RIPT: A Receiver-initiated Reservation-based Protocol for Underwater Acoustic Networks

Nitthita Chirdchoo, *Student Member, IEEE,* Wee-Seng Soh, *Member, IEEE,* and Kee Chaing Chua, *Member, IEEE*

***Abstract*— Although there are many MAC protocols that have been proposed for terrestrial wireless networks with a wide variety of aspects, these protocols cannot be applied directly in underwater acoustic networks due to the channel's uniqueness of having low data rate and long propagation delay. In order to achieve a high throughput, both characteristics must be taken into account in the MAC design. We propose a random access MAC protocol for multi-hop underwater acoustic networks based on receiver reservation, which we shall call the "Receiver-initiated Packet Train" (RIPT) protocol. It is a handshaking-based protocol that addresses the channel's long propagation delay characteristic by utilizing receiver-initiated reservations, as well as by coordinating packets from multiple neighboring nodes to arrive in a packet train manner at the receiver. Our simulation results have confirmed that the RIPT protocol can achieve our goal of having high and stable throughput performance while maintaining low collision rate.**

***Index Terms*— Underwater acoustic communication, Underwater acoustic telemetry, Access protocols, Communication systems, Data communication, Multiaccess communication, Simulation.**

I. Introduction

The issue of medium access control (MAC) has been widely studied for terrestrial wireless networks, especially ad-hoc networks, due to many interesting applications that involve wireless sensor networks. Ad-hoc networks are useful in applications where there is no infrastructure, and no centralized control. In order to cover the whole area of interest, the size of the network may grow large and become a multi-hop network. It is widely known that in such networks, hidden and exposed terminal problems are the main causes of low throughput. The hidden-terminal problem causes high collision rate, while the exposed-terminal problem causes a node to become over-conservative when transmitting packets. These problems tend to result in under-utilization of the channel.

In terrestrial wireless networks, there are two main approaches in MAC protocol designs to alleviate the abovementioned problems. The first approach is the use of a busy signal to inform the hidden node about an ongoing transmission. Upon hearing the busy signal, the hidden node will avoid accessing the channel until the busy signal ends. This could help resolve the hidden and exposed terminal problems to some extent, depending on the variation of those techniques that fall within this category. Examples of such protocols are BTMA [1], RI-BTMA [2], and DBTMA [3]. In order to utilize the busy signal approach, every node needs to be equipped with more than one transceiver. This may not be feasible for some applications such as sensor networks, in which cost is a major concern. The second approach, on the other hand, uses a handshaking mechanism to reduce the hidden and exposed terminal problems without requiring any additional hardware. "Handshaking" refers to the exchange of multiple small control packets prior to transmitting a longer data packet. This approach has been studied extensively, and many ad-hoc MAC protocols are designed based on this idea. MACA [4] was the first MAC protocol that uses the handshaking mechanism. Some other examples of handshaking-based MAC protocols are MACAW [5], MACA-BI [6] and the widely used IEEE 802.11 protocol.

While there are many MAC protocols proposed to-date addressing the hidden and exposed-terminal problems for terrestrial multi-hop wireless networks, none of them are directly applicable to multi-hop underwater acoustic networks. This is because these terrestrial MAC protocols are designed for high speed radio communication, and they typically assume that the propagation delay is negligible. In contrast, underwater communication mainly uses acoustic channel with a low propagation speed of approximately 1500 m/s, thus resulting in significantly longer propagation delay. Another unique characteristic of underwater acoustic channel that affects the MAC protocol's performance is its narrow available bandwidth, which leads to low data rate. Specifically, the amount of available bandwidth depends on the communication range; a long-range system that operates over several tens of kilometers may have a bandwidth of only a few kilohertz, while a short-range system operating over several tens of meters may have more than a hundred kilohertz of bandwidth [7].

Both of the abovementioned characteristics of the acoustic channel, namely, long propagation delay and narrow available bandwidth, are the key factors that prevent the direct application of the terrestrial MAC protocols in underwater. The narrow available bandwidth implies that it may not be practical to set aside a separate frequency band for transmitting busy signals. The long propagation delay, on the other hand, makes it very expensive to transmit multiple control packets (e.g., RTS/CTS frames) before *every* data packet transmission. In fact, both [8] and [9] have shown that such a technique offers a lower throughput than the well-known Aloha protocol when applied in underwater acoustic networks.

The high latency overhead introduced by the control packets of handshaking-based protocols implies that the channel's

N. Chirdchoo, W.-S. Soh, and K. C. Chua are with the Department of Electrical & Computer Engineering, National University of Singapore (E-mail: {g0500102, weeseng, eleckc}@nus.edu.sg).

Manuscript received March 1, 2008; revised July 15, 2008. The research reported in this paper was supported by the Ministry of Education of Singapore, AcRF Tier 1 funding, under Grant No. R-263-000-370-112.

utilization may be improved if multiple data packets in the form of a packet train can be transmitted for every set of handshake. This is one of the key motivations underlying our proposed MAC protocol, which we call the "Receiver-initiated Packet Train" (RIPT) protocol. While the RIPT protocol is also handshaking-based, the key difference here is that the reservations are receiver-initiated. As will be explained in Section III later on, the use of receiver-initiated reservations is crucial in reducing data packet collisions in the presence of long propagation delays. Another novel concept of the RIPT protocol is that the "packet train" that arrives at the receiver after each set of handshaking is actually formed by transmissions from multiple neighboring nodes. This is built on the assumption that every node knows the inter-node propagation delay between itself and each of its immediate neighbors, so that it can schedule its transmissions accordingly to ensure that a packet train can be formed at the receiver. This design results in high channel utilization, as well as low data packet collisions.

The remainder of this paper is organized as follows. In Section II, we describe briefly some related work in MAC protocol design for underwater acoustic networks. We then present in Section III the RIPT protocol that we propose for underwater networks with distributed topology. Section IV describes the simulations that were carried out to compare the performance of the proposed scheme with several others. In Section V, we provide further insights into our proposed scheme, and finally, we give our conclusions in Section VI.

II. Related Work

Currently, the research efforts in underwater MAC protocols are still in their infancy stage. Some work in the literature, such as [10], has adopted a centralized control approach, which requires a master node to configure the data scheduling, and pass the control messages to its slaves. On the other hand, the distributed control approach, in which each node decides on its own whether to send out a packet, appears to be more attractive. This is largely due to the latter's advantages, such as scalability, faster response to topology changes, smaller number of control messages, as well as immunity from node (master node) failure. In [11], Rodoplu and Park propose a MAC protocol that achieves energy efficiency by reducing collisions. The protocol illustrates a novel idea of how an underwater network can achieve a locally synchronized schedule without acquiring the absolute-timing information. Specifically, each node schedules by itself the time to transmit the next packet randomly, and broadcasts this information by attaching it to the current data packet along with its transmission cycle. Upon hearing the broadcast, the other nodes will know when to wake up for the subsequent packet, and they may go to sleep at other times. However, in order to operate at a low collision rate, each node requires a small duty cycle, which makes it difficult to achieve high throughput.

