

# UWAN-MAC: An Energy-Efficient MAC Protocol for Underwater Acoustic Wireless Sensor Networks

Min Kyoung Park, *Member, IEEE*, and Volkan Rodoplu, *Member, IEEE*

**Abstract**—In this paper, we propose a distributed, scalable, energy-efficient media access control (MAC) protocol that works despite long, unknown propagation delays of the underwater acoustic medium. This protocol can be used for delay-tolerant applications such as underwater ecological sensor networks between energy-limited nodes. Our protocol differs significantly from ALOHA, multiple access with collision avoidance (MACA), and the media access protocol for wireless LANs (MACAW) in that energy is the main performance metric in our case rather than bandwidth utilization. We show that under a realistic underwater sensor network scenario, our MAC protocol wastes only 4% of the transmit energy and only 1.5% of the receive energy due to collisions, when the average number of neighbors is four, and the duty cycle is 0.004. This distributed, scalable MAC protocol has the potential to serve as a primer for the development of energy-efficient MAC protocols for future underwater sensor networks.

**Index Terms**—Energy, media access control (MAC), protocol, sensor, underwater acoustic, wireless.

## I. INTRODUCTION

THE development of media access control (MAC) protocols for underwater wireless acoustic networks (UWANs) is challenging due to energy limitations, long propagation delays, low data rates, and the difficulty of synchronization in underwater environments [1]–[5]. Our aim in this paper is to develop an energy-efficient MAC protocol that will operate under large propagation delays in underwater ecological sensing and monitoring applications. We concentrate on a dense network of hundreds of sensor nodes with small node spacing, e.g., up to 100 m. Such short distances extend battery life by using low-power transmissions [6]–[8]. In addition, we exploit the idea of “sleep mode” [6], [9] to save the nodes’ energy via relatively low duty cycles. In this paper, we focus on delay-tolerant data such as conductivity–temperature–depth (CTD) measurements, which are the fundamental measurements for oceanography, limnology, and marine science applications [10].

The recent design of energy-efficient MAC protocols concentrated on terrestrial sensor networks [9], [11]–[15] and the techniques that have been developed are not suitable for the challenging underwater acoustic communication medium [16] that

experiences a very large propagation delay of 1 s over 1.5 km. In past acoustic network deployments, frequency-division multiple access (FDMA) was used, e.g., in the 1998–1999 SeaWeb [17] and in [18], but was found to be restrictive and inefficient in terms of bandwidth utilization. SeaWeb 2000 [4] favored a carrier sense multiple access/collision avoidance (CSMA/CA) solution with ready-to-send/clear-to-send (RTS/CTS) exchange; however, the problem with the application of handshaking protocols [19]–[23] to underwater sensor networks is the energy consumption overhead caused by the RTS/CTS packets.

A recent underwater networking solution [24] proposed by Xie and Gibson, seeks to achieve predictable end-to-end delays in the network by use of a base station that computes routes for all underwater sensors. This obviates the RTS/CTS exchange and significantly reduces both the propagation delay and jitter along each route; however, there are issues regarding the scalability of this centralized solution to a large number of nodes. The topology establishment phase uses a clever design for the distribution of code-division multiple access (CDMA) pseudonoise sequences to a tree of nodes in the network; however, the near–far problem [25]–[27] that would arise from the use of CDMA codes in such a network has not been addressed.

In the recent literature for terrestrial sensor networks, power-efficient delay-aware medium access control protocol for sensor networks (PEDAMACS) [12] is an energy-efficient MAC protocol in which the nodes are synchronized by a common base station. PEDAMACS uses time-division multiple access (TDMA) for access, and due to its centralized structure, achieves high utilization of the bandwidth as well as energy efficiency. Self-configuring protocols that use a distributed TDMA scheme were proposed in [13]. The sensor-MAC (S-MAC) protocol [11] improves upon the existing 802.11 distributed coordinated function (DCF). Recent papers [9], [14] improve upon S-MAC by proposing fast path schemes that leverage upper-level topology information to allow fast routing, and bypass the delay associated with the length of the sleep–listen cycle. The problem with the application of all of these terrestrial protocols to underwater acoustic networks is their assumption that the propagation delay between the nodes is very small. Further, the recent improvements [9], [14] to S-MAC use a scheme that converges to a global schedule for the entire network, which would be untenable for an underwater acoustic network. In contrast, our protocol does not require global schedules.

The rest of this paper<sup>1</sup> is organized as follows: In Section II, we describe our energy-efficient MAC protocol. In Section III,

<sup>1</sup>A part of the results of this paper was presented by the authors in [28] at the OCEANS 2005 Conference. The main contributions of this paper over the conference paper are the detailed new simulation results and the delay analysis, in Section IV.

Manuscript received March 30, 2006; revised December 28, 2006; accepted March 9, 2007.

**Associate Editor: J. Preisig.**

M. K. Park was with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560 USA. She is now with the Samsung Advanced Institute of Technology, Yongin-si 449-712, Korea (e-mail: mkpark@gmail.com).

V. Rodoplu is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560 USA (e-mail: vrodoplu@ece.ucsb.edu).

Digital Object Identifier 10.1109/JOE.2007.899277

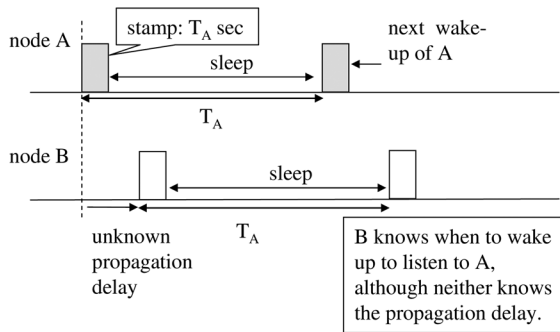


Fig. 1. Basic idea of the UWAN-MAC protocol.

we address the channel variations that may occur due to mobility or platform motion and propose enhancements to the basic protocol. In Section IV, we evaluate the performance of our MAC protocol in terms of the fraction of energy wasted, the total energy consumption, and the delay. In addition, we exhibit network self-reconfiguration upon the deployment of new nodes into an existing network. In Section V, we discuss the implications of this work for the design of future underwater acoustic sensor networks.

## II. PROPOSED MAC PROTOCOL FOR UWANS

### A. Basic Idea

The key assumption in the design of the UWAN-MAC protocol is that the power consumption during the sleep mode is less than that during the idle listening mode, at each node. Hence, there is an incentive to put the nodes to sleep, as in [6] and [9], to conserve energy. Further, we only address networks of stationary nodes in this paper. Mobility would induce further variations in the propagation delay, and we plan to address this in our future work.

The basic idea of the protocol is illustrated in Fig. 1, which explains how to achieve a locally synchronized schedule even in the presence of long, unknown propagation delays.

1) *Determination of “Listen” Cycles:* In Fig. 1, node *A* broadcasts a SYNC packet (the shaded rectangle in the figure) at the beginning of its cycle period, and then goes to sleep by turning off its transceiver circuits to save energy. This SYNC packet announces node *A*’s transmission cycle period “ $T_A$ .” Assume that node *B* is located near node *A*. When node *B* joins the network, it first listens to the channel for this SYNC packet to achieve frame synchronization with node *A*. (The white rectangles in the figure indicate node *B*’s receptions of node *A*’s SYNC packets.) After achieving frame synchronization with node *A*, node *B* decodes the length of the transmission cycle period  $T_A$  from the message. This *explicit stamping of the transmission cycle period*, rather than any absolute wake-up time, allows node *B* to wake up at exactly the correct time in the next cycle to listen to node *A* without any knowledge of the value of the propagation delay, as long as the propagation delay remains fixed from one cycle to the next and the clock drift is not significant in one cycle. This localized protocol holds for any pair of nodes (*A*, *B*). Further, this scheduling algorithm does not require any adjustments to the nodes’ clocks since absolute timing information is obviated. (The use of a relative

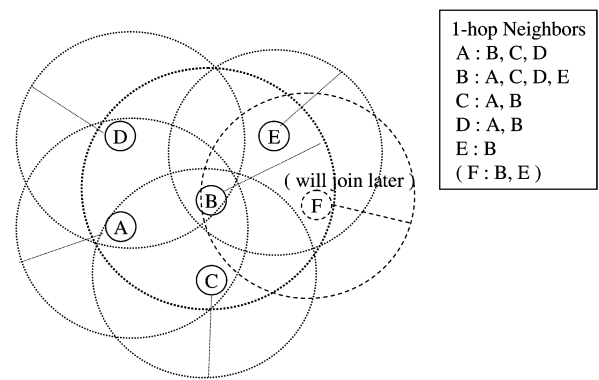


Fig. 2. Example network configuration.

time stamp is the same as in S-MAC [9]; however, as we will explain shortly, the determination of the transmit start times is different.)

