

# Handling Triple Hidden Terminal Problems for Multi-Channel MAC in Long-Delay Underwater Sensor Networks

Zhong Zhou<sup>†</sup>, Zheng Peng<sup>†</sup>, Jun-Hong Cui<sup>†</sup>, and Zaihan Jiang<sup>‡</sup>  
 {zhongzhou,zhengpeng,jcui}@engruconn.edu, zaihan.jiang@nrl.navy.mil

<sup>†</sup>Computer Science & Engineering, University of Connecticut, USA

<sup>‡</sup>Acoustic Division, U.S. Naval Research Lab, Washington DC, USA

## Abstract

In this paper, we investigate the multi-channel MAC problem in underwater acoustic sensor networks. To reduce hardware cost, only one acoustic transceiver is often preferred on every node. In a single-transceiver multi-channel long-delay underwater network, new hidden terminal problems, namely multi-channel hidden terminal and long-delay hidden terminal (together with the traditional multi-hop hidden terminal problem, we refer to them as “triple hidden terminal problems”), are identified and studied in this paper. Based on our findings, we propose a new MAC protocol, called CUMAC, for long delay multi-channel underwater sensor networks. CUMAC utilizes the cooperation of neighboring nodes for collision detection, and a simple tone device is designed for distributed collision notification, providing better system efficiency while keeping overall cost low. Analytical and simulation results show that CUMAC can greatly improve the system throughput and energy efficiency by effectively solving the complicated triple hidden terminal problems.

## I. INTRODUCTION

Recently underwater acoustic sensor networks have received a rapid growing interest both in academia and industry [9], [3], [16], [7]. However, due to the unique characteristics of underwater acoustic channels (limited available bandwidth, long propagation delays and extensive time-varying multi-path effects, etc.), building underwater sensor networks encounters grand challenges at almost every level of the protocol stack, among which efficient medium access control (MAC) is one of the most fundamental issues.

In this paper, we propose a new multi-channel MAC protocol for long delay underwater sensor networks. Although many previous studies [14], [10], [18], [6], [21] focus on single-channel networks, the multi-channel technology has been shown more promising [20], [13], [8], [27], [19], [22] in both terrestrial and underwater networks. Further, recent acoustic communication advances [11] make it possible to utilize multiple channels in underwater networks. “AquaNetwork” modem from DSPCOMM [2], for example, is such a product that could be purchased off the shelf. In short, both the academia and industry are ready to move on to the multi-channel approach.

Before we can fully enjoy the benefits of the multi-channel technology, however, some critical issues should be noticed and solved. Firstly, MAC protocols for single-channel networks can not be directly used in multi-channel networks because of their low efficiency [24], [20], [17]. Secondly, the high expense of

underwater transceivers (usually several hundred to thousand dollars each [4]) make the multi-transceiver based MAC protocols [27], [8], [24] very costly. Lastly, the triple hidden terminal problems (which include two new problems: multi-channel and long-delay hidden terminal problems and the traditional multi-hop hidden terminal problem, to be discussed in Section II) inherent in the single-transceiver multi-channel long-delay underwater environment should be well addressed. Without careful design, these problems can greatly degrade the overall system performance. In summary, the new multi-channel long delay network scenario calls for innovative solutions in the MAC layer.

In this paper, we propose a protocol, called Cooperative Underwater Multi-channel MAC (CUMAC), for long-delay underwater sensor networks. By using a single transceiver, we can alleviate the budget burden of the underwater system. Our work is the first to identify the triple hidden terminal problems i.e. multi-hop, multi-channel and long-delay hidden terminal problem, that are inherent in the single-transceiver multi-channel long-delay network scenario. We solve these complicated problems with the introduction of a cooperative collision detection scheme. An inexpensive tone device (as in [21]), is adopted to further improve the system efficiency. Our CUMAC solution features low cost, high network throughput, low energy consumption and on-demand channel assignment. Our claims are backed by both analysis and simulation results.

The rest of this paper is organized as follows. In Section II, we discuss the triple hidden problems. Then in Section III, we present the system model. After that, we describe and analyze CUMAC in detail in Section IV and Section VI. Simulation results are presented in Section VII. Finally, we review some related work in Section VIII, followed by our conclusions and future work in Section IX.

## II. TRIPLE HIDDEN TERMINAL PROBLEMS

RTS/CTS based multi-channel MAC protocols have been extensively researched for terrestrial radio networks [24], [8], [17], [15]. In these studies, the communication link is divided into one control channel and multiple data channels and channel assignment is integrated into the RTS/CTS handshaking process on the control channel. For single-transceiver multi-channel long-delay underwater networks, however, these approaches are not efficient, because, except for the traditional multi-hop hidden terminal problem for the single channel network, they will suffer from two new hidden terminal problems that are inherent in the new network scenario: multi-channel and long-delay hidden terminal problems.

### A. Multi-channel Hidden Terminal Problem

Multi-channel hidden terminal problem was firstly discovered in [20] for nodes with single transceivers. If each node has only one transceiver, it can work either on the control channel or on a data channel, but not on both. This essentially causes the multi-channel hidden terminal problem. As shown in Fig. 1, when node  $a$  and node  $b$  do handshaking on the control channel, node  $c$  and node  $d$  are communicating with each other on data channel 2. Thus, node  $c$  and node  $d$  do not know the channel that is selected by node  $a$  and node  $b$  (i.e. data channel 1). Later, when node  $c$  wants to send a data packet to node  $d$ , it initiates a handshaking process on the control channel. Since node  $d$  does not know that channel 1 has been used by others at this time, it may select the same channel and thus create a collision.

Multi-channel hidden terminal problem can be solved by having one dedicated transceiver listening on the control channel continuously and thus, at least two transceivers are needed on every node. Because

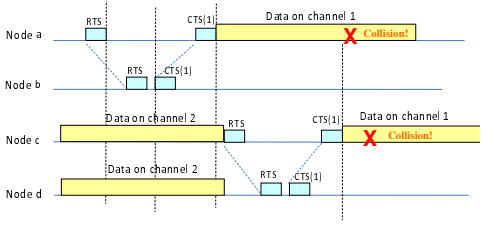


Fig. 1. Multi-channel hidden terminal problem

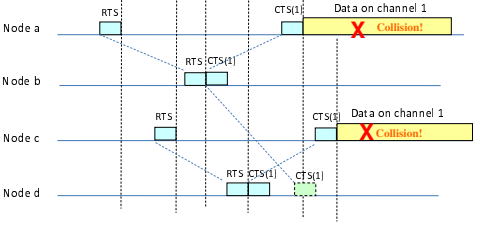


Fig. 2. Long-delay hidden terminal problem

of the high expense of underwater acoustic transceivers, it is quite costly to equip multiple transceivers on each node. To reduce system cost, our CUMAC solution targets at the network scenario where there is only one acoustic transceiver on every node .

### B. Long-delay Hidden Terminal Problem

The long propagation delays of the underwater acoustic channel introduce another kind of hidden terminal problem. As shown in Fig. 2, at the beginning, all nodes are listening to the control channel. Node *a* starts its handshaking process with node *b* on the control channel and then selects channel 1 for communication. Later, node *c* and node *d* also negotiate on the control channel for their data transmission. Let us assume the CTS message from node *b* arrives at node *d* after it selects its own data channel and sends its CTS message back to node *c*. In this case, node *d* does not know that the channel has already been used by node *b*. It may select channel 1 to communicate with node *c* and thus create a collision. We call this delay-related hidden terminal problem as “long-delay hidden terminal problem”. This problem is usually negligible in terrestrial radio networks due to the high propagation speed of radio signal. For long-delay underwater acoustic networks, however, this problem has to be well addressed.

In short, new solutions are demanded in order to effectively solve the triple hidden terminal problems in single-transceiver multi-channel long-delay underwater networks. Our CUMAC protocol is proposed to tackle these new challenges.

## III. SYSTEM MODEL

In this paper, we use the following multi-channel underwater acoustic network model:

- There are multiple channels in the network, with each channel having equal bandwidth. One control channel is dedicated for control message exchange. Other channels (i.e. data channels) are for data

transmission. Every node knows the common control channel and listens to it when there are no data to send or receive.

