MACA-MN: A MACA-based MAC Protocol for Underwater Acoustic Networks with Packet Train for Multiple Neighbors

Nitthita Chirdchoo, Wee-Seng Soh, Kee Chaing Chua
Department of Electrical & Computer Engineering
National University of Singapore, Singapore
Email: {g0500102, elesohws, eleckc}@nus.edu.sg

***Abstract*— Unlike the terrestrial wireless networks that utilize the radio channel, underwater networks use the acoustic channel, which poses research challenges in the medium access control (MAC) protocol design due to its low bandwidth and high propagation delay characteristics. Since most of the MAC protocols for wireless terrestrial networks have been designed with negligible propagation delay in mind, they generally perform poorly when applied directly in underwater acoustic networks, especially for the case of handshaking-based protocols. In this paper, we propose a MACA-based MAC protocol with packet train to multiple neighbors (MACA-MN). It improves the channel utilization by forming a train of packets destined for multiple neighbors during each round of handshake, which greatly reduces the relative proportion of time wasted due to the propagation delays of control packets. This approach also reduces the hidden terminal problem. Our simulations show that the MACA-MN is able to achieve much higher throughput than the MACA protocol.**

I. Introduction

Although underwater communication has been around for several decades, it mainly utilizes point-to-point communication, which limits the growth of its applications. Due to the needs of new emerging applications such as oceanographic data collection, pollution monitoring, offshore exploration, and disaster prevention, etc. [1], underwater sensor networks have been extensively studied in recent years, especially in the area of underwater medium access control (MAC). This is because in any wireless network, MAC design is an integral part for achieving the desired network performance.

Unlike the terrestrial wireless networks that utilize the radio channel, underwater networks use the acoustic channel, which poses new research challenges when dealing with the MAC design. The underwater acoustic channel is characterized by low bandwidth and high propagation delay. The acoustic channel's bandwidth is both frequency and communication range dependent [2], [3]. Specifically, a long-range system that operates over tens of kilometers may have a bandwidth of only a few kilohertz, while a short-range system operating over tens of meters may have a hundred kilohertz of bandwidth [1]. On the other hand, the low speed of sound in underwater causes its propagation delay to be around 0.67 s/km; this is very high compared to the terrestrial wireless networks, which often assume negligible propagation delay. Due to the above-mentioned uniqueness of underwater networks, the wide variety of MAC protocols previously proposed for wireless terrestrial networks do not perform well in underwater. The underwater MAC protocols must account for the low bandwidth and high propagation delay characteristics.

In [4], we proposed two MAC protocols, namely, Aloha-CA and Aloha-AN. These two protocols utilize the information obtained from overheard packets to calculate the busy durations of other nodes, which would be useful for scheduling DATA packet transmissions. For Aloha-CA, when a node has a packet to transmit, it must first check the busy durations of other nodes whether its transmission would cause any collision. If a collision can occur, it will defer its transmission for some random time. In the case of Aloha-AN, when a node has a packet to transmit, it also performs a collision check in a similar way as Aloha-CA. If no collision is foreseen, it will notify the other nodes about its pending data transmission using a small notification packet. After a "lag time", the node will recheck the busy durations again before transmitting the DATA packet. Aloha-AN is shown to achieve higher and more stable throughput than Aloha-CA, because each node can maintain the other nodes' status more accurately. Nevertheless, Aloha-AN's performance decreases significantly when implemented in multi-hop networks. In a multi-hop network, Aloha-AN can no longer maintain the other nodes' status accurately due to the presence of hidden terminals, which result in high collisions.

In terrestrial wireless networks, handshaking-based protocols are common. In such protocols, a node schedules its transmissions according to the control packets (e.g., request-to-send (RTS)/clear-to-send (CTS)) it hears. These control packets also notify other neighbors about the ongoing transmission, including the hidden nodes, and could reduce collisions significantly. Recently, some handshaking-based MAC protocols have been proposed for underwater networks as well. For example, Guo *et al.* introduce the propagation-delay-tolerant collision avoidance protocol (PCAP) in [5]. It requires clock synchronization between the neighboring nodes, just like those in [6]. In order to improve channel utilization, it allows a sender to perform other actions during the long wait between the RTS and CTS frames. Although its maximum throughput is 20%, which is higher than what the conventional handshaking protocol can achieve in underwater, this is merely comparable to Aloha's throughput. Molins and Stojanovic propose in [7] a slotted random access MAC protocol, which, yet again, requires clock synchronization. It is also handshaking-based, but an RTS or CTS frame can only be transmitted at the beginning of each time slot. Although the protocol achieves

guaranteed collision avoidance for its data packets, the long slot length requirement and the handshaking mechanism itself affect the throughput.