In this protocol, a collision may occur in the following two ways. First, a “transmit–receive collision” may occur in which a node is transmitting while other nodes’ packets arrive at its receiver and collide with the node’s own transmission. Since the transmit power is usually much larger than the power required for successful packet reception, the received packet would not be decoded properly at the node in this case. Second, a “receive–receive collision” may occur if more than two packets arrive at a node and overlap in duration, in which case the node cannot decode either of the packets.

2) *Determination of Transmit Start Times:* The topology control layer keeps track of the neighbors of a node for whose transmissions a node needs to wake up at the proper time. Even though the listen times are predetermined using the previously described scheme, the initial transmission time is selected at random and independently by each node. However, once a node chooses a certain transmission start time, it sticks to its schedule by transmitting its data at that time again in the next cycle. Since the nodes have clocks that are at random offsets from each other, if the cycle period is much longer than the transmit duration, then the probability of collisions will be small. (This will be quantitatively demonstrated in Section IV.)

### B. Initialization of the MAC Protocol

We use the network in Fig. 2 as an example network configuration throughout this paper. In this figure, each node sets its transmission range by using localized topology control [6], and each circle indicates the transmission range of the node at its center.<sup>2</sup> For this given network topology, the broadcasting of control packets in the initialization period is illustrated in Fig. 3. In this figure, every node’s time axis is shown separately. A node’s own transmission is shown as shaded, and the receptions of that transmission at the other nodes are shown as white rectangles with a label on top of the white rectangle that indicates from which node that packet was received. In the initialization period, for the distribution of the transmission schedules among the nodes, each node *i* broadcasts its SYNC packet, and

<sup>2</sup>The transmission ranges are circular in this diagram for illustration purposes to indicate neighbor relationships. In reality, these will not be circular, and the proposed MAC protocol does not depend on the circularity of these ranges.

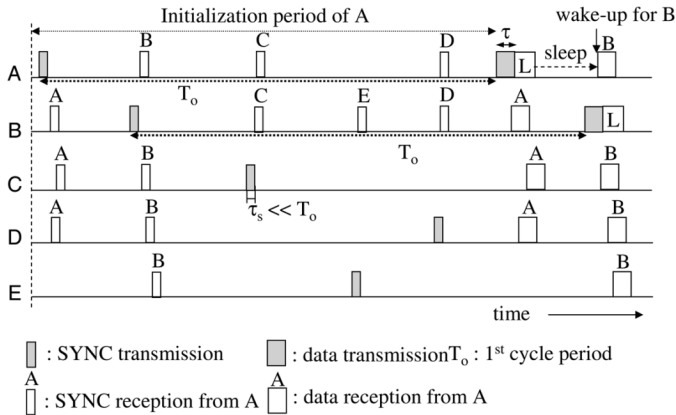


Fig. 3. Initialization phase for the nodes in Fig. 2 carried out at the initial network deployment.

remains awake until the beginning of the next cycle to receive its neighbors' SYNC packets. The SYNC packet contains the node's transmission cycle period  $T_i$  which tells its neighbors that it will transmit data again after this time period. The cycle period  $T_i$  is initially fixed to the same value  $T_o$  for all nodes in order for them to initialize their transmission/listen schedules.

In Fig. 3, every node selects its transmission start time at random in the interval  $[0, T_o)$  and broadcasts its SYNC packet to its neighbors. Because node A's start time happens to be the first, we see in this figure that node A broadcasts its SYNC packet first. Then, node A's neighbors, namely, nodes B, C, and D, receive node A's SYNC packet during initialization which allows them to schedule their wake-up times, which will take place *after the initialization period*. Node A will also schedule particular wake-up times as shown in this figure for its neighbor nodes B, C, and D after it hears their schedules during this initialization period. Note that the "duty cycle," namely, the ratio of the transmit duration  $\tau$  to the cycle period  $T_o$ , is in fact very low, (e.g.,  $\tau/T_o = 0.004$ ), and the packets in this figure have not been drawn to scale.

Each node maintains its neighbor table in the topology control layer. In our protocol, each node runs its initialization periodically (e.g., every ten cycles).

C. Data Transmission After Initialization

After the initialization phase is complete, each node knows when it needs to wake up again to receive data from its neighbors. After this initialization, nodes follow their established schedules and begin sending data. Fig. 4 shows the structure of the data transmission packet and the listen period to listen to any potential newcomers during the data transmission phase. The transmit duration has been shaded and is followed by a listen duration. The transmit duration has three distinct parts: "missing," "SYNC," and "data Tx." The "data Tx" corresponds to the part where actual data is being sent. We now explain the first two control functions: In our protocol, each node  $i$  compares the neighbor list with the list of nodes from which node  $i$  has successfully received signals. After this comparison, node  $i$  generates the "missing node list" and sends the list of "missing" neighbor nodes in the header of the data packet in its next cycle.

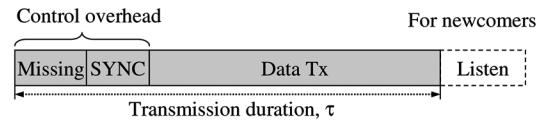


Fig. 4. Transmission packet structure and listen duration for newly joining nodes in the data transmission phase.

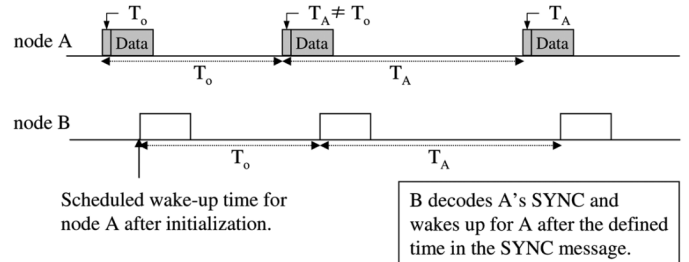


Fig. 5. Modification of the cycle period during the data transmission phase.

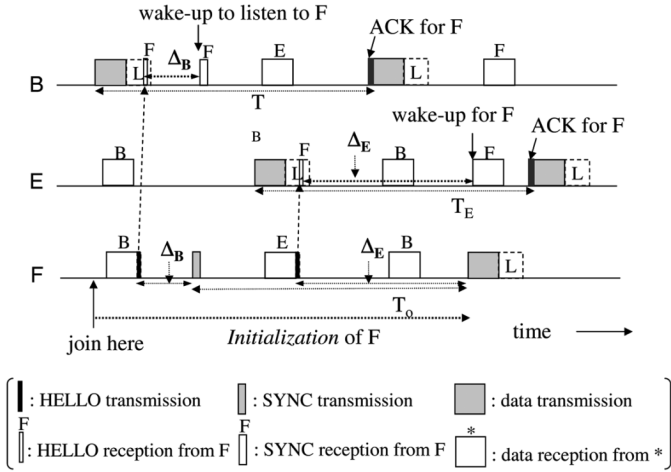
During regular operation, every node keeps sending in its SYNC header a cycle period stamp, which may be different from the one ( $T_o$  seconds) in the initialization period. By using the SYNC message in the header, a node has the option to change its current cycle period, and its neighbors can decode the modified SYNC message and change their wake-up times for that node. Fig. 5 shows this modification of a cycle period during the data transmission phase. In this figure, node A changes its next cycle period from  $T_o$  to  $T_A$ , and node B decodes the modified SYNC message and wakes up to listen to node A after the new cycle period  $T_A$ . If a node B loses contact with node A during this modification, it will recover node A as a neighbor via the missing neighbor list. This recovery procedure will be detailed in Section II-E.

We now explain the listen duration shown in Fig. 4. After the transmission of data, a node does not go to sleep right away but rather goes into an idle listening mode. In the listening mode, the node is still awake but operates at low power. If it hears something, it will go into the receive mode. This additional listen duration is used to hear newcomers and improves robustness. A scenario in which a new node joins will be discussed in Section II-D in detail. The length of the listen duration is chosen as a practical parameter which involves a tradeoff: A very long duration decreases the energy efficiency of the protocol due to idle period energy consumption, but a very short duration might not be enough to catch some of the newcomers' messages.

D. Handling Newcomers

We use the topology shown in Fig. 2 to show how the protocol handles newcomers. In this figure, node F joins the network, and nodes B and E become its neighbors. Fig. 6 shows the packet transmissions that occur in this scenario. After node F joins the network, whenever it hears from another node, it sends a HELLO packet back to that neighbor to inform the neighbor of its transmission schedule.

