R-MAC: An Energy-Efficient MAC Protocol for Underwater Sensor Networks *

Peng Xie and Jun-Hong Cui
Computer Science & Engineering Department
University of Connecticut, Storrs, CT 06269
{*xp, jcui*}@*engr.uconn.edu*

Abstract

Underwater sensor networks are significantly different from terrestrial sensor networks in that sound is mainly used as the communication medium. The long propagation delay and limited bandwidth of acoustic channels make the existing MAC protocols designed for radio networks either unpractical or not energy efficient for underwater sensor networks. In this paper, we propose a reservation-based MAC protocol, called R-MAC. The major design goals of R-MAC are energy efficiency and fairness. R-MAC schedules the transmissions of control packets and data packets to avoid data packet collision completely. The scheduling algorithms not only save energy but also solve the exposed terminal problem inherited in RTS/CTS-based protocols. Furthermore, the scheduling algorithms allow nodes in the network to select their own schedules, thus loosening the synchronization requirement the protocol. Additionally, R-MAC supports fairness. By simulations, we show that R-MAC is an energy efficient and fair MAC solution for underwater sensor networks.

1 Introduction

Underwater sensor network has emerged as a powerful technique for aquatic applications, and it has attracted more and more attention from the networking research community recently [15, 10, 2, 6, 4].

Due to the dense network deployment and the shared communication medium, an efficient medium access control (MAC) protocol is very important to the final performance of underwater sensor networks. Different applications have different requirement on MAC protocols. In this paper, we aim to design a MAC protocol for long term aquatic monitoring applications [4]. This type of applications are not sensitive to the end-to-end delay and generate sporadic traffic unevenly distributed spatially and temporally. For such applications, sensor nodes are usually deployed densely, with tens to hundreds of meters apart. The most important goal of the MAC design for such underwater sensor networks is to *resolve data packet collision efficiently in terms of energy consumption*. This is because sensor nodes in underwater sensor networks are usually powered by batteries, and it is difficult to change or recharge these batteries in harsh underwater environments. *Fairness* is another goal of our MAC protocol, as it is very important for in-network data processing. Since data are usually generated by the neighboring sensor nodes that are highly temporally related, it is desirable to deliver temporal related data to some node at the same time for in-network processing. Other properties such as end-to-end latency, throughput and channel utilization are desirable but not hard requirements in our design.

In short, for our targeted applications, the design goals of our MAC protocol is to *resolve data packet collision and support fairness in an energy efficient way*.

There are numerous MAC protocols for designed radio networks. However, we can not directly adapt these protocols to underwater sensor networks due to the significant difference between underwater sensor networks and terrestrial (radio) sensor networks. In underwater sensor networks, acoustic channel is used as the communication method. The propagation speed of sound in water is about 1500 m/s, which is 5 orders of magnitude lower than that of radio. The low propagation speed results in a high propagation delay even for communication between two neighbors. Moreover, the available bandwidth of acoustic channels is typically less than 15 KHz, which is much narrower compared with that of RF channels. These unique acoustic communication characteristics pose many challenges when applying existing protocols in underwater sensor networks.

In general, MAC protocols can be roughly divided into two categories: contention-free protocols and contention-based protocols. Contention-free protocols include TDMA, FDMA and CDMA, where communication channels are separated in time, frequency or code domains. It is common wisdom that FDMA is unsuitable for underwater sensor networks because of the narrow available bandwidth. There are some researches on TDMA and CDMA for underwater net-

---

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

works. However, some problems inherent in these methods have not been well addressed in acoustic networks. For example, the synchronization problem in TDMA and near-far problem in CDMA. Thus, the feasibility of these protocols in underwater sensor networks is unclear.

Contention-based protocols includes random access methods [1, 11, 12] and collision avoidance methods [7, 3, 5]. In a random access protocol, the sender sends packets without coordination. Thus packet avoidance is totally probabilistic. While in a collision avoidance protocol, the sender and receiver capture the medium through control packet exchange before data transmission.

There are many collision avoidance protocols, among which RTS/CTS-based protocols are widely used. The performance of random access methods and RTS/CTS-based approaches in underwater sensor networks is determined by many factors. [17] has shown that RTS/CTS-based methods outperform random access approaches (in terms of energy efficiency) in a network with short transmission ranges, dense neighborhood and bursty traffic. These properties are close to those in our targeted underwater sensor networks. Thus, it appears promising for us to explore the RTS/CTS based approaches.

However, the long propagation delay in underwater sensor networks makes it too energy consuming to use RTS/CTS to eliminate the data packet collisions completely. As shown in [5], in order to completely avoid data packet collision, two conditions have to be satisfied: 1)the duration of RTS should be greater than the maximum propagation delay; and 2) the duration of CTS should be greater than that of RTS plus twice the maximum propagation delay plus the hardware transmit-to-receive transmission time. It is easy to satisfy these two conditions in terrestrial radio sensor networks. However, it is too expensive to do so in underwater sensor networks where the maximum propagation delay is usually very long, as makes the size of control packets unacceptably large.

If we relax the requirement of completely avoiding data packet collisions, we can use a modified RTS/CTS-based method as in [17]. Although the modified method can not avoid data packet collisions completely, it can reduce the data packet collisions to a low level. However, the modified RTS/CTS method is not energy efficient. In order to avoid packet collisions, a node has to listen the channel during the active time period. On the other hand, the high propagation delay prolongs the active time period. Therefore, the high propagation delay results in more energy waste on overhearing and idle state when RTS/CTS exchange is used in underwater sensor networks.