In [12], Morns *et al.* propose two scheduling protocols to control data packet transmission and arrival times. One protocol is based on Code Division Multiple Access (CDMA), while the other one is based on Time Division Multiple Access (TDMA). However, both protocols require clock synchronization between all the nodes. Also, the time slot allocation for individual nodes becomes hard to manage when the number of nodes grow. Guo *et al.* introduce the propagation-delay-tolerant collision avoidance protocol (PCAP) in [8], which is a handshaking-based protocol. It also requires clock synchronization between neighboring nodes, just like those in [12]. Besides the requirement of request-to-send (RTS) and clear-to-send (CTS) frames, 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 typically, it is merely comparable to Aloha's throughput. Molins and Stojanovic propose in [13] 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.

Recently, Aloha-based protocols have also gained the interest of underwater network researchers due to their simplicity. The throughput analysis of Aloha in underwater is recently presented in [14] and [15]. In [16], we proposed two Aloha-based MAC protocols, namely, Aloha with collision avoidance (Aloha-CA) and Aloha with advance notification (Aloha-AN). Both protocols are capable of exploiting the acoustic channel's long propagation delay, leading to superior throughput performance and lower collision rates when compared with other Aloha-based variants. In Aloha-CA, when a node overhears a packet's header, it calculates the busy durations of all its neighbors that would result from this packet. While the idea is simple, it is shown to improve throughput moderately. The Aloha-AN, on the other hand, transmits a notification packet (NTF) prior to each data packet. The NTF packet helps the node's neighbors to avoid transmitting packets that would collide with the impending packet at their respective receivers. This helps Aloha-AN achieve high and stable throughput, reaching maxima of approximately 32% and 66% over a 2400 bps channel in a single-hop network, when the packet sizes are 2400 bits and 9600 bits, respectively. Note that, although both Aloha-CA and Aloha-AN work well in single-hop networks, they do not adequately address the hidden and exposed terminal problems in multi-hop networks.

III. The Receiver-initiated Packet Train (RIPT) Protocol

A. Overview

Although the RTS/CTS mechanism is widely used for alleviating the hidden and exposed terminal problems in terrestrial multi-hop networks, they suffer from two main drawbacks when they are applied in underwater acoustic networks. Firstly, the need for at least one full round-trip exchange of control packets prior to sending every data packet introduces considerable latency, due to the long propagation delay. This leads to under-utilization of the channel, and low throughput. Secondly, the long propagation delay also seriously impacts the ability

of the RTS/CTS handshake mechanism to resolve the hidden terminal problem, because it now takes much longer for a node to receive RTS and CTS packets from its neighbors, which extends the vulnerable period. This leads to higher collision rate, and again, low throughput. For the first drawback, some previously proposed protocols [13], [17] attempt to increase the channel utilization by sending a train of packets after each successful handshake. Note that the packet train concept has also been proposed for terrestrial wireless networks in [18]. For the second drawback, it appears that receiver-initiated reservations are better at avoiding collisions in the presence of long propagation delay, since the receiver has accurate information on its own current state.

The important observations above lead us to propose the RIPT protocol. While it also seeks to alleviate the hidden and exposed terminal problems through a handshaking mechanism, it does a better job at avoiding collisions by utilizing receiver-initiated reservations. In order to improve channel utilization, we propose the idea of "*multiple-node polling*", in which multiple nodes are allowed to transmit data packets to a single receiver within each round of handshake. By assuming that every node knows the propagation delays between itself and its neighboring nodes, the transmissions can be scheduled in such a way that the data packets will be received by the receiver in the form of a packet train. This is different from the packet train approach in [13], [17], [18], in which the train of packets that a receiver receives are sent by a single transmitter.

We now give more insights into the RIPT protocol's design. As discussed in [15], the performance of high latency networks, such as underwater networks, is affected by both space and time uncertainty. The space uncertainty is caused by the nodes' locations, which result in different propagation delays, while the time uncertainty is caused by the randomness of packet arrivals. The RIPT overcomes the space uncertainty by carefully scheduling the data packet transmissions to avoid collision at the receiver, using the knowledge of inter-node propagation delays. For the time uncertainty, we realize that those techniques widely used in terrestrial MAC (e.g., synchronizing transmission, sensing channel [15]) are not appropriate in underwater since they only remove the uncertainty at the transmitter, but not at the receiver. Moreover, unlike other transmitter-initiated protocols that encounter two types of data packet collision, namely, "transmit-receive collision" and "receive-receive collision", a receiver-initiated approach only experiences receive-receive collision. This is because a receiver knows exactly when the current handshake will end, and how long it should defer its own transmission in order to avoid a transmit-receive collision. The above reasonings explain why we have adopted a receiver-initiated approach in designing the RIPT protocol. Note that transmit-receive collision refers to the scenario whereby an incoming packet arrives at a node while it is transmitting. In this case, the incoming packet will not be heard. On the other hand, receive-receive collision occurs when two or more packets arrive at a receiver simultaneously, causing all packets to be corrupted.

Instead of the typical 3-way (RTS/CTS/DATA) handshake found in protocols such as MACA, the RIPT protocol utilizes a receiver-initiated 4-way (RTR/SIZE/ORDER/DATA) handshake. The RTR (Ready-To-Receive) packet serves to inform all of the initiating receiver's neighbors that the receiver node is ready to act as a receiver for a certain duration of time. A series of SIZE packets will then be sent by the receiver's neighbors to inform the receiver about the number of packets that each neighbor wishes to send to it. The receiver then sends an ORDER packet, which informs its neighbors the relative order to transmit their data packets, and how many packets they are allowed to transmit. Finally, the respective neighbors transmit their DATA packets.

TABLE I
NOTATIONS USED FOR EXPLAINING THE RIPT PROTOCOL.