In this work, we propose an asynchronous random access MAC protocol, namely, MACA-MN (MACA with packet train for Multiple Neighbors). The protocol utilizes a handshaking-based approach in order to help avoid collisions and alleviate the hidden terminal problem in multi-hop underwater networks. In addition, the MACA-MN can overcome the low throughput problem suffered by typical handshaking-based protocols (such as MACA), by transmitting a train of packets during each round of handshake. Although the idea of packet train has been used in several MAC proposals such as in [7] and [8], our work goes one step further as the packet train is actually formed for multiple neighboring nodes simultaneously. In addition, Unlike the work proposed in [5] and [7], our MACA-MN does not require any synchronization.

The remainder of this paper is organized as follows. In Section II, we explain our proposed protocol in detail. Next, Section III describes the simulations that were carried out to compare the performance of the proposed scheme with several others, and provides further discussions. Finally, we give our conclusions in Section IV.

II. The Proposed Protocol: MACA-MN

A. How the Protocol Works

In this section, we explain how the MACA-MN protocol works. Table I shows the notations that are used, while Fig. 1 illustrates how the handshake is carried out. Similar to the widely known MACA protocol, we employ a three-way handshake (RTS/CTS/DATA). A node that wishes to transmit its data packets will first initiate a handshake to its intended neighbor(s) by broadcasting an RTS packet. However, in contrast to the MACA protocol, our RTS packet can simultaneously request for DATA transmission to multiple neighbors. Specifically, the RTS packet contains the receiver's node ID, the number of DATA packets the sender wishes to transmit to each of its neighbors, as well as the inter-node propagation delay from the sender to its intended receivers. In static networks, each node may obtain the inter-node propagation delay in the initialization phase during which synchronization can be assumed. For example, each node can broadcast small packets with its node ID and timestamp, which can then be used by other nodes to calculate the inter-node propagation delay by comparing the timestamp with a node's local clock. For mobile networks, if the individual nodes are aware of their location coordinates, they can exchange such information, which can then be used to calculate the inter-node propagation delay.

When an intended receiver hears the RTS packet, it will respond with a CTS packet, provided that it is currently not involved in a handshake with another node, and is also not required to remain silent. Note that there is an important modification on how the neighbors should respond with their CTS packets compared with the original MACA protocol, because our MACA-MN needs to handle more than one CTS packet from multiple neighbors. The rule of thumb here is that it will transmit the CTS packet immediately upon receiving the RTS packet, subject to the condition that the CTS packet will not result in a collision with another node's CTS packet at the sender, which may happen when two or more nodes have similar distances from the sender. A collision of CTS packets is costly for the MACA-MN, because it will leave those requested DATA transmission time slots idle, thus leading to low throughput. Fortunately, such collisions can be easily avoided using the inter-node propagation delay information provided by the sender. The solution to this problem is, after a node calculates and learns that if transmitting a CTS packet immediately will cause collision with an earlier CTS packet sent by another node, it will defer sending its CTS packet to the next earliest possible time instead. This concept is illustrated in Fig. 1. As can be seen, if both neighboring nodes #2 and #3 respond with their CTS packets immediately upon hearing the RTS packet, their CTS packets will collide. When such a situation arises, the conflict can be resolved by granting priority to the node that has shorter propagation delay from the sender, and to the node with the smaller node ID when the propagation delays are equal. Here, neighboring node #3 defers transmitting its CTS packet, and ensures that its CTS packet will only arrive at the sender after neighboring node #2's CTS packet has been completely received. Having resolved the time to transmit its CTS packet, each neighboring node will also compute the busy duration at the sender that will be caused by all the CTS packets sent from the sender's neighbors. The end of this busy duration is denoted by t_{busy} (see Fig. 1), which is the time at which the receiver finishes receiving the CTS packet sent from its most distant neighbor.

If a neighboring node y is not one of the intended receivers as indicated in the RTS packet, after computing t_{busy}, it determines $t_{\text{silent},y}$ using (1). The node then avoids transmitting control packets until $t_{\text{silent},y}$ is over, in order to allow the sender to finish receiving the CTS packets from multiple neighbors.

$$t_{\text{silent},y} = t_{\text{busy}} - D_y \tag{1}$$

TABLE I
Notations Used for Explaining the MACA-MN Protocol.