In this paper, we concentrate on the collision avoidance approaches for underwater sensor networks. We propose a reservation-based MAC protocol, called R-MAC, to achieve the objectives of energy efficiency and fairness. In R-MAC, we do not use the RTS/CTS message exchange to avoid data packet collisions. Instead, we schedule the transmission of control packets and data packets at both the sender and the receiver to avoid data packet collisions completely. The scheduling algorithms address the problem caused by the high propagation delay efficiently. Moreover, R-MAC supports fairness. Additionally, we adopt a new ARQ technique, burst-based acknowledgment, in R-MAC to improve channel utilization. The burst-based acknowledgment combined with the scheduling algorithms solves the exposed terminal problem and improves the network throughput.

The rest of the paper is organized as follows. In Section 2, we present R-MAC and the scheduling algorithms in details. In Section 3, we evaluate the performance of R-MAC using simulations. In Section 4, we briefly discuss some related work. In Section 5, we conclude our work and discuss some future directions.

2 R-MAC Protocol Design

In this section, we first brief the basic ideas of R-MAC, then we describe the three phases of R-MAC in details. After that, we focus on the scheduling algorithms on both sender and receiver. Finally, we discuss the fairness issue.

2.1 Overview of R-MAC

In R-MAC, to reduce the energy waste on idle state and overhearing, each node works in listen and sleep modes *periodically*. The durations for listen and sleep are the same for all nodes, and each node *randomly* selects its own schedule, as means that no centralized scheduling and synchronization are required in R-MAC. For any node, if there is no traffic in its neighborhood, it simply listens and sleeps periodically. When a node (i.e., sender) wants to send data to another node (i.e., receiver), R-MAC employs a reservation-based approach to synchronize, in a distributed way, transmissions to avoid data collisions.

R-MAC has three phases, namely, *latency detection*, *period announcement*, and *periodic operation*. The first two phases are used to synchronize nodes in the neighborhood and the third one is for listen/sleep operations. A node in the latency detection phase detects the propagation latency to all its neighbors. In the period announcement phase, each node randomly selects its own listen/sleep schedule and broadcasts this schedule. The data (if there are any) are transmitted in the periodic operation phase. Next we discuss the three phases in details.

2.2 Phase One: Latency Detection

In this phase, all nodes power on. Each node randomly selects a time to broadcast a control packet, called *Neighbor Discovery packet*, denoted as *ND*. Upon receiving NDs from its neighbors, a node records the arrival times of these

NDs, then randomly selects a time to transmit an acknowledgment packet, denoted as *ACK-ND*, which has the same packet size as ND, for each of the NDs it receives. In each ACK-ND, the node specifies the duration from the arrival time of the ND packet to the transmission time of this ACK-ND packet, I_2. After receiving an ACK-ND, a node computes the interval from the time that the corresponding ND packet is transmitted to the arrival time of the ACK-ND, I_1. Then the propagation latency, L, between the two nodes can be calculated as $L = \frac{I_1 - I_2}{2}$. Therefore, the propagation latency L between two nodes is the interval from the time the first node sends the first bit of a packet to the time the second node receives the last bit of the packet.

An example is illustrated in Figure 1. Node A sends an ND packet and records the time. Upon receiving this packet, node B randomly delays some time period I_B and sends an ACK-ND packet back to node A. Node B specifies in the ACK-ND packet the time interval I_B, its ID and the ID of ND packet. Upon receiving this packet, node A computes the time interval from the transmission time of its ND packet to the arrival time of ACK-ND packet, I_A. Then node A calculates the propagation latency to node B as $L_{AB} = \frac{I_A - I_B}{2}$.

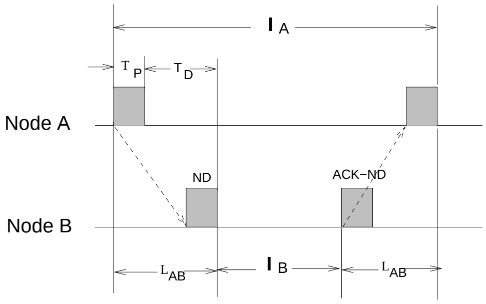

Figure 1. Latency measurement

Propagation latency L includes transmission delay and propagation delay. Since the system and network architecture in sensor nodes are relatively simple, the transmission delay is mainly determined by the hardware and the size of the packet. Thus its variance is negligible. The propagation delay is mainly depends on the sound speed in water, which might be affected by many factors such as temperature and pressure. However, for a short time period, it is reasonable to assume that the sound speed in water is constant, i.e., propagation delay does not change in a short time period. Therefore, propagation latency L is accurate and stable for a short time period. With time going, latency measurements become more and more inaccurate due to clock drift and varying sound speed. Thus, it is desirable that these measurement will be updated after a period of time, as can be done through the message exchange in the third phase.

After the latency detection phase, each node records the propagation latencies to all of its neighbors.

2.3 Phase Two: Period Announcement

In this phase, each node randomly selects its own start time of the listen/sleep periodic operations (i.e., the third phase) and broadcasts this time (we also call it *schedule*). After receiving broadcast packets, each node converts the received times (schedules) to its own time (schedule).