- Each node has only one acoustic transceiver, which can dynamically switch to different channels including control channel and data channels. Further, each node is equipped with an inexpensive out-of-band tone device which can send and receive tone signal (similar to the tone hardware in [21]).
- For the ease of presentation and analysis, we use a circular transmission range  $R$  for each node. Two nodes do not interfere with each other if their distance is larger than  $R$ . The propagation speed of the acoustic signal is  $v$ . And thus, the maximal propagation time  $T$  for the acoustic signal from a node to reach its transmission range equals  $\frac{R}{v}$ . If a node receives signal from more than one nodes simultaneously on the same channel, a collision happens and packets get lost.
- Every node is assumed to know its own location information by some localization protocols. This is a reasonable assumption since localization is also needed by networking functions such as Georouting [25] and applications such as environmental monitoring [5], [9], [12].

#### IV. PROTOCOL DESCRIPTION

In this section, we first give an overview of CUMAC, and then we present the two key techniques it employs: cooperative collision detection and tone pulse sequence. After that, we discuss how CUMAC effectively solves the triple hidden terminal problems.

##### A. Protocol Overview

In CUMAC, when a node has packets to send, it will initiate a channel negotiation process which consists of a RTS/Beacon/CTS control message exchange on the control channel. During this process, the receiving node cooperates with its neighboring nodes for channel selection and collision detection. After a data channel has been successfully selected, both the sender and the receiver switch to the selected data channel for data transmission. Therefore, CUMAC can be divided into two phases: RTS/Beacon/CTS Channel Negotiation and Data Transmission. We brief each phase as follows:

- **RTS/Beacon/CTS Channel Negotiation:** Upon data to send, a node first sends a RTS message out, which includes some useful information such as its available data channel set, on the control channel. After receiving the RTS message, the receiver will select a data channel based on the received RTS message and its own perceived channel conditions. Then it broadcasts a *beacon message* on the control channel and seek the cooperation from its neighboring nodes for collision detection. We term this technique as *cooperative collision detection* (to be described in Section IV-B). The receiver will then start a timer and wait for the responses from its neighbors. If no collisions are detected before the timer times out, the receiver sends a CTS message back to the sender to inform it of the data channel that has been selected. Otherwise, the receiver selects another data channel and broadcasts a beacon message again, performing another round of cooperative collision detection. Note that in a dense network, the multiple responses from neighboring nodes will easily congest the control channel. To overcome this problem, CUMAC employs a technique called *tone pulse sequence* with the assistance of an out-of-band tone device on each node. This technique will be discussed in Section IV-C.
- **Data Transmission:** This phase of the protocol is quite straightforward. After a successful RTS/Beacon/CTS channel negotiation process, both the sender and the receiver switch to the selected data channel for data transmission. Afterwards, both of them will switch back to the control channel.

## B. Cooperative Collision Detection

1) *Basic Idea*: In CUMAC, a cooperative collision detection scheme is designed to alleviate the hidden terminal problems discussed in Section II. In this scheme, neighboring nodes cooperate with each other to select a suitable data channel and detect potential collisions on the selected data channel. The key idea of cooperation is the following: Since a node will listen to the control channel if it does not have data to send or receive, it can obtain the channel usage information of its neighboring nodes by overhearing the control channel, and such information can be effectively leveraged by other neighboring nodes to detect any potential collision.

Taking Fig. 3(a) as an example. While node  $b$  is sending data to node  $e$  on data channel 1, node  $s$  tries to communicate with node  $d$  and sends a RTS message to it. After node  $d$  receives the RTS from node  $s$ , it selects a data channel (e.g. channel 1) which is deemed to be free by itself at this time<sup>1</sup>. Node  $d$  then broadcasts a beacon message on the control channel to inform its neighboring nodes of the channel it will use for data transmission. However, since both node  $b$  and node  $e$  are on data channel 1, they can neither hear the beacon message from node  $d$  on the control channel nor tell node  $d$  about the potential collision on data channel 1. Fortunately, as shown in the figure, some of the neighboring nodes of both node  $b$  and node  $d$  might know the status of data channel 1 by overhearing the messages on the control channel. For example, in Fig. 3(a), node  $a$  is a neighbor of both node  $b$  and node  $d$ . And  $a$  knows that data channel 1 has been used by the communication between node  $e$  and node  $b$  through overhearing their handshaking process. Thus, when node  $a$  hears the beacon message from node  $d$ , it can notify node  $d$  of the potential collision on data channel 1. In a dense network, such a cooperative collision detection scheme is expected to be quite effective in suppressing the multi-channel hidden terminal problem. As the example in Fig. 3(a), node  $c$ , node  $a$  and node  $h$  can all judge the possible collision on data channel 1, and any of them can inform node  $d$  to avoid the collision.

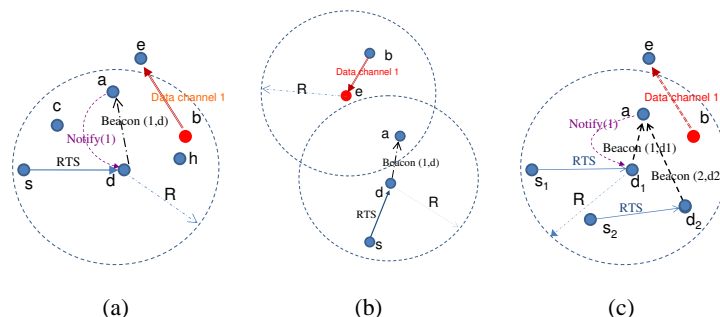


Fig. 3. Cooperative collision Detection: (a) The basic idea; (b) Invalid multi-hop scenario; (c) Problem with tone signal

To put this cooperative collision detection idea into practice, however, two critical problems, *heterogeneous collision region* and *redundant collision notification*, need to be addressed.

2) *Heterogeneous Collision Region*: In a multi-hop network scenario, the collision region of every node is different. Then how can a node tell there will be collisions on its neighboring nodes? For example, as shown in Fig. 3(b), node  $a$  knows that node  $e$  and node  $b$  are using data channel 1 and it later

<sup>1</sup>Node  $d$  might miss the handshaking process between node  $b$  and node  $e$  because of the single transceiver architecture and thus it does not know that channel 1 has already been used.

receives a beacon message from node  $d$  saying data channel 1 is selected for data transmission. Fig. 3(b) shows us that these two communication pairs ( $e$  and  $b$ ,  $s$  and  $d$ ) are far enough to use data channel 1 simultaneously without collision, which is quite different from the case in Fig. 3(a). However, how can the cooperative node (node  $a$  in the examples) identify different situations? We term this problem as *heterogeneous collision region problem*.

In CUMAC, location information of each node is needed and could be included in the control messages such as RTS/Beacon/CTS. Thus, a node can obtain the locations of all its known data senders and receivers. Then, it can calculate the distances between these nodes and judge whether collision among them will happen or not. For example, as in Fig. 3(a), node  $a$  knows the locations of nodes  $s$ ,  $d$ ,  $b$  and  $e$  by overhearing the control channel. Node  $a$  thus can calculate the distances between these nodes and find that node  $b$  and node  $d$  are close enough to interfere with each other.

3) *Redundant Collision Notification*: In a dense network, it is possible that multiple neighboring nodes perceive the same collision event, which stimulates all of them to notify the ongoing communication pair. Multiple collision notification messages thus might be generated. For example, as shown in Fig. 3(a), besides node  $a$ , nodes  $c$  and  $h$  might also find the potential collision on data channel 1 and will notify node  $d$ . Apparently, it is unwise to transmit all these notification messages on the control channel directly since these messages will overwhelm the channel. This will significantly increase the collision probability of the control channel and degrade the system performance. Therefore, such redundant collision notification should be suppressed.

To address this problem, in CUMAC, an out-of-band tone device is required on each node and it is used for the collision notification purpose. The simplest method is that if a node detects collision, it sends out a tone signal for notification. However, such a naive method does not work in the multi-channel network scenario. As multiple data channels are used concurrently, multiple tone signals might exist simultaneously. Then how can a node with only one tone device differentiate multiple tone signals for different channels and different communication pairs? For example, as show in Fig. 3(c), node  $s_1$  and  $d_1$  are trying to communicate with each other on data channel 1 and node  $s_2$  and  $d_2$  are planning to use data channel 2 at the same time. Both  $d_1$  and  $d_2$  send out a beacon message to ask for help from their neighbors. At this time, node  $a$  finds collisions on data channel 1 and then it will send its tone signal out to inform node  $d_1$ . However, at the same time,  $d_2$  is also waiting for the response from its neighboring nodes and it can also receive the tone signal for node  $d_1$ . Because tone signals cannot be decoded, how can node  $d_2$  know that the received tone signal is not for it? In CUMAC, we introduce the technique of *tone pulse sequence* to solve this problem.