Notation	Description
t_j	Time at which the neighbor of order j finishes receiving the RTR packet
$t_{\text{SIZE},j}$	Time at which the neighbor of order j starts transmitting its SIZE packet
t_{busy}	Time at which receiver finishes receiving last SIZE packet
$t_{\text{out,rcv}}$	Timeout at receiver
$t_{\text{out},x}$	Timeout at node x
$t_{\text{tx},x}$	Time at which node x starts transmitting DATA packet
$t_{\text{rx},x}$	Time at which node x's DATA packet first arrives at receiver
t_{rx}	Time at which first DATA packet within the packet train may arrive at receiver
M_{train}	Number of DATA slots currently reserved at receiver
$M_{\text{train,max}}$	Maximum allowable value for M_{train}
N_{b}	Number of broadcast packets (if any) from the receiver
$N_{\text{slots},i}$	Number of DATA slots allocated to the i^{th} node to transmit
D_x	Propagation delay between node x and the receiver
$D_{x(j)}$	Propagation delay between the receiver and node $x(j)$ which has order j
D_{max}	Maximum D_x among all first-hop neighbors of the receiver
$D_{x,y}$	Propagation delay between node x and node y
n	Average number of first-hop neighbors per node
n_{hidden}	Average number of hidden terminals per node
T_{RTR}	Transmission time of each fixed-length RTR packet
T_{ORDER}	Transmission time of each fixed-length ORDER packet
T_{SIZE}	Transmission time of each fixed-length SIZE packet
T_{DATA}	Transmission time of each fixed-length DATA packet
T_{avg}	Average time interval between initiating RTRs at a node
T_{guard}	Guard time to protect against any estimation error in the inter-node propagation delays

B. How the Protocol Works

As mentioned earlier, the RIPT protocol requires every node to know the inter-node propagation delay between itself and each of its immediate neighbors. Therefore, the protocol works best in either a static network, or one with limited mobility but every node can determine its own position. For static networks, such inter-node propagation delays can be estimated during initialization, in which all nodes take turns to broadcast some control packets to its neighbors. Upon hearing such a packet from one of its neighbors, a node can calculate its propagation delay by comparing the timestamp on the packet with its local clock. Although this procedure requires time synchronization

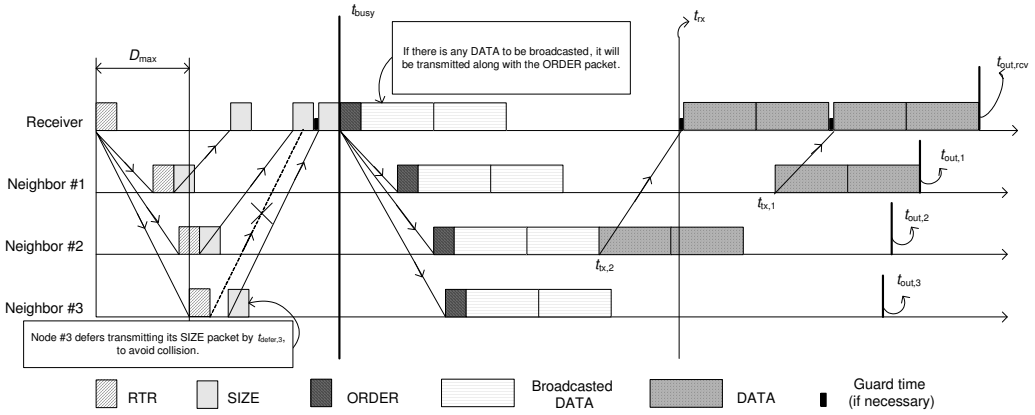

Fig. 1. 4-way handshaking with multiple-node polling.

among all the nodes, the assumption is quite reasonable because the initialization stage is short, and thus any clock drift will be negligible if the synchronization is carried out right before deployment. Note that the RIPT protocol no longer requires time synchronization beyond the initialization stage. For the case where each node has positioning capability, messages could be exchanged between neighboring nodes to update each other about their locations, which can then be used for computing the inter-node propagation delays.

We now explain how the RIPT protocol works. Table I shows the notations that are used, while Fig. 1 shows an example of how the 4-way handshake is carried out.

1) 4-way handshake initiation by the RTR packet: when an idle node wishes to become a receiver, it initiates the 4-way handshake by broadcasting an RTR packet. In order to avoid any confusion, we shall clarify that the terms "receiver", and "senders", refer to the initiating node that intends to be a receiver, and its immediate neighbors that have packets to send to it, respectively. The RTR packet contains the initiating receiver's node ID, the number of DATA slots reserved at the receiver (M_{train}), and the inter-node propagation delay from itself to each of its neighbors, if necessary; for the case of a static network, the inter-node propagation delay information can be exchanged during initialization, and does not need to be retransmitted with every RTR packet. Note that these inter-node propagation delays will be used by each neighboring node to compute the time at which it needs to send its SIZE packet, which will be explained later in Section III-B.2. In order to accommodate the need to broadcast DATA packets, the RTR packet also includes a flag to indicate whether the receiver has any DATA packet to broadcast, as well as a field that indicates the total number of DATA packets it will broadcast (N_{b}). As mentioned earlier, the RTR packet serves to inform all of the receiver's neighbors that the former is ready to act as a receiver for a certain duration of time.

2) Transmission slot request using the SIZE packet: when a neighboring node hears the RTR packet, it needs to respond with a SIZE packet. The rule of thumb is to transmit the SIZE packet immediately upon receiving the RTR packet, subject to the condition that it will not collide with another node's SIZE packet at the receiver. Any such collision will be costly because the receiver will not allocate any DATA slot to a neighboring node if it does not hear the latter's SIZE packet, and will result in low throughput. Fortunately, such collisions can be easily avoided if the inter-node propagation delay between the receiver and each of its neighbors are known to all of these neighbors. Note that the overhead incurred to maintain this information is of the order of $O(n^2)$ per node (where n is the average number of first-hop neighbors per node), if it is statically maintained at each node. If this information is provided by the RTR packet instead, then the overhead is in the order of $O(n)$. The information allows each neighboring node to compute the time at which it is supposed to transmit its SIZE packet without colliding with other SIZE packets at the receiver. The node first arranges the inter-node propagation delays between the receiver and each of the neighboring nodes in ascending order. If there are multiple nodes having the same propagation delay, the conflict is resolved by granting priority to the node with the smaller node ID. Suppose the node finds that it has the order j, and t_j is the time at which it finishes receiving the RTR packet. The time at which it should transmit its SIZE packet is given by

$$t_{\text{SIZE},j} = \max[t_j, (t_{\text{SIZE},j-1} + D_{x(j-1)} + T_{\text{guard}} + T_{\text{SIZE}} - D_{x(j)})], \quad (1)$$

where $t_{\text{SIZE},1} = t_1$, $D_{x(j)}$ is the propagation delay between the receiver and the neighboring node $x(j)$ that has order j, and T_{guard} is a small guard time that can be inserted to protect against any estimation error in the inter-node propagation delays. We will discuss more about how the value of T_{guard} may be chosen in Section V, but it should be noted for now that the amount of T_{guard} required is usually very small compared to the DATA packet's transmission time.

In order to better understand the above algorithms, we shall look at the example in Fig. 1. As can be seen, if both neighboring nodes #2 and #3 respond with their SIZE packets immediately upon hearing the RTR packet, their SIZE packets will collide at the receiver. Here, neighboring node #3 defers transmitting its SIZE packet, so as to ensure that it will only arrive at the receiver after neighboring node #2's SIZE packet has been completely received.