As shown in Figure 2, node A, randomly selects its listen/sleep schedule, and announces this schedule by broadcasting a synchronization packet, denoted as *SYN*. There are two fields in this packet: node A's ID and time interval I_A, which specifies the interval from the time to send SYN to the beginning time of its third phase. Upon receiving a SYN packet from node A, node B calculates the time interval from the arrival time of this SYN packet to the starting time of its third phase, I_B. Then node B converts the schedule of node A relative to its own schedule by $I_B - I_A + L_{AB}$, where L_{AB} is the propagation latency from node A to node B.

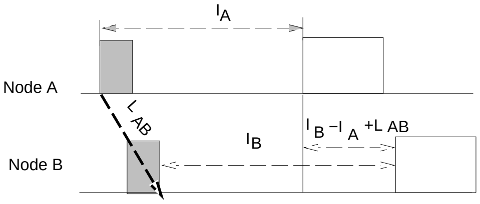

Figure 2. Schedule conversion

In the second phase of R-MAC, each node records the schedules of its neighbors relative to its own schedule. In the real implementation, the first two phases can run multiple rounds to make sure that all the nodes in the network have complete information about their neighbors.

2.4 Phase Three: Periodic Operation

In this phase, nodes wake up and sleep periodically. We call one listen/sleep cycle as one *period*. All nodes have the same periods. One period, denoted as T_0, consists of two parts: listen window, denoted as T_L, and sleep window T_S. Thus $T_0 = T_L + T_S$.

In R-MAC, nodes communicate through *REV/ACK-REV/DATA/ACK-DATA* message exchange, where *REV* denotes the reservation packet, *ACK-REV* is the acknowledgment packet for *REV*, and *ACK-DATA* is the acknowledgment packet for data *DATA*. All the control packets in R-MAC, namely REV, ACK-REV and ACK-DATA have the same size, which is much smaller than that of data packets. When a node has data to send, it first sends a REV to reserve a time slot at the receiver. If the receiver is ready for data transmission, it will notify all its neighbors about the reserved time slot by ACK-REVs. Upon receiving ACK-REVs, all the nodes other than the sender keep silent in their corresponding time slots, and the sender can send data at the reserved time slot. In R-MAC, data are transmitted in a burst. A node queues its data for the same receiver until it captures the channel, then injects all the queued data. The receiver sends back an ACK-DATA to the sender at the end of the burst transmission. In other words, in order to reduce the control packet overhead and improve the channel utilization, the receiver acknowledges per burst instead of per packet. We refer to this technique as *burst-based acknowledgement* in this paper.

R-MAC treats ACK-REVs as the highest priority packets and reserves the first part of the listen window exclusively for ACK-REV packets. We call this reserved part *R-window*, denoted as T_R, which is the maximum possible duration of a control packet. The rationale of this design is the following: ACK-REV is used by the receiver to notify its neighbors of not interrupting the subsequent data transmission. If a node misses an ACK-REV, it possibly interferes the subsequent data transmission. As for the case of missing a REV or ACK-DATA, no data collisions will be caused. In R-MAC, nodes only have to sense the channel in their R-windows to get the information about the subsequent data transmission. If a node receives an ACK-REV in its R-window, then this node knows the duration of the subsequent data transmission and keeps silent during that time period. However, when a node senses the channel busy in its R-window, but can not receive an ACK-REV clearly (i.e., there is a ACK-REV collision), it will back off. When a node is in backoff state, it still needs to sense the channel in its R-window and updates the usage information of the channel in its neighborhood.

Since R-windows are designed for receiving ACK-REVs, all other types of packets (including REV, DATA, ACK-DATA) have to avoid R-windows. To achieve this purpose, in R-MAC, all nodes have to carefully schedule the transmission of control and data packets. The scheduling algorithms at both the sender and receiver should guarantee that only ACK-REVs can propagate to any node in its R-window, and all other control packets such as REVs and ACK-DATAs are scheduled to arrive at the target in its listen window and data packets are scheduled to arrive at the intended receiver in its reserved time slot. We discuss the scheduling algorithms next.

2.5 Scheduling at Sender

When a node has queued data packets and is in idle state, it then schedules to send a REV to the intended receiver so that the REV arrives in the receiver's listen window and, at the same time, avoids the R-windows of all its neighbors.

The sender first maps the whole listen window of the intended receiver and the R-windows of its neighbors into its own time line, then marks all the mapped R-windows which fall in the mapped listen window. After that, it divides the unmarked part (i.e., the available part) of the mapped listen window into slots by the duration of one control packet and randomly selects one slot as the time to transmit the REV. The REV specifies the required data duration and the offset of the to-be-reserved time slot to the beginning of its current period. When the sender calculates the duration needed to transmit the queued data packets, it has to count the time to skip the mapped R-windows of its neighbors.