Notation	Description
n	Total number of receivers in the current handshake
t_{busy}	Time at which sender finishes receiving last CTS packet
$t_{\text{silent},x}$	Time until which Node x must remain silent
$t_{\text{rx},x(j)}$	Time at which DATA's 1st bit would arrive at receiver $x(j)$; DATA packets are sent to $x(1)$, $x(2)$, …, $x(n)$ in sequence
$t_{\text{out},x}$	Time at which Node x releases itself from current handshake
c_x	Total number of DATA packets to be sent to Node x
Dx	Inter-node propagation delay between Node x & sender
D_{max}	Maximum inter-node propagation delay
Dx, y	Inter-node propagation delay between Node x & Node y
M_{train}	Minimum packet threshold to trigger RTS transmission
T_{max}	Maximum time threshold to trigger RTS transmission
T_{DATA}	Transmission time of each fixed-length DATA packet

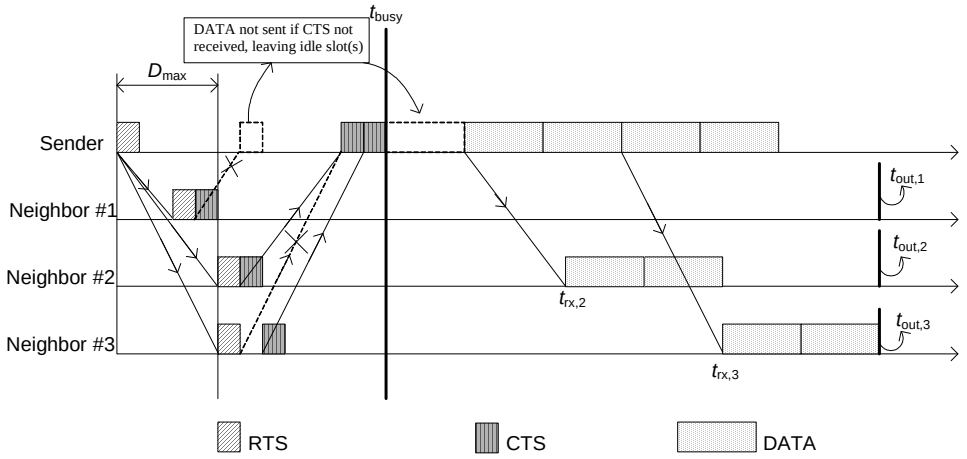

Fig. 1. The three-way handshake in MACA-MN.

On the other hand, if Node $x(j)$ is the j^{th} intended receiver node, it uses t_{busy} to calculate $t_{\text{rx},x(j)}$ and $t_{\text{out},x(j)}$, which are the expected arrival time of the DATA packet intended for itself, and the time to release itself from the current handshake, respectively, using the following equations:

$$t_{\text{rx},x(j)} = t_{\text{busy}} + D_{x(j)} + \sum_{i=1}^{j-1} c_{x(i)} \cdot T_{\text{DATA}} \tag{2}$$

$$t_{\text{out},x(j)} = t_{\text{busy}} + \sum_{i=1}^{n} c_{x(i)} \cdot T_{\text{DATA}} + D_{\max} \tag{3}$$

Once $t_{\text{rx},x(j)}$ is obtained, the intended receiver can go to sleep and wake up just to receive the DATA packets intended for itself. Note that $t_{\text{out},x(j)}$ must be large enough to allow all receivers to finish receiving their DATA packets in the current handshaking loop; any new transmission from this node beyond the timeout will not interfere with any other receiver in this handshaking loop.

Each intended receiver, Node $x(j)$, shall attach its $t_{\text{out},x(j)}$ to its CTS packet, in order to notify its timeout to the two-hop neighbors (hidden terminals) of the sender. At time t_{busy}, the sender starts its DATA transmission to those receivers that have responded with their CTS packets, starting from the node with the least inter-node propagation delay, Node $x(1)$. In cases where the sender has requested to send DATA packets to some neighbors, but it has not heard their CTS replies (either because of packet corruption, or because those neighbors have decided not to respond to the sender's RTS), the sender still keeps the order of DATA transmission by leaving the DATA slots for those unheard neighbors idle, as shown in Fig. 1. Here, it is assumed that the CTS from Neighbor #1 fails to reach the sender. Thus, the sender, which is assumed to have requested to transmit one DATA packet to Neighbor #1, will leave the first DATA slot idle. By doing so, the computation in (2) is still valid even if there is any missing CTS. Another important point to note here is that, a node always gives higher priority to those packets that it is relaying, over its own newly generated packets. This is because the relay packets have already consumed channel resources to reach this far, and it would be wasteful if they are discarded. This strategy tends to achieve higher network throughput.