Having resolved the time to transmit its SIZE packet, the neighboring node will also compute the busy duration at the receiver that will be caused by all the SIZE packets sent from

the receiver'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 SIZE packet sent from its most distant neighbor of order n, where n is the number of first-hop neighbors that the receiver has. Every first-hop neighboring node can then calculate t_{busy} locally as follows:

$$t_{\text{busy}} = t_{\text{SIZE},n} + D_{x(n)} + T_{\text{SIZE}}. \tag{2}$$

The value of t_{busy} will then be used to compute $t_{\text{out,rcv}}$, which is the time at which the receiver is expected to finish receiving the entire packet train. Specifically,

$$t_{\text{out,rcv}} = t_{\text{busy}} + 2D_{\max} + T_{\text{ORDER}} + (N_{\text{b}} \cdot T_{\text{DATA}}) + (M_{\text{train}} \cdot T_{\text{DATA}}) + n \cdot T_{\text{guard}}. \tag{3}$$

We now describe the information contained within a neighboring node's SIZE packet. It contains the number of relay DATA packets, as well as the number of its own DATA packets that it wishes to transmit to the receiver. It also contains its own timeout, calculated as

$$t_{\text{out},x} = t_{\text{out,rcv}} - D_x, \tag{4}$$

where D_x is the propagation delay between the receiver and the node itself. Note that $t_{\text{out},x}$ is the timeout that node x sets to release itself from the current handshaking loop. This timeout needs to be large enough to allow the receiver to finish receiving all the DATA packets in the current handshaking loop. However, it does not need to be as large as $t_{\text{out,rcv}}$, because of the propagation delay D_x between the receiver and itself. It simply needs to be large enough such that any transmission from this node beyond the timeout will not interfere with the receiver.

The SIZE packet serves two purposes. Besides informing the receiver about the number of relay and new DATA packets to be transmitted, it also informs each of the receiver's second-hop neighbors (i.e., its hidden nodes) to avoid initiating an RTR handshake until a certain timeout. For a second-hop neighbor (say, node y), upon receiving the SIZE packet sent by the first-hop neighbor (say, node x), its timeout is

$$t_{\text{out},y} = t_{\text{out},x} + D_{x,\,y}, \tag{5}$$

where $D_{x,\,y}$ is the propagation delay between node x and node y. The need for the second-hop neighbors to avoid becoming receivers is key for achieving a stable throughput. If the second-hop neighbors were to act as receivers, they may lose some DATA packets due to collisions arising from the first-hop neighbors' transmissions.

It is also important to note that the RIPT protocol still functions properly even when some of the receiver's neighbors miss the RTR broadcast. When such a case arises, the only impact on RIPT is that the particular neighbor will not respond with a SIZE packet, and subsequently, it will not be allocated any DATA slot for the current round of handshake.

3) Transmission order broadcast through the ORDER packet: after the receiver has acquired all the SIZE packets from its neighbors, it allocates its available DATA slots (i.e., M_{train}) using a simple strategy. The rule of thumb is to prioritize all relay DATA packets over new DATA packets, because the relay packets have already consumed channel resources to reach the intermediate nodes, and it would be wasteful if they were to be discarded due to buffer overflow. We now explain the assignment strategy using the example shown in Table II, where $M_{\text{train}} = 4$. First, each of the neighboring nodes is assigned a unique priority randomly. Their requirements are then sorted according to decreasing node priority. Next, the receiver runs through the "relay packets" column according to the node priorities, and accommodate as many relay packets as possible. If there are still available DATA slots after considering the relay packets, the node will then run through the "new packets" column, and assign the remaining DATA slots accordingly. Upon completing the slot assignment, the receiver then transmits the ORDER packet, which contains the total number of DATA slots assigned to each neighboring node, the order of transmission, the broadcast flag, as well as the number of DATA packets to broadcast. Notice that the receiver resends the information on broadcast packets so as to improve the chances of its neighbors to be ready for them. Immediately after transmitting the ORDER packet, the receiver transmits its broadcast packets, if any.

TABLE II

AN EXAMPLE ILLUSTRATING THE SLOT ASSIGNMENT STRATEGY, WHERE $M_{\text{TRAIN}} = 4$.

Priority	Node ID	Relay packets	New packets	Slots assigned
1	Neighbor #2	0	3	2
2	Neighbor #1	2	1	2
3	Neighbor #3	0	3	0
		Total number of slots assigned		4

4) DATA train transmission: upon hearing the ORDER packet, a node that has been allocated at least one DATA slot must compute the time at which it shall start its DATA transmission, so that its packets will form a packet train at the receiver with the other senders' packets. Note that the transmission start time must take into account the propagation delay to the receiver. For instance, if node x's DATA packet is expected to reach the receiver at time $t_{\text{rx},x}$, it shall start transmitting the packet at

$$t_{\text{tx},x} = t_{\text{rx},x} - D_x. \tag{6}$$

Suppose node x is assigned by the receiver as the l^{th} node to transmit, we can obtain $t_{\text{rx},x}$ as

$$t_{\text{rx},x} = t_{\text{busy}} + 2D_{\max} + T_{\text{ORDER}} + (N_{\text{b}} \cdot T_{\text{DATA}}) + (l-1) \cdot T_{\text{guard}} + \sum_{i=1}^{l-1} N_{\text{slots},i} \cdot T_{\text{DATA}}, \tag{7}$$

where $N_{\text{slots},i}$ is the number of slots allocated to the i^{th} node.

C. Adaptive Train Size

In actual implementation, the packet train size M_{train} for each handshaking loop should not be held constant, because the offered load would fluctuate with time. When the load is low, only a few neighbors may have DATA packets to transmit, or, in the worst case, no neighbor has any DATA packet to

transmit. On the other hand, when the load is high, many neighbors may wish to transmit DATA packets. A self-adaptive algorithm would allow each node to adapt the M_{train} parameter according to the current load observed. In particular, the total number of DATA packets that all neighbors wish to transmit during the current handshaking loop could be used to predict a suitable M_{train} value for the future handshaking loop.

We now describe a possible approach as follows. If the receiver finds that its M_{train} is not large enough to accommodate all the slot requests from its neighbors, it increases M_{train} by 2 for the next round. If it finds that there are insufficient slot requests to fill up its M_{train}, it decreases M_{train} by 1. The main reason why the algorithm is more conservative when decreasing M_{train} is due to the relative reliability of the above two triggers. To understand this, we need to be aware that the sum of slot requests computed from those SIZE packets that it receives does not always reflect the true number of packets that its neighbors wish to send. The inaccuracy may arise because some neighbors' SIZE packets might have been corrupted, or it may be because some neighbors are required to remain silent as they are currently involved in other handshaking loops. Although the computed sum may not be accurate, if it happens to be higher than the current M_{train}, there is no ambiguity that the current M_{train} is indeed too small. On the other hand, if the sum is less than M_{train}, the receiver cannot be sure whether its M_{train} is indeed too large, because there might be missing information. Note that the change in M_{train} only affects the next round of handshake. Also, there should be a maximum limit for M_{train}, so as to avoid any receiver from capturing the channel for too long.

