment with collisions. Hence, one must stop the ACKs at some level. For energy minimization, this is usually close to the level at which we stopped in this paper, namely, not sending any ACKs at all.

The propagation delay estimation is a second mechanism

that can be incorporated into this protocol optimization program. In this case, at any point in time, the links, as modeled by Node $k$, fall into two categories: Those whose delays $k$ knows, either via its own measurement or this information's reaching $k$ via the chain of control packets, and those that $k$ does not know. We assume that each node knows the propagation delay distribution, but not the values of the propagation delays. (The distribution can be taken as uniform by a node, between a minimum and a maximum, if there is no other information.) An additional term is added in this case to the objective function, that averages over the distribution of the links whose delays are not currently known by $k$. Hence, the constraints that involve $\tau_{ab}$'s in the program remain the same, and correspond to realizations if they are random, and to the actual values if the values are known. Finally, a set of constraints is added to complete a handshake for the propagation delay estimation.

Finally, we discuss how the protocol optimization program can be generalized further by allowing each node to initiate its own thread (rather than allowing only Node 1 to do so). Note that we did not utilize the timing information, which is in fact part of the general control packet structure. The time stamps indicate also when the current transmission starts, according to the sender's own clock. Since the content of a node's past control information receptions is included in its own control packet, every node can determine at any time, which of the multiple threads it has received was the earliest one. Hence, at each point, only a single thread at each node can be kept alive. Then, a node can run its energy minimization program (the one that we presented), not for the global thread as in this paper, but for the thread that is currently alive at time $I$, and determine its optimal output accordingly.

# 6. CONCLUSIONS

The protocol optimization framework that we presented in this paper can be seen as a first step toward the challenging problem of automating the design of networking protocols[5] for underwater networks. Such automation has much promise to completely change the way we design protocols, dramatically reduce the long design cycles, and quickly find near-optimal protocols, most of which we might miss in our hand-designs. Our current framework allows the designers to describe the protocol space within which they are searching for protocols, via a set of constraints. Our eventual aim is to build a library of such protocol optimization programs, that will be publicly available, upon which more sophisticated optimization programs with more constraints can be built for different types of applications, starting from a base case. This joint effort by researchers across the globe is expected to dramatically reduce the long cycles in the design of customized protocols for specific applications. For underwater acoustic networks, it will likely result in the fast development of high-performance MAC protocols that address propagation delays, energy efficiency as well as maximum throughput.

# 7. REFERENCES


[1] I. F. Akyildiz, D. Pompili, T. Melodia, "State-of-the-art in protocol research for underwater sensor networks", in *Proc. ACM WUWNet 2006*, pp. 7-16, Sep. 2006.

[2] L. Badia, M. Mastrogiovanni, C. Petrioli, S. Stefanakos, and M. Zorzi, "An optimization framework for joint sensor deployment, link scheduling and routing in underwater sensor networks," in *Proc. of the 1st ACM Workshop on Underwater Networks (WUWNet'06)*, pp. 56–63, Sep. 2006.

[3] M. K. Park and V. Rodoplu, "UWAN-MAC: An energy-efficient MAC protocol for underwater acoustic wireless networks," *IEEE J. Oceanic Engineering*, vol. 32, no. 3, pp. 710-720, July 2007.

[4] P. Xie and J. Cui, "R-MAC: an energy-efficient MAC protocol for underwater sensor networks," in *Proc. of International conference on Wireless Algorithms, Systems and Applications (WASA'07)*, pp. 187–198, Aug. 2007.

[5] X. Guo, M. R. Frater, and M. J. Ryan, "A propagation-delay-tolerant collision avoidance protocol for underwater acoustic sensor networks," in *Proc. of the MTS/IEEE OCEANS 2007-Asia Pacific*, pp. 1–6, May 2007.

[6] B. Peleato and M. Stojanovic, "A MAC protocol for ad-hoc underwater acoustic sensor networks," in *Proc. of the 1st ACM Workshop on Underwater Networks (WUWNet'06)*, pp. 113–115, Sep. 2006.

[7] A. F. Harris III, M. Stojanovic, and M. Zorzi, "When underwater acoustic nodes should sleep with one eye open: Idle-time power management in underwater sensor networks," in *Proc. of the 1st ACM Workshop on Underwater Networks (WUWNet'06)*, pp. 105–108, Sep. 2006.

[8] M. J. Fischer, N. A. Lynch, M.S. Paterson, "Impossibility of distributed consensus with one faulty process", *J. ACM*, vol. 32, no. 2, 1985, pp. 374-382.

[9] V. Rodoplu, and A. Aminzadeh Gohari, "Challenges: Automated Design of Networking Protocols," in *Proc. of ACM International Conference on Mobile Computing and Networking (MobiCom'08)*, Sept. 2008.


[5] See [9] for a discussion of further challenges.