When a non-participating node y overhears a CTS packet sent by node x, it updates its silent period, $t_{\text{silent},y}$, using the parameter $t_{\text{out},x}$ found within the received CTS packet:

$$t_{\text{silent},y} = t_{\text{out},x} - D_{x,y}. \tag{4}$$

This silent period ensures that node y will not transmit any packet that would collide with the data reception at node x.

After a node releases itself from the current handshake, it is required to back off before initiating another handshake. The randomly chosen backoff interval allows the node to overhear other nodes' statuses, which could help avoid collisions.

B. When to Trigger RTS Attempts

In MACA-MN, a node uses two independent parameters to trigger its RTS attempts, namely, $T_{\max}$ and M_{train}. If a node has not initiated the RTS packet for a time duration $T_{\max}$ from the time it last finished transmitting a DATA packet, the node starts attempting to transmit an RTS packet. However, if a node has accumulated at least M_{train} DATA packets, it will also trigger its RTS attempts. In the high load case, if there are more than M_{train} DATA packets awaiting for transmission, the sender is allowed to transmit only M_{train} packets in each handshake. The purpose of limiting the packet train size to M_{train} is to help the network maintain fairness and shorter delay. Note that even after the triggering condition for making RTS attempts has been met, the node still has to ensure that it is not currently involved in any other handshake, and that it is not supposed to remain silent at this time. Otherwise, it will defer its RTS transmission to the first instant it can start doing so.

C. Backoff Algorithm

When a node needs to be backed off, it randomly chooses a slot from a constant window size, and then multiplies this value with the maximum propagation delay. Here, we use a constant window size instead of the popular binary exponential

backoff (BEB) algorithm. In BEB, the window size normally starts with a small value (e.g., 1), which only enlarges when the expected CTS packet is not returned. Thus, the window size usually tends to be large when the network load is high, during which there are a lot of contentions. In the case of our MACA-MN, a node tends to initiate a handshake with multiple neighbors simultaneously when the network load is high. Thus, the probability of not receiving CTS from at least one neighbor is quite rare, and the window size would not increase if BEB were used. This could lead to low throughput. Therefore, we have chosen to use a constant window size here.

III. Simulations and Results

A. Simulation Model

Our simulation model consists of 36 static sensor nodes arranged in a grid topology, with a grid spacing of 700 m. However, instead of precisely placing each node at a grid intersection point, we introduce some degree of randomness by allowing each node to deviate from the grid intersection point by a maximum of 10% of its grid spacing, in both the vertical and horizontal directions. The transmission range is such that each node will have exactly eight neighboring nodes. Note that we have adopted the wraparound strategy, to eliminate boundary effects. For routing, we focus on the effects of two-hop routes on throughput and packet collision performance. For each packet that is generated by a node, we randomly pick its target destination as any of the node's 16 two-hop neighbors with equal probability. None of its single-hop neighbors can be picked as a target destination. Also, we apply static routing, and distribute the routes evenly. All nodes are equipped with half-duplex and omnidirectional modems which operate at a fixed data rate and a communication range of 2400 bps and 1225 m, respectively.

In the simulations, we assume that the traffic load is divided evenly among all the nodes according to the Poisson distribution. The acoustic propagation speed used in this performance study is 1500 m/s. The channel is also assumed to be error-free, so that all packet losses are purely due to the MAC protocol's performance. We also do not implement ACK for any of the schemes studied in the simulations, thus there is no retransmission for lost packets. All control packets (i.e., CTS, RTS) have the same size of 100 bits, while all DATA packets are 2400-bit long. The buffer size for both new packets and relayed packets are set to 100 each, and the constant window size for backoff is set to 32. We choose to benchmark our protocol with two other previously proposed schemes, namely, Aloha-AN [4] and MACA [9]. For both of these benchmarking protocols, we set the control packet length (i.e., NTF packet for Aloha-AN, and RTS/CTS packets for MACA) to 64 bits, and the maximum window size of BEB in MACA is 64, while keeping all other parameters the same. Note that all the protocols in our simulation study are random access MAC protocols that do not require any time synchronization.