D. When to Initiate an RTR Packet

Because the RIPT protocol requires a node that wishes to act as a receiver to initiate the handshaking loop by broadcasting an RTR packet, the timing of initiating RTR packets is an important issue. Although a traffic prediction scheme might be useful for helping a node to schedule the proper time to initiate the handshaking loop, it is beyond the scope of our study. Here, we simply pick the exponential distribution for the time between RTR-initiations, with an average of T_{avg}.

In order to avoid the same node from acting as a receiver successively before other neighboring nodes have a chance at playing the role, we make use of a "fairness bit" at each node. If a node has just been released from a handshaking loop while acting as a receiver, it will set this bit to '0'. While in this state, it will not initiate any RTR packet. The fairness bit can only be reset once the node has served as a sender in any subsequent handshaking loop. However, if the node's fairness bit has been set to '0' for longer than a threshold time t_{limit}, it will reset the fairness bit back to '1' to avoid any deadlock.

IV. Simulations and Results

A. Simulation Model

Our simulation model consists of 36 static nodes arranged in a grid topology, as shown in Fig. 2. However, instead of precisely placing each node at a grid intersection point, we introduce some degree of randomness by allowing each node

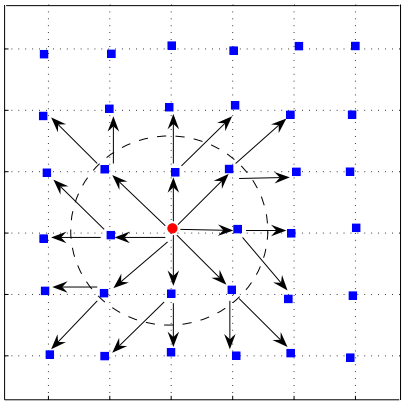

Fig. 2. Our simulation network topology. Note that the nodes are not placed precisely at the grid intersection points. Also, the arrows in the figure only show the routes between a single node and its 16 two-hop neighbors.

to deviate from the grid intersection point by a maximum of 10% of its grid spacing, in both the x and y directions. The deviations from the grid intersection points are introduced here in order to ensure that the network topology resembles a real scenario, whereby the nodes are usually non-equidistantly placed. Note, however, that the RIPT still works even if the neighboring nodes are equidistant. The transmission range of each node is assumed to be 1.75 times the grid spacing, such that each node has exactly eight neighbors within its range. In order to avoid edge effects, we have adopted the wraparound strategy, such that even the nodes at the boundaries will have eight one-hop neighbors. Note that, in a real scenario where edge effects exist, we expect the throughput to be higher than our simulation results; this is because the nodes at the network edge usually have lower number of hidden and exposed nodes, thus resulting in lower number of collisions.

We assume that the traffic load is divided evenly among all nodes according to the Poisson distribution. For routing, in order to make it easier to interpret the results, we consider two-hop routes only, rather than varying number of hops. For each packet that is generated by a node, we randomly pick its destination to be any of the node's 16 two-hop neighbors with equal probability. Also, we apply static routing here. The arrows in Fig. 2 show the routes originating from *one* particular node (the round node) to each of its 16 two-hop neighbors. We do not show the two-hop routes for each of the remaining 35 nodes when they behave as source nodes, but their two-hop routes have exactly the same pattern.

We also assume that all the nodes are equipped with half-duplex, omnidirectional modems, with a fixed data rate of 2400 bps. The acoustic propagation speed is assumed to be 1500 m/s. Since the RIPT is designed to perform independently of the physical layer, and our simulation study only focuses on the RIPT's performance in the MAC layer, we do not specify the modulation scheme used. Here, 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 simulated, thus there is no retransmission for lost packets. All control packets (i.e., RTR, SIZE, and ORDER) have the same size of 100 bits,

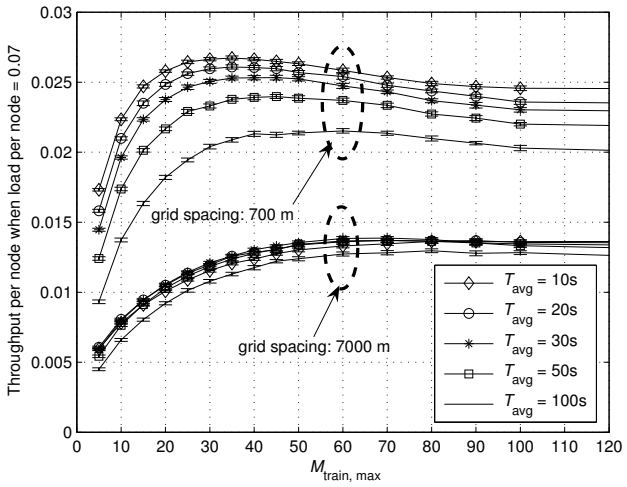

Fig. 3. The effect of $M_{\text{train,max}}$ and T_{avg} on throughput.

while DATA packets are 2400-bit long. The buffer size for both new packets and relayed packets are set to 100 each, and the parameter M_{train} is initialized to 1. We choose to benchmark our protocol with two previously proposed schemes, namely, Aloha-AN [16] and MACA [4]. For both of these protocols, we set the control packet length (i.e., NTF packet for Aloha-AN, and RTS/CTS packets for MACA) to 64 bits, 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. We also investigate both 700 m and 7000 m grid spacings to evaluate the performance of the protocols under different average propagation delays.

B. Simulation Results

The simulation duration for each data point was 8×10^5 s, so that the 95% confidence interval for any throughput per node is within ±0.0002 from its sample mean. Here, we adopt the definition of "throughput" from [9], and define "throughput per node" as the average throughput over 36 nodes as follows:

$$\text{Throughput per node} = \frac{1}{36}\left[\frac{\text{No. of Packets Received/Simulation Time}}{\text{Data Rate/Packet Length}}\right] \quad (8)$$

1) Factors Affecting the RIPT's Performance:

