Design of a Propagation-Delay-Tolerant MAC Protocol for Underwater Acoustic Sensor Networks

Xiaoxing Guo, Michael R. Frater, *Member, IEEE*, and Michael J. Ryan, *Senior Member, IEEE*

***Abstract*—Underwater acoustic sensor networks (UASNs) can be employed in a vast range of applications, retrieving accurate and up-to-date information from underneath the ocean's surface. Although widely used by terrestrial sensor networks, radio frequencies (RFs) do not propagate well underwater. Therefore, acoustic channels are employed as an alternative to support long-distance and low-power communication in underwater sensor networks even though acoustic signals suffer from long propagation delay and have very limited bandwidth. In this paper, we introduce an adaptive propagation-delay-tolerant collision-avoidance protocol (APCAP) for the media access control (MAC) sublayer of UASN. The protocol includes an improved handshaking mechanism that improves efficiency and throughput in UASN where there is a large propagation delay. The mechanism guarantees nodes that can potentially interfere with a forthcoming transmission are properly informed. It also allows a node to utilize its idle time while waiting for messages to propagate, which is otherwise wasted by most existing MAC protocols. The simulation results indicate that where employed by UASN, APCAP exhibits good performance and outperforms the other MAC protocols examined in this paper.**

***Index Terms*—Acoustic communication, MAC, underwater networks, wireless sensor networks.**

NOMENCLATURE

APCAP	adaptive propagation-delay-tolerant collision-avoidance protocol;
CML	compulsory maximum latency;
CSMA	carrier sense multiple access;
CTS	clear-to-send;
LLC	logical link control;
MAC	media access control;
MLP	MAC level pipelining;
MSDU	MAC service data unit;
NAV	network allocation vector;
PCAP	propagation-delay-tolerant collision-avoidance protocol;
PCAP-CC	PCAP with both CML_{CTS} and CML_{data};
PCAP-RCW	PCAP-CC with RX CTS window;
PCAP-TDW	PCAP-CC with TX DATA window;
RF	radio frequency;
RTS	request-to-send;
TX	transmit/transmission;
UASN	underwater acoustic sensor network;
WSN	wireless sensor network;
RX	receive/reception.

Manuscript received March 26, 2007; revised June 09, 2008; accepted January 06, 2009. First published March 27, 2009; current version published May 13, 2009.

Associate Editor: J. Preisig.

The authors are with the School of Information Technology and Electrical Engineering, University College, the University of New South Wales, Canberra, ACT 2600, Australia (e-mail: x.guo@adfa.edu.au; m.frater@adfa.edu.au; m.ryan@adfa.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JOE.2009.2015164

I. INTRODUCTION

RECENT advances in embedded system and communication technologies have pushed a higher level of functionality into ever-smaller devices capable of sensing and wireless communication. Networks formed by these devices, known as WSNs, have been attracting a tremendous amount of research effort due to their potential in both civil and military domains [1]–[5]. For their proximity to the immediate physical environment, wireless sensors can provide localized measurements and detailed information that are difficult to obtain through traditional instrumentation.

As over 70% of Earth's surface is covered by water, it is desirable to deploy underwater sensor networks to support applications such as oceanic research, disaster recovery, and navigation. Although RF-based WSNs have been widely adopted by onshore applications, RF signals deliver very poor performance underwater, providing transmission ranges of merely a few meters at the typical RF sensor transmission power [6]. Optical communication for underwater sensor networks using light waves has also been investigated [7], however these methods either require high precision or high power if the distances between sensor nodes are large. Consequently, acoustic networks enabled by sound waves become ideal alternatives since acoustic signals propagate well through water and require much less power underwater than RF and light signals for the same communication range.

A UASN is a communications network comprising wireless sensors operating in the acoustic band; and from these sensors, readings of certain physical phenomena can be gathered by one or more sink nodes. As a special type of WSN, a UASN is similar to these terrestrial sensor networks in terms of architecture and functionality. However, acoustic signals travel much slower than RF (approximately 1500 m/s underwater) and have very limited bandwidth, delivering data rates of merely a few kilobits per second. Moreover, UASNs are envisaged to monitor larger areas than those on the ground. As a result, propagation delay becomes significant in UASN. Coupled with low data rates, the

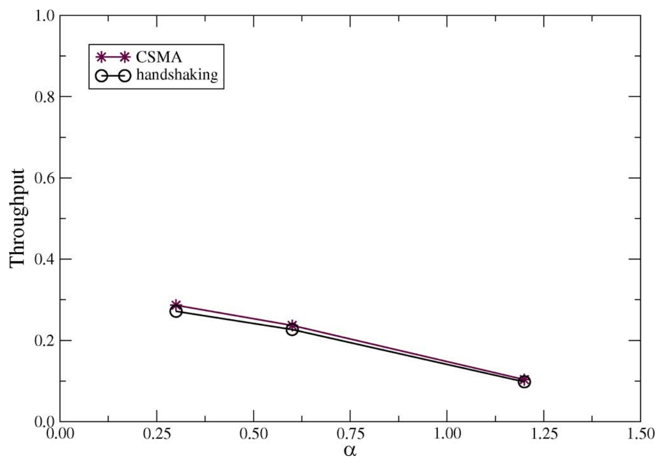

Fig. 1. Throughput versus α of CSMA and conventional handshaking at an offered load of 0.5.

long propagation delay means that UASNs greatly challenge networking concepts developed for their RF-based terrestrial counterparts in which the propagation delay is usually negligible.

In a UASN, nodes share a common multiaccess channel, and each node needs to contend for the channel to transmit a frame into it. Therefore, a MAC protocol is necessary to regulate the behavior of the nodes when they compete for the channel. Since the performance of a network is severely constrained if it lacks a well-defined MAC protocol, the design of MAC protocols with efficient collision-avoidance mechanisms is essential. For a pioneering wireless network system called ALOHAnet, the ALOHA protocol [8] was proposed, which allows nodes to grasp the channel whenever they have data to send. An improvement to the original unslotted ALOHA protocol is slotted ALOHA [9], which divides time into discrete slots and a node can only start transmission at the beginning of a slot. Unlike ALOHA that allows nodes to transmit at will, protocols that make nodes listen to the channel before they access it are called carrier sense multiple access CSMA protocols [10]. CSMA is widely used by both wired networks (e.g., LAN) and wireless networks (e.g., Wi-Fi or ZigBee-compliant networks). With CSMA, a node may only start transmitting if it senses that the channel is free. However, since the transmission range of a node in a wireless network is limited, CSMA is subjected to the hidden terminal problem, which means that carrier sensing cannot indicate the real status of the channel if competing nodes are out of each other's transmission range. A more sophisticated MAC protocol, called MACA, was proposed by Karn [11] to avoid the hidden terminal problem. With MACA, a node wishing to transmit performs handshaking with the destination using RTS and CTS frames before transmitting the actual data frame. As a result, the RTS/CTS frames can provide the source and the destination with channel status and inform the potential interferers of the forthcoming transmission.

Although the aforementioned protocols and the mechanisms they employ are widely adopted by RF-based networks, they become less efficient as the propagation delay increases in a UASN. For example, the performance of CSMA and MACA's RTS/CTS handshaking is severely constrained if the propagation delay is large. Kleinrock and Tobagi [12] analyzed the throughput-delay characteristics of CSMA and indicated that the efficiency of CSMA drops dramatically as the propagation delay grows. Meanwhile, work in [13] also indicates that, with CSMA, the propagation delay of a network considerably affects its performance. In addition, Matsuno *et al.* [14] showed that the performance of a MACA-like protocol that employs RTS/CTS handshaking is also severely constrained by a long propagation delay. The impact of long propagation delay on CSMA and handshaking is presented in Fig. 1, in which the results are obtained from a simulation using the same model as described in Section III with a fixed offered load of 0.5. The offered load and throughput used in Fig. 1 are defined in (10) and (11) in Section III. We also define the ratio of the propagation delay to the frame length as

$$\alpha = \frac{P}{\frac{K}{\text{data rate}}} = \text{data rate}\frac{P}{K} \tag{1}$$

where P is the propagation delay, and K is the size of a frame in bytes.[1] It can be seen from Fig. 1 that the throughput delivered by CSMA decreases as α increases (that is, as the data rate increases in our normalized simulations). This means that for a certain propagation delay, CSMA delivers a higher throughput if nodes transmit at a lower data rate. In the meantime, handshaking delivers even lower throughput than CSMA at the same data rate, though almost negligibly so. This is because handshaking incurs extra RTS and CTS frames before sending a data frame; and worse, when handshaking fails, the time spent on propagating the control frames (i.e., RTS and CTS frames) is wasted. The results shown in Fig. 1 are in accordance with those presented in [12] and [14].