B. Simulation Results

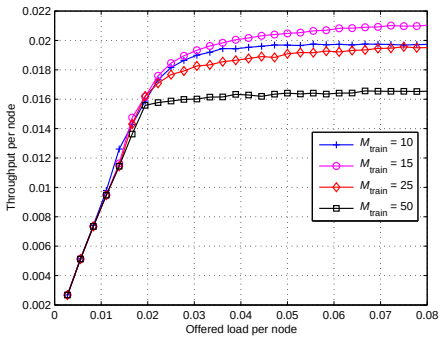

Fig. 2. The effect of different M_{train} when T_{max} = 10 s.

1) Factors Affecting MACA-MN's Performance: As mentioned previously, there are two parameters used to trigger the handshake attempts, namely, T_{max} and M_{train}. The varying of the parameter T_{max} does not significantly change the maximum throughput (simulation results not shown due to limited space), because T_{max} is only responsible for triggering handshakes when the network is operating at low load. As the load shifts to the high load region, the M_{train} parameter plays the dominant role in triggering the RTS attempts, and eventually determines the maximum throughput. Thus, the M_{train} parameter should be carefully chosen, by considering the following factors: the maximum propagation delay, and the buffer size available inside each node.

Although it seems that we might be able to improve the throughput by increasing the size of the M_{train} parameter, this may not always be true, as shown in Fig. 2, due to two reasons. Firstly, recall that a sender should prioritize relay packets over its own new packets in order to achieve good throughput. Suppose the M_{train} parameter is unbounded, the throughput performance would actually degrade because a node will then always attempt to transmit both new and relay packets in each handshake, without prioritizing the latter. Thus, a smaller M_{train} parameter would allow a node to transmit more relay packets than new packets when the network load is high. In addition to the priority issue, a large M_{train} would also make it more difficult to maintain the network's fairness. Secondly, the M_{train} parameter must be adjusted according to a node's available buffer size. A suitable value of M_{train} for a particular buffer size depends on the total number of immediate neighbors that a node has. For a higher number of neighbors, the M_{train} parameter should be smaller, so that each node would participate in multiple handshakes while acting as a receiver, prior to taking the role of a sender. As a guideline, we propose that the total transmission time of M_{train} packets must be larger than the maximum propagation delay, while the value of M_{train} should be limited to the available buffer size divided by the total number of immediate neighbors.

2) Performance Comparison Against Aloha-AN and MACA: As can be seen in Fig. 3, MACA-MN always outperforms both Aloha-AN and MACA significantly, while being able to maintain a stable throughput at high load. Among the three schemes, MACA performs the worst in terms of maximum

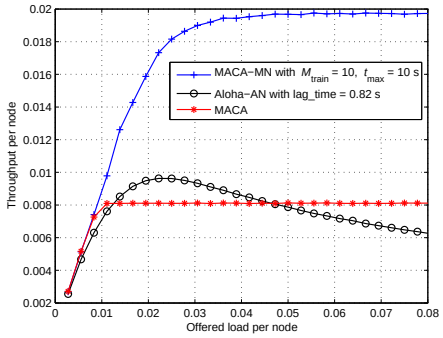

Fig. 3. Throughput comparison with MACA and Aloha-AN.

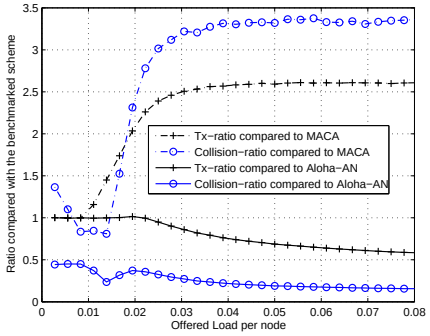

Fig. 4. Tx- and Collision-ratios compared to MACA and Aloha-AN.