- **$M_{\text{train,max}}$ and T_{avg}:** Fig. 3 shows how the parameters $M_{\text{train,max}}$ and T_{avg} affect the RIPT's throughput when the offered load per node is 0.07. Note that this is the offered load that is high enough to cause the RIPT's throughput to saturate. From the figure, we can observe that when T_{avg} becomes large, the throughput actually decreases. Ideally, T_{avg} should be as small as possible, in order to reduce the packet delays. For the case where the grid spacing is 700 m, we observe that the suitable range of $M_{\text{train,max}}$ varies with T_{avg}. For example, when T_{avg} is 10 s, the suitable range of $M_{\text{train,max}}$ would be approximately [25,40]. However, when T_{avg} increases to 100 s, the suitable range of $M_{\text{train,max}}$ also increases to [35,70]. This is intuitive since a larger T_{avg} would imply that there are more DATA packets waiting to be transmitted in each handshake. When $M_{\text{train,max}}$ is outside these suitable ranges, we see that the RIPT's throughput deteriorates. Furthermore, for any T_{avg}, the throughput initially increases as $M_{\text{train,max}}$ increases, but begins to decrease when $M_{\text{train,max}}$ is too large. This can be explained as follows. If $M_{\text{train,max}}$ is too small, the network actually spends more time exchanging control packets rather than transmitting DATA packets, which results in low throughput. On the other hand, if $M_{\text{train,max}}$ is too large, the throughput may also be low due to the higher chances of collisions, and also due to more unutilized reserved slots. Note that, despite using a receiver-initiated handshaking approach, the RIPT still encounters collisions, just like other transmitter-initiated handshaking approaches. Collisions can occur between the various combinations of control packets and DATA packets. When a node transmits longer train of packets, there is a higher chance that it may miss the control packets from its other neighbors, thus losing the opportunity to keep an accurate view of its neighbors' status (e.g., when they are acting as receivers); this in turn reduces the node's capability to avoid collisions. In addition, if a neighboring node misses an RTR packet, it will not be able to transmit DATA packets to the receiver, even if it has many packets to send; this may result in unutilized reserved slots at the receiver. Another point worth mentioning about $M_{\text{train,max}}$ is its effect on packet delay. Although not shown here, we have found via simulations that the packet delay tends to increase when $M_{\text{train,max}}$ becomes larger.
- **Inter-node Propagation Delay:** When the grid spacing is increased from 700 m to 7000 m, we can see from Fig. 3 that the throughput becomes less sensitive to the variation in T_{avg}. However, changes in $M_{\text{train,max}}$ still produce significant changes in the throughput. The suitable ranges of $M_{\text{train,max}}$ have also increased compared to the previous case. This is intuitive since more DATA packets should be transmitted in each round of handshake in the presence of longer propagation delay, in order to stay efficient. It should also be mentioned that the overall throughput has also dropped significantly compared to the previous case. This is a common observation among handshaking-based MAC protocols.
- **Packet Length:** We have also performed simulations for the case where the packet length is increased to 4800 bits. However, we do not show the results here because similar conclusions can be made as above. Nevertheless, it should be mentioned that a larger DATA packet length is observed to reduce the effects of $M_{\text{train,max}}$ on throughput.

2) Performance Comparison Against Aloha-AN and MACA: For a better understanding of the RIPT protocol's performance when compared against Aloha-AN and MACA, we use four metrics as our performance measure, as follows:

- **Throughput:** As can be seen in Fig. 4, the RIPT outperforms both MACA and Aloha-AN significantly as the load increases. The throughput of Aloha-AN, when implemented in the current multi-hop network setting,

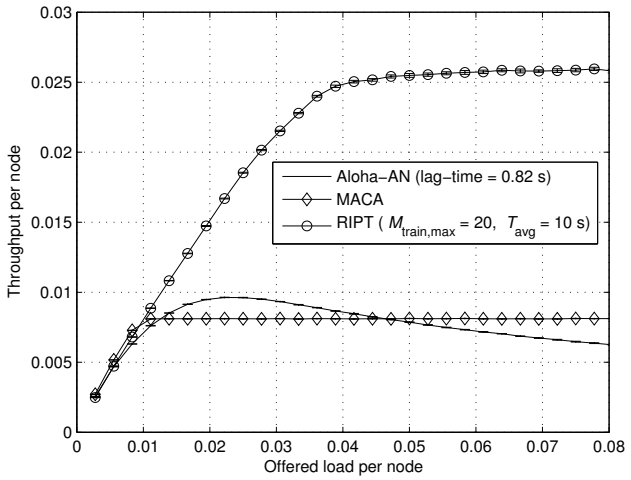

Fig. 4. Comparing the throughput of RIPT, MACA, and Aloha-AN (grid spacing = 700 m).

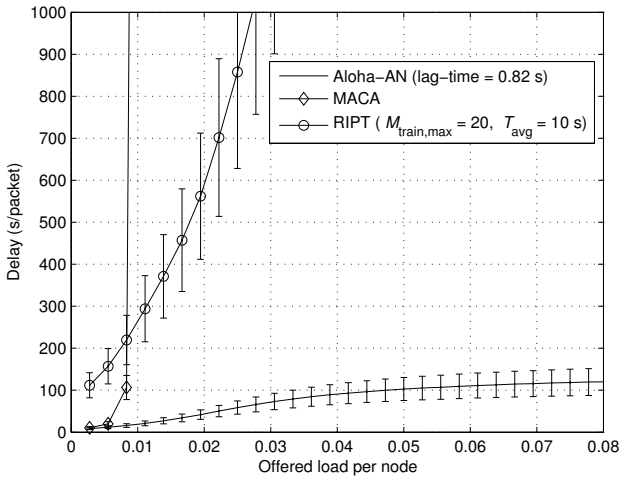

Fig. 6. Comparing the packet delays of RIPT, MACA, and Aloha-AN.

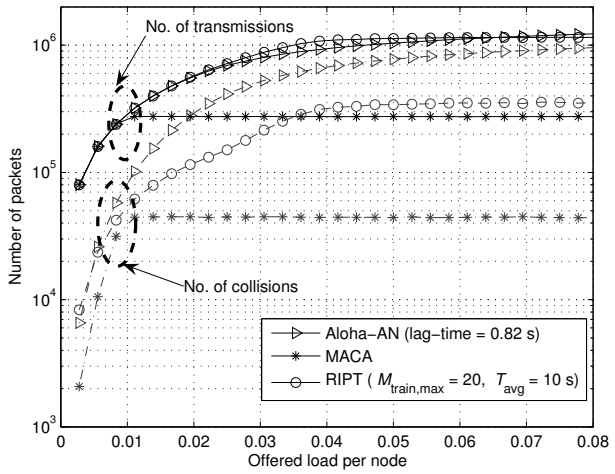

Fig. 5. Comparing the number of DATA packet transmissions and the number of DATA packet collisions of RIPT, MACA, and Aloha-AN. Note that the simulation duration is 8×10^5 s for every point.

becomes lower than that of MACA at high load, although the latter is designed for terrestrial networks. This is because Aloha-AN does not address the hidden terminal issue in its design, which becomes worse when the load is high. The results obtained from the study of MACA has proven that handshaking-based schemes could help reduce collisions in multi-hop underwater networks by alleviating the hidden terminal problem. It also guarantees a stable throughput at high load. However, as seen in Fig. 4, its throughput is much lower when compared to our RIPT protocol. This is largely due to MACA's inefficiency in underwater since it only transmits a single data packet per round of handshake, which suffers from under-utilization of the channel when the propagation delay is high. In contrast, our RIPT protocol improves channel utilization by forming a packet train at the receiver for each round of handshake.