### C. Tone Pulse Sequence

1) *Basic Idea*: *Tone pulse* is defined as a short tone signal and **tone pulse sequence** is a sequence of  $n$  periodic tone pulses, where  $n$  is a predefined system parameter. In CUMAC, a node who has detected collision on a data channel will send out a tone pulse sequence at a specific time. With careful design, tone pulse sequences destined for different nodes will arrive at a node at different time. If a node knows the schedule of the tone pulse sequence destined for itself, this node can pick its tone pulse sequence out and make a correct decision.

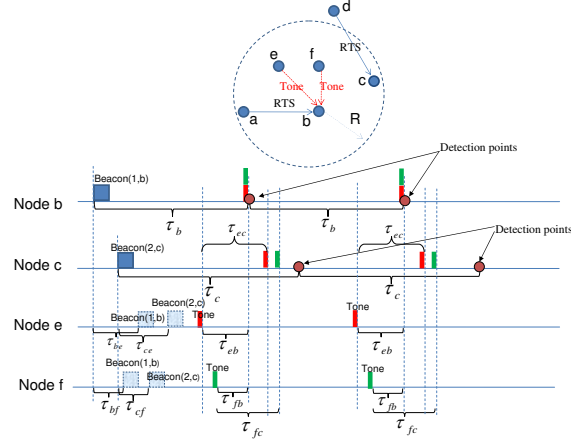


Fig. 4. Mechanism of tone pulse sequence

In CUMAC, a node, say  $i$ , which has received a RTS and selected a data channel for its data receiving will first broadcast a beacon message, on the control channel. This beacon message includes this node's location information and the periodicity  $\tau_i$  of its tone pulse sequence. Node  $i$  then listens for the tone pulse every  $\tau_i$  starting from the time it sends its beacon message out. And we define these time points where node  $i$  listens for the tone pulse as the **detection points** of node  $i$ . For example, if node  $i$  sends its beacon message out at time  $t_i$ , then  $t_i + k\tau_i, 1 \leq k \leq n$  are its detection points.

After a neighboring node receives a beacon message, it will first estimate whether there will be collisions based on the beacon message and its perceived data channel conditions. If a neighboring node detects a collision, based on the location information, it will calculate the time to send its tone pulse sequence in order to make them arrive at node  $i$  on the expected detection points. For example, if the distance between node  $i$  and a neighboring node  $j$  is  $d_{ij}$  and at time  $t_j$  node  $j$  receives the beacon message, then, at time  $k\tau_i - 2\frac{d_{ij}}{v} + t_j, (k = 2\lceil \frac{d_{ij}}{v\tau_i} \rceil)$ , node  $j$  will send out its first tone pulse, where  $v$  is the propagation speed of the underwater acoustic signal. After that, node  $j$  will send out its tone pulse every  $\tau_i$  until  $n$  tone pulses (i.e., a tone pulse sequence) are sent out.

Tone pulses from different sources destined for the same node will arrive at the destination on its detection points. If multiple tone pulses arrive at the same time, they add up and the destination can detect the existence of the tone pulses. Since the duration of a tone pulse is small and every node randomly chooses the periodicity for its tone pulse sequence, the probability that the tone pulse sequence destined for one node overlaps with the tone pulse sequences destined for another node is quite small.

For example, as shown in Fig. 4, node  $b$  and node  $c$  select channel 1 and 2 respectively for data receiving. And they send out beacon messages to inform neighboring nodes of their selected channels. Both node  $e$  and  $f$  find that channel 1 is not available and send out tone pulse sequences back to node  $b$ . Since node  $b$ 's location is known by  $e$  and  $f$  through its beacon message,  $e$  and  $f$  can calculate the time to send their tone pulse sequences out. And as shown in this figure, the tone pulses from different neighbors (node  $e$  and  $f$ ) will arrive at node  $b$  at its detection points. From Fig. 4, it is clear that the tone pulse sequences from node  $e$  and  $f$  arrive at node  $c$  at times different from the detection points of node  $c$ . Thus, node  $c$  will not mistake the tone pulse for node  $b$  as its own.

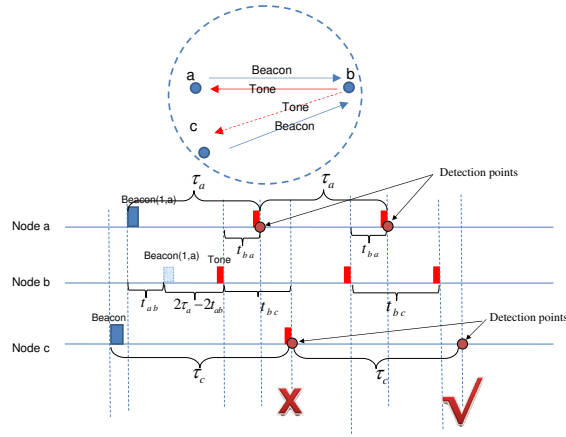


Fig. 5. Strength of tone pulse sequence

2) *Why Tone Pulse Sequence?*: In CUMAC, multiple tone pulses (a tone pulse sequence) are needed to reduce its error probability. As shown in Fig. 5, node  $a$  and node  $c$  send out their beacon messages and node  $b$  finds potential collision for node  $a$ . While for node  $c$ , no potential collisions are detected on its selected data channel. Although the tone pulse that is sent by  $b$  at time  $t_a + \tau_a - t_{ab}$  arrives at node  $a$  at time  $t_a + \tau_a$ , it arrives at node  $c$  at time  $t_c + \tau_c$ , which happens to be a detection point of node  $c$ . It is clear that if node  $c$  makes its judgment simply based on this single tone pulse, it will make a wrong decision.

Therefore, in CUMAC, every node which has detected possible collisions for node  $i$  will send out  $n$  tone pulses (a tone pulse sequence) periodically according to its received periodicity  $\tau_i$  to let them arrive at node  $i$  at its detection points. With high probability, different nodes will have different  $\tau_i$ . Thus, it is highly unlikely that all  $n$  tone pulses destined for one node fall on the detection points of other nodes. For example, as shown in Fig. 5, although the first tone pulse destined for node  $a$  falls on one detection point of node  $c$ , the second tone pulse does not, which prevents the error from happening on node  $c$ . In Section VI, we will analyze the false alarm and false approval probability of our protocol.

Because of the space limit, we omit the implementation details of our CUMAC in this paper. Interested readers can refer to our technical report [26].

#### D. Discussions

With the import of the RTS/Beacon/CTS handshaking process and the cooperative collision detection scheme, CUMAC can greatly alleviate, if not eliminate, the triple hidden terminal problems. As the traditional multi-hop hidden terminal problem (which has been well studied in the literature) can be easily addressed by a typical RTS/CTS handshaking process, in the following, we mainly discuss how CUMAC handles the two new hidden terminal problems: multi-channel and long-delay hidden terminal problems.

First, the multi-channel hidden terminal problem is caused by the incomplete channel usage information available to each node, which is the result of the single transceiver architecture. In CUMAC, the cooperative collision detection scheme asks for help from neighboring nodes. Collisions on the data channels can only happen when all neighboring nodes do not perceive the channel condition correctly, which is highly

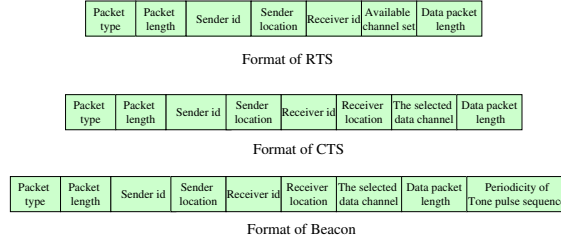


Fig. 6. Packet format of control messages

unlikely in a dense network. Thus, the multi-channel hidden terminal problem is greatly suppressed in CUMAC.

Second, the long-delay hidden terminal problem originates from the fact that the long propagation delays of underwater acoustic channels prevent nodes from getting updated data channel conditions timely. In CUMAC, the RTS/Beacon/CTS handshaking process effectively addresses this problem. Before a data channel is used by a node for data communication, this node first reserves this channel and sends a beacon message out to inform the neighboring nodes. Only after neighboring nodes get this information and agree, this channel can be used. Thus, the long-delay hidden terminal problem is also alleviated.