For the transmission time of this HELLO packet, the node selects its transmission time slot uniformly among  $M$  time slots, where  $M$  is the number time slots and a slot lasts for the duration of a HELLO packet. (In the simulations, we use  $M =$


 Fig. 6. New node  $F$  joins the network.

10; however, the choice of the parameter  $M$  would be system dependent.) This random selection of a HELLO transmission among  $M$  multiple time slots avoids the possible collisions from the HELLO packets from the other newcomers, in case an existing node has multiple newcomer neighbors who simultaneously enter the network. Hence, the duration of the listen period is chosen as  $M$  times the HELLO packet duration plus twice the maximum internodal propagation delay.

In Fig. 6, since node  $F$  happens to receive data from node  $B$  first, it selects its HELLO packet transmission time among these  $M$  times at random when the data reception is complete, and transmits a HELLO packet back to node  $B$ . (In the figure, the selection of the transmission time of the HELLO packet is omitted, not to make the figure too complex.) This HELLO packet from  $F$  contains a time stamp  $\Delta_B$ , which is defined as the number of seconds (according to  $F$ 's clock) from the beginning of the HELLO packet to the beginning of the scheduled SYNC packet of node  $F$ . By the same argument as in the basic idea of the protocol (in Section II-A), this mechanism allows node  $B$  to wake up to listen to node  $F$  exactly  $\Delta_B$  seconds after it receives node  $F$ 's HELLO packet. After this, in the figure, at its selected transmission start time, node  $F$  broadcasts its SYNC packet to achieve initialization with its neighbors  $B$  and  $E$ . When node  $B$  decodes this SYNC packet of node  $F$ , it schedules its wake-up time to receive future data transmissions from node  $F$ . To acknowledge the successful reception of this HELLO packet, node  $B$  sends back an acknowledgment (ACK) to node  $F$  at  $B$ 's next data transmission time, and node  $F$  ensures its synchronization with node  $B$  by keeping sending its HELLO packet until it hears the ACK from node  $B$ . A similar procedure is repeated for neighbor node  $E$  in the figure. At the end, node  $F$ 's data transmission begins at its scheduled time, namely,  $T_o$  seconds after its SYNC packet.

### E. Handling Node and Synchronization Failures

A node may not receive data at a scheduled wake-up time due to a bad channel condition from the sender, or due to a sender node failure. Whenever the receiver node does not hear from the sender at a scheduled time, the receiver puts the sender in its missing node list, as explained in Section II-C. For instance,

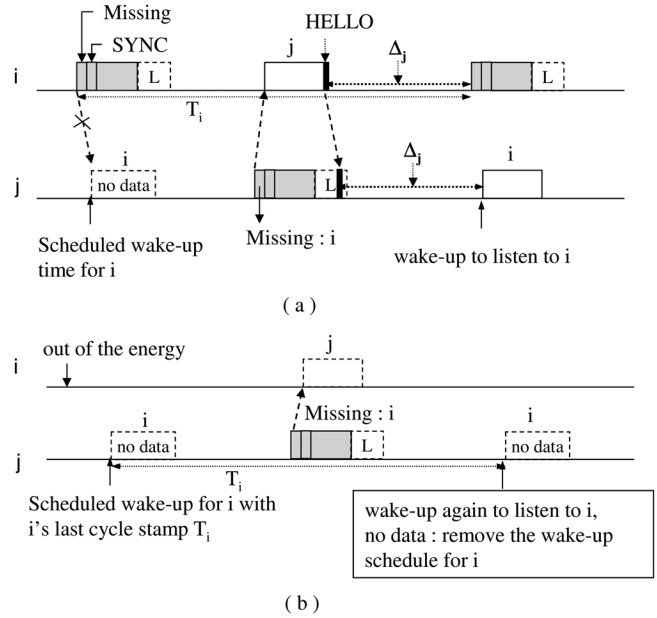


Fig. 7. (a) Handling loss of synchronization. (b) Handling a node failure.

in Fig. 7(a), assume that a data delivery from node  $i$  to node  $j$  has failed in cycle  $t_m$  of node  $i$ . Then, node  $j$  will announce via its missing header that it has not heard from node  $i$ . In case the delivery failed due to a bad link state from node  $i$  to node  $j$ , node  $i$  will decode the missing header from node  $j$ , and discover its own delivery failure or possible loss of synchronization with node  $j$ . Then, node  $i$  sends a HELLO message to node  $j$ , as if node  $i$  were a new neighbor of node  $j$ . Then, since node  $j$  is in listen mode after its data transmission, it will hear this HELLO message from node  $i$ , and will reschedule its wake-up time for the next cycle.

In Fig. 7(b), when node  $i$  fails (e.g., it runs out of energy), node  $j$  will announce  $i$  in its missing list. However, since node  $i$  is no longer available in the network, node  $j$  will not be able to hear back from node  $i$  in cycle  $t_{m+1}$ . Then, node  $j$  will remove node  $i$ 's wake-up time from its table of wake-up times, starting at cycle  $t_{m+2}$ . In other words, in this figure, if a node does not hear from a neighbor for two consecutive cycles of the neighbor, it deletes its wake-up schedule for the neighbor to save energy. In general, node  $j$  can count the number of cycles for which no transmissions from node  $i$  have been heard, and this number of cycles may be set as a configuration parameter to trade off energy consumption versus protocol robustness.

### III. ADDRESSING CHANNEL VARIATIONS

So far, we have assumed that the propagation delay remains the same between cycles, but the propagation delay in reality varies due to channel fluctuations caused by the relative motion of the transmitter, receiver, or significant scattering surfaces, and platform motion. In this section, we model propagation delays in the network as random variables. However, arbitrarily large, random propagation delays are difficult to address in design, and do not occur in practice. Hence, we will also assume that the nodes also know the value or the order of magnitude of the maximum propagation delay between the nodes in the network.

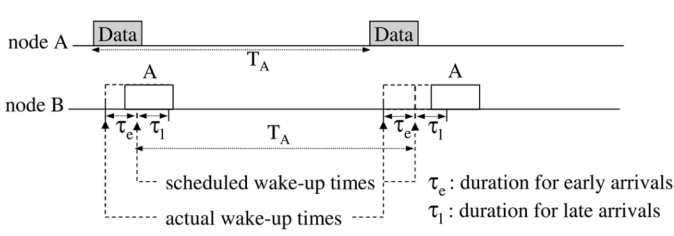


Fig. 8. Listen durations placed before and after the scheduled wake-up time.

(This was assumed so far only in the practical choice of the listen duration in the basic protocol.) For example, for densely deployed underwater networks, the maximum propagation delay between the nodes is about 70 ms for distances of up to about 110 m. When we discuss simulation results in Section IV, we will report the results of the protocol both with and without these additional assumptions on the channel model.

We now describe the improvements that we propose that use this more realistic channel model. First, to handle random propagation delays, we let each node place listen durations:  $\tau_e$  for the early arrival and  $\tau_l$  for the late arrival of data from a neighbor. For example, in Fig. 8, we show the listen mode duration of node *B* for any existing neighbor *A*. Node *B* can catch node *A*'s data, even when the propagation delay from node *A* to node *B* does not remain exactly the same as in the previous cycle. In this figure, although node *A*'s data in the first cycle arrives earlier and its data in the second cycle arrives later than the scheduled wake-up time of node *B* for node *A*, node *B* can still receive them.

Second, by exploiting the maximum propagation delay information, we present an improvement of our protocol to reduce receive–receive collisions: Each node *i* places a certain “guard time” on both sides of its transmit duration. This guard time is different from the guard times employed in TDMA systems, which are allocated by the base station depending on the multipath channel conditions. The guard times in this section are chosen by each node in a completely localized manner to reduce the collision rate. For example, Fig. 9(a) shows an example of a receive–receive collision in the presence of propagation delays. To avoid this collision, if  $\tau_1$  or  $\tau_2$  is less than a predefined guard time  $\tau_g$ , then node *A* or *B* reselects its transmission start time to satisfy the condition that its transmit duration  $\tau$  plus  $2\tau_g$  does not overlap with any other wake-up time. Fig. 9(b) suggests one possible solution for the case where  $\tau_2 < \tau_g < \tau_1$ . No collision occurs because node *B*'s newly selected transmission start time has been applied.