- **Number of DATA Packet Transmissions and Collisions:** In Fig. 5, the RIPT transmits approximately as many DATA packets as the Aloha-AN, while having less number of collisions. This confirms that the RIPT avoids collisions by maintaining more accurate information about the receiver compared to Aloha-AN. When comparing the RIPT against MACA, the RIPT transmits much more packets than MACA. This arises from our technique of using multiple neighbors to form a packet train at the receiver, which is much more efficient than MACA. Despite being able to offer a high and stable throughput, the RIPT suffers from much higher number of collisions than MACA. Although the RIPT's handshaking-based mechanism can greatly alleviate the hidden terminal problem in multi-hop networks, it cannot resolve the problem completely. Thus, whenever a collision occurs, a large number of DATA packets within a packet train may be corrupted. In contrast, the MACA only transmits a single DATA packet during each round of handshake, and hence, it loses less packets in a collision.
- **Delay:** Fig. 6 shows the delay performance of the three schemes. At very low load (below 0.01), the RIPT has the worst delay performance. This is because of its receiver-initiated approach, whereby a sender needs to wait until there is a handshake initiated by the receiver before it can attempt transmit a DATA packet to the latter. Moreover, its packet train tends to be very short when the load is low, thus making the overhead of its 4-way handshake mechanism more significant. However, beyond a load of 0.01, its delay becomes shorter than the MACA. This is the point where the average packet train size has grown large enough to overcome the overheads incurred by both the 4-way handshake, and the average waiting time for the handshake initiation by the receiver. The Aloha-AN is seen to have the best delay performance among all the three schemes. This is due to the fact that it is not a handshaking-based protocol, and only uses a one-way notification mechanism.

V. Discussion

An important point that we would like to stress about the RIPT protocol is that, a neighboring node can compute the time at which it needs to transmit its SIZE packet and its DATA packet train, based on the time at which it receives the RTR packet; in other words, there is no need for absolute clock synchronization. In a way, this bears some similarity with the MAC protocol proposed in [11], in which the neighboring nodes can achieve a locally synchronized schedule without absolute clock synchronization. Therefore, the purpose of the guard times in the RIPT protocol is merely to buffer any error in the inter-node propagation delays that were previously estimated. These small guard times are required at strategic instances within the 4-way handshake. It is important to note that we do not require the guard time to be inserted between every DATA packet that is sent; it is only inserted between the string of packets that are transmitted from different neighbors. Also, the guard time only needs to be as large as the maximum expected error in the inter-node propagation delay estimation, which may be in the order of tens of milliseconds. Note, also, that it is possible for the RIPT protocol to correct any error in the delay estimates through the following enhancement. A receiver can examine the timings at which the SIZE packets are arriving from its neighbors, and calculate their deviations from the expected arrival times. The receiver can then include the timing corrections the next time it broadcasts an RTR packet to these neighbors.

Earlier in Section IV-B, we have seen that if $M_{\text{train,max}}$ is too large, it may result in low throughput as well. In the following, we provide some guidelines for selecting an appropriate $M_{\text{train,max}}$. As the receiver and the senders need to exchange control packets before the DATA packet train can be transmitted, the inter-node propagation delays thus introduce some amount of fixed-cost (C_{fix}) in each handshake. Assuming that the receiver does not have any packet to broadcast, and also assuming the worst-case scenario whereby all the n first-hop neighbors are located at D_{max} from the receiver, the 4-way handshake incurs a fixed-cost overhead of

$$C_{\text{fix}} = 4D_{\text{max}} + T_{\text{RTR}} + nT_{\text{SIZE}} + T_{\text{ORDER}}. \tag{9}$$

Thus, the total transmission time of the DATA packet train must be longer than C_{fix}, in order to justify this overhead. Besides the above constraint, $M_{\text{train,max}}$ should also be able to accommodate all the DATA packets waiting to be transmitted in each of the receiver's neighbors. Assuming that the network is operating in the high load region, at which all nodes are backlogged. Thus, in each handshake, $M_{\text{train,max}}$ should ideally be large enough to accommodate $(n_{\text{hidden}} + n)$ DATA packets, where n is the average number of first-hop neighbors per node, and n_{hidden} is the average number of hidden terminals per node. Thus, the size of $M_{\text{train,max}}$ can be calculated as

$$M_{\text{train,max}} = \left\lceil \max\left(\frac{C_{\text{fix}}}{T_{\text{DATA}}}, n_{\text{hidden}} + n\right)\right\rceil. \tag{10}$$

So far, we have assumed a scenario whereby all nodes are statically deployed, and set up at the same time. We now discuss how the RIPT can be modified to handle network dynamics caused by new nodes joining an existing network. Suppose a new node, Node y, wishes to be considered by a receiver as one of its possible transmitting neighbors. In order to cope with this, a receiver's 4-way handshake can include an additional listening interval, T_{join}, right after the time at which it expects the last bit of the last SIZE packet, which we denote by $T_{\text{SIZE\_end}}$. In order to declare its presence, Node y first listens to the receiver's RTR packet, and calculates the value of $T_{\text{SIZE\_end}}$ from the inter-node propagation delay information attached within the RTR packet. Next, it transmits a short JOIN packet that will be received at the receiver some time within the T_{join} interval. This can be ensured if T_{join} is larger than the maximum propagation delay between one-hop neighbors, which depends on the transmission range. Since clock synchronization is no longer available, Node y must try to estimate the propagation delay between itself and the receiver as half of the round-trip time (RTT) instead. The RTT can be obtained using a technique that exchanges time-stamped messages (as nicely described in [19]); in our case, the ORDER packet is used to piggyback the time-stamped message in the reverse direction. Once Node y obtains the propagation delay to the receiver, it transmits this information to the receiver during the next round of handshake, to be received within the receiver's T_{join} interval again. It can then be formally included as one of the receiver's neighbors. Note that a receiver does not need to include T_{join} in every handshake; it can use a flag within its RTR packet to indicate whether the current 4-way handshake includes the interval T_{join}. In this way, the receiver can control the overhead incurred, which is a tradeoff with how soon a new node can join as its neighbor.

Although we have omitted the effects of channel packet losses (e.g., due to bit errors) in our discussion so far, we will now discuss them briefly. Note that, a neighboring node can calculate when to transmit its SIZE packet and DATA train, so long as it has received the RTR packet and the ORDER packet correctly. If these packets were corrupted, the neighboring node simply does not get to transmit any DATA train in the current round of handshake, which causes the throughput to drop. Nevertheless, the other neighboring nodes can still proceed without any timing conflict. Conversely, if a receiver does not receive a neighbor's SIZE packet correctly, it will not allocate any DATA slot for that neighbor, which also reduces the throughput. This, however, only has an isolated effect on that neighbor, but not other neighbors.