Since MAC protocols devised for RF-based wireless networks cannot provide satisfactory performance where employed by underwater acoustic networks, considerable research effort has been invested to overcome the effects of long propagation delay and its impact on underwater acoustic networks.

Foo *et al.* [15] proposed a MAC protocol for underwater acoustic networks, which is derived from MACA. By introducing a priority system, their protocol deals with deciding the action to be taken when a node sends an RTS frame but receives an RTS frame from a third node. However, lacking the effort to accommodate the propagation delay, their protocol has the same limitation as MACA, which lies in the timing and inadequate information provided by the RTS and CTS frames. Shahabudeen and Chitre [16] later studied the performance of amended classical MAC protocols used by RF wireless networks in an underwater environment. However, except to keep the protocols simple, the work does not attempt to resolve the MAC sublayer issues in underwater acoustic networks by accommodating the propagation delay. In addition, the study does not consider very long propagation delay, and therefore, the protocols are only valid when distances between nodes are short.

Where employed by an underwater acoustic network, the problems encountered by the aforementioned protocols can be

[1] 1 byte = 8 bits

summarized as *the efficiency problem* and *the delivery problem*. The efficiency problem is that a transmitting node is simply idle during the propagation of its own message and the message it is waiting for, whereas the essence of the delivery problem is that RTS and CTS frames cannot reach potential interferers on time and thus are unable to inform them of the forthcoming transmission.

Independent work by Guo *et al.* [17] as well as Molins and Stojanovic [18] both considered using additional delay to deal with the delivery problem. The idea of the protocol in [18], called slotted FAMA, is to set the slot size to equal the maximum single-trip propagation delay that may exist in a network. Unlike slotted FAMA, the approach taken in [17] is to fix the interval between sending a message and receiving the response to the maximum round-trip propagation delay. With the protocol in [17], a transmitting node knows exactly when the response will arrive and may start processing other frames if the interval allows, meaning that it also offers a solution to the efficiency problem. Recent work by Peleato and Stojanovic [19] includes another protocol that deals with both the efficiency and the delivery problems. The idea of the protocol is for a source to add a waiting period between receiving the CTS frame and sending the data frame, and the length of such waiting period may vary depending on the source–destination distance measured during handshaking. In addition, the protocol allows the destination to send a warning frame to the source to cancel a transmission if it receives an RTS frame from a third node. Recently, a new class of MAC protocols, called T-Lohi, has been proposed in [20], where the authors seek to solve the problem faced by CSMA, which is called space-time uncertainty in the literature. Also offering a solution to the delivery problem, T-Lohi is a reservation-based MAC protocol that uses short contention tones to reserve the channel for competing nodes. However, T-Lohi requires a node to be idle and listen to the channel in each contention round when competing for the channel, and the listening period lasts at least the maximum single-trip propagation time plus the time to detect the contention tone. As a result, its throughput is limited to a maximum of 0.66 of the channel capacity.

In addition to those protocols that dynamically allocate channel resource by handshaking, schedule-based protocols have also been widely considered for underwater acoustic networks. For example, Xie and Gibson [21] considered MAC and routing issues in underwater acoustic networks, and proposed a protocol that employs master nodes to handle resource allocation, addition/removal of nodes, routing, etc. It is clear that the employment of master nodes makes sophisticated control possible, but such a scheme does not suit a network whose topology is rapidly changing. Rodoplu and Park [22] also studied MAC sublayer issues in underwater acoustic networks where nodes have low duty cycles, for which they proposed a listening time scheduling scheme. The key idea of the listening time scheduling scheme is to let a transmitting node tell others its own schedule, such that other nodes will wake up at the proper time to listen to it. However, the scheme does not support burst transmission and only works well if each node is stationary and only periodically has data to transmit.

For a UASN that is designed to have simpler nodes than normal underwater acoustic networks, a MAC protocol that dynamically allocates channel resources while taking into account the propagation delay is desired. In this paper, we introduce an adaptive MAC protocol for UASN, which concentrates on accommodating the propagation delay while delivering good network performance. The original contributions of this paper are as follows. First, we briefly show that existing MAC protocols involving handshaking become inefficient where employed by UASN, which is caused by the efficiency and delivery problems. We then propose an adaptive propagation-delay-tolerant collision-avoidance protocol (APCAP), which includes an improved handshaking mechanism that improves throughput in networks with high propagation delay. To solve the efficiency and delivery problems, the new handshaking mechanism 1) guarantees proper delivery of handshaking signals to prevent potential interferers from interfering with a transmission; and 2) allows a node to utilize its idle time while waiting for messages to propagate, which is otherwise wasted by most existing MAC protocols.

II. An Adaptive Propagation-Delay-Tolerant MAC Protocol for UASN

Work in [12]–[14] showed that propagation delay severely constrains the performance of CSMA and handshaking, which are widely adopted by RF-based wireless networks. This means that conventional MAC protocols designed for RF networks are unable to deliver promising performance in a UASN unless nodes are placed very closely. In this section, we discuss how the long propagation delay can be accommodated with respect to avoiding collisions. We then introduce a MAC sublayer collision avoidance protocol for UASN, which performs very well even when the propagation delay is large.

In the following discussion, we consider a UASN that is randomly deployed over a fixed area. Each sensor node may know its own location, but it does not have any knowledge about where another node is, unless sufficient information is provided through exchanging data between the two nodes. For example, a node is able to calculate its distance to another if the latter sends a frame to it and the frame indicates its departure time. In addition, all nodes share a multiaccess channel, where colliding signals from multiple nodes will cause collisions regardless of their strength and signal-to-noise ratios. Moreover, collisions are the only source of frame losses. For each node in a UASN, the MAC sublayer processes MSDUs submitted from the LLC sublayer. An MSDU contains the actual data to be transported by the network in the form of data frames. MSDUs are first queued by MAC and then processed on a first-come–first-served (FCFS) basis.

A. CSMA and Handshaking, Keep, Discard, or Improve?

First, we discuss whether CSMA and handshaking can be adopted by a UASN and whether they need any improvement to fit into a long-propagation-delay environment.

As a reactive collision avoidance mechanism, nodes performing CSMA overhear channel output to conclude whether

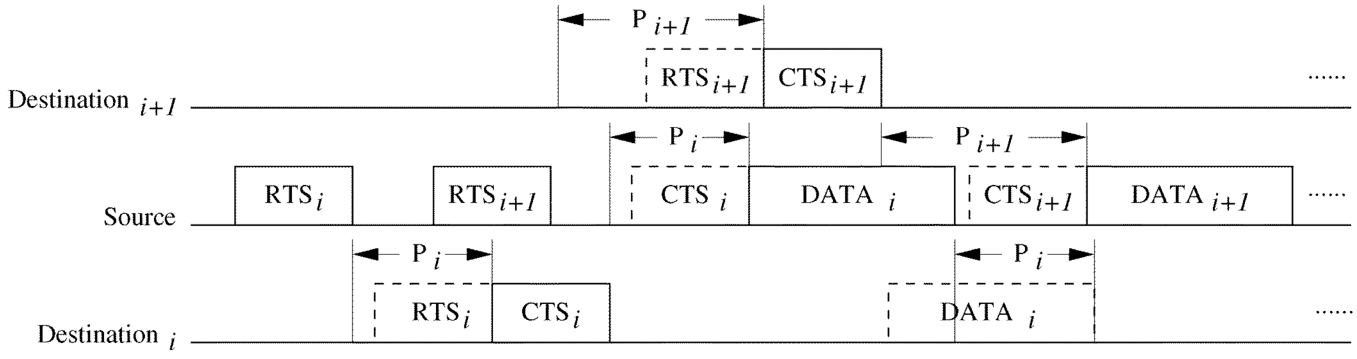

Fig. 2. Illustration of MLP.