## V. PROTOCOL DETAILS AND IMPLEMENTATION

In CUMAC, a data channel might be in the following three states:

- Free: this channel is free and can be used or reserved.
- Reserved: this channel is reserved by some nodes for data packet transmissions.
- Busy: this channel is busy in transmitting data packets.

Every node will maintain a channel usage table to record the usage information of data channels. Every item in the channel usage table records some useful information of a data channel such as the channel status, the length of its data packet, the source node, the destination node and their locations. As shown in Fig. 3(b), it is possible that a node (for example, node  $a$  in Fig. 3(b)) might perceive multiple non-interfering communication pairs on a data channel in its neighboring area. These pairs should be recorded as different items in the channel usage table. Thus, multiple items might exist in a node's channel usage table for a channel.

At the beginning, no data channels have been used and all channels are in free status. Thus, the channel usage table of every node is empty. When a node has packets to send, it sends a RTS message out, which includes its available channel set, its  $id$ , location and data length. The available channel set of a node consists of the free data channels which do not appear in its channel usage table. When the intended receiver, let's say node  $i$ , receives the RTS, it selects one channel randomly from the **intersection** of its own available channel set and that of the source node which is contained in the RTS message. Then, Node  $i$  sends a beacon message out, which includes the selected channel, the data packet length, the  $ids$  and locations of the sender and the receiver. The periodicity of its tone pulse sequence  $\tau_i$ , which has been discussed in Section IV-C, is also included in the beacon message. After that, node  $i$  inserts a corresponding item in its channel usage table and sets the channel in the reserved status.

The receiver (node  $i$ ) then waits for the tone pulses and sets a timer which will expire after  $2T + n\tau_i$ , where  $T$  is the maximal propagation delay. If less than  $n$  tone pulses are received at its detection points before it times out, this node believes that there will be no collisions on the selected data channel and will send a CTS message out. The CTS messages includes the selected data channel, the data packet length, the locations as well as the *ids* of the sender and the receiver. The packet format of all control messages is shown in Fig 6. Node  $i$  also changes the corresponding item in its channel usage table to busy. Then, this node will switch to the data channel. After receiving the CTS message, the data sender also switches to the data channel and transmits its data. After the data transmission, both of the sender and the receiver will switch back to the control channel.

If  $n$  or more tone pulses are received on the detection points of the receiver before it times out, the receiver will delete the corresponding item in its channel usage table. Then it selects another channel and repeat the above process. After  $k$  channel selection trials, if this node still can not get a right channel which is approved by all its neighbors for data receiving, it gives up and will not send beacon messages out any more. On the other hand, when the sender times out on receiving CTS message, it will randomly backoff for some time and resend a RTS to the receiver. The optimal backoff strategy is beyond the scope of this paper and in our simulation, we adopt the exponential backoff strategy.

A node who receives a beacon message correctly detects whether collisions will happen on the selected data channel based on its own channel usage table. If no collision is detected between this communication event and other perceived communication events, a new item will be inserted into this node's channel usage table and set its status to be reserved. If a collision is detected between this communication event and other on-going data packets (i.e. busy items in the node's channel usage table), this node will send out a tone pulse sequence to inform the corresponding receiver to select another channel for data receiving. If this communication event is predicted to collide with other in-negotiation transmission events (reserved items in the node's channel usage table), this node will compare the new communication event with the existing colliding ones. The one who sends out its beacon message first will be deemed to be effective. This node will then send tone pulse sequences out to notify the non-effective communication pairs and delete the corresponding items in its channel usage table.

If an item in the channel usage table corresponding to the receiver  $i$  has been in the reserved status for more than  $2T + n\tau_i$  and its CTS message has not been received, this item will be deleted. When the CTS message is received, this item will be changed to the busy status. It should be noted that the RTS/CTS message includes the length of the subsequent data packets, which indicates how long this channel will be occupied for data transmissions. Every node who has received the CTS message will set the corresponding item in its channel usage table to the busy status for the period for the data transmission and then delete it.

After a receiver sends its CTS to the sender and switches to the selected data channel, it will start a timer which will expire after  $2T$ . If it does not receive any data from the sender before it times out, the CTS message might get lost and the receiver switches back to the control channel. It then deletes the corresponding item in its channel usage table. After that, it will broadcast a **REVOKE** message out to cancel the channel reservation. The **REVOKE** message is the same as the Beacon message except that the last tow fields are set to null. This receiver then reselects another data channel and sends out a new beacon message. neighboring nodes who receive the **REVOKE** message will check its channel usage table

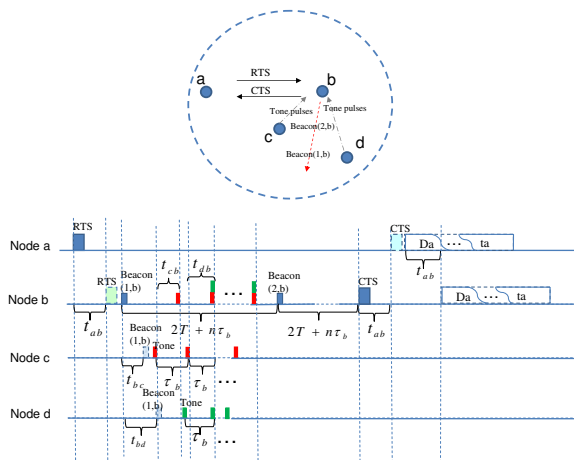


Fig. 7. An example of CUMAC process

and delete the corresponding item.

One example of the process of CUMAC is shown in Fig. 7. Here, node  $a$  wants to send a packet to node  $b$ . And at this time, node  $c$  and node  $d$  are also listening to the control channel. Node  $a$  first sends out a RTS message to  $b$  and after  $b$  gets RTS message, it will select a channel (e.g. data channel 1) and send a beacon message out. Then, node  $b$  inserts a new item about data channel 1 in its channel usage table and set its status reserved. After node  $c$  and  $d$  receive the beacon message, both of them judge there will be collisions on the selected data channel and will send out their tone pulses at the specified time. The tone pulses from node  $c$  and  $d$  will arrive at node  $b$  at its detection points. Thus, node  $b$  knows that the selected data channel 1 can not be used. Node  $b$  then deletes the corresponding item about data channel 1 from its channel usage table and selects another data channel (e.g. data channel 2). After receiving the beacon message for data channel 2 from node  $b$ , node  $c$  and  $d$  do not detect collision at this time. Then, both node  $c$  and  $d$  will insert a new item about data channel 2 into their channel usage tables. No tone pulses will be sent out now. After some time, node  $b$  sends out its CTS message to inform node  $a$  the selected data channel 2. And finally, both node  $a$  and  $b$  switch to the data channel 2 for data transmission.

## VI. ANALYSIS

As discussed earlier, CUMAC has effectively solved the triple hidden terminal problems. However, the benefits do not come without any cost. Two kinds of errors: *false alarm* and *false approval*, naturally arise in the cooperative collision detection scheme. In this section, through analysis, we will show that these errors are bounded and negligible compared to the benefits of CUMAC, esp. in dense networks.

For simplicity, we make the following assumptions. In the network, nodes follow a uniform distribution with a density  $\rho$ . Thus in a 3-D underwater sensor network, the average neighbors of a node is  $\frac{4\pi R^3}{3}\rho$ . The traffic of each node follows an identical independent (iid) Poisson process with the parameter  $\lambda$ . For the tone pulse sequence technique, we set the duration of the tone pulse to be  $t_{tp}$ . We denote the periodicity of the tone pulse sequence for node  $i$  as  $\tau_i$ , and  $\tau_i$  is in a range of  $[\tau_{\min}, \tau_{\max}]$  and takes discrete values

among  $\tau_{\min} + 2k \times t_{tp}$ , ( $0 \leq k \leq s$ ), where  $s = \frac{\tau_{\max} - \tau_{\min}}{2t_{tp}}$ , with equal probability (we also use this setting of  $\tau_i$  in our simulations).

1) *False Alarm Probability*: *False alarm* means that even though there will be no collision on the selected data channel, CUMAC mistakenly believes that there will be collisions and asks for data channel reselection. Clearly false alarm increases the traffic on the control channel, which will degrade the system performance.