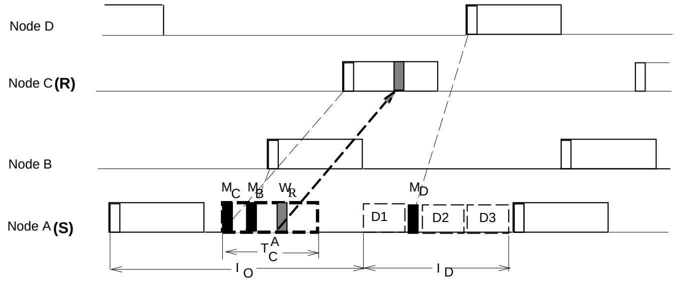

Figure 3. Scheduling at sender

Figure 3 illustrates the scheduling algorithm at the sender. In this figure, node A is the sender and node C is the intended receiver. Node A maps the listen window of node C, the R-windows of nodes C, B and D into to its time line as T_C^A, M_C, M_B and M_D, respectively. When node A schedules to send a REV to node C, it randomly selects a slot from the unmarked part of T_C^A, denoted as W_R, to send the REV packet. The REV packet will arrive at node C during its listen window without interfering other nodes in their R-windows. In the REV packet, node A specifies the offset of the reserved time slot to the beginning of its current period, denoted as I_O, and the duration of the reserved time slot, denoted as I_D, which includes the transmission time of all the data packets ($D1$, $D2$ and $D3$ in the figure) and the time to skip the mapped R-window of node D, M_D.

The scheduling algorithm at the sender guarantees that no REV arrives in any node's R-window in its neighborhood. Furthermore, once the channel is granted, no data packet arrives in any node's R-window since the sender avoids all the R-windows of its neighbors when it transmits data packets.

2.6 Scheduling at Receiver

Once a reservation is selected, the receiver first schedules the transmission of ACK-REVs, and based on this schedule, arranges a time slot for the selected reservation. During the reserved time slot, the receiver powers on and waits for incoming data packets. After a pre-defined interval, if it does not receive any data packets as scheduled[1], it simply quits the receiving state and goes back to the periodic

---

[1] It is possible that the sender receives another ACK-REV (from another pair of transmission) before the ACK-REV from this receiver, and thus can not speak. In this case, the sender has to back off and resend a REV later.

listen/sleep. After the receiver receives data packets in a burst, it schedules the transmission of ACK-DATA so that the ACK-DATA arrives in the sender's listen window.

2.6.1 Scheduling Algorithm for ACK-REVs

To avoid data transmission interference, the receiver should guarantee that before the sender receives its ACK-REV, all the receiver's neighbors have already been notified the reserved time slot. Therefore, when the sender receives the ACK-REV from the receiver, it is already granted the channel for the subsequent data transmission.

The receiver sends ACK-REV packets to its neighbors in their mapped R-windows to guarantee that these ACK-REVs arrive during their R-windows. However, the mapped R-window to send ACK-REV to the sender is the earliest mapped R-window which is greater than $S_i = M_i + 2 \times L_i$, i is any of the receiver's neighbors other than the sender, where M_i is the mapped R-window of node i at the receiver and L_i is the propagation latency from the receiver to node i. Here we introduce an additional one-way delay to handle the following case: after the receiver sends out an ACK-REV to one neighbor, it is possible that the neighbor transmits an ACK-REV (for another pair of transmission) to the receiver. In such case, the receiver checks if the reserved time slot specified in the incoming ACK-REV conflicts with its transmission of ACK-REVs. If yes, the receiver stops scheduling to send ACK-REVs. Otherwise, the receiver records the reserved time slot, continues to transmit its ACK-REVs and prepares for incoming data packets.

In some cases, the mapped R-windows at the receiver possibly overlap each other. In an even rare case, the receiver's own R-window overlaps with the mapped R-window of some neighbor. These special cases can be effectively handled in the second phase of R-MAC: period announcement. In the second phase, if a node finds out either one of these cases occurs, the node re-schedules its periodic sleep/listen. Since the duration of a R-window is very short compared with the duration of one period, for example, in our implementation of R-MAC, $T_R \leq 10ms$ and $T = 1s$, it only takes a few rounds for nodes to randomly select their schedules to avoid such cases.

2.6.2 Scheduling Algorithm for Reserved Time Slot

When the receiver determines the reserved time slot, it has to leave enough time for the ACK-REV to reach the sender and for the data packets to propagate from the sender to the receiver. Since the offset of the reserved time slot within a period of the sender is already specified in the REV packet, the receiver computes the offset of the reserved time slot to its own period and arranges the reserved time slot according to the transmission time of its ACK-REVs. The reserved time slot is the earliest time slot that is greater than $S_s = M_s + 2 \times L_s$, where M_s is the mapped R-window of the sender on the receiver and T_s is the propagation latency from the receiver to the sender. Therefore, when the sender receives the ACK-REV, it has enough time to deliver its data packets to the receiver in the reserved time slot.

In each ACK-REV packet, the receiver specifies the interval from the transmission time of this ACK-REV to the reserved time slot and the duration of the reserved time slot.

2.6.3 Scheduling Algorithm for ACK-DATA

When the receiver receives all the data packets, it schedules to acknowledge the data burst. R-MAC treats REV and ACK-DATA in the same way. That is, the receiver uses the same scheduling algorithm for REVs to schedule an ACK-DATA. It needs to make sure that the ACK-DATA arrives at the sender in its listen window. In the ACK-DATA packet, the receiver indicates whether a packet is received or corrupted by a bit vector.

2.6.4 Giving an Example

Now, we use an example to illustrate the scheduling at the receiver. Referring to Figure 4, again, node A is the sender and node C is the receiver. M_A, M_B, and M_D are the mapped R-windows at node C for nodes A, B, and D respectively. Node C transmits ACK-REVs for node B and D first, then transmits ACK-REV for node A. S_D is the earliest time that M_A could be scheduled and S_A is the lower bound for the final reserved time slot.