they can transmit (bearing in mind that what matters in a network are collisions that occur at the receiver). In a long-propagation-delay UASN, an acoustic signal from one source may arrive at different nodes at different times depending on their distances to the source (we call this property the *asynchronous arrivals* of the acoustic signal). As a result, if a node senses that the channel is busy, it does not necessarily mean that it cannot access it, which is also claimed by the authors of [12] and [23]. For example, if the destination is located much closer to the source than the potential interferer, the reception of a frame may have already been completed at the destination when a potential interferer receives the signal. On the other hand, if a frame takes much longer to reach a potential interferer than to reach the destination, the potential interferer may regard the channel as idle, start transmitting a frame into it, and cause a collision, which is also indicated in [20] as the space-time uncertainty problem. Since the deployment of nodes in a UASN is rather random over a large geographical region, CSMA behaves unpredictably. Since MAC protocols devised for RF-based networks cannot provide satisfactory results where employed by underwater acoustic networks, therefore it would be better to exclude CSMA in a collision avoidance protocol for UASN unless the propagation delays between nodes are only a small fraction of the frame length.

We now examine handshaking. With handshaking, the RTS/CTS frames are transmitted as probes to: 1) notify the destination of the forthcoming transmission; and 2) alert the potential interferers. In fact, handshaking is utilized to reserve channel capacity for data frames using smaller RTS/CTS frames (these frames are referred to as the control frames). A potential interferer should be notified before it can cause a collision. Handshaking works perfectly in an RF network because a radio signal from a source reaches different nodes virtually simultaneously; and the propagation delay in an RF-based network is normally just a few hundredths of a frame length. However, in a UASN, the asynchronous arrivals mean that some interferers may have not received the "notification" by an RTS or a CTS frame while the transmission of a data frame has commenced, which is the delivery problem as mention earlier in this paper. Compared to the problem that CSMA encounters, handshaking can be improved by deliberately delaying a pending response. For example, additional delay can be added that allows a CTS frame to reach all the potential interferers located within the transmission of the destination node (assume the considered UASN only has symmetrical links) before the time the data frame reaches it.

B. An Improved Handshaking Mechanism

Following the above discussion, we expect that a useful collision avoidance protocol would not employ CSMA but would have an improved handshaking mechanism that is tolerant of the propagation delay. We now introduce the improvements and show how they help the protocol to accommodate the impact of the long propagation delay in UASN.

1) MAC Level Pipelining: With conventional handshaking as in [11], the source node transmits an RTS frame, waits for a CTS frame replied by the destination, and then starts transmitting the data frame. Both the RTS and the CTS frames should inform the potential interferers of the forthcoming transmission of the data frame. However, in a long-propagation-delay UASN, if the destination starts transmitting the CTS frame as soon as it receives the RTS frame, and if the source starts transmitting the data frame as soon as it receives the CTS frame, some potential interferers may still cause collisions because either the RTS or the CTS frame may have not reached them in time. A straightforward idea is to: 1) let the destination delay the transmission of the CTS frame when it receives the RTS frame; and 2) let the source delay the transmission of the data frame when it receives the CTS frame. As a consequence, both the RTS and the CTS frames have sufficient time to reach the potential interferers and thus prevent them from interfering with the reception of the corresponding pending frames.

However, since the propagation delay in a UASN is already very high, further delaying frames can make the network even less efficient. Notice that the propagation delay in UASN results in a large time gap between the transmission of a frame and the reception of the response. In addition, the extra delay proposed above makes that gap even larger, like the gap between RTS$_i$ and CTS$_i$ in Fig. 2 at the source. It would then be ideal if a node could utilize these large gaps that are normally wasted by idling by conventional MAC protocols. In other words, a node may transmit frames for the other MSDU in the queue while waiting for a pending response, which means a node may process frames for different MSDU in an interlaced order rather than just sequentially. We call this way of processing MSDU MLP and an example is shown in Fig. 2, where P_i and P_{i+1} are the propagation delays of the source to nodes i and $i+1$. In Fig. 2, the source starts to transmit RTS$_{i+1}$ to destination node $i+1$ after

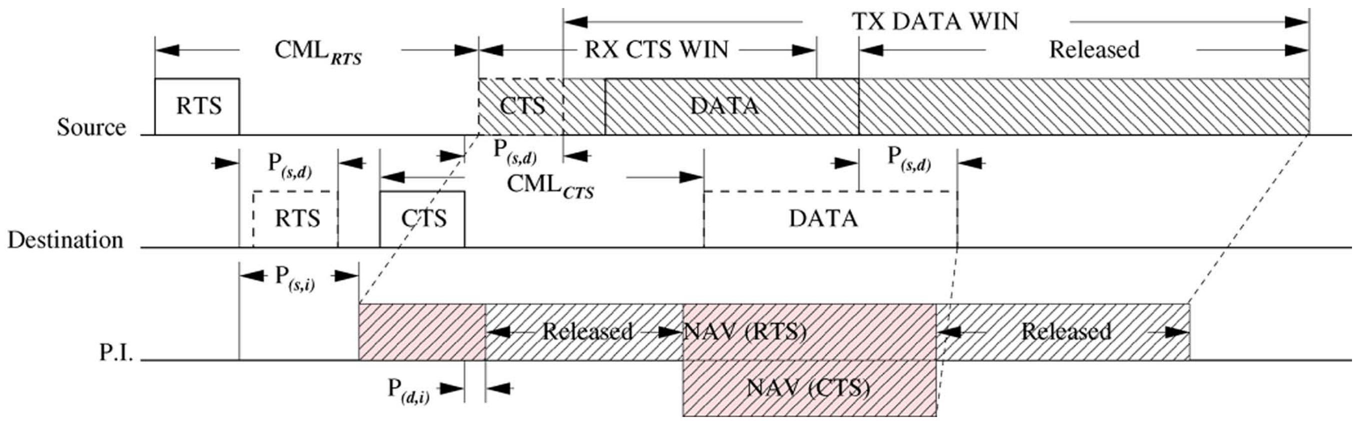

Fig. 3. Timing of APCAP; PI is a potential interferer.

the departure of RTS_i but before the arrival of CTS_i. Notice that in all timing diagrams included in this paper, transmission and reception are represented by solid boxes and dashed boxes, respectively.

In summary, MLP gives a node an opportunity to process multiple MSDU while waiting for a pending response that may take a long time to arrive. Therefore, it offers a solution to the efficiency problem by reducing the time wasted when requiring a node to remain idle while frames are propagating.

2) Compulsory Maximum Latency: With MLP, when frames for different MSDU are processed out of order, it is vital that the gap between a frame and its response is at least equal to the length of the frame to be transmitted/received in it. In other words, if a node starts processing another MSDU while the response of the current one arrives, the node is unable to receive the response correctly. The premise of performing MLP is that the pipelined actions should not clash with each other. The term *action* used here is the process of transmitting or receiving a frame. For example, transmission of an RTS frame (TX RTS) and reception of a CTS frame (RX CTS) are two actions that a source node takes when it performs handshaking.

It can be seen that the colliding actions under MLP are caused by random propagation delay between nodes. For example, P_i and P_{i+1} in Fig. 2 may vary significantly depending on the distances between nodes. Recall that we assumed that a node does not hold the information about its distance to an arbitrary peer unless it receives a frame from it. Therefore, the node is unable to know whether the space between a frame and its consequent frame[2] is large enough to allow other frames to be transmitted/received. A straightforward idea to avoid clashing actions is to force a node to delay its response such that the response is received by the waiting node after it has completed another action in the gap.