throughput. Although Aloha-AN can achieve higher maximum throughput when compared to MACA, its throughput actually decreases as the load increases, due to the presence of hidden terminals in multi-hop networks. For a better understanding of the MACA-MN's good throughput performance, let us look at the Transmission-ratio (Tx-ratio) and Collision-ratio [4] shown in Fig. 4. When compared to MACA, MACA-MN has a higher number (i.e., Tx-ratio $>$ 1) of DATA transmissions, due to its use of packet train for multiple neighbors. By transmitting multiple packets in each handshake, the long propagation delay has less detrimental effect on the network as fewer handshakes need to be activated. We can also notice that the number of collisions in MACA-MN is higher than that of MACA (Collision-ratio $>$ 1). This is due to the fact that the hidden terminal problem in MACA-MN cannot be completely resolved. When a collision occurs while transmitting a train of DATA packets in MACA-MN, the number of packets corrupted tends to be higher than that of MACA (1 packet) in each handshake, thus leading to higher number of collisions. Nevertheless, the MACA-MN's good channel utilization still outweighs its losses due to the hidden terminal problem, which explains its good throughput. In addition to the throughput enhancement, Fig. 5 shows that MACA-MN can also help alleviate the large delay encountered in MACA.

When compared to Aloha-AN, MACA-MN transmits less DATA packets, and achieves fewer collisions. This indicates that MACA-MN encounters less hidden and exposed terminal problems compared to Aloha-AN. Being better at restraining itself from transmitting DATA packets that would otherwise result in collision, the MACA-MN is more energy efficient

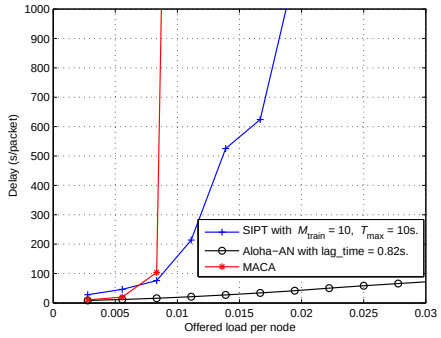

Fig. 5. Delay comparison with MACA and Aloha-AN.

than Aloha-AN. In terms of packet delay, Aloha-AN has the lowest delay among all three schemes, because it only utilizes a one-way notification mechanism. However, its throughput suffers seriously because its one-way notification mechanism does nothing to alleviate the hidden terminal problem.

IV. Conclusion

In this paper, we propose an asynchronous random access handshaking-based protocol for multi-hop underwater networks, namely MACA-MN. Besides adopting the widely known three-way handshake, it features the simultaneous transmission of a train of packets to multiple neighbors, which significantly alleviates the detrimental effect of long propagation delay on network throughput. The MACA-MN is shown to achieve high and stable throughput. This throughput enhancement can be attributed to two main reasons: the channel's utilization improvement resulting from the use of long packet train, and the reduction of the hidden-terminal problem in multi-hop networks. In order to achieve a high maximum throughput, the M_{train} parameter must be carefully chosen, by considering the total number of immediate neighbors of each node, and the node's buffer size.

References

[1] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Elsevier's Journal of Ad Hoc Networks*, vol. 3, no. 3, 2005, pp. 257–279.

[2] M. Stojanovic, "Recent Advances In Hi-Speed Underwater Acoustic Communications," in *IEEE J. Oceanic Eng.*, vol. 21, 1996, pp. 125-36.

[3] D. B. Kilfoyle and A. B. Baggeroer, "The state of the Art In Underwater Acoustic Telemetry," in *IEEE J. Oceanic Eng.*, vol. 25, 2000, pp. 4-27.

[4] N. Chirdchoo, W. S. Soh, K. C. Chua, "Aloha-based MAC Protocols with Collision Avoidance for Underwater Acoustic Networks," in *Proc. IEEE INFOCOM 2007*, May 2007.

[5] X. Guo, M. R. Frater, and M. J. Ryan, "A propagation-delay-tolerant collision avoidance protocol for underwater acoustic sensor networks," in *Proc. MTS/IEEE OCEANS'06*, 2006.

[6] I. P. Morns, O. R. Hinton, A. E. Adams, and B. S. Sharif, "Protocol for sub-sea communication networks," in *Proc. MTS/IEEE OCEANS'97*, 1997, pp. 2076–2082.

[7] M. Molins, M. Stojanovic, "Slotted FAMA: a MAC protocol for underwater acoustic networks," in *Proc. MTS/IEEE OCEANS'06*, 2006.

[8] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proc. IEEE INFOCOM 2002*, Jun. 2002.

[9] P. Karn, "MACA-a new channel access method for packet radio," in *Proc. ARRL/CRRL*, 22 Sept, 1990.