BiC-MAC: Bidirectional-Concurrent MAC Protocol with Packet Bursting for Underwater Acoustic Networks

Hai-Heng Ng, Wee-Seng Soh, Mehul Motani
Department of Electrical and Computer Engineering
National University of Singapore
Email: {g0600158, elesohws, motani}@nus.edu.sg

***Abstract*—Underwater communication mainly relies on acoustic waves. Its unique characteristics, such as slow propagation speed and low bit rate-distance product, present both challenges and opportunities for Media Access Control (MAC) protocol design. In existing sender-initiated handshaking based MAC protocols, each successful handshake only allows an initiating sender to transmit single or multiple consecutive packets to its intended receiver. In a long propagation delay environment, these unidirectional transmissions often result in poor channel utilization. By exploiting the channel's characteristics, we present a novel approach of concurrent, bidirectional data packet exchange to improve the data transmission efficiency. To further amortize the high latency overhead, we adopt a packet bursting idea, in which a sender-receiver (S-R) node pair can exchange multiple rounds of bidirectional packet transmissions. We then design an asynchronous handshaking based MAC protocol, which we call BiC-MAC (Bidirectional-Concurrent MAC with packet bursting). Via extensive simulations and comparisons, we show that BiC-MAC can significantly increase the channel utilization and offer performance gains in terms of throughput and delay.**

I. Introduction

Compared to its terrestrial counterpart, underwater MAC protocol design faces even more formidable challenges due to the unique characteristics of acoustic communication such as slow propagation speed and low bit rate-distance product [1]. The low bandwidth capacity implies that a single-channel protocol is more preferable, while slow propagation speed calls for new packet transmission methodology.

Among the existing MAC protocols, there is a strong focus on sender-initiated handshaking based protocols (i.e., Request-to-Send (RTS)/Clear-to-Send (CTS) based protocols), as they can offer benefits such as the carrying of useful information in the control packets, and the ability to alleviate hidden node problem. Sozer *et al.* [2] propose a 3-way RTS/CTS/DATA handshake MAC protocol with error detection via Stop-and-Wait Automatic Repeat Request (ARQ). In [3], Molins and Stojanovic propose the Slotted-FAMA, which uses a 4-way RTS/CTS/DATA/ACK handshake with a time-slotting mechanism to avoid any data collision. Peleato and Stojanovic [4] propose another 3-way handshake protocol called DACAP. It allows a sender to use different handshake lengths for different receivers so as to minimize the average handshake duration. Recently, Guo *et al.* [5] introduce a 3-way handshake protocol called APCAP, which seeks to improve channel utilization by allowing a sender to initiate another handshake with a different receiver, while waiting for the CTS reply. In [8], we propose the MACA-U protocol to serve as a more appropriate benchmarking protocol; its key feature is the incorporation of state transition rules to account for the long propagation delay. The aforementioned MAC protocols only transmit a single data packet unidirectionally upon each successful handshake; coupled with the multi-way control packet exchange, the channel utilization is usually poor. To improve the performance, several works utilize the packet train technique, where a sender can transmit multiple back-to-back packets for each successful handshake. Shahabudeen *et al.* [6] propose the MACA-MCP, where they assume that each node simultaneously uses three acoustic modems that operate at different frequency bands and different ranges. It employs a 4-way handshake, with packet train enhancement. Chirdchoo *et al.* [7] propose the MACA-MN, which sends multiple data packets to multiple neighbors in a 3-way handshake. In [9], we present the ROPA protocol, where a sender can invite its one-hop neighbors (appenders) to opportunistically append their own packets. The sender can coordinate the packet appending so that those packets will arrive in a collision-free packet train manner. Although all these packet-train-based protocols can improve the performance to some extent, the data transmission is still inefficient due to the high latency overhead. More importantly, the unidirectional transmission approach fails to fully exploit the opportunities offered by the underwater acoustic channel.

Here, we present a novel approach to improve the channel utilization. Our idea is inspired by the observation that, in underwater networks, two nodes may transmit to each other at around the same time, and yet, not collide with each other. In this case, the transmission duration of each data packet should be much shorter than the inter-nodal propagation delays; fortunately, this condition can often be easily satisfied in underwater. We then propose the idea of *"concurrent, bidirectional data packet exchange"* to optimize the data transmission efficiency, in which a sender-receiver (S-R) node pair is allowed to transmit data packets to each other for every successful handshake. This bidirectional transmission should occur in a concurrent manner so that the transmissions are tightly packed. To further amortize the high latency overhead, we adopt a *"packet bursting"* idea that allows the S-R node pair to exchange multiple rounds of bidirectional packet transmissions. This is different from the normal packet train approach; in our approach, the entire set of data packets are actually transmitted over several discontinuous packet bursts.