It must be noted that extra delay is also required to deliver the control frames to alert all the potential interferers before they may cause collisions. Consequently, the extra delay should also result in a gap of fixed length between a control frame and its consequent frame; and such fixed length gaps should: 1) allow the control frame to reach all the potential interferers within range; and 2) give the node performing MLP sufficient time to transmit/receive other frames. More precisely, the extra delay is to be mandatorily added: 1) at the source between transmitting the RTS frame and receiving the corresponding CTS frame; and 2) at the destination between transmitting the CTS frame and receiving the data frame.

The fixed length gap between a control frame and its consequent frame is called CML, and is illustrated in Fig. 3, where the gap at the source between TX RTS and RX CTS is called $\mathrm{CML}_{\mathrm{RTS}}$, and that at the destination between TX CTS and RX DATA is called $\mathrm{CML}_{\mathrm{CTS}}$. We assume that the propagation delay is solely accountable for the network latency, and CML is designated to allow the RTS and CTS frames, both transmitted at the maximum power level, to reach all interferers located within the transmission ranges of the source and the destination. If a node does not know the distance it is from another, the worst case scenario should be considered. The physical specifications of sensor nodes as well as the environmental factors determine that a node in a UASN should only be able to communicate reliably with another within a certain range if the maximum power level is used. Let $R_{\max}$ denote the maximum transmission range over all nodes in a UASN, then the maximum (possible) single-trip propagation delay $P_{\max}$ is $P_{\max} = R_{\max}/v$, where v is the lowest speed of sound waves travelling underwater, which is measured over the region of the UASN. Then, with respect to this worst case scenario, we define the length of a CML space as $\mathrm{CML}_{\mathrm{RTS}} = \mathrm{CML}_{\mathrm{CTS}} = 2P_{\max}$. However, under certain scenarios when precise $P_{\max}$ is impossible to obtain, CML can be set to a value that is larger than any possible $2P_{\max}$ in reality although this may impact setup delay, but not throughput.

Since we assumed that nodes do not have any global topology knowledge, the key to enabling CML is accurate time keeping, which allows a node to measure the distance to another node. Each node in the same UASN then maintains a clock providing the node with an absolute time reference. A control frame should carry its departure time, hence upon reception of this frame, a recipient is able to calculate its distance to the transmitter by

[2]For a particular MSDU, a consequent frame is a frame that is the immediate result of another frame. For example, from the source's point of view, the (reception of the) CTS frame is the consequent frame of (the transmission of the) RTS frame. Likewise, the data frame is the consequent frame of the CTS frame.

comparing the time stamp against its local clock.[3] The source and destination nodes then run CML as follows.

At the time a source node transmits an RTS frame, it assumes that the destination is located at the boundary of $R_{\max}$. As a consequence, it would expect a responded CTS to arrive after $2P_{\max}$, i.e., a $\mathrm{CML}_{\mathrm{RTS}}$ space as shown in Fig. 3. In the meantime, the RTS frame reaches all the potential interferers located closer than $R_{\max}$ and sets the network allocation vectors (NAV) of these nodes, which indicate the time they can or cannot access the channel. After receiving the RTS frame, the destination then uses the distance information measured from the RTS frame to calculate the time to reply with a CTS frame such that the CTS frame reaches the source after a $\mathrm{CML}_{\mathrm{RTS}}$ space counted from the departure of the RTS as shown in Fig. 3.

Likewise, the destination also runs a $\mathrm{CML}_{\mathrm{CTS}}$ to deliver the CTS frame to potential interferers before they may cause a collision. The destination, in replying to the source with a CTS frame, also proposes a time for the source to transmit the data frame such that it arrives after a $\mathrm{CML}_{\mathrm{CTS}}$ space counted from the departure of the CTS frame, which is also shown in Fig. 3. By utilizing the distance information measured from the CTS frame, a potential interferer then sets its NAV to avoid colliding with the reception of the data frame at the destination.

In summary, CML solves the delivery problem brought by the asynchronous arrivals in a UASN: $\mathrm{CML}_{\mathrm{RTS}}$ guarantees that any potential interferer located within the transmission range of the source will not interfere with the reception of the CTS and the data frames, while $\mathrm{CML}_{\mathrm{CTS}}$ allows the CTS frame to inform the potential interferers in the range of the destination that they should not transmit while the data frame is being received. Although timing error has an impact on CML, synchronization schemes designed for high-latency networks such as [24] can be employed to offset such an impact and maintain the correctness of CML.

3) RX CTS Window: According to CML, the CTS frame for an MSDU should arrive at the source node after a $\mathrm{CML}_{\mathrm{RTS}}$ space when the RTS frame starts being transmitted. In other words, CML specifies that the interval between the departure of an RTS frame and the arrival of its corresponding CTS frame should be equal to $\mathrm{CML}_{\mathrm{RTS}}$. It is clear that, in this case, the receive time for the CTS frame is solely proposed by the source without taking the availability of the destination into account. However, upon reception of the RTS frame, the destination may not be available to respond if it happens to transmit/receive a different frame at the RX CTS time proposed by the source node.

As a result, the source node should not strictly require the CTS frame to reach it as specified by $\mathrm{CML}_{\mathrm{RTS}}$. Instead, it reserves an *RX CTS window* shown as a shaded slot in Fig. 3 for the source. With the RX CTS window, the destination can then choose to transmit the CTS frame at any time as long as its arrival fits into the source's RC CTS window. Assume the destination node happens to transmit/receive another frame when it has to send the CTS frame to meet the schedule of the source. The worst case is that the destination is busy with the longest frame that should be served first on an FCFS basis, and the first symbols of the two frames overlap. With respect to the worst case scenario, the destination should not transmit the CTS frame until the longest frame is completely served. We further assume that the data frame with the largest allowed payload is the longest of all frames. The source node then reserves the RX CTS window from the time it starts transmitting the RTS frame plus $\mathrm{CML}_{\mathrm{RTS}}$. The length of the window, which includes the time to receive the whole CTS frame, is

$$L_{\mathrm{RX\_CTS\_WIN}} = L_{\mathrm{CTS}} + L_{\mathrm{DATA}^*} \tag{2}$$

where L_{CTS} and L_{DATA^*} are the lengths of the CTS frame and the largest data frame, respectively. Therefore, the RX CTS window is between $T_{d_{\mathrm{RTS}}} + \mathrm{CML}_{\mathrm{RTS}}$ and $T_{d_{\mathrm{RTS}}} + \mathrm{CML}_{\mathrm{RTS}} + L_{\mathrm{CTS}} + L_{\mathrm{DATA}^*}$, where $T_{d_{\mathrm{RTS}}}$ is the time that the first symbol of the RTS frame departs from the source node.

In summary, the RX CTS window allows the destination to negotiate with the source on the time to transmit the CTS frame, which gives both the source and the destination more flexibility, and therefore, reduces the chance that the destination fails to transmit the frame at the time nominated by the source because it is unavailable.

4) TX DATA Window: With MLP and CML, a source node reserves an RX CTS window to receive the CTS frame and, likewise, it also needs to reserve a *TX DATA window* to transmit the data frame. However, due to the lack of the global topology information, the source does not know its distance from the destination at the time that the RTS frame is transmitted. With the presence of the RX CTS window, the time that the source starts to receive the CTS frame is rather flexible as long as it fits in the window. This means that the time for the source to start transmitting the data frame is indeterminate at the time that the RTS frame leaves it. Again, by considering the worst case scenario, we define the start and end points of the TX DATA window as follows.

Upon reception of an RTS frame, the destination nominates a time to receive the data frame with respect to both $\mathrm{CML}_{\mathrm{CTS}}$ and the TX DATA window. It then inserts this nominated time value in the CTS frame and replies to the source. Since the TX DATA window is reserved by the time that the RTS frame starts being sent, at which time the source node does not know its distance to the destination, we then define that a TX DATA window starts as soon as the CTS frame is actually received. Therefore, the earliest time at which the TX DATA window can start is $T_{d_{\mathrm{RTS}}} + \mathrm{CML}_{\mathrm{RTS}} + L_{\mathrm{CTS}}$, which is shown in Fig. 3 and is also the case when the CTS frame is received at the earliest time allowed in the RX CTS window.