Overall, the RIPT is suitable for delay-tolerant underwater applications that are required to operate at high load, such as undersea exploration and data collection. In particular, it is efficient for networks in which every node has a large number of neighbors. This is because the RIPT allows multiple neighbors to transmit to a receiver at one go to form a packet train, in contrast to other handshaking-based protocols that would require every neighbor to perform dedicated handshake with the receiver. Furthermore, due to its receiver-initiated handshaking nature, the RIPT would be more appropriate for applications in which the offered load does not fluctuate too rapidly. Otherwise, the system parameter M_{train} may not adapt fast enough, which leads to inefficiency.

Although the RIPT protocol has shown good throughput and delay performance with our current settings, we would like to

comment on some possible future studies as follows:

- **Determining the time between RTR initiations:** Although it is currently chosen to be an exponentially distributed random variable with an average of T_{avg}, the value of M_{train} can potentially help the node determine the suitable time between RTR initiations, as these two parameters have shown some correlation.
- **Performance benchmarking:** It is also interesting to see how the RIPT compares against other MAC protocols that are also designed for underwater multi-hop networks, such as Slotted FAMA [13].

VI. Conclusion

With the use of the acoustic channel in underwater communications, underwater networks are characterized by its long propagation delay and low data rate. Because of these unique properties, the MAC protocols designed for terrestrial networks cannot be directly applied in underwater. In this paper, we have proposed and studied a new MAC protocol for multi-hop underwater acoustic networks – RIPT. The RIPT protocol is a random access handshaking-based protocol that addresses the channel's long propagation delay characteristic by utilizing receiver-initiated reservations, as well as by coordinating packets from multiple neighboring nodes to arrive in a packet train fashion. We have confirmed through simulations that the RIPT can achieve high and stable throughput with proper values of packet train size, M_{train}, as well as average time between handshake initiations, T_{avg}.

References

[1] L. Kleinrock, F. A. Tobagi, "Packet switching in radio channels: Part I-carrier sense multiple access modes and their throughput-delay characteristics," *IEEE Trans. Comun.*, 23, Dec. 1975, pp. 1400-1416.

[2] C. Wu, V. O. K. Li, "Receiver-Initiated Busy-Tone Multiple Access in packet radio networks," in *Proc. ACM SIGCOMM Conferrence*, 1987, pp. 336-342.

[3] Z. J. Hass, J. Deng, "Dual Busy Tone Multiple Access (DBTMA)-a multiple access control scheme for ad hoc networks," *IEEE Trans. Commun.*, 2002, pp. 975-984.

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

[5] V. Bhargavan, A. Demers, S. Shenker, L. Zhang, "MACAW-A Media Access protocol for wireless Lans," in *Proc. ACM SIGCOMM*, 1994, pp. 212-225.

[6] F. Talucci and M. Gerla, "MACA-BI (MACA By Invitation) A wireless MAC protocol for high speed ad hoc networking," in *Proc. ICUPC*, 1997.

[7] 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.

[8] 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.

[9] S. Shahabudeen and M. A. Chitre, "Design of networking protocols for shallow water peer-to-peer acoustic networks," in *Proc. IEEE/OES OCEANS'05 Europe*, 2005.

[10] G. G. Xie and J. H. Gibson, "A network layer protocol for UANs to address propagation delay induced performance limitations," in *Proc. MTS/IEEE OCEANS'01*, 2001, pp. 2087–2094.

[11] V. Rodoplu and M. K. Park, "An energy-efficient MAC protocol for underwater wireless acoustic networks," in *Proc. MTS/IEEE OCEANS'05*, 2005.

[12] 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.

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

[14] L. F. M. Vieira, J. Kong, U. Lee, and M. Gerla, "Analysis of Aloha Protocols for Underwater Acoustic Sensor Networks," in *Proc. ACM WUWNet 2006*, Los Angeles, CA, Sep. 2006.

[15] A. Syed, W. Ye, B. Krishnamachari, and J. Heidemann, "Understanding Spatio-Temporal Uncertainty in Medium Access with ALOHA Protocols," in *Proc. ACM WUWNet 2007*, Montreal, Quebec, Canada, Sep. 2007, pp. 41–48.

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

[17] M. Stojanovic, "Optimization of a Data Link Protocol for an Underwater Acoustic Channel," in *Proc. IEEE OCEANS'05*, June 2005.

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

[19] A. A. Syed and J. Heidemann, "Time Synchronization for High Latency Acoustic Networks," in *Proc. INFOCOM 2006*, April 2006, pp. 1–12.

Nitthita Chirdchoo received the B.Eng (Hons) degree in telecommunication engineering from the Suranaree University of Technology, Thailand, in 1999. In 2000 she was awarded the scholarship by Royal Thai Government to study at University of Pittsburgh, where she obtained her M.Sc. degree in Telecommunication in 2002. Since 2005, she has been pursuing a Ph.D. degree in Electrical and Computer Engineering at National University of Singapore (NUS) under the Research Scholarship granted by NUS. Her research interest includes underwater sensor networks and wireless sensor networks.

Wee-Seng Soh (S'95–M'04) received the B.Eng. (Hons) and M.Eng. degrees in electrical engineering from the National University of Singapore in 1996 and 1998, respectively. In 1998, he was awarded the Overseas Graduate Scholarship by the National University of Singapore to study at Carnegie Mellon University, Pittsburgh, PA, where he received the Ph.D. degree in electrical and computer engineering in 2003.

Since 2004, he has been with the Department of Electrical and Computer Engineering, National University of Singapore (NUS), where he is currently an Assistant Professor. Prior to joining NUS, he was a Postdoctoral Research Fellow in the Electrical Engineering and Computer Science Department, University of Michigan. His research interests are in wireless ad hoc and sensor networks, cellular networks, as well as underwater acoustic networks.

Kee Chaing Chua received a PhD degree in Electrical Engineering from the University of Auckland, New Zealand in 1990. He joined the National University of Singapore (NUS) as a Lecturer in 1990 and is now a Professor in the Department of Electrical & Computer Engineering. From 1995 to 2000, he was seconded to the Center for Wireless Communications (now part of the Institute for Infocomm Research), a national telecommunication R&D centre funded by the Singapore Agency for Science, Technology and Research as its Deputy Director. From 2001 to 2003, he was on leave of absence from NUS to work at Siemens Singapore where he was the founding head of the Mobile Core R&D Department funded by Siemens' ICM Group. He served as the Faculty of Engineering's Vice Dean for Research from 2003 to 2006. From 2006 to March 2008, he was seconded to the National Research Foundation as a Director. He returned to NUS in April 2008 to serve as the Vice Dean for Research in the Faculty of Engineering.

Dr Chua has carried out research in various areas of communication networks and has published more than 200 papers in these areas in international refereed journals and conferences. His current research interests are in wireless networks (in particular wireless sensor networks), underwater networks and optical burst switched networks. He has also been an active member of the Institute of Electrical & Electronics Engineers (IEEE), Inc., and is a recipient of an IEEE 3rd Millennium medal.