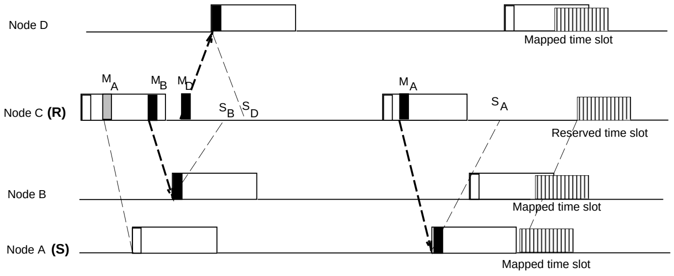

Figure 4. Scheduling at receiver

2.7 Fairness

Fairness is a desirable property for a MAC protocol. R-MAC supports fairness in two aspects. First, an intended receiver provides equal opportunities for all its neighbors to make reservations. Second, an intended receiver randomly selects one reservation from the reservations it collects. The first aspect is automatically guaranteed by the REV/ACK-REV series. The second aspect is granted by introducing additional waiting time at the receiver to get all possible REVs from its neighbors (Due to space limit, this technique is omitted from this paper. Interested readers can find more details from our technique report [16]). As we can see, R-MAC supports fairness at the cost of lengthening end-to-end delay. In our targeted applications, however, end-to-end delay is not important compared with energy efficiency and fairness, as long as it satisfies the application requirements.

3 Simulation Study

In this section, we evaluate the performance of R-MAC using simulations. By comparing with T_u-MAC (a revised version of T-MAC for underwater sensor networks), we demonstrate the energy efficiency of R-MAC. We also conduct experiments to explore the fairness and channel utilization of R-MAC. Due to space limit, we do not include these results in this paper. Interested readers can refer to our technical report [16].

3.1 Simulation Settings

We implement R-MAC in Aqua-Sim, an NS-2 based simulator for underwater sensor networks, developed at the Underwater Sensor Network (UWSN) Lab at the University of Connecticut *(http://uwsn.engr.uconn.edu/)*.

Unless specified otherwise, we use the following parameters in the simulations. We set the size of control packets to 5 Bytes and the data packet size to 60 Bytes. The period length is 1 second and the listen window is 100 ms. The bit rate is 10 Kbps. The data generation follows a Poisson process with an average rate as λ, called *traffic rate*. The maximum transmission range is 90 meters. The interference range is the same as the transmission range. The maximum number of data packets allowed in a burst is 3. In our simulations, we set the energy consumption parameters based on a commercial underwater acoustic modem, UMW1000, from LinkQuest [8]: the power consumption on transmission mode is 2 Watts; the power consumption on receive mode is 0.75 Watts; and the power consumption on sleep mode is 8 mW.

Performance Metrics: We define two metrics: *energy consumption* and *end-to-end delay*. Energy consumption is measured by the average energy consumed for each successfully delivered packet. End-to-end delay is the average time interval from the source to the destination for each successfully delivered packet.

3.2 Implementation of T_u-MAC

As we mentioned earlier, there are no energy efficient MAC solutions for underwater sensor networks with unevenly distributed traffic in the literature. We choose to implement T-MAC [14] as a reference because both T-MAC and R-MAC can adapt to unevenly distributed traffic with high energy efficiency, though T-MAC is designed for radio-based sensor networks. We apply the idea of T-MAC in underwater sensor networks to show that it is not efficient to simply adapt radio-based network protocols in underwater network environment.

To make T-MAC feasible for underwater sensor networks, we implement a revised version of T-MAC, referred to as T_u-MAC. We make three major revisions: First, we modify the active time, *TA*, to incorporate propagation delay, which is non-negligible in underwater sensor networks; Second, we modify the RTS/CTS method adopted in T-MAC. As shown in [17], the original RTS/CTS method is not suitable for underwater sensor networks due to long propagation delays. In order to make it practical in underwater sensor networks, we make the following change: when the sender receives a CTS from the intended receiver, it has to wait until the CTS propagates through the whole transmission range of the receiver; Third, since carrier sensing does make much sense in underwater sensor networks, T_u-MAC does not adopt this technique. Instead, it has the following design: if a node senses the channel busy when it starts to transmit a packet, it will back off for a random time interval between one data packet duration and two data packets durations.

3.3 Energy Efficiency

We evaluate the energy efficiency of R-MAC and T_u-MAC using a star topology as shown in Figure 5. In this network, node 0 is the only receiver and all other nodes are senders. All the senders can hear each other. We measure the energy consumption of R-MAC and T_u-MAC at different traffic rates. To make the comparison fairly, we choose the same parameters for R-MAC and T_u-MAC wherever this rule applies. It should be also noted that the ARQ technique adopted in the original T-MAC is different from that used in R-MAC. We thus also implement ARQ per burst, i.e., burst-based acknowledgement, for T_u-MAC.

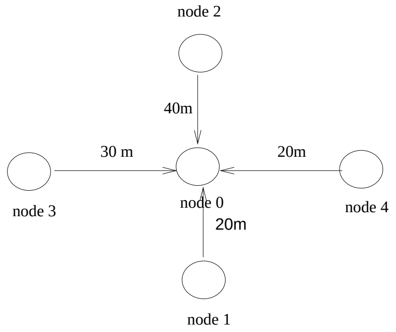

Figure 5. Star topology for energy efficiency

To analyze how R-MAC achieves energy efficiency, in the following we first measure the average time spent on transmitting, receiving, and idle state per successfully delivered packet, for each we call transmitting overhead, receiving overhead and idle overhead respectively.