We now derive the end point of the TX DATA window. Let $T_{a_{\mathrm{CTS}}}$ denote the actual time that the CTS frame arrives at the source. To respond to the RTS frame, the destination must calculate the time that the CTS frame is sent such that the source can complete receiving it in the RX CTS window. Therefore, the worst case is that the responded CTS frame is received by the source at the latest allowed time in the RX CTS window, which is

$$T_{a_{\mathrm{CTS}}} = T_{d_{\mathrm{RTS}}} + \mathrm{CML}_{\mathrm{RTS}} + L_{\mathrm{DATA}^*}. \tag{3}$$

[3]A commercial RTC/TCXO/crystal integration device can provide accuracy of ±2.0 ppm from 0 °C to 40 °C [25].

In addition, assume that $P_{(s,d)} = 0$, where $P_{(s,d)}$ denotes the propagation delay between nodes s and d. With respect to $\mathrm{CML}_{\mathrm{CTS}}$, the reserved TX DATA window should contain at least one $\mathrm{CML}_{\mathrm{CTS}}$ and one L_{DATA}. With the latest time of arrival of the CTS frame shown in (3), the TX DATA window should finish at $T_{d_{\mathrm{RTS}}} + \mathrm{CML}_{\mathrm{RTS}} + L_{\mathrm{DATA}^*} + \mathrm{CML}_{\mathrm{CTS}} + L_{\mathrm{DATA}}$, which is in fact the end of the RX CTS window plus $\mathrm{CML}_{\mathrm{CTS}}$ and plus L_{DATA}. Consequently, the TX DATA window is between $T_{d_{\mathrm{RTS}}} + \mathrm{CML}_{\mathrm{RTS}} + L_{\mathrm{CTS}}$ and $T_{d_{\mathrm{RTS}}} + \mathrm{CML}_{\mathrm{RTS}} + L_{\mathrm{DATA}^*} + \mathrm{CML}_{\mathrm{CTS}} + L_{\mathrm{DATA}}$, which means that the window starts from the earliest and ends at the latest possible times that the data frame is sent. Notice that, as shown in Fig. 3, the TX DATA window overlaps with the RX CTS window but this does not mean that the processes of receiving the CTS and transmitting the data frames may overlap. This overlap is solely because of the way the TX DATA window is defined, as it literally means the period within which the data frame may be transmitted. In addition, as shown in Fig. 3 for the source, the unused portion of the TX DATA window will be released immediately after the data frame is received.

In summary, the TX DATA window allows the destination to nominate with the source on the time to transmit the data frame with respect to its own schedule. As with the RX CTS window, this reduces the possibility that the destination is unavailable at a time strictly proposed by the source to receive the data frame.

5) Setting NAVs With RTS and CTS Frames: With conventional handshaking, an RTS/CTS frame typically has a “duration” field that indicates to a potential interferer the duration for which it should keep silent. In an RF-based network, the duration value is roughly the length of the pending frame and the inhibition starts immediately when the RTS/CTS frame is received. However, in a UASN, due to the asynchronous arrivals of acoustic signals, the duration values carried by RTS/CTS frames are not able to inform the potential interferers of the appropriate time that they should remain silent. Unlike conventional handshaking, we discard the duration field in an RTS or CTS frame; instead, each frame carries a field indicating its departure time, which is the time a frame leaves the transmitter. With the departure time of an RTS or CTS frame, a potential interferer is able to calculate the exact time during which it should remain silent to protect the pending frames.

First, we show how the NAV of a potential interferer is set by an RTS frame. The NAV of a potential interferer is set by an RTS frame to protect: 1) the pending CTS frame and 2) the DATA frame in case the pending CTS frame cannot be received. Let $\mathrm{NAV}_i^{\mathrm{start}}$ and $\mathrm{NAV}_i^{\mathrm{end}}$ denote the start and end points of the NAV of a potential interferer i. Clearly, with the information provided by the RTS frame, $\mathrm{NAV}_i^{\mathrm{start}}$ should address the earliest time that the CTS frame may arrive at the source, while $\mathrm{NAV}_i^{\mathrm{end}}$ should assume that the data frame may reach the destination at the latest time allowed. Consequently, we have

$$\mathrm{NAV}_i^{\mathrm{start}} = T_{d_{\mathrm{RTS}}} + \mathrm{CML}_{\mathrm{RTS}} - P_{(i,s)}. \tag{4}$$

Because the potential interferer is unable to determine its distance to the destination with only an RTS frame, the worst case scenario is considered in determining $\mathrm{NAV}_i^{\mathrm{end}}$. It is clear that only nodes distributed closer than $P_{\max}$ can directly communicate with each other. Therefore, for a node i that has received the RTS frame and may cause a collision with the reception of the data frame, there must be

$$P_{(s,i)} \le P_{\max} \tag{5}$$

and

$$P_{(i,d)} \le P_{\max}. \tag{6}$$

Thus, the latest time for any such node i to collide with the data frame is when i is at exactly the same location of d and there is $P_{(s,i)} = P_{(s,d)} = P_{\max}$. Consequently, with respect to this latest possible time of causing a collision, we have

$$\begin{aligned}\mathrm{NAV}_i^{\mathrm{end}} = T_{d_{\mathrm{RTS}}} &+ \mathrm{CML}_{\mathrm{RTS}} + L_{\mathrm{DATA}^*} \\ &+ \mathrm{CML}_{\mathrm{CTS}} + L_{\mathrm{DATA}} + P_{\max}.\end{aligned} \tag{7}$$

The NAV set by an RTS frame is illustrated as the shaded block NAV(RTS) in Fig. 3 for a potential interferer.

Then, we show how the NAV is set upon reception of a CTS frame. When a potential interferer i receives a CTS frame, it is able to calculate its distance to the destination. Assume that there is no dedicated acknowledgement frame, then the data frame is the last one of a series of frames for an MSDU. Let $T_{a_{\mathrm{DATA}}}$ denote the determinate time chosen by the destination to receive the data frame with respect to the TX DATA window. The CTS frame then prevents node i from interfering with the reception of the data frame by setting the NAV of i as

$$\mathrm{NAV}_i^{\mathrm{start}} = T_{a_{\mathrm{DATA}}} - P_{(i,d)} \tag{8}$$

and

$$\mathrm{NAV}_i^{\mathrm{end}} = T_{a_{\mathrm{DATA}}} + L_{\mathrm{DATA}} - P_{(i,d)} \tag{9}$$

and as shown in Fig. 3, NAV(CTS) overrides the $\mathrm{NAV}_i^{\mathrm{start}}$ and $\mathrm{NAV}_i^{\mathrm{start}}$ set by the RTS frame and the extra time reserved by NAV(RTS) is also released.

In summary, the departure times carried by the RTS and CTS frames inform potential interferers of the exact time that they should inhibit their transmission to protect the pending frames. In addition to CML, this helps to reduce the collisions that would be unavoidable using conventional MAC mechanisms devised for RF-based networks.

C. Adaptive Propagation-Delay-Tolerant Protocol

Having proposed the improved handshaking with MLP, CML, RX CTS window, and TX DATA window, we are now ready to assemble an adaptive MAC protocol, which is referred to as the APCAP. The timing diagram of APCAP is shown in Fig. 3.

APCAP is a MAC sublayer collision avoidance protocol that exploits an improved handshaking mechanism using RTS/CTS frames. However, the protocol does not employ carrier sense due to the reasons illustrated in Section II-A. Once an MSDU from the LLC sublayer arrives, MAC puts it into a queue and serves it as soon as its schedule and NAV allow and the transmitter is idle. Each node has a clock that indicates the absolute time that is universal in one UASN. Both RTS and CTS frames, which are transmitted at the maximum power level, should indicate their departure time.