In CUMAC, for a node (say node  $i$ ) which has sent out its beacon message and is waiting for the response from its neighbors, if it receives  $n$  tone pulses which are destined for other nodes at its  $n$  detection points, node  $i$  will make a false decision and false alarm happens. To get the upper bound for the false alarm, we assume that all neighbors of node  $i$  are on the control channel (this is the worst case because some nodes might be on the data channel and will not be able to send tone pulse sequences). Since nodes are uniformly distributed in the network, the tone pulses destined for other nodes arrive at node  $i$  following a uniform distribution.

If another node, say node  $j$ , selects the same periodicity as that of node  $i$  and a tone pulse destined for node  $j$  arrives at node  $i$  at the detection point of node  $i$ , it is self-evident that the following tone pulses for node  $j$  will also arrive at node  $i$  at  $i$ 's detection points. If more than  $n$  tone pulses destined for node  $j$  fall in the detection periods of node  $i$ , false alarm happens on node  $i$ . The probability that node  $j$  selects the same periodicity of its tone pulse sequence as that of node  $i$  will be  $1/s$  since every node selects the periodicity for its tone pulse sequence independently. And the probability that a tone pulse destined for node  $j$  arrives at node  $i$  on the detection point of node  $i$  will be  $\frac{2t_{tp}}{\tau_i}$ . Thus, we can get the probability  $P_{B_{ij}}$  that node  $i$  makes a false alarm based on the tone pulse sequence which is destined to another node  $j$  as

$$P_{B_{ij}} = \frac{1}{s} \left( 1 - \left( 1 - \frac{2t_{tp}}{\tau_i} \right)^{\frac{4}{3}\pi R^3 \rho} \right) \quad (1)$$

Since we assume that the input traffic for every node follows an identical independent Poisson process, the probability  $P(k)$  that there are  $k$  other ongoing channel negotiations which might interfere with node  $i$  can be written as

$$P(k) = \frac{\left( \frac{4}{3}\pi R^3 \rho \lambda 4T \right)^k}{k!} e^{-\left( \frac{4}{3}\pi R^3 \rho \lambda 4T \right)} \quad (2)$$

where  $T$  is the maximal propagation delay. Since in CUMAC, every node sends out its beacon message and chooses the periodicity of its tone pulse sequence independent of others, we can get the upper bound of the probability of this kind of false alarm  $P_B$  as

$$P_B = \sum_{k=1}^{\infty} \left( 1 - (1 - P_{B_{ij}})^k \right) P(k) \quad (3)$$

The above false alarm is brought about by the tone pulse sequence destined for another node. In practice, however, it is possible that multiple tone pulses destined for different nodes happen to arrive at node  $i$  on its  $n$  detection points, which also leads to false alarm. By assuming that all neighbors are on the control channel, we can approximately derive the upper bound of the probability for this kind of false alarm  $P_A$ .

If there are other  $k$  on-going channel negotiations in the neighborhood of node  $i$ , at most  $\frac{4}{3}\pi R^3 \rho k$  tone pulse sequences will be generated. Set  $P(A|K = k)$  to be the probability that node  $i$  makes this kind of

false alarm under the condition that there are other  $k$  channel negotiations in progress.

$$P(A|K = k) = 1 - \sum_{l < n} C_l^{\frac{4}{3}\pi R^3 k \rho} \left(\frac{2t_{tp}}{\tau_i}\right)^l \left(1 - \frac{2t_{tp}}{\tau_i}\right)^{\frac{4}{3}\pi R^3 k \rho - l} \quad (4)$$

Since the input traffic of every node is assumed to be identical independent Poisson process, we can get

$$\begin{aligned} P_A &= \sum_{k=1}^{\infty} P(K = k)P(A|K = k) \\ &= \sum_{k=0}^{\infty} \frac{\left(\frac{4}{3}\pi R^3 \rho \lambda 4T\right)^k}{k!} e^{-\frac{4}{3}\pi R^3 \rho \lambda 4T} \times \\ &\quad \left(1 - \sum_{l < n} C_l^{\frac{4}{3}\pi R^3 k \rho} \left(\frac{2t_{tp}}{\tau_i}\right)^l \left(1 - \frac{2t_{tp}}{\tau_i}\right)^{\frac{4}{3}\pi R^3 k \rho - l}\right) \end{aligned} \quad (5)$$

Thus, the overall false alarm probability  $P_{f1}$  can be got

$$P_{f1} \leq P_A + P_B \quad (6)$$

Our simulation results in Section VII show that the false alarm probability under normal network conditions is very small, usually less than 0.1.

2) *False Approval Probability*: **False approval** means that although there exist potential collisions on the selected data channel, CUMAC fails to detect them, which finally results in the collisions on the data channel. Since data packets usually are much longer than control packets, collisions on the data channel will lead to relatively high resource (such as bandwidth and energy) wastage.

Due to the use of a cooperative collision detection scheme, in CUMAC, a false approval only happens if no neighbors oppose to a false decision. We denote the probability that a node stays on the control channel as  $p_c$  and its average collision probability on the control channel as  $p_{col}$ . A neighbor can correctly receive a control message only if it is on the control channel and does not perceive collisions on the control channel. And the probability of such an event is  $p_c(1 - p_{col})$ . Only if a neighbor can receive the beacon/CTS messages correctly from both colliding communication pairs, it can detect collisions. Thus, the false approval probability  $P_{f2}$  is

$$P_{f2} = [1 - (p_c(1 - P_{col}))^2]^{\frac{4\pi R^3}{3}\rho}. \quad (7)$$

Now we estimate the probability of  $p_c$ . For one data packet transmission, averagely, at least  $\frac{1}{(1-p_{col})^2}$  times of RTS/Beacon/CTS exchange need to be performed, and every RTS/Beacon/CTS exchange will take at least  $4T + t_{rts} + t_{cts} + t_b$ , where  $t_{rts}$ ,  $t_{cts}$  and  $t_b$  are the transmission time for a RTS, CTS and beacon message respectively. Then we can get

$$p_c \geq \frac{\frac{1}{(1-p_{col})^2}(4T + t_{rts} + t_{cts} + t_b) + \frac{1}{\lambda}}{\frac{1}{(1-p_{col})^2}(4T + t_{rts} + t_{cts}) + t_b + \frac{1}{\lambda} + t_D} \quad (8)$$

where  $t_D$  is the transmission time for a data packet. Here  $\frac{1}{\lambda}$  is included because a node will stay on the control channel when it has no traffic (also recall that the input traffic is a Poisson process with parameter  $\lambda$ ). For the collision probability on the control channel  $p_{col}$ , as in [8], by assuming  $t_{rts} = t_{cts} = t_b = t_c$ ,

we can approximate aggregated traffic of RTS/Beacon/CTS on the control channel as a Poisson process with parameter  $\lambda + 2(1 - p_{col})\lambda$ . Then we can have

$$p_{col} = 1 - e^{-\frac{4\pi R^3}{3}\rho(1+2(1-p_{col}))\lambda t_c}, \quad (9)$$

which can be used to calculate  $p_{col}$  numerically.

## VII. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our CUMAC via simulations.

### A. Simulation Settings

Based on NS-2, UConn UWSN Lab has developed a simulation package, called Aqua-Sim, for underwater sensor networks [1]. We implemented CUMAC in Aqua-Sim. Both random and fixed topology networks are investigated. Unless specified otherwise, the simulation parameters are as follows. There are 8 channels in the network and the bandwidth of each channel is set to  $1kbps$ . The propagation speed of the acoustic signal is  $1500m/s$ . The transmission range of every node is set to be  $500m$ . The length of a tone pulse is set to 2ms. The periodicity of the tone pulse sequence for each node is in a range of  $[12ms, 60ms]$  and takes discrete values among  $12 + 4k$ , ( $0 \leq k \leq 12$ ) ms with equal probability. The number of tone pulses in a tone pulse sequence is 3. The average data packet length is 300 bytes. The average transmitting and receiving power of the acoustic transceiver is set to be 0.6 Watt and 0.2 Watt. And the idle listening power is 0.02 Watt. Every simulation runs for 500 seconds and every data point is the average of 100 simulations.

For comparison purpose, we implemented two other multi-channel MAC protocols, random multi-channel MAC protocol (*random scheme* for short) and RTS/CTS based multi-channel MAC protocol (*RTS/CTS scheme* for short).