The results for transmitting overhead are shown in Figure 6. From this figure, we can see that the transmitting overhead of R-MAC is higher than that of T_u-MAC when the traffic rate is low, but it becomes less as the traffic rate continues to increase. This is because that the receiver in R-MAC has to transmit an ACK-REV to each of its neighbors, while the receiver in T_u-MAC just needs to send one CTS for all of its neighbors. Thus when the traffic rate is low, the number of packets in a burst is small, which means that the probability that data packets in T_u-MAC collide is very low. Thus the transmitting overhead of R-MAC is higher than that of T_u-MAC. As the traffic rate keeps increasing, the number of data packets in a burst becomes larger, which leads to the reduction of transmitting overhead in R-MAC. On the other hand, when the data burst size is lifted, the data packet collision in T_u-MAC becomes more and more significant, which causes the increase of transmitting overheads in T_u-MAC. When the traffic rate is relatively high (greater than 0.14 pkts/s as shown in the figure, the data packet collision is the dominating source for the transmitting overheads in T_u-MAC. Therefore, in this case, the transmitting overhead of R-MAC is lower than that of T_u-MAC.

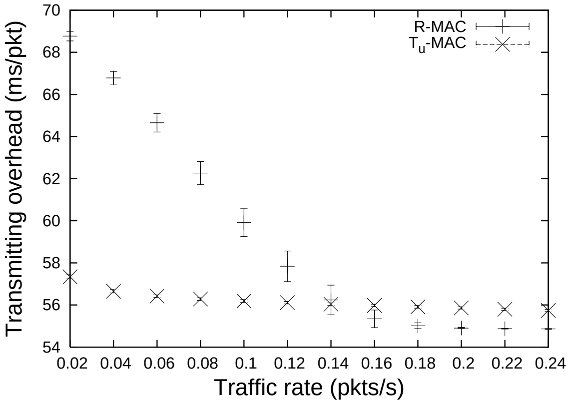

Figure 6. Transmitting overhead

The results for receiving overhead are shown in Figure 7. As we can see, the receiving overhead for R-MAC is only half that of T_u-MAC. Such significant difference on the receiving overhead is attributed to the random schedule in R-MAC. In T_u-MAC, all nodes listen at the same time; therefore, when a node sends packets, all its neighbors overhear these packets, resulting in higher receiving overhead. On the other hand, in R-MAC, nodes have different schedules. When a node sends packets, it is most likely that other nodes are in their sleep state.

Figure 8 shows the results of idle overhead. From this figure, we can observe that idle overhead decreases with the growth of traffic rate and the idle overhead of R-MAC is only half of that of T_u-MAC. This is mainly caused by two factors. First, the method adopted in T_u-MAC to reduce end-to-end delay results in longer idle time on each node. In T_u-MAC, nodes have to stay idle for possible future traffic when they find the channel is captured by other nodes. Second, the long active time, TA, of each node also contributes to the high idle overhead of T_u-MAC.

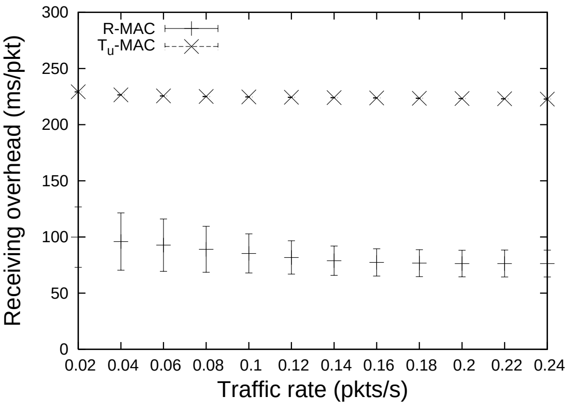

Figure 7. Receiving overhead

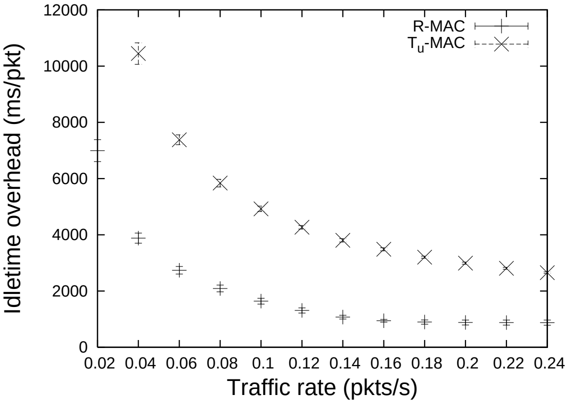

Figure 8. Idle overhead

We convert the overhead shown in Figure 6, Figure 7 and Figure 8 into energy consumption per data packet based on the parameters of UMW1000 modems, and the results are plotted in Figure 9. From this figure, we can observe that under low traffic rate such as 0.02 pkts/s, even R-MAC has higher transmission overhead (per data packet), it is still more energy efficient than T_u-MAC due to its lower receiving overhead and less idle overhead per data packet. As the traffic rate increases, the difference between R-MAC and T_u-MAC is enlarged. This is mainly due to the transmitting inefficiency of T_u-MAC when the traffic rate is high.