We now build an explicit algorithm to carry out this guard time policy. Let node *i* follow its current transmission schedule, which consists of the transmission start time and the cycle period. Then, if the duration between a node *i*'s selected transmission start time (or end time) and its wake-up time is less than the guard time, the node reselects its new data transmission time such that it does not overlap with the existing wake-up time schedules for its neighbor nodes. To state this precisely, let  $X_i$  denote node *i*'s current transmission start time, which can be fixed as 0;  $T_i$ , node *i*'s current cycle period;  $W_{ik}$ , the wake-up time of node *i* for its neighbor node *k*;  $\tau$ , the data transmission duration;  $\tau_g$ , the guard time duration; and  $X'_i$ , node *i*'s new

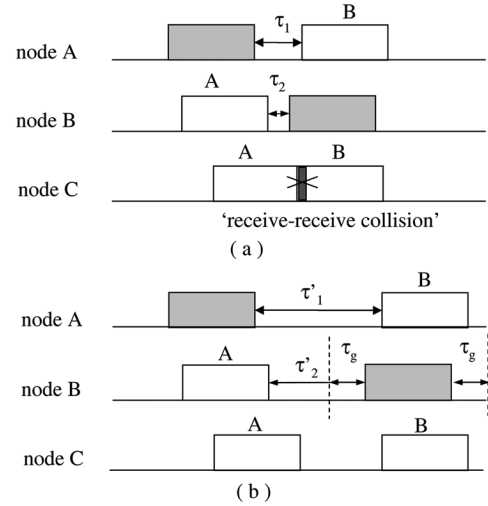


Fig. 9. Scenario of a receive–receive collision with propagation delays and collision avoidance using the guard time policy. (a) Receive–receive collision due to the propagation delays. (b) Guard time solution for the case  $\tau_2 < \tau_g < \tau_1$ .

transmission start time. Each node sets its new schedule to satisfy the following: If  $X_i - W_{ik} < \tau_g$  or  $W_{ik} - (X_i + \tau) < \tau_g$  for any *k*, then select  $X'_i \in (X_i, X_i + T_i)$  at random, subject to the following condition: For every *k*,  $X'_i - W_{ik} \geq \tau_g$  or  $W_{ik} - (X'_i + \tau) \geq \tau_g$ .

The simplest choice of the guard time duration is twice the maximum propagation delay, which can be the same for all nodes. A node announces this new transmission schedule to its neighbors using the SYNC header, and follows the new schedule starting with the following cycle.

#### IV. PERFORMANCE EVALUATION

We distinguish between two types of energy used. The “transmit energy” is the energy spent by the transmitter to broadcast a packet, and the “receive energy” is the energy consumed by a receiver to receive a packet. As performance metrics, we define the fraction of transmit energy wasted due to collisions as the transmit energy wasted due to collisions divided by the total transmit energy used. Similarly, we define the fraction of receive energy wasted as the receive energy wasted due to collisions divided by the total receive energy used.

For the fraction of transmit energy wasted due to collisions, when two packets collide, we assume, rather conservatively, that the two nodes failed in their transmissions, and that they wasted their transmission energies. Even though the same packets may have been successfully delivered to the other neighbors, we still assume that the two nodes wasted their energies. This gives us an upper bound on the fraction of energy wasted. This is the simplest way to define such a metric for the MAC protocol without having to make assumptions about the particular routing protocol used.

For the fraction of receive energy wasted due to collisions, the metric is a more accurate description of the receive energy usage, since the receive energy is indeed associated with only that receiver and no others. Hence, we measure the receive energy consumed by a receiver for the failed deliveries due to collisions. The fraction of receive energy wasted is at most equal to

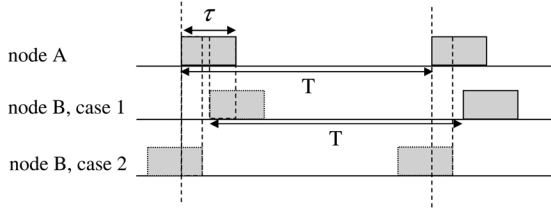


Fig. 10. Two cases in which a packet collides with another packet in a cycle.

the fraction of transmit energy wasted (however, the magnitudes of the two energies can be dramatically different). Note that for the special case in which all the packets have the same duration, this fraction of receive energy wasted due to collisions reduces to the collision rate, which is the total number of packets that collide at receivers divided by the total number of packet transmissions attempted. Hence, tracking the fraction of receive energy wasted is more general and can be specialized if necessary.

In this section, we first examine two cases without any topology control: the case with no propagation delays and the case with long propagation delays, and quantify the impact of propagation delay on the fractions of each type of energy consumption. Second, we evaluate the fractions of both transmit and receive energy wasted due to collisions in a more realistic network scenario with topology control, and measure the energy consumed in each of the operation modes. Finally, we analyze the delay performance.

#### A. No Propagation Delays

The reason we consider this artificial case first is that analytical results can be derived for this case and shed light on the simulation results obtained for the actual case with propagation delays. In this section, we assume that all nodes share the same medium so that all of them can hear from each other. (In Section IV-C, we will remove this assumption under a realistic network scenario.)

In the absence of propagation delays, we first compute the probability that a collision occurs. Each node generates its transmission start time uniformly in the interval  $[0, T)$  in an independent identically distributed (i.i.d.) fashion, and its data transmit duration is  $\tau$ . Fig. 10 shows the two cases in which two nodes' transmissions collide in the interval  $[0, T)$ . Taking node  $A$ 's transmission as a fixed reference in this figure, we see that node  $B$ 's transmission in the first cycle collides either with  $A$ 's transmission in the first cycle shown, or with  $A$ 's transmission in the next cycle. Hence, the probability that node  $A$ 's transmission collides with the node  $B$ 's is  $2\tau/T$ , since the transmission start times are uniformly distributed. Thus, the probability that a node's transmission collides with at least another node's transmission over 1 cycle is as follows:

$$\begin{aligned}
 &P[\text{A node collides with at least one other node}] \\
 &= 1 - P[\text{A node does not collide with any other node}] \\
 &= 1 - \left[ \left( 1 - \frac{2\tau}{T} \right)^{N-1} \right] \quad (1)
 \end{aligned}$$

where  $N$  is the number of nodes in this network. Assuming that each node uses the same energy for transmission and that each

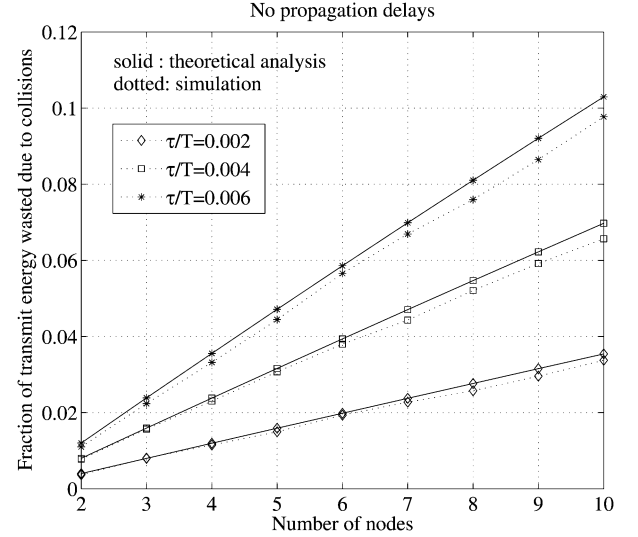


Fig. 11. Fraction of transmit energy wasted due to collisions versus the number of nodes, without propagation delay.

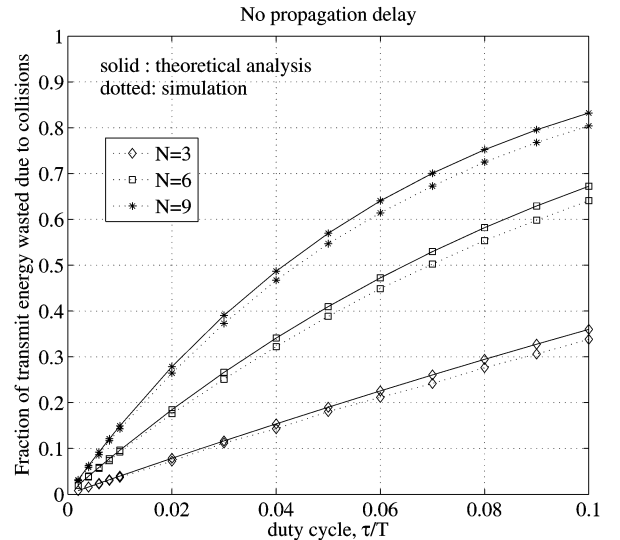


Fig. 12. Fraction of transmit energy wasted due to collisions versus the duty cycle  $\tau/T$ , without propagation delay.

transmission is independent of other transmissions, the expected value of the total transmit energy wasted due to collisions is  $N(1 - [(1 - 2\tau/T)^{N-1}]E)$ , where  $E$  is the value of the energy for each transmission, and the total value of the transmit energy used for all nodes is  $NE$ . Thus, the fraction of transmit energy wasted due to collisions is  $(1 - [(1 - 2\tau/T)^{N-1}])$ .