- *Random multi-channel MAC protocol*: In this protocol, when a node has data to send, it first randomly chooses a data channel and then sends out a small control packet on the control channel to inform the receiver of its selection. After that, this node switches to the selected data channel and sends out its data packet immediately. If the receiver gets the control packet correctly, it switches to the data channel for data receiving. This random protocol is analyzed for underwater acoustic networks in [27].
- *RTS/CTS based multi-channel MAC protocol*: In this protocol, if a node has data to send, it starts with a RTS on the control channel to inform the receiver. After the RTS been received correctly, the receiver will send a CTS message which includes the selected data channel back to the sender. Both the sender and the receiver then switch to the selected data channel for data transmission. This protocol is similar to the protocol in [24], [8] which is designed for terrestrial sensor networks and analyzed in [27] for underwater acoustic networks.

For all three protocols, we measure two metrics: *average network throughput* and *average energy consumption per byte*. The first metric is defined as the average number of successfully transmitted data bytes per second. The second metric is obtained by dividing the overall energy consumption in the network by the successful transmitted data bytes, which is measured in milli-joule per byte.

## B. Simulation Results

1) *Results in random networks:* We simulate a random network where 20 underwater nodes are uniformly distributed in a  $300m \times 300m \times 300m$  area. A sending node randomly chooses one of its neighbors as the receiver.

**Impact of input traffic:** In this set of simulations, we change the input traffic of every node from 0.002 to 0.06 packet per second. As shown in Fig. 8, for all three protocols, the throughput increases rapidly with the input traffic at first. And then, it decreases. For example, the throughput of CUMAC achieves its maximal when the input traffic increases to 0.04 packet per second. After that, because of the increasing collisions in the network, the throughput decreases slowly with the input traffic. Fig. 8(a) shows us that our CUMAC can achieve much higher throughput than the other two. For example, when the input traffic is 0.03 packet per second, the throughput of CUMAC is about 160 bytes per second. While that of the other two is less than 100 bytes per second.

From Fig. 8(b), we can see that for all three protocols, the average energy consumption per byte decreases at first with the increase of the input traffic. This is because with the increase of the input traffic, less energy will be wasted on the idle listening (the idle listening power consumption in our simulation is 0.02 Watt). However, with the increase of the input traffic, the collision probability in the network will also increase, which results in the degradation of energy efficiency. Fig. 8(b) also shows us that CUMAC can achieve much higher energy efficiency than the other two. For the random scheme, a node sends its packets in a totally random manner. There is no method to suppress the collision in the network. Thus, its energy efficiency is the worst. For the RTS/CTS scheme, the RTS/CTS exchange before the data transmission reduces the collisions on the data channels to some extent. However, the RTS/CTS scheme suffers from the multi-channel and the long-delay hidden terminal problems which are inherent in the single-transceiver multi-channel long-delay underwater network. As for CUMAC, its cooperative collision detection scheme along with the tone pulse sequence mechanism suppresses the triple hidden terminal problems efficiently, which results in higher throughput and better energy efficiency.

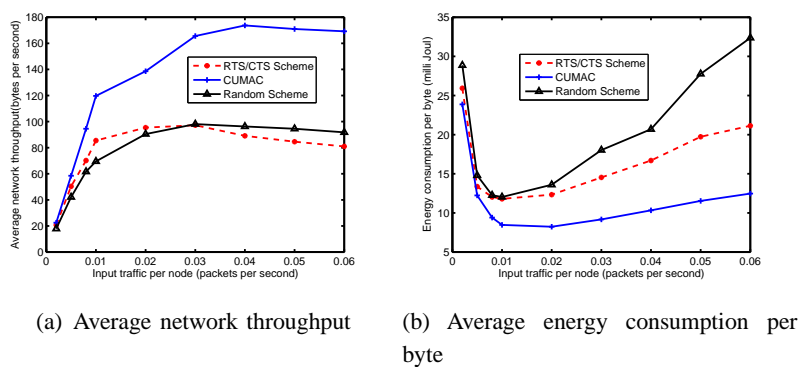


Fig. 8. Performance comparison with different input traffic

**Impact of number of channels:** In this set of simulations, we set the input traffic of every node to 0.02 packet per second and change the number of channels from 4 to 16. Without surprise, as shown in Fig. 9, both the network throughput and the energy efficiency improve with the number of channels. This is because the more the data channels, the less the collision probability on the data channels since the net input traffic to every data channel will be reduced. However, the improvements on the system

performance will slow down with the increase of the number of data channels. This is because the system's performance is jointly determined by the control channel and the data channels. With the increase of the number of data channels, the system will be more constrained by the control channel. Thus, the impacts of increasing the number of data channels become less and less. To further improve the system performance, the bandwidth of the control channel should change dynamically with the network conditions, which is an interesting research topic and will be studied in our future work.

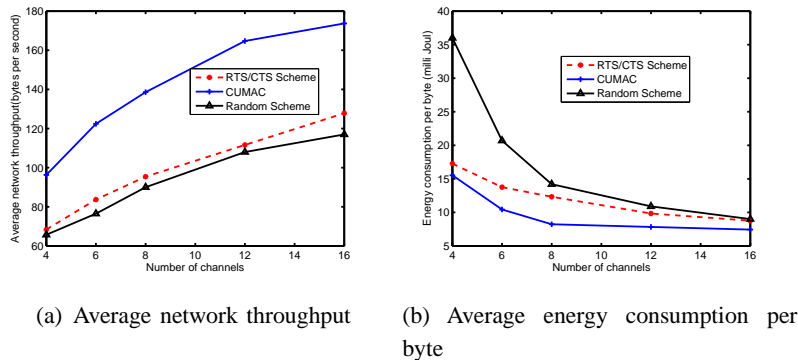


Fig. 9. Performance comparison with varying number of channels

**Impact of data packet length:** In this set of simulations, we fix the input traffic of every node to 0.02 packet per second and change the data packet length from 200 bytes to 600 bytes<sup>2</sup>. As shown in Fig. 10, in CUMAC and the RTS/CTS scheme, the network throughput increases monotonically with the packet length. This is reasonable since the longer a data packet is, the more data will be transmitted on the data channels for a successful handshaking process on the control channel. Consequently a higher throughput can be achieved.

For the random scheme, when the packet length is small (less than 500 bytes), the network throughput increases slowly with the packet length, but decreases afterwards. This is because for the random scheme, the length of data packets has double effects. On one hand, longer data packets may contribute to higher collision probability on the data channels, which might lead to the decrease of the throughput. On the other hand, potentially, with longer data packets, one successful data packet transmission contributes more throughput than the case with shorter data packets. When the average length of data packets is short, the second factor dominates the first, therefore the network throughput will increase. But with the increase of data packet length, the first factor plays a major role and thus the network throughput will decrease.

From Fig. 11(b), we can also observe that for all three protocols, the average energy consumption decreases with the increase of the average data packet length, which means that the longer the data packet, the higher the energy efficiency. Thus, in practice, we may increase the length of data packets for high energy efficiency.

**Impact of tone pulse sequence length:** In this set of simulations, we change the number of tone pulses in a tone pulse sequence ( $n$ ) from 1 to 5. Fig. 11 illustrates the strength of our tone pulse sequence mechanism. For example, when the input traffic is 0.03 packets per second, the throughput for the system with single tone pulse is only 118 bytes per second. And by simply increasing the number of tone

<sup>2</sup>Here, the packet length does not necessarily mean the length of one data segment, it might be the total length of a series of small data segments. Here, we define the packet length as the total data bytes followed a control messages exchange

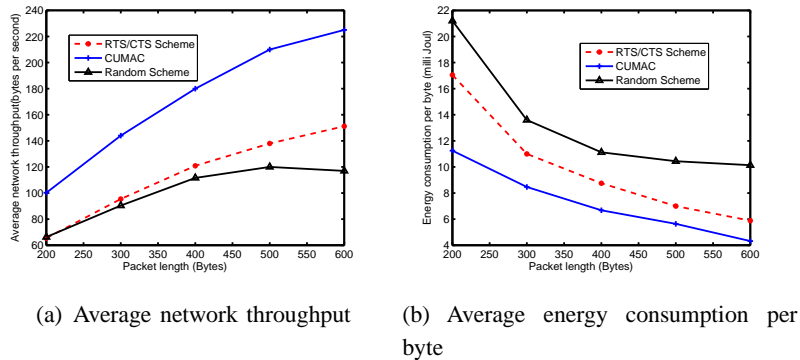


Fig. 10. Performance comparison with varying data packet length

pulses in a tone pulse sequence to 3, we can improve the throughput to more than 160 bytes per second. And at the same time, the system's energy consumptions also decrease. This is because our tone pulse sequence mechanism can effectively reduce the false alarm in the system and thus reduce the unnecessary re-negotiation on the control channel.