From this set of simulations, we can conclude that R-MAC is much more energy efficient than T_u-MAC under various traffic rates. This tells us that we can not directly apply T-MAC, a well designed MAC solution for radio-

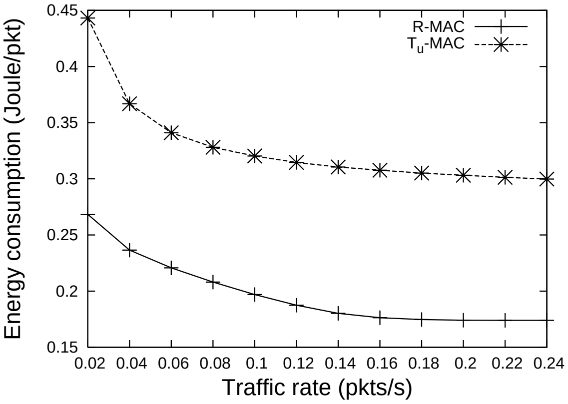

Figure 9. Energy consumption

based sensor networks, in underwater network scenarios. The conclusion also indicates that R-MAC can achieve high energy efficiency under unevenly distributed traffic.

3.4 End-to-End Delay

In this set of simulations, we evaluate the end-to-end delays of R-MAC and T_u-MAC in a multi-hop network. The topology is shown in Figure 10, where node 0 is the receiver and node n is the only sender. The distance between any two adjacent nodes is 80 meters. The sender sends data at the average rate of 0.1 packet per second (a light traffic rate). We vary the number of hops by adding different number of nodes between the sender and the receiver. For each number of hops, we compute the average end-to-end delay of all the packets received by the receiver.

The end-to-end delay results are plotted in Figure 11. From this figure, we can see that the end-to-end delay of both R-MAC and T_u-MAC increases linearly as the number of hops increases. These results are consistent with our analysis (reported in our technical report [16]): at light traffic, R-MAC at most takes $5T_0$ to transmit one packet per hop, and T_u-MAC takes about T_0. This is mainly due to two factors: 1) T_u-MAC explicitly introduces a technique to reduce end-to-end delay (as described earlier) with the cost of sacrificing energy efficiency; 2) R-MAC introduces additional delay (one period) to guarantee fairness. In other words, R-MAC trades end-to-end delay for energy efficiency and fairness.

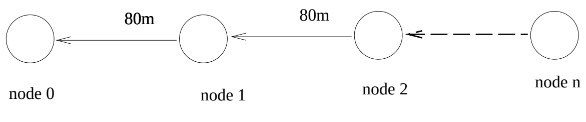

Figure 10. Line topology for end-to-end delay

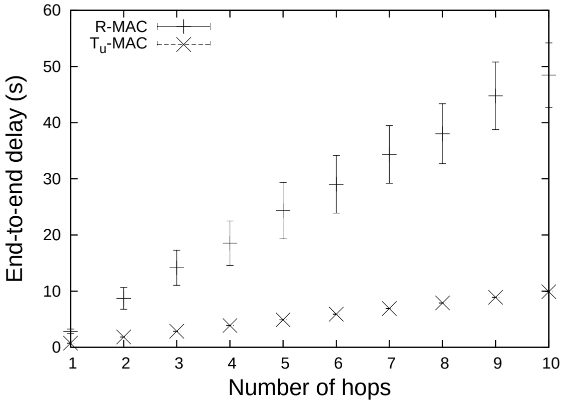

Figure 11. End-to-end delay

We also run simulations for high traffic rate. In fact, due to the good scheduling of R-MAC, the increase of end-to-end delay in R-MAC is much smaller than the increase in T_u-MAC. For example, for a traffic rate of 1 packet per second, the end-to-end delay of T_u-MAC for 10-hops is around 100 seconds, while the result of R-MAC is only about 200 seconds. As we increase the traffic rate to very high, the end-to-end delays of R-MAC and T_u-MAC are very close. This is because the packet queueing delay is dominating when the transmission competition is very intensive. Moreover, the delivery ratio of T_u-MAC drops much more significantly at high traffic than that of R-MAC. Thus, if we combine the delay results with the energy results, we can conclude that R-MAC has more advantages in the networks with high data traffic.

4 Related Work

Many widely used MAC protocols, such as IEEE 802.11, for radio networks are based on the RTS/CTS approach. The RTS/CTS approach was first proposed in MACA[7]. Then a variant protocol, MACAW, was proposed in [3]. This protocol adopts backoff and ARQ techniques in addition to the RTS/CTS control message exchange. Later, carrier sensing was combined with RTS/CTS in a new protocol, called FAMA, in [5].

A major problem with these RTS/CTS based protocols is energy efficiency: too much energy is wasted in idle state since all nodes keep powered on all the time.

With the emerging of sensor networks, energy-efficient MAC becomes a hot topic. PAMAS proposed in [13] makes an improvement to save energy by putting nodes into sleep state when the nodes are prohibited from sending any packet. PAMAS improves MACA without sacrificing the throughput and end-to-end delay. However, in order to turn off/on the nodes intelligently, PAMAS uses a separate signaling channel as a control channel, as is not desirable in sensor networks.