The simulation results displayed in Figs. 11 and 12 under the scenario with no propagation delay are close to the analytical result above. The slight difference between the analytical and simulation results is due to the discretization of the time axis in the simulations. In this simulation, every node is a neighbor of every other node in the network, and the result is averaged over 10 000 runs. For the data transmission duration,  $\tau = 200$  ms is used for all simulations throughout this section. In Fig. 11, as the number of nodes increases from two to ten nodes, the fraction of transmit energy wasted due to collisions is shown for different

values of the duty cycle  $\tau/T$ . In this regime of network operation with a low duty cycle, writing down the Taylor expansion of (1) for low duty cycles shows that the fraction of transmit energy wasted is a linear function of the number of nodes, as can also be seen in this plot. For six nodes, which is a typical number of neighbors for a node if topology control were applied, the figure shows that 96% of the transmit energy is spent in successful transmission for a duty cycle  $\tau/T = 0.004$ . In Fig. 12, as the duty cycle  $\tau/T$  increases from 0.002 to 0.1, the fraction of energy wasted due to collisions is shown for different numbers of nodes. Hence, this graph indicates the limits of operation of this MAC protocol, as a function of the duty cycle. For example, for  $N = 9$  nodes, if the target fraction of energy wasted due to collisions is 15%, a duty cycle of about 1% is required.

### B. Long Propagation Delays

Now, we show the impact of long propagation delays on the fraction of energy wasted due to collisions in our protocol. In the simulations of this section, propagation delays between any two nodes are independent and uniformly distributed between 5 and 75 ms, corresponding to internodal distances up to about 110 m, since the inverse speed of sound in water is 0.67 ms/m. In this subsection, we use our basic protocol without any enhancements such as listen durations and guard times, which were described in Section III.

The results are displayed in Figs. 13 and 14. The conclusion of these two graphs is twofold: First, when we compare the fraction of transmit energy wasted due to collisions with that in the case without propagation delays, the fraction is larger in the case with propagation delays. Since there was no delay between any two nodes in Section IV-A, the transmissions and receptions could be represented on the same global time axis, and both the transmit–receive collisions and the receive–receive collisions could occur, but both the transmit–receive and the receive–receive collisions occurred at exactly the same time. In contrast, under our protocol, a different kind of receive–receive collision that is not coincident with a transmit–receive collision is added in the presence of propagation delays. An example of this collision is illustrated in Fig. 9(a). In this example, due to the different propagation delays on the links, two packets transmitted from node  $A$  and  $B$  collide at node  $C$  even though neither of the two transmissions collides at node  $A$  or  $B$ . Thus, propagation delays worsen the transmit energy efficiency of the protocol over the case without propagation delays. There is as much as 20% loss in transmit energy efficiency due to propagation delays, for the numbers of nodes and the duty cycles considered.

Second, we see that the fractions of receive energy wasted with propagation delays and the one without propagation delays are roughly the same. The reason is that although there is propagation delay in this section, each transmission arrives independently and consumes the same duration on the time axis from the viewpoint of receivers. Hence, at each receiver, the received packets are simply the same packets as in the case of no propagation delays, but shifted from their arrival times by the amount of the random propagation delay.

As stated earlier, the guard time policy is not used in this section, and we use only the basic protocol without any enhancements. If the guard time policy were applied, all of the

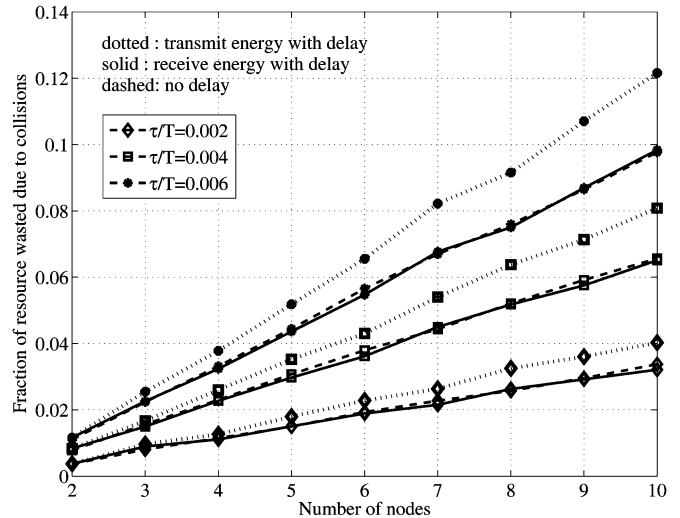


Fig. 13. Fraction of energy wasted due to collisions versus the number of nodes in the presence of propagation delays.

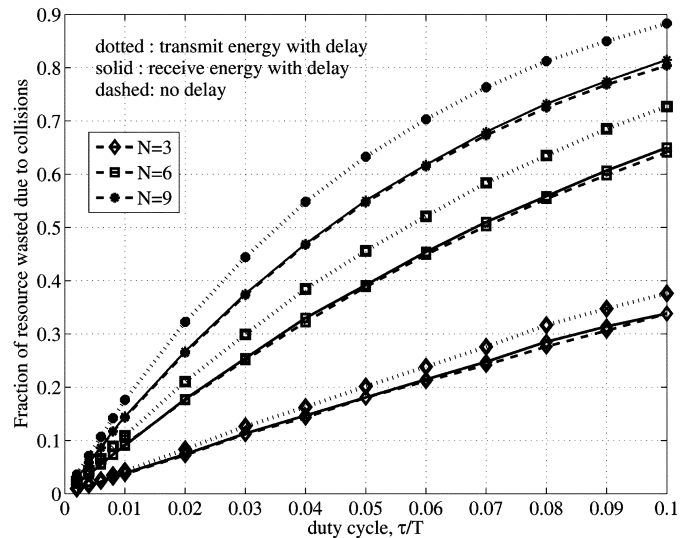


Fig. 14. Fraction of energy wasted due to collisions versus the duty cycle  $\tau/T$  with propagation delays.

receive–receive collisions would be avoided, since in this section, all of the nodes share the same medium using low duty cycles. We will apply the protocol enhancements of Section III in Section IV-C.

### C. Realistic Network Scenario

So far, in the simulations, we have assumed that all of the nodes are neighbors of each other and that every node could reach another. However, in a large scale network, topology control limits the transmission range of each node as in Fig. 2 to improve performance. In this section, we assume that each node determines its neighbors using localized neighbor search, and that the values of the channel gains between the nodes that are neighbors are available to those nodes via measurement.

In the simulation setup, 15–30 nodes are randomly deployed over a  $500 \times 500\text{-m}^2$  area, which gives internodal distances of about 100 m for 25 nodes. To compute the channel gains, we fix the reference received power. Then, the channel gain is

equivalent to the required minimum transmission power. We use the energy consumption model from [5] in which the transmit energy consumption model is given as

$$E = P_0 T_p A(r) = P_0 T_p r^k a^r \quad (2)$$

where  $E$  denotes the transmit energy,  $P_0$  is the reference received power required to decode the received packet,  $T_p$  is the packet duration,  $r$  is the distance from the source, and  $k$  is the spreading factor which is 1 for cylindrical, 1.5 for practical, and 2 for spherical spreading. In (2),  $a = 10^{\alpha(f)}$  is the frequency-dependent term obtained from the absorption coefficient  $\alpha(f)$ . By Thorp's expression [29] in decibels per kilometer

$$\alpha(f) = \frac{0.11f^2}{1+f^2} + \frac{44f^2}{4100+f^2} + 2.75 \times 10^{-4} f^2 + 0.03 \quad (3)$$

where  $f$  is in kilohertz. For our simulations, we chose the parameters as  $f = 25$  kHz and  $k = 1.5$ . There are four modes of operation at each node: transmit, receive, idle listening, and sleep. In this section, we will use the power consumption specifications of the Woods Hole Oceanographic Institution (WHOI, Woods Hole, MA) micromodem [30]. The power consumption for the transmit, receive, and idle listen modes is 10 W, 3 W, and 80 mW, respectively, for this modem. Since we turn off the entire modem during the sleep mode in our protocol, we take the sleep power to be 0 mW.