A summary of APCAP follows. A source node transmits an RTS frame when it has an MSDU to send. In the meantime, it reserves two windows to receive the CTS frame and to transmit the data frame. Upon reception of the RTS frame, the destination calculates its distance to the source, selects a time to transmit the CTS frame according to the RX CTS window proposed by the source, and chooses a time to receive the data frame with respect to the TX DATA window nominated by the source. If the destination is unable to realize the request due to a schedule conflict, it discards the RTS frame. Then, the source ceases the MSDU if it does not receive the CTS frame at the end of the RX CTS window. An interferer sets its NAV when it receives the RTS frame, and waits for the CTS frame. If it receives the CTS frame, it overrides the NAV set by the RTS frame using the new values indicated by the CTS frame. CML intervals are invoked at the source waiting for the CTS frame and at the destination waiting for the data frame. As a result, RTS and CTS frames can be delivered to the farthest potential interferer before they may cause collisions.

The suggested frame formats of APCAP are specified below.

- **RTS frame.** In an RTS frame, the `Departure Time` is the time the frame starts being sent by MAC. `S-D Distance` is the distance between the source and the destination, which is constantly 0 in an RTS frame. `MSDU ID` is a numerical ID that is unique to each MSDU. `Dest Address` is the address of the destination node.
- **CTS frame.** In addition to the fields in the RTS frame, a CTS frame carries an extra 2-B $T_{a_{\mathrm{DATA}}}$ value indicating the expected arrival time of the data frame, which is chosen by the destination with respect to the TX DATA window of the source.
- **DATA frame.** A data frame contains an `I-source Address` and an `I-dest Address` that are the addresses of the origin node of the MSDU and the destination it is addressed to. The `Frame Body` of the data frame contains the actual MSDU and is variable between 0 and 2312 B.

The sizes of the RTS, CTS, and data frames are 20, 22, and 24 to 2336 B, respectively.

III. Simulation Studies

In what follows, we provide a quantitative analysis of APCAP by showing its performance when different components of it are enabled and disabled. We then evaluate the performance of APCAP against selected MAC protocols that can be employed by UASN. We also study the effect of different data rates on the performance of APCAP and finally show the energy efficiency of the protocol.

TABLE I
APCAP Variants

APCAP variants	1	2	3	4	5
PCAP [17]	√	√			
PCAP-CC	√	√	√		
PCAP-RCW	√	√	√	√	
PCAP-TDW	√	√	√		√
APCAP	√	√	√	√	√

A. Effect of APCAP Components on Network Performance

First, we study the effect of different APCAP components on network performance. Variants of APCAP that feature different combinations of the improvements introduced in Section II are examined. For notation simplicity, the variants of APCAP are referred to as the protocols show in Table I, where components $1 = \mathrm{MLP}$, $2 = \mathrm{CML}_{\mathrm{RTS}}$, $3 = \mathrm{CML}_{\mathrm{CTS}}$, $4 =$ RX CTS window, and $5 =$ TX DATA window, respectively.

We study the performance of APCAP variants in terms of throughput, which is the total amount of data successfully transported by the network within a given period of time. Let S denote the throughput of a network, and G denote the overall offered load. S and G are then defined as follows:

$$G = \frac{NL_{\mathrm{DATA}}}{\tau} \tag{10}$$

and

$$S = \frac{ML_{\mathrm{DATA}}}{\tau} \tag{11}$$

where τ is the duration of the simulation, N is the total number of data frames for all MSDU at all nodes in a UASN during τ, and M is the total number of data frames successfully received.

The settings used in our simulations are as follows. Each network instance is of a modest scale, comprising 20 underwater sensor nodes. All nodes are randomly deployed in a 2-D area of 4500×4500 m^2, resulting in a maximum possible internode distance of 6363 m. The maximum transmission range $R_{\max}$ is then set to 6363 m, meaning that all nodes are able to hear each other. There is one sink node located at the geographical center of the area, collecting data from the sensor nodes. All nodes are equipped with half-duplex omnidirectional transponders of a uniform type, and share one multiaccess channel. MSDUs are generated at nodes according to a Poisson distribution. The speed of sound is 1500 m/s. The frame formats are as defined in Section II-C with the body of a data frame set to 1000 B in all simulations. However, we set CML to 9 s to absorb the impact of uncertainty in the speed of sound. Note that no acknowledgement scheme is implemented, and for all simulations, we assume that unsuccessful transmission is solely caused by collisions. In other words, other factors that may lead to the failed transmission of a frame are ignored. All simulation results are obtained through applying the protocols to three network instances of different topologies and the duration of each simulation is set to 10 000 s. The throughput offered by APCAP variants under a data rate of 2400 b/s is presented in Fig. 7.

It can be seen from Fig. 7 that original PCAP [17] delivers the lowest throughput as neither $\mathrm{CML}_{\mathrm{CTS}}$ nor RX CTS and TX

Octets:	2	4	2	2	2	4	4
	Frame Control	Departure Time	S–D Distance	Source Address	Dest Address	MSDU ID	CRC

Fig. 4. RTS frame.

Octets:	2	4	2	2	2	4	2	4
	Frame Control	Departure Time	S–D Distance	Source Address	Dest Address	MSDU ID	$T_{a_{DATA}}$	CRC

Fig. 5. CTS frame.

Octets:	2	4	2	2	2	2	2	4	0–2312	4
	Frame Control	Departure Time	S–D Distance	I–source Address	I–dest Address	Source Address	Dest Address	MSDU ID	Frame Body	CRC

Fig. 6. DATA frame.

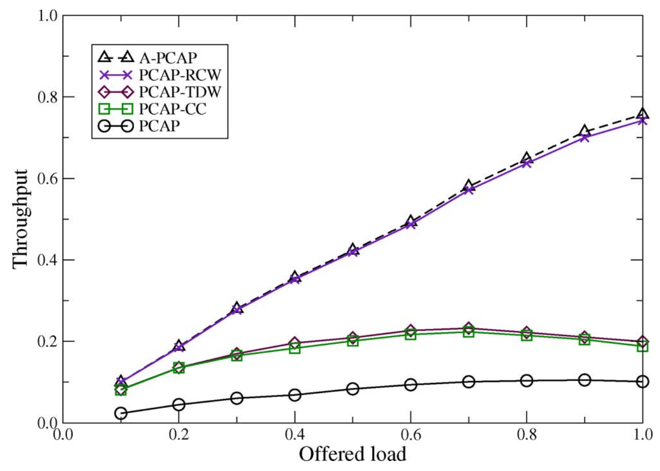

Fig. 7. Throughput versus offered load of APCAP variants.

DATA windows are implemented. The inclusion of $\mathrm{CML_{CTS}}$ in PCAP-CC provides throughput about twice of that delivered by original PCAP. Then, the TX DATA window is enabled in PCAP-TDW and the throughput is only improved marginally over PCAP-CC. Then, the RX CTS window in PCAP-RCW significantly boosts the throughput because of its ability to allow the destination to negotiate with the source on the occasion to transmit the CTS frame. APCAP, which is the protocol with all the proposed improvements enabled, has the highest throughput.

These results indicate that RX CTS window gives the destination node flexibility to respond to the source with a CTS frame with respect to its own schedule, and has the highest gain. In a UASN, all nodes are working "blindly" until sufficient information is extracted from incoming frames. Then, it is effective to allow nodes to negotiate the time for transmission and reception. Compared to original PCAP, PCAP-RCW keeps the probability of a TX CTS action colliding with other actions at the destination low. The RX CTS window is particularly helpful when many nodes are attempting to transmit to a common node. Since the improvement brought about by the inclusion of TX DATA window is small, it might be dropped by applications requiring low complexity. As indicated by the performance of PCAP-CC,

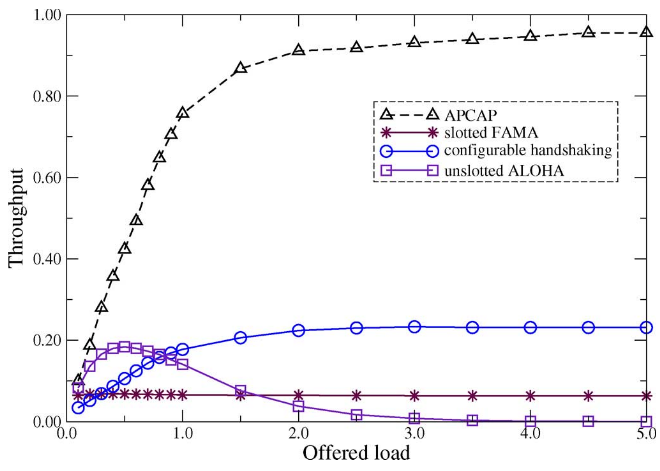

Fig. 8. Throughput versus offered load of APCAP, slotted FAMA [18], configurable handshaking [19], and ALOHA.