S-MAC proposed in [18] uses the in-channel signal to control the node to listen and sleep periodically. S-MAC reduces energy consumption on listening significantly. However, the fixed duty cycle in S-MAC is undesirable since the traffic in sensor networks varies with locations and time. T-MAC proposed in [14] improves S-MAC by adopting an adaptive duty cycle scheme to reduce the energy consumption while maintaining a reasonable throughput.

Our R-MAC design has benefited from the previous techniques in various aspects. However, as shown in Section 3, we cannot directly apply radio-based ideas into underwater network scenarios. Instead, new MAC solutions are required for underwater sensor networks. In the literature, there are only few MAC protocols proposed for underwater networks. We discuss two recent proposals as follows.

A random access based MAC protocol for underwater sensor networks is proposed in [12]. Due to the fundamental limitation of random access, this protocol only works well for the networks with very low and evenly distributed traffic. Moreover, this protocol has high overhearing and idle waste since for a given node all its neighbors have to wake up for the possible traffic, even when this node sends nothing.

Recently, a modified FAMA, called slotted FAMA, is proposed in [9] for underwater acoustic networks. In slotted FAMA, time is divided into slots, and all packets including control packets and data packets are sent at the beginning of a slot. In this way, the lengths of RTS and CTS are not determined by the propagation delay as is in the original FAMA, thus making the protocol feasible for underwater acoustic networks. It should be noted that energy efficiency is not the design goal of slotted FAMA.

5 Conclusions and Future Work

In this paper, we propose an energy efficient MAC protocol, R-MAC, for underwater sensor networks. R-MAC carefully schedules the transmissions of control and data packets to avoid data packet collisions. The scheduling algorithms not only avoid data packet collisions completely, but also solve the exposed terminal problem. In R-MAC, each node adopts periodic listen/sleep to reduce the energy waste in idle state and overhearing. Moreover, R-MAC loosens the synchronization requirement by allowing each node randomly selects its own schedule. Additionally, R-MAC supports fairness. Finally, the burst-based acknowledgment technique reduces the control packet overhead further and improves the channel utilization.

We plan to integrate some coding schemes in the burst-based acknowledgment to make R-MAC more robust and energy efficient under error-prone communication channels. Considering the harsh underwater environments, we believe that the combination of coding and ARQ is promising for reliable underwater acoustic communications.

References

[1] N. Abramson. The ALOHA System -Another Alternative for Computer Communications. *AFIP Press*, Vol 37, 1970.

[2] I. F. Akyildiz, D. Pompili, and T. Melodia. Challenges for Efficient Communication in Underwater Acoustic Sensor Networks. *ACM SIGBED Review*, Vol. 1(1), July 2004.

[3] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. Macaw:A media access protocol for wireless lans. In *ACM SIGCOMM'94*, 1994.

[4] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou. Challenges: Building Scalable Mobile Underwater Wireless Sensor Networks for Aquatic Applications. *Special Issue of IEEE Network on Wireless Sensor Networking*, May 2006.

[5] C. L. Fullmer and J. Garcia-Luna-Aceves. Floor Acquisition Multiple Access (FAMA) for Packet-Radio Networks. In *ACM SIGCOMM'95*, Cambridge, MA, USA, August 1995.

[6] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li. Research Challenges and Applications for Underwater Sensor Networking. In *IEEE Wireless Communications and Networking Conference*, Las Vegas, Nevada, USA, April 2006.

[7] P. Karn. MACA - a new channel access method for packet radio. In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pages 134–140, 1990.

[8] LinkQuest. http://www.link-quest.com/.

[9] M. Molins and M. Stojanovic. Slotted FAMA: a MAC protocol for underwater acoustic networks. In *IEEE OCEANS'06*, Sigapore, May 2006.

[10] J. G. Proakis, E. M. Sozer, J. A. Rice, and M. Stojanovic. Shallow Water Acoustic Networks. *IEEE Communications Magazines*, pages 114–119, November 2001.

[11] L. Roberts. Aloha Packet System with and without slots and captures . In *Tech. Rep Ass Note 8*, Standford Research Institute, Advanced Research Project Agency, Network Information Center, 1972.

[12] V. RoDoplu and M. K. Park. An Energy Efficiency MAC Protocol for Underwater Wireless Acoustic Networks . In *IEEE Oceans Conference*, Washington DC, 2005.

[13] S. Singh and C. S. Raghavendra. PAMAS-Power Aware Multi-Access Protocol with Signalling for Ad. In *ACM Computer Comm. Rev.*, pages 5–26, July 1998.

[14] T. van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *ACM SenSys'03*, Los Angeles, California, USA, November 2003.

[15] G. G. Xie and J. Gibson. A Networking Protocol for Underwater Acoustic Networks. In *Technical Report TR-CS-00-02*, Department of Computer Science, Naval Postgraduate School, December 2000.

[16] P. Xie and J.-H. Cui. An energy-efficient mac protocol for underwater sensor networks. *UCONN CSE Technical Report: UbiNet-TR06-06*, December 2006.

[17] P. Xie and J.-H. Cui. Exploring Random Access and Handshaking Techniques in Large-Scale Underwater Wireless Acoustic Sensor Networks. In *MTS/IEEE Oceans'06*, Boston, MA, September 2006.

[18] W. Ye, J. Heidemann, and D. Estrin. Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks. *IEEE/ACM TRANSACTIONS ON NETWORKING*, Vol.12(3), June 2004.