Each node maintains its neighbor table, which is the list of neighbors within its transmission range. This list is used to schedule the receiver node to wake up to listen to its neighbors. Assume that the maximum transmission power is the same for all nodes and that the generation of propagation delay is the same as in Section IV-B. We set a maximum transmission power to enable a node to have 4–6 neighbors on average in the simulations. Each node generates its transmission start time uniformly in the duration  $[0, T)$  in an i.i.d. fashion. For the control packet durations, we set  $\tau_s = 0.1\tau$  for the SYNC packets and  $\tau_h = 0.1\tau$  for the HELLO packets, where  $\tau$  is the data transmission duration. The average duty cycle  $\tau/T$  is 0.004. Each transmit energy for different types of packets is proportional to the corresponding packet length.

To test the robustness of our protocol, we change the network topology by deploying five new nodes into the existing network after cycle 3 and show how our protocol adapts to this change. In this simulation, a network operates with  $N_o$  nodes (in Fig. 15,  $N_o \in \{15, 20, 25\}$ ) in the beginning. From the simulations, we obtained that the average number of neighbors is about three for  $N = 15$ , four for  $N = 20$ , five for  $N = 25$ , and six for  $N = 30$ .

Fig. 15 shows both the fractions of transmit energy and of receive energy wasted due to collisions per cycle. In this figure, every node is in its initialization phase in cycle 1. In this initialization, each node broadcasts only SYNC packets to its neighbor nodes within its transmission range. We see that the fractions of energy wasted are not significant because the duration of control packets is relatively short, compared with that of the data packets. Due to the longer packet durations for data transmissions, the fractions of both transmit and receive energy wasted become much larger in cycle 2. However, the fraction of receive

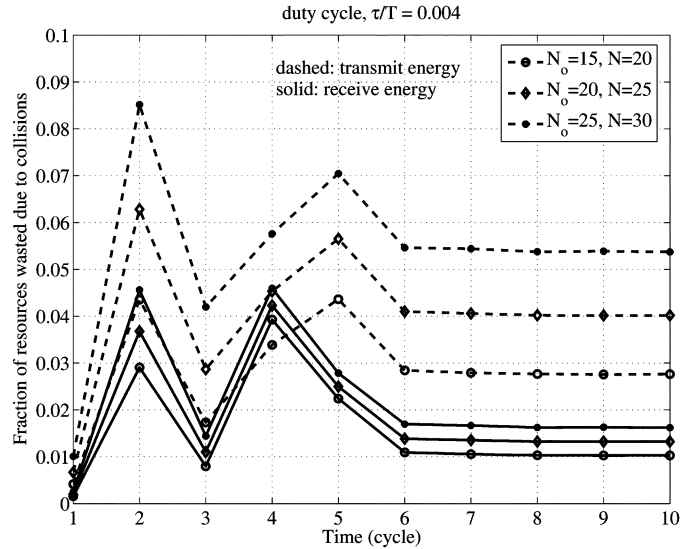


Fig. 15. Fraction of resource wasted due to collisions: Five new nodes are added at cycle 4, where  $N_o$  is the initial number of nodes deployed, and  $N$  is the number of nodes after this addition.

energy wasted is much lower than that of the transmit energy wasted.

As an enhancement of our protocol, nodes refine the transmission/listen schedules using a guard time, as explained in Section III. A node modifies its transmission schedule if the time difference between its own transmission and its receptions from the neighbors is less than the guard time. Now, in cycle 3 of Fig. 15, the fractions of transmit and receive energy wasted due to collisions are significantly reduced by the guard time policy. In the simulations, we have used twice the maximum propagation delay as the guard time duration. We see that, by using guard times, we can achieve roughly a 50% performance improvement in the fraction of transmit energy wasted due to collisions, and roughly a 60% improvement in the fraction of receive energy wasted due to collisions. The guard time durations are not able to avoid all of the collisions because there remain collisions that are caused by the hidden terminal problem. For example, assume that nodes  $A$  and  $B$  share the same medium and nodes  $B$  and  $C$  share another medium, but that nodes  $A$  and  $C$  are not neighbors of each other. In this case, if nodes  $A$  and  $C$  transmit at the same time, then the two transmissions collide at node  $B$ . Since nodes  $A$  and  $C$  do not have any information on the transmission schedules of each other, they cannot correct their transmission schedules by using their guard time policies. Hence, even though the guard time durations can reduce some receive–receive collisions [as shown in Fig. 9(a)], there are still collisions due to this hidden terminal problem in our protocol.

If we compute the average of the fraction of wasted resources over ten cycles without adding any new nodes, then our MAC protocol wastes about 4% of the transmit energy due to collisions, and only about 1.5% of the receive energy due to collisions, when the average number of neighbors is four. (These average values are obtained from the simulation results and are not shown in Fig. 15.)

In cycle 4, we add five new nodes to the network. These new nodes send the HELLO packets whenever they hear from the

existing nodes, and they also send their SYNC packets for initializing the transmission/wake-up schedules. Regardless of this change, the existing nodes keep sending their DATA packets at the scheduled transmission times. However, because of this addition of the newcomers' control packets (that is, HELLO and SYNC packets), more collisions occur in cycle 4. Therefore, the fractions of energies wasted due to collisions increase. Note that the slope of the fraction of transmit energy wasted is more gradual than that of the fraction of receive energy wasted. The reason is that the newly generated collisions in this cycle are mainly collisions between the existing DATA packets and the control packets which have short durations, and this type of collision has much more impact on the fraction of receive energy wasted than on that of transmit energy wasted. For example, let us assume that there are two sets of nodes  $\{i, j, k, 1\}$  and  $\{j, k, 1, 2\}$ , where nodes  $i, j$ , and  $k$  are the existing nodes and nodes 1 and 2 are the newcomers, and each of these sets of nodes shares a common medium. In addition, assume that newcomer 2 sends its SYNC packet which overlaps with node  $i$ 's DATA packet transmission, and as a result, there are collisions between DATA and SYNC packets in the common receiver nodes  $j, k$ , and 1. Then, in this case, the transmit energy spent for one DATA packet and for one SYNC packet has been wasted, and the receive energy for three DATA packets and three SYNC packets has been wasted. Thus, the fraction of receive energy wasted due to collisions is more significantly increased but is still lower than the fraction of transmit energy wasted due to collisions. Here, note that even though this SYNC packet is not successfully delivered to the other nodes, the newcomer still sends its HELLO packet whenever it hears from the other nodes. Thus, both the existing nodes and the newcomers can successfully maintain the transmission/reception schedules.

In cycle 5, the new nodes start sending their own DATA, so that all the nodes transmit DATA packets which are ten times longer than the control packets. Because of longer packet durations, there are more chances for any newly generated DATA packet of a newcomer to collide with the DATA packets of the existing nodes. Thus, the fraction of transmit energy wasted increases. However, in Fig. 15, the fraction of receive energy wasted due to collisions decreases in cycle 5. The reason is that the total receive energy increases faster than the receive energy wasted due to collisions does. Even though there is a failed DATA reception from a newcomer at a receiver, there may still be successful deliveries in other receivers, which have not been taken into account in the case of the fraction of transmit energy wasted.

Beginning in cycle 6, all the nodes apply their guard times with the newly discovered schedules for the newcomers, and this reduces the fraction of the energy wasted due to collisions in cycle 6. From cycle 6 onwards, all nodes have refined their transmission schedules with guard times, and our protocol eventually gives the similar or slightly better performance over the initial network for  $N = 20, 25, 30$  nodes (which is shown in cycle 3 in the same graph). Hence, the protocol is self-configuring in the sense that it converges to a high energy efficiency even after the addition of new nodes.

In addition to the fraction of energy wasted due to collisions, we evaluate the performance of our protocol in terms of the energy consumption. In the simulations, we normalize each component with the transmit power consumption, so that the

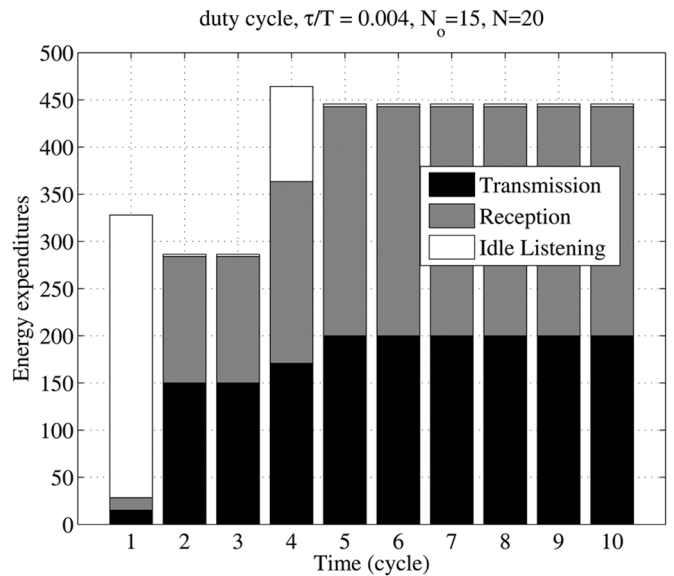


Fig. 16. Total energy consumption over ten cycles.