From Fig. 11, we can see that when the length of the tone pulse sequence reaches some large values, the system performance decreases slowly. This is because, on the one hand, when the length of the tone pulse sequence is large, further increasing it cannot contribute much to the decrease of the false alarm in the system. On the other hand, the duration for a RTS/Beacon/CTS handshaking process in our CUMAC increases slowly with the increase of the tone pulse sequence length, which contributes to the reduction of the system throughput. It is clear in Fig. 11 that the benefits of our tone pulse sequence mechanism is more significant when the input traffic is heavy. This is reasonable since with the increase of the input traffic, more RTS/Beacon/CTS handshaking processes will be initiated on the control channel. This will contribute to the increase of the false alarm probability. Consequently, our tone pulse sequence mechanism which aims to reduce the false alarm probability plays a more important role in the system.

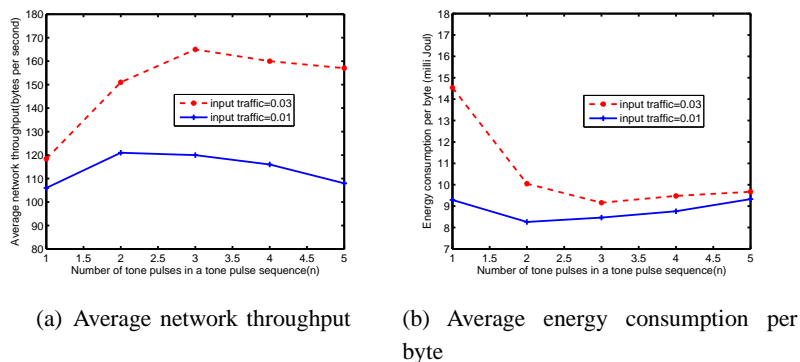


Fig. 11. Performance comparison with varying tone pulse sequence length

**False alarm and false approval probability:** In this set of simulations, we change the input traffic of every node from 0.002 to 0.06 packets per second. Fig. 12 shows that with the increase of the input traffic, both the false alarm and the false approval probability increase. Compared to the false alarm, the value of false approval is much smaller (almost 0). This means that in a network with relatively high node density, the RTS/Beacon/CTS handshaking process of CUMAC can almost eliminate the collision

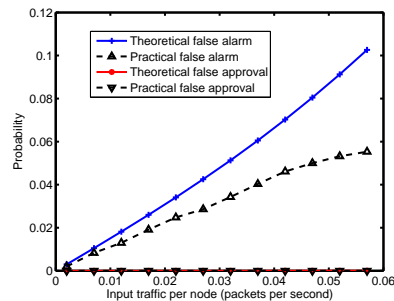


Fig. 12. False alarm and false approval: analysis vs simulation:

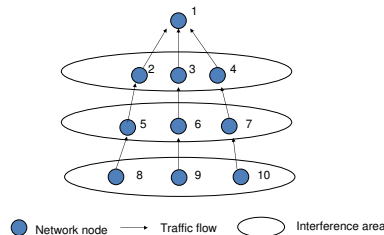


Fig. 13. A star topology multi-hop network

on the data channel, which results in high energy efficiency. As in Fig. 12, the false alarm probability is low in most cases, usually, less than 10%. To further improve the performance of our protocol, we can dynamically adjust the bandwidth of the control channel to minimize the false alarm probability, which is another interesting research topic and is our future work.

Fig. 12 demonstrates us that our theoretical analysis results in Section VI can be safely served as the upper bounds for both errors. Our upper bounds are tight when the input traffic is low and they become looser with the increase of the input traffic. This is because in our analysis, we assume that all nodes are on the control channel, which can be approximately satisfied when the input traffic is low. However, with the increase of the input traffic, more and more nodes are involved in the data transmission on the data channels rather than control channel, which contributes to the untightness of our upper bounds in the heavy traffic-loaded network.

2) *Results in star-topology networks:* In this set of simulations, we configure a multi-hop star network topology as shown in Fig. 13, where node 1 is the sink node. Node 8, 9 and 10 are the source nodes to generate traffic towards node 1. There are 4 channels in the network. A VBF-like routing protocol [25] is used here to route the traffic as show in Fig. 13. Nodes in the same interference area interfere with each other. This topology is quite similar to the topology around the sink node in a practical underwater network. For this topology, we use the end-to-end throughput to measure its performance, which is defined as the average number of successfully transmitted data bytes per second from the source nodes (i.e. 8, 9 and 10) to the sink node (node 1).

Fig. 14 demonstrates the advantages of CUMAC in such a practical multi-hop network environment. The performance of the random scheme in this network is poor (the end-to-end throughput is almost 0) and we ignore its results here. Compared to the RTS/CTS scheme, CUMAC can improve the end-to-end throughput by almost 40% with much higher energy efficiency. In this simulation, we also observed that although the simulated network is not dense (only 10 nodes), it gets saturated when the input traffic

of every source node is only about 0.05 packet per second. One of the main reasons for such a quick saturation lies in the limited capacity of the sink node. Here, although the sink node works as the traffic aggregator, it only has one acoustic transceiver. When the sink node is communicating with one node, the other nodes have to wait, which will greatly degrade the system performance. One way of solving this problem is to equip the sink node with multiple acoustic transceivers if the extra cost is allowable. We would like to explore this topic in our future work.

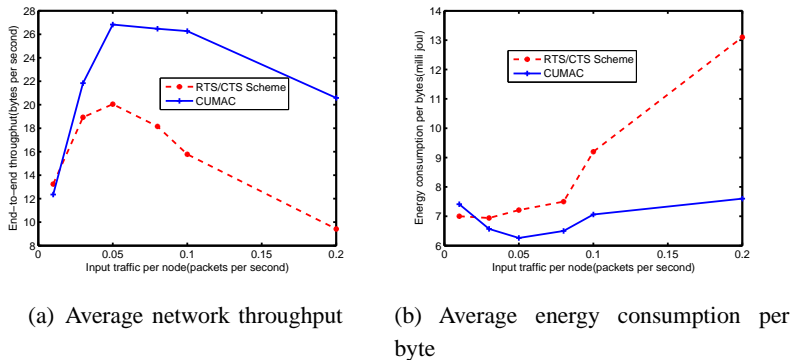


Fig. 14. Performance comparison with different input traffic in the star topology multi-hop network

### C. Summary

We have also conducted simulations to evaluate the performance of our CUMAC with other parameters such as the average packet length and the number of data channels in different network settings. Due to space limits, these results are not included here. Interested readers can refer to our technical report [26]. Our simulation results clearly show us that our CUMAC can achieve high performance in single-channel long-delay underwater sensor networks. Its RTS/Beacon/CTS handshaking process along with the cooperative collision detection can effectively suppress the hidden terminal problems. Although false alarm and false approval exist, they are bounded and usually negligible compared with the benefits of CUMAC. And the tone pulse sequence technique is an effective and efficient means to improve the performance.

## VIII. RELATED WORK

In this section, we first briefly review some related work on multi-channel MAC protocols for terrestrial radio wireless networks. Then, we discuss some research on multi-channel MAC protocols for underwater acoustic networks and show their differences from our work.

Multi-channel MAC protocols have long been investigated for terrestrial radio wireless networks. In [23], the authors propose a receiver initiated multi-channel MAC protocol based on frequency hopping for multi-hop wireless networks. Strict synchronization among nodes is needed for this protocol. The authors of [20] propose a multi-channel MAC protocol with a single transceiver. They identify the multi-channel hidden terminal problem and solve it with a synchronized MAC protocol which splits the time into channel negotiation phase and data transmission phase. In [8], the authors study the performance of multi-channel MAC protocols with Aloha-like reservation on a dedicated control channel for wireless networks. In [13], the authors analyze and compare four different multi-channel MAC protocols for single-hop wireless network. Their analysis and simulation results show that different protocols have different properties and

thus are preferred in different network scenarios. All these studies are conducted for radio communication where propagation delay is negligible.