Based on the above ideas, we propose a novel asynchronous single-channel handshaking based MAC protocol, which we call BiC-MAC. Although the handshake is sender-triggered, the intended receiver can exploit this opportunity to initiate potentially multiple rounds of concurrent, bidirectional data exchange when it has data packets in return. By using this approach, both nodes in the S-R node pair share only one set of communication overhead, e.g., backoff time, delay incurred by handshake, etc., and the proportion of time spent on control signaling is significantly reduced. Furthermore, the data packet transmission efficiency is greatly enhanced because both nodes can access the channel simultaneously after the floor reservation. Note that, a versatile MAC framework is conceived so that BiC-MAC can operate in three possible bidirectional transmission modes to cater for the fact that the S-R node pair may not intend to exchange the same number of packets. Lastly, if the receiver does not have any data packet in return, normal unidirectional packet transmissions can still be performed. In short, MACA with packet train can be viewed as a subset of BiC-MAC. To the best of our knowledge, BiC-MAC is the first handshaking based MAC protocol that utilizes a comprehensive concurrent, bidirectional transmission MAC framework for exchanging data packet bursts in underwater acoustic networks.

The remainder of this paper is organized as follows. In Section II, we describe the BiC-MAC protocol in detail. Next, we present our simulations and results in Section III. Finally, we give our conclusions and future work in Section IV.

II. Protocol Design

A. General Assumptions

We consider static multi-hop underwater acoustic networks where each node is equipped with a single half-duplex underwater acoustic modem. However, the "static" nodes still have some limited degree of movement as they are typically anchored to the seabed, and subjected to a maximum sway distance caused by underwater currents. Every node is assumed to know its estimated propagation delay from each of its one-hop neighbors, as well as each of these neighbors' maximum propagation delay from the latter's one-hop neighbors. During network initialization, this information is estimated via round-trip time measurements of control packets, and disseminated to the neighbors. Note that any estimation errors or fluctuations caused by sway movements are typically very small as compared to the propagation delays. In addition, our protocol can accommodate such errors via the use of small guard times.

B. How the BiC-MAC Protocol Works

Table I defines the notations used in explaining the BiC-MAC protocol, while Fig. 1(a)–1(c) depict the timing diagrams of BiC-MAC for the three typical scenarios.

1) ***Channel Reservation Phase:*** As can be seen, the three typical scenarios have identical sequence of control packet exchanges. In BiC-MAC, an idle node adopts a hybrid of "batch-by-size" and "batch-by-time" strategies to determine when to trigger its RTS attempt. After the triggering condition

TABLE I
NOTATIONS USED FOR EXPLAINING THE BIC-MAC PROTOCOL

Notation	Description
T_{DATA}	Transmission time of each fixed-length data packet
T_x	Transmission time of each fixed-length control packet of type x, where $x \in \{\text{RTS}, \text{CTS}, \text{NTF}\}$
τ_{max}	Maximum propagation delay that corresponds to a node's maximum transmission range
$\tau_{\text{max},i}$	Maximum propagation delay between node i and its one-hop neighbors
$\tau_{i,j}$	Inter-nodal propagation delay between node i and node j
S_{burst}	Threshold number of accumulated data packets for triggering an RTS attempt
T_{max}	Time threshold for triggering an RTS attempt
t_{ref}	Starting reference time for the S-R node pair's bidirectional data exchange
$d_{\text{wts},i}$	Wait-to-Start; duration that node i must wait before engaging in the bidirectional data exchange
$d^{x}_{\text{silent},i}$	Duration that node i must remain silent after overhearing a type x control packet, where $x \in \{\text{RTS}, \text{CTS}, \text{NTF}\}$
$d_{\text{busy},x}$	Busy duration information that is carried within a type x control packet, where $x \in \{\text{RTS}, \text{CTS}, \text{NTF}\}$
$d^{\text{rx}}_{\text{busy},i}$	Busy duration that node i spent from t_{ref} until it finishes receiving the last intended data packet
$d_{\text{toc},i}$	Time-of-Completion; duration that node i spent from t_{ref} until it is released from the handshake, i.e., upon completing its bidirectional data transmissions/receptions
T_{guard}	Guard time to accommodate any inter-nodal propagation delay's estimation error, and the transceiver's transmit-receive turnaround time
k_i	Number of data packets that node i transmits either in a Complete Round (CR) or a Residual Round (RR)
k_{max}	Maximum number of data packets that a node is allowed to transmit in a CR so as to avoid TX-RX collisions
n_{CR}	Total number of CRs required in the current bidirectional data exchange
d_{CR}	Duration of a single CR in the current bidirectional data exchange
d_{RR}	Duration of a single RR in the current bidirectional data exchange
n_i	Total number of data packets (both relayed and new) that node i intends to transmit in current handshake
$n_{x,i}$	Number of data packets of type x that node i intends to transmit in current handshake, where $x \in \{\text{relay}, \text{new}\}$
$\mathcal{N}_i$	Set of node i's one-hop neighboring nodes

has been satisfied, the sender shall initiate its contention timer according to the Binary Exponential Backoff (BEB) algorithm, and then broadcast an RTS packet to its one-hop neighbors upon the timer expiry (see Section II-C). In the RTS packet, the sender announces its intended total number of data packets to be transmitted for the current handshake, n_{S}. Since a node may transmit both relayed and self-originated traffic, n_{S} can be specified as two separate bit-fields, i.e., $n_{\text{relay,S}}$ and $n_{\text{new,S}}$.

Upon receiving the RTS packet, a receiver checks whether it has data packets in return, before responding with a CTS packet. Note, however, that the intended receiver may only transmit the CTS response provided it is currently not involved in any other handshake, and is also not required to remain silent. In the scenario where the receiver is interested in