$\mathrm{CML_{CTS}}$ also improves the throughput compared to PCAP, especially when the offered load is not high. The results imply that each of the improvements proposed for APCAP is effective and efficient in terms of improving throughput of a UASN where the propagation delay is significant.

B. *Performance of APCAP Against Selected MAC Protocols*

Since CSMA and conventional handshaking are ineffective and inefficient in a UASN as discussed in Section I, unslotted ALOHA becomes an alternative that is immediately available for a UASN due to its simplicity and time efficiency. In addition to unslotted ALOHA, slotted FAMA [18] and configurable handshaking [19] are also implemented to compare to APCAP. For configurable handshaking, $t_{\min}$ is set to $2P_{\max}$ and $D = \infty$,[4] and the slot size for slotted FAMA is $P_{\max}$. The data rate for this part of the simulations is also fixed to 2400 b/s, and to further study the behavior of APCAP, the offered load is extended to 5.0. The throughput delivered by APCAP against the selected protocols is presented in Fig. 8.

It can be seen from Fig. 8 that APCAP offers the highest throughput of the four protocols studied. The throughput of APCAP peaks at about 0.95 when the offered load is close to 5.0. Configurable handshaking demonstrates the second-best performance but is still lower than APCAP under all offered load settings. This is mainly because configurable handshaking does not allow the source to process other MSDU while waiting for its RTS and the responded CTS to propagate. In addition, without RX CTS and TX DATA windows, configurable handshaking gives neither the source nor the destination the flexibility to best utilize their available time slots. Configurable handshaking outperforms unslotted ALOHA when the offered load exceeds 0.9 and outperforms slotted FAMA when the offered load is greater than 0.3. Slotted FAMA only surpasses unslotted ALOHA when the offered load is above 1.5, which is largely due to its long slot size and restrictive control over the media (transmission can only occur at the beginning of a slot).

[4]In [19], $t_{\min}$ is defined as the minimum handshake length. D is the difference in distance, which is essentially the signal-to-interference ratio (SIR) translated into distance.

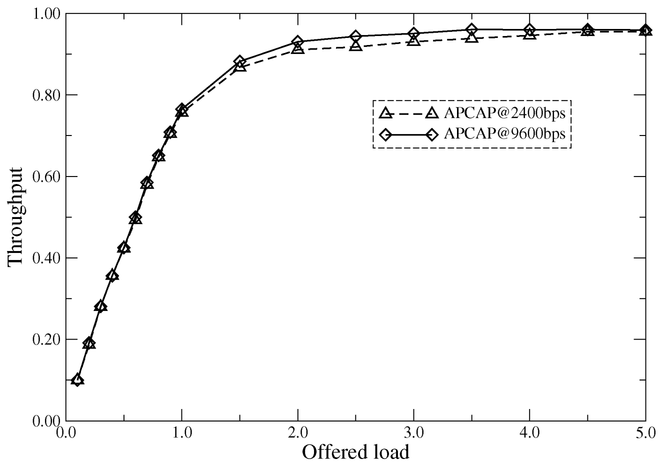

Fig. 9. Throughput versus offered load of APCAP under different data rates.

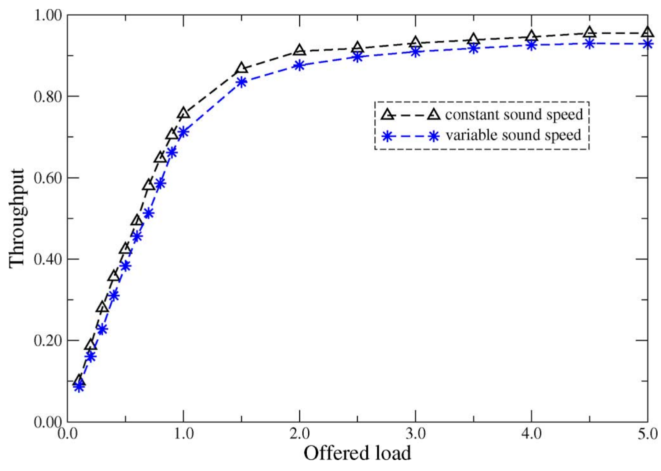

Fig. 10. Throughput versus offered load of APCAP under variable sound speed (data rate is set to 2400 b/s).

C. *Adaptability and Energy Efficiency of APCAP*

In what follows, APCAP's adaptability to data rate change, variable sound speed, and energy efficiency are examined.

1) Data Rate Change: Fig. 9 shows that the throughput offered by APCAP is only marginally different when the data rate changes from 2400 to 9600 b/s. The difference becomes more evident when there is a modest offered load, but reaches its maximum of 0.03 of the total channel capacity when the offered load is 2.5. This indicates that APCAP, unlike CSMA and handshaking as shown in Fig. 1, is insensitive to the change of α, which is defined in (1). This is the result that no matter what the data rate is, APCAP always attempts to utilize idle time slots at nodes, which greatly improves the overall efficiency of the network. The throughput of APCAP under both 2400 and 9600 b/s peaks at about 0.95 when the offered load is close to 5.0.

2) Variable Sound Speed: Then, we examined the impact of variable sound speed on the performance of APCAP. Since APCAP measures the distances between nodes to coordinate the transaction of data frames and to ensure that interferers are properly blocked, the accuracy of such measurements may affect the performance of the protocol significantly. Although we have discussed in Section II-B that CML can be set to larger than the theoretical value to absorb the uncertainty in sound speed, variability of the speed of sound may still cause interferers to be blocked at improper time and lead to collisions. To review the effect of variable sound speed on APCAP, for each request processed in the simulation, a random sound speed normally distributed with mean 1500 m/s and standard deviation 50 m/s is assigned to each transmission. The results are shown in Fig. 10.

It can be seen from Fig. 10 that the variability of sound speed has an apparent but limited impact on the performance of APCAP. Overall, the throughput delivered by APCAP is lower when the sound speed is varying compared to when it is constant. The maximum decline in throughput caused by variable sound speed is about 3% of the total channel capacity under our simulation settings.

3) Energy Efficiency: Since MAC may discard a data frame for an MSDU if handshaking is unsuccessful, the offered load G does not necessarily reflect the actual load to the channel.

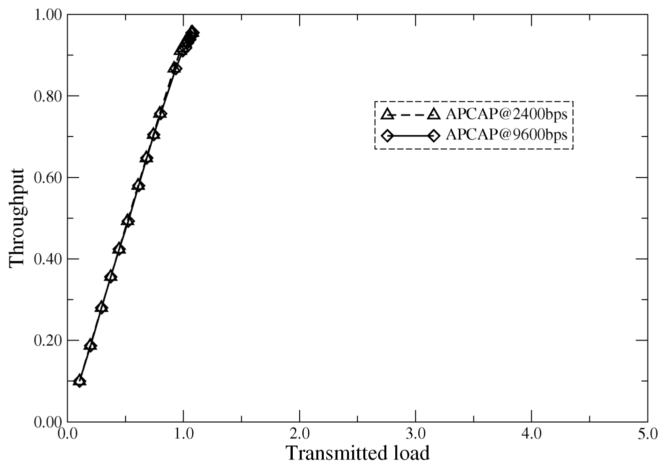

Fig. 11. Throughput versus transmitted load of APCAP under different data rates.