TABLE I  
PERCENTAGE ENERGY CONSUMPTION OF DIFFERENT  
OPERATION MODES OVER TEN CYCLES

Cycle \ Category	Transmission	Reception	Idle Listening
Cycle 1	4.6	4.1	91.3
Cycle 2~3	52.4	46.8	0.8
Cycle 4	36.8	41.5	21.7
Cycle 5 ~ 10	44.9	54.5	0.7

ratio of the power consumption in the different modes becomes transmit : receive : idle : sleep = 1 : 0.3 : 0.008 : 0. The resulting Fig. 16 shows the total energy expenditure over ten cycles, and Table I summarizes the percentages of the energy consumption over different modes. In cycle 1, since all the nodes send only the SYNC packets and are awake to initialize the transmission schedules, the majority of the energy expenditure is in the idle listen mode. In cycles 2 and 3, the nodes send and receive DATA packets but are in the idle listening mode for a short amount of time. The total amount of the listen duration in these cycles is twice the maximum propagation delay in addition to  $M\tau_h$ , where  $M = 10$  in the simulations. Here, note that we do not count the short listen durations for the early and late arrivals of all receptions because we assume that there is no clock drift in one cycle in the simulations. Now, in cycle 4, the energy consumption for the idle listen mode increases because the five new nodes perform their initializations. In this cycle, the energy consumption for the transmission and reception modes also increases, since the existing nodes keep sending their DATA packets as the newcomers transmit their control packets. From cycle 5 onwards, both the transmission and reception energies increase, but the portion of the energy consumption for the idle listening mode decreases again. Because there is no retransmission policy in our protocol when collisions occur, the energy consumptions stay the same from cycle 2 to 3, and also from cycle 5 onwards. Hence, the plots of the magnitude of energy consumption and the fraction of energy wasted, in each mode, are to be used jointly to understand the performance of the protocol.

#### D. Analysis of Delay Performance

In this section, we analyze the delay performance of the protocol. In general, the delay performance depends critically on the routing protocol as well as the MAC protocol. Many routing protocols will be able to run on top of this MAC protocol, and will contribute different delays. To separate the effect of the delay that results from the MAC layer itself, in this section, we do not model the delay induced by traffic congestion. This is a reasonable assumption when the nodes generate only tiny amounts of data, and at the end of this section, we give a rough estimate of the network size for which this delay analysis remains valid.

First, we analyze the delay for a linear arrangement of  $L$  nodes, where the source node  $k_0$  is assumed to be the left-most node in this arrangement, and the destination node  $k_{L-1}$  is the right-most node. We assume that the basic protocol, without any enhancement via guard times, is used. We now derive the expected value of the delay that a packet generated at  $k_0$  experiences until it reaches its destination  $k_{L-1}$ .

We assume that every packet at every node is generated at a time that is uniformly distributed on the interval  $[0, T)$ . We also assume that a node sends out the generated packet, at the first opportunity possible. (The assumption of no network congestion is invoked here.) Let  $G$  denote the generation time of the packet, and let  $X_0$  denote the transmit start time of the first transmission of node  $k_0$  that occurs after  $G$ . Now, both  $X_0$  and  $G$  are uniformly distributed on  $[0, T)$ . When we condition on the packet generation time  $G$ , the distribution of  $X_0$  is uniformly distributed on the interval  $[G, G + T)$  (since the uniform distribution of the transmit start times holds true for the subsequent interval as well); hence, the expected value of the delay from the generation time to the first transmit start time of  $k_0$  that occurs after the generation time is  $T/2$ .

Now, for each intermediate node, we calculate the average delay that the packet experiences to get from node  $k_i$  to the next node in the arrangement  $k_{i+1}$ . Since the transmit start times of all of the nodes are uniformly distributed on  $[0, T)$  and repeated at fixed intervals of  $T$  at each interval afterwards, the same argument that was used for the generation time in the previous paragraph applies; that is, in this case, conditioned on  $X_{k_i}$ ,  $X_{k_{i+1}}$  is uniformly distributed on the interval  $[X_{k_i}, X_{k_i} + T)$ ; hence, the expected value of the delay from the transmission of the packet from  $k_i$  to the transmission of the same packet from  $k_{i+1}$  is  $T/2$ .

Hence, for a linear arrangement of  $L$  nodes, the expected value of the delay from the generation time of the packet to its delivery at the destination node is  $(L-1)(T/2)$  plus the average propagation delay from node  $k_{L-2}$  to  $k_{L-1}$ . Since the propagation delay (in the last hop) is assumed to be much smaller than  $T$ , we may ignore this term and say that the average MAC delay on a linear arrangement of  $L$  nodes is  $(L-1)T/2$ .

Now, this result applies to *any* path of  $L$  nodes on which a packet is delivered in the network, independent of the network topology. Hence, this result is, in fact, general. For example, assume that the maximum number of hops in the network is ten, and the cycle period  $T = 10$  s. Then, the average MAC protocol delay is 50 s, over the largest number of hops in the network. For scientific data collection applications that require delays ranging

from minutes to hours, this delay would definitely be tolerable. Note that the standard deviation of the average delay decreases as  $\sim 1/\sqrt{L}$  due to the law of large numbers; hence, the average calculated becomes more accurate as the number of nodes in the network grows large. When the guard time policy is applied, it can be shown that this calculated average MAC delay holds approximately, if the nodes in the network have been deployed randomly onto a fixed area.

Now, we turn to the analysis of the number of nodes for which the previously described analysis is valid, that is, for which there is no congestion in the network. We will carry out the analysis for the same example of the linear arrangement of  $L$  nodes. Assume that every node in this arrangement generates  $g$  bits per second on average, to be transferred to node  $k_{L-1}$ , which is taken to be the collection site for all the nodes. Then, the penultimate node, namely, node  $k_{L-2}$  in this arrangement, carries the heaviest traffic, which is  $g(L-1)$  bits per second on average. Assume that each node uses the same modem, with a data rate of  $R$  bits per second. Since each node is awake only for a fraction  $\tau/T$ , the data rate available at each node is  $R\tau/T$ . Hence, at node  $k_{L-2}$ , we must satisfy  $g(L-1) \leq R\tau/T$ . As an example, assume that  $L = 11$ , that is, ten hops, and  $R = 5.4$  kb/s, which is the maximum data rate of the WHOI modem, and  $\tau/T = 0.004$ . Then, the maximum generation rate at each node that can be supported is 7.8 kb/h. This is actually a very reasonable generation rate for sensors that collect CTD and other basic sensor measurements. Even with this rate, there can be congestion in the network since our rate calculation was based on the average, and did not account for the variance. However, it gives a ballpark figure for the generation rates that can be carried without congestion. The analysis can be generalized to other topologies besides the linear arrangement. For example, for a single collection site, and  $N$  randomly deployed sensor nodes around it, the number of hops is on the order of  $\sqrt{N}$ . Then,  $L$  is replaced by  $\sqrt{N}$  in the previous analysis. For the previous example, a random network of roughly 100 nodes can be supported at this generation rate.

#### V. CONCLUSION

The MAC protocol proposed in this paper can serve as a primer for the development of underwater sensor network MAC protocols that have to operate under long, unknown propagation delays. The use of relative time stamps, not only in the transmission of data but also in the establishment of communication with newcomers, allows the nodes to operate in a synchronized environment where they can sleep and save energy the remainder of the time when they are not communicating. The development of such energy-efficient MAC protocols for underwater sensor networks will facilitate the deployment of hundreds of nodes in this harsh communication medium.

This work serves both the development of underwater sensor networks for marine science applications, as pursued for example in the marine and aquatic long-term ecological research sites around the world, and the development of robust communication protocols for underwater surveillance applications where underwater vehicles must operate with high energy efficiency while remaining connected with other vehicles and sensors in their vicinity.

The directions for future research in this area include the development of integrated protocol stacks for underwater communications that take into account the long, unknown propagation delays in the medium. Such work must successfully integrate the decades of accomplishments in the physical layer of underwater acoustic communications with advances at the network layer and is likely to lead to the development of novel protocol stacks that work under large, unknown delays.