Multi-channel MAC protocols for underwater acoustic networks have also aroused significant research interest recently. For example, in [19], a hierarchical multi-channel MAC protocol is proposed for clustered underwater networks where TDMA is used for the intra-cluster communication and CDMA is used for the inter-cluster communication. Strict synchronization among all nodes is needed in this scheme. In [22], the authors utilize CDMA as the multiple access technique. A RTS/CTS handshaking scheme is employed for every channel before actual data transmission. In this scheme, CDMA spreading codes are distributed first by some predefined algorithm and every node is assumed to get a unique spreading code among its one-hop neighbors. In [27], the authors analyze two generalized multi-channel MAC protocols for underwater sensor networks. The long propagation delay of underwater acoustic channel is taken into account in their analysis model. Their results demonstrate the great benefits of multi-channel MAC protocols. However, in [27], at least two acoustic transceivers are assumed to be available on every node. Different from all the existing work, this paper considers a cost-effective network architecture where one and only one transceiver is needed on each node. We identify the triple hidden terminal problems which are inherent in the single-transceiver multi-channel long-delay underwater scenario and propose a new protocol to effectively deal with them.

## IX. CONCLUSIONS AND FUTURE WORK

In this paper, we identified and investigated the triple hidden terminal problems in single-transceiver multi-channel long-delay underwater sensor networks. Based on our findings, we proposed CUMAC, a new multi-channel MAC protocol. By employing a novel cooperative collision detection scheme and a tone pulse sequence technique, CUMAC handles the triple hidden terminal problems well and greatly reduces the collision in the network. Simulation results show that CUMAC can significantly improve the network throughput with high energy efficiency in both one-hop and multi-hop networks.

As for the future work, we would like to pursue our studies in two directions: 1) Investigate the optimal bandwidth allocation strategy for the control channel and data channels. 2) Implement CUMAC in underwater sensor network testbeds and evaluate its feasibility and practicality in the real world.

## REFERENCES

- [1] Aqua-Sim: Underwater Acoustic Network Simulator. In <http://uwsn.engr.uconn.edu/aquasim.tar.gz>.
- [2] AquaNetwork: Underwater wireless modem with networking capability. In [http://bwww.dspcomm.com/products\\_aquanetwork.html](http://bwww.dspcomm.com/products_aquanetwork.html).
- [3] I. F. Akyildiz, D. Pompili, and T. Melodia. State of the Art in Protocol Research for Underwater Acoustic Sensor Networks. *ACM Mobile Computing and Communication Review*, 11:11–22, October 2007.
- [4] B. Benson, G. Chang, D. Manov, B. Graham, and R. Kastner. Design of a Low-cost Acoustic Modem for Morred Ocenographic Applications. In *Proceedings of ACM WUWNet'06*, pages 71–78, September 2006.
- [5] X. Cheng, H. Shu, and Q. Liang. A Range-differenece Based Self-positioning Scheme for Underwater Acoustic Sensor Networks. In *Proceedings of International Conference on Wireless Algorithms, Systems and Applications (WASA)*, volume 1, pages 38–43, Aug 2007.
- [6] N. Chirdchoo, W.-S. Soh, and K. C. Chua. Aloha-based MAC Protocols with Collision Avoidance for Underwater Acoustic Networks. In *Proceedings of INFOCOM'07, Mini-Symposium*, pages 2271 – 2275, 2007.
- [7] M. Chitre, S. Shahabudeen, and M. Stojanovic. Underwater Acoustic Communications and Networking: Recent Advances and Future Challenges. *Marine Technology Society Journal*, 42(1):103–116, Spring 2008.
- [8] Y. S. Han, J. Deng, and Z. J. Haas. Analyzing Multi-Channel Medium Access Control Schemes with ALOHA Reservation. *IEEE Transaction on Wireless Communications*, 5(8):2143–2152, August 2006.

- [9] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li. Research Challenges and Applications for Underwater Sensor Networking. In *Proceedings of IEEE Wireless Communications and Networking Conference*, pages 228–235, Las Vegas, Nevada, USA, Apr. 2006.
- [10] K. B. Kredo and P. Mohapatra. A Hybrid Medium Access Control Protocol for Underwater Wireless Networks. In *Proceedings of ACM WUWNet'07*, September 14 2007.
- [11] B. Li, S. Zhou, M. Stojanovic, L. Freitag, and P. Willett. Multicarrier Communication over Underwater Acoustic Channels with Nonuniform Doppler Shifts. *IEEE Journal of Oceanic Engineering*, 33(2), Apr. 2008.
- [12] L. Liu, S. Zhou, and J.-H. Cui. Prospects and Problems of Wireless Communications for Underwater Sensor Networks. *Wiley Wireless Communications and Mobile Computing, Special Issue on Underwater Sensor Networks*, Aug. 2008.
- [13] J. Mo, H.-S. Wilson, and J. Walrand. Comparison of MultiChannel MAC Protocols. *IEEE Transaction on Mobile Computing*, 7(1):50–65, January 2008.
- [14] M. Molins and M. Stojanovic. Slotted FAMA: A MAC Protocol for Underwater Acoustic Networks. In *Proceedings of the IEEE Oceans Conference*, pages 1–7, May 2006.
- [15] A. Nasipuri and J. Mondhe. Multi-channel MAC with Dynamic Channel Selection for Ad Hoc Networks. *Technical report*, <http://www.ece.uncc.edu/~anasipur/>, 2004.
- [16] J. Partan, J. Kurose, and B. N. Levine. A Survey of Practical Issues in Underwater Networks. In *Proceedings of ACM WUWNet'06*, Los angeles, CA, USA, Sept. 2006.
- [17] J. S. Pathmasuntharam, A. Das, and A. K. Gupta. Primary Channel Assignment based MAC(PCAM)- A Multi-Channel MAC Protocol for Multi-hop Wireless networks . In *IEEE Proceedings of Wireless Communications and Networking Conference(WCNC)*, pages 1110–1115, 2004.
- [18] B. Peleato and M. Stojanovic. Distance Aware Collision Avoidance Protocol for Ad-Hoc Underwater Acoustic Sensor Networks. *IEEE Communication Letters*, 11(12):1025–1027, December 2007.
- [19] F. Salva-Garau and M. Stojanovic. Multi-Cluster Protocol for Ad Hoc Mobile Underwater Acoustic Networks. In *Proceedings of the IEEE Oceans Conference*, pages 91–98, 2003.
- [20] J. So and N. Vaidya. Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using A single Transceiver. In *Proceedings of ACM MobiHoc*, pages 222–233, May 2004.
- [21] A. Syed, W. Ye, and J. Heidemann. T-Lohi: A New Class of MAC Protocol for Underwater Acoustic Sensor Networks. In *Proceedings of IEEE INFOCOM'08*, 2008.
- [22] H.-X. Tan and W. K. G. Seah. Distributed CDMA-based MAC protocol for Underwater Sensor Networks. In *Proceedings of the 32th IEEE Conference on Local Computer Networks*, pages 26–36, 2007.
- [23] A. Tzamaloukas and J. J. Garcia-Luna-Aceves. A Receiver-Initiated Collision-Avoidance Protocol for Multi-Channel Networks. In *Proceedings of IEEE INFOCOM*, pages 189–198, 2001.
- [24] S.-L. Wu, C.-Y. Lin, Y.-C. Tseng, and J.-P. Sheu. A New Multi-channel MAC Protocol with On-demand Channel Assignment for Multi-hop Mobile Ad Hoc Networks. In *Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN)*, May 2000.
- [25] P. Xie, L. Lao, and J.-H. Cui. VBF: Vector-based Forwarding Protocol for Underwater Sensor Networks. In *Proceedings of IFIP Networking*, May 2006.
- [26] Z. Zhou, Z. Peng, J.-H. Cui, and Z. Jiang. Handling Triple Hidden Terminal Problems for Multi-Channel MAC in Long-Delay Underwater Sensor Networks. *UCONN CSE Technical Report: UbiNet-TR09-02, August 2009. URL: http://www.cse.uconn.edu/~jcui/publications.html*.
- [27] Z. Zhou, Z. Peng, J.-H. Cui, and Z. Shi. Analyzing Multi-channel MAC Protocols for Underwater Acoustic Sensor Networks. *UCONN CSE Technical Report: UbiNet-TR08-02, August 2008. URL: http://www.cse.uconn.edu/~jcui/publications.html*.