In the following, we observe another parameter, called transmitted load, as an additional performance measure. The transmitted load G_t is defined as follows:

$$G_t = \frac{K_{\text{DATA}}\,L_{\text{DATA}} + K_{\text{RTS}}\,L_{\text{RTS}} + K_{\text{CTS}}\,L_{\text{CTS}}}{\tau} \quad (12)$$

where K_{DATA}, K_{RTS}, and K_{CTS} are the numbers of data, RTS and CTS frames transmitted by MAC over all nodes regardless of their outcomes. Since wireless transmission is the major source of power consumption in a UASN, G_t also directly reflects the power efficiency of a protocol in terms of the actual amount of bits transmitted into the channel. Also note that while offered load is expressed purely in terms of data frames, the transmitted load also includes RTS and CTS frames, since each of these requires energy for transmission. The results are shown in Fig. 11.

Fig. 11 indicates that APCAP achieves a high degree of energy efficiency. The actual load to the channel never exceeds about 1.1 no matter what the data rate and offered load are. This means that APCAP preserves energy for underwater sensors by disallowing a node to transmit data frames after a failed handshake.

IV. Conclusion and Remarks

In this paper, we have discussed the ineffectiveness and inefficiency of carrier sense and conventional RTS/CTS handshaking if they are used in long-propagation-delay UASN. With a series of improvements to the conventional MAC mechanisms, a collision avoidance protocol, called APCAP, is proposed. The protocol is specially designed to accommodate the long propagation delay in UASN and is flexible as well as adaptive to both the offered load and the availability of destinations. The protocol is very useful for networks where nodes only have the minimal knowledge of the global network topology. The experimental results also indicate that APCAP is superior to those existing protocols if employed by a UASN.

It should also be noticed that since APCAP deliberately delays transmissions of CTS and data frames, end-to-end delay from the source to the destination over multiple hops can increase. However, this could be potentially overcome by an end-to-end scheduling protocol, but this is beyond the scope of this paper.

References

[1] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[2] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, “System architecture directions for networked sensors,” *ACM SIGOPS Operat. Syst. Rev.*, vol. 34, no. 5, pp. 93–104, 2000.

[3] J. Hill and D. Culler, “Mica: A wireless platform for deeply embedded networks,” *IEEE Micro*, vol. 22, no. 6, pp. 12–24, Nov./Dec. 2002.

[4] K. Kalpakis, K. Dasgupta, and P. Namjoshi, “Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks,” *Comput. Netw.*, vol. 42, no. 6, pp. 697–716, 2003.

[5] Y. Yao and J. Gehrke, “The cougar approach to in-network query processing in sensor networks,” *ACM SIGMOD Record*, vol. 31, no. 3, pp. 9–18, 2002.

[6] I. F. Akyildiz, D. Pompili, and T. Melodia, “Underwater acoustic sensor networks: Research challenges,” *J. Ad Hoc Netw.*, vol. 3, no. 3, pp. 257–279, 2005.

[7] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, “Data collection, storage, and retrieval with an underwater sensor network,” in *Proc. ACM SenSys*, 2005, pp. 154–165.

[8] N. Abramson, “Development of the ALOHANET,” *IEEE Trans Inf. Theory*, vol. IT-31, no. 2, pp. 119–123, Mar. 1985.

[9] L. G. Roberts, “Aloha packet system with and without slots and capture,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 5, no. 2, pp. 28–42, 1975.

[10] A. S. Tanenbaum, *Computer Networks*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996, pp. 243–338.

[11] P. Karn, “MACA—A new channel access method for packet radio,” in *Proc. ARRL/CRRL Amateur Radio 9th Comput. Netw. Conf.*, 1990, pp. 134–140.

[12] L. Kleinrock and F. A. Tobagi, “Packet switching in radio channels: Part I-Carrier sense multiple-access modes and their throughput-delay characteristics,” *IEEE Trans. Commun.*, vol. COM-23, no. 12, pp. 1400–1416, Dec. 1975.

[13] H. Takagi and L. Kleinrock, “Throughput analysis for persistent CSMA systems,” *IEEE Trans. Commun.*, vol. COM-33, no. 7, pp. 627–638, Jul. 1985.

[14] H. Matsuno, H. Ishinaka, and T. Shigeyasu, “Effect of propagation delay and RTS packet recognition time on MACA,” *Electron. Commun. Jpn.*, vol. 88, no. 1, pt. 1, pp. 21–31, 2005.

[15] K. Y. Foo, P. R. Atkins, T. Collins, C. Morley, and J. Davies, “A routing and channel-access approach for an ad hoc underwater acoustic network,” in *Proc. MTS/IEEE OCEANS Conf.*, 2004, pp. 789–795.

[16] S. Shahabudeen and M. Chitre, “Design of networking protocols for shallow water peer-to-peer acoustic networks,” in *Proc. OCEANS Conf. Europe*, 2005, pp. 628–633.

[17] X. Guo, M. Frater, and M. Ryan, “A propagation-delay-tolerant collision avoidance protocol for underwater acoustic sensor networks,” in *Proc. OCEANS Asia Pacific IEEE–Singapore Conf./Exhib.*, Singapore, May 16–19, 2006.

[18] M. Molins and M. Stojanovic, “Slotted FAMA: A MAC protocol for underwater acoustic networks,” in *Proc. OCEANS Asia Pacific IEEE–Singapore Conf./Exhib.*, May 16–19, 2006.

[19] B. Peleato and M. Stojanovic, “A MAC protocol for ad hoc underwater acoustic sensor networks,” in *Proc. 1st ACM Int. Workshop UnderWater Netw.*, 2006, pp. 113–115.

[20] A. A. Syed, W. Ye, and J. Heidemann, “T-Lohi: A new class of MAC protocols for underwater acoustic sensor networks,” Inf. Sci. Inst., Univ. Southern California, Los Angeles, CA, Tech. Rep. ISI-TR-638b, 2007.

[21] 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 Conf.*, 2001, pp. 2087–2094.

[22] V. Rodoplu and M. K. Park, “An energy-efficient MAC protocol for underwater wireless acoustic networks,” in *Proc. MTS/IEEE OCEANS Conf.*, 2005, pp. 1198–1203.

[23] L. Badia, M. Mastrogiovanni, and M. Zorzi, “An optimization framework for joint sensor deployment, link scheduling and routing in underwater sensor networks,” in *Proc. 1st ACM Int. Workshop UnderWater Netw.*, 2006, pp. 56–63.

[24] A. Syed and J. Heidemann, “Time synchronization for high latency acoustic networks,” in *Proc. IEEE Infocom*, Apr. 2006, pp. 1–12.

[25] Maxim Integrated Products [Online]. Available: http://www.maxim-ic.com

Xiaoxing Guo was born in Taiyuan, China, in 1980. He received the B.S. degree in computer science and engineering from the National University of Defense Technology, Hunan, China, in 2002, and the M.Phil. degree in computer science from the Australian National University, Canberra, A.C.T., Australia. Currently, he is working towards the Ph.D. degree at the University of New South Wales, Australian Defence Force Academy, Canberra, A.C.T., Australia.

His research interests include wireless networks, web search engines, and digital libraries. He is currently employed by the National Library of Australia.

Michael R. Frater (S'88–M'91) received the B.Sc. and B.E. degrees from the University of Sydney, Sydney, N.S.W., Australia, in 1986 and 1988, respectively, the Ph.D. degree from the Australian National University, Canberra, A.C.T., Australia, in 1991 and the M.H.Ed degree from the University of New South Wales, Canberra, A.C.T., Australia, in 1996.

Since 1991, he has been employed by the University of New South Wales, Australian Defence Force Academy, Canberra, where he is currently Professor of Electrical Engineering, and Head of the School of Information Technology and Electrical Engineering. His research interests lie in the fields of audiovisual and multimedia communications and communications systems.

Dr. Frater has served as an Associated Editor of the IEEE Transactions on Image Processing.

Michael J. Ryan (M'95–SM'06) received the B.E., M.Eng.Sc., and Ph.D. degrees from the University of New South Wales, Canberra, N.S.W., Australia in 1981, 1989, and 1996, respectively.

Since 1998, he has been employed by the University of New South Wales, Australian Defence Force Academy, where is he currently a Senior Lecturer in the School of Information Technology and Electrical Engineering. His research interests lie in the fields of systems engineering and communications systems.