#### ACKNOWLEDGMENT

The authors would like to thank R. Schmitt, S. Holbrook, and A. Brooks at the Marine Science Institute, and S. MacIntyre and L. Washburn at the Institute for Computational Earth Systems Science, University of California in Santa Barbara (UCSB), for explaining the current roadblocks and the need for networking solutions for underwater ecological sensing networks.

#### REFERENCES

- [1] D. B. Kilfoyle and A. Baggeroer, "The state of the art in underwater acoustic telemetry," *IEEE J. Ocean. Eng.*, vol. 25, no. 1, pp. 4–27, Jan. 2000.
- [2] I. F. Akyildiz, D. Pompili, and T. Melodia, "Challenges for efficient communication in underwater acoustic sensor networks," *ACM SIGBED Rev.*, vol. 1, no. 2, pp. 3–8, Jul. 2004.
- [3] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad Hoc Netw.*, vol. 3, no. 3, pp. 257–279, Feb. 2005.
- [4] J. G. Proakis, E. M. Sozer, J. A. Rice, and M. Stojanovic, "Shallow water acoustic networks," *IEEE Commun. Mag.*, vol. 39, no. 11, pp. 114–119, Nov. 2001.
- [5] E. M. Sozer, M. Stojanovic, and J. G. Proakis, "Underwater acoustic networks," *IEEE J. Ocean. Eng.*, vol. 25, no. 1, pp. 72–83, Jan. 2000.
- [6] V. Rodoplu and T. H. Meng, "Minimum energy mobile wireless networks," *IEEE J. Select. Areas Commun.*, vol. 17, no. 8, pp. 1333–1344, Aug. 1999.
- [7] V. Rodoplu and T. H. Meng, "Bits-per-joule capacity of energy-limited wireless ad hoc networks," *Proc. IEEE GLOBECOM*, vol. 1, pp. 16–20, Nov. 2002.
- [8] V. Rodoplu and T. H. Meng, "Bits-per-joule capacity of energy-limited wireless networks," *IEEE Trans. Wireless Commun.*, vol. 6, no. 3, pp. 857–865, Mar. 2007.
- [9] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 3, pp. 493–506, Jun. 2004.
- [10] Lake metabolism project, [Online]. Available: <http://www.lakemetabolism.org>
- [11] W. Y. J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. IEEE INFORCOM*, Jun. 2002, pp. 1567–1576.
- [12] J. Shu and P. Varaiya, "PEDAMACS: Power efficient and delay aware medium access protocol for sensor networks," *Inf. Res. Frontiers*, vol. 5, no. 1, pp. 29–37, 2003.
- [13] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for self-organization of a wireless sensor network," in *IEEE Personal Commun.*, Oct. 2000, vol. 7, pp. 16–27.
- [14] Y. Li, W. Ye, and J. Heidemann, "Energy and latency control in low duty cycle MAC protocols," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2005, pp. 676–682.
- [15] C. Enz, A. El-Hoiydi, J.-D. Decotignie, and V. Pereis, "WiseNET: An ultralow-power wireless sensor network solution," *IEEE Computer*, vol. 37, no. 8, pp. 62–70, Aug. 2004.
- [16] V. H. Poor and G. W. Wornell, *Wireless Communications*. Englewood Cliffs, NJ: Prentice-Hall, 1998, ch. 8.
- [17] J. Rice, B. Creber, C. Fletcher, P. Baxley, K. Rogers, K. McDonald, D. Rees, M. Wolf, S. Merriam, R. Mehio, J. Proakis, K. Scussel, D. Porta, J. Baker, J. Hardiman, and D. Green, "Evolution of Seaweb underwater acoustic networking," in *Proc. MTS/IEEE OCEANS*, Sep. 2000, vol. 3, pp. 2007–2017.
- [18] A. Benson, J. Proakis, and M. Stojanovic, "Towards robust adaptive acoustic communications," in *Proc. MTS/IEEE OCEANS*, Sep. 2000, vol. 2, pp. 1243–1284.
- [19] M. Green and J. Rice, "Handshake protocols and adaptive modulation for underwater communications networks," in *Proc. MTS/IEEE OCEANS*, Sep. 1998, vol. 1, pp. 487–491.
- [20] S. M. Smith, J. C. Park, and A. Neel, "A peer-to-peer communication protocol for underwater acoustic communication," in *Proc. IEEE OCEANS*, Oct. 1997, vol. 1, pp. 268–272.
- [21] R. Creber, J. Rice, P. Baxley, and C. Fletcher, "Performance of undersea acoustic networking using RTS/CTS handshaking and ARQ retransmission," in *Proc. MTS/IEEE OCEANS*, Nov. 2001, vol. 4, pp. 2083–2086.
- [22] G. Lapiere, L. Chevallier, F. Gallaud, and G. Ayela, "Design of a communication protocol for underwater acoustic modems and networks," in *Proc. MTS/IEEE OCEANS*, Nov. 2001, vol. 4, pp. 2220–2226.
- [23] K. Foo, P. Atkins, T. Collins, C. Morley, and J. Davies, "A routing and channel-access approach for an ad hoc underwater acoustic network," in *Proc. MTS/IEEE OCEANS*, Nov. 2004, vol. 2, pp. 789–795.
- [24] G. G. Xie and J. Gibson, "A networking protocol for underwater acoustic networks," Naval Postgrad. School, Monterey, CA, Tech. Rep. TR-CS-00-02, 2000.
- [25] R. Kohno, R. Meidan, and L. Milstein, "Spread spectrum access methods for wireless communications," *IEEE Commun. Mag.*, vol. 33, no. 1, pp. 58–67, Jan. 1995.
- [26] M. Stojanovic, J. G. Proakis, J. A. Rice, and M. D. Green, "Spread-spectrum methods for underwater acoustic communications," *Proc. MTS/IEEE OCEANS*, vol. 2, pp. 650–654, Sep. 1998.
- [27] L. Freitag, M. Stojanovic, S. Singh, and M. Johnson, "Analysis of channel effects on direct-sequence and frequency-hopped spread-spectrum acoustic communication," *IEEE J. Ocean. Eng.*, vol. 26, no. 4, pp. 586–593, Oct. 2001.
- [28] V. Rodoplu and M. K. Park, "An energy-efficient MAC protocol for underwater wireless acoustic networks," in *Proc. MTS/IEEE OCEANS*, Sep. 2005, vol. 2, pp. 1198–1203.
- [29] L. Berkhovskikh and Y. Lysanov, *Fundamentals of Ocean Acoustics*. New York: Springer-Verlag, 1982.
- [30] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball, "The WHOI micro-modem: An acoustic communications and navigation system for multiple platforms," in *Proc. MTS/IEEE OCEANS*, Sep. 2005, vol. 2, pp. 1086–1092.



**Min Kyoung Park** (S'02–M'06) received the B.S. degree in electronics engineering from Chun-nam National University, Gwang-ju, Korea, in 1998, the M.S. degree from the Department of Information and Communication, Gwang-ju Institute of Science and Technology (GIST), Gwang-ju, Korea, in 2000, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California, Santa Barbara (UCSB), in 2003 and 2006, respectively.

Currently, she works for the Samsung Advanced Institute of Technology, Yongin-si, Korea. In 2000, she earned a Korean patent for a direct digital frequency synthesizer at GIST. During 2001–2003, she studied communications at the UCSB and developed a beamforming algorithm for frequency-hopping spread-spectrum systems. During 2003–2006, she designed MAC protocols for underwater networks, and developed methods for quality-of-service delivery in terrestrial sensor and ad hoc networks.



**Volkan Rodoplu** (S'94–M'03) received the B.S. degree (*summa cum laude*) from Princeton University, Princeton, NJ, in 1996 and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA, in 1998 and 2003, respectively, all in electrical engineering.

In summer 1998, he worked for Texas Instruments, Dallas, TX, on multiuser detection and interference cancellation algorithms, and during 2000–2001, for Tensilica, Inc., Santa Clara, CA, on communication algorithms and architectures for reconfigurable processors. In 2003, he joined the Department of Electrical and Computer Engineering at the University of California, Santa Barbara, as an Assistant Professor. His research focuses on the design of minimum energy wireless networks, investigating both the theoretical limits of minimum energy transmission, and the practical delivery of minimum energy networking solutions.

Dr. Rodoplu is the recipient of the National Science Foundation CAREER Award (2007), the University of California Regents' Junior Faculty Fellowship (2006), the Andreas Bechtolsheim Stanford Graduate Fellowship (1997–2000), Stanford University Department of Electrical Engineering Outstanding Service Award (2000), the John W. Tukey Award of the American Statistical Association (1996), the George B. Wood Legacy Jr. Prize (1996), and the G. David Forney Jr. Prize from Princeton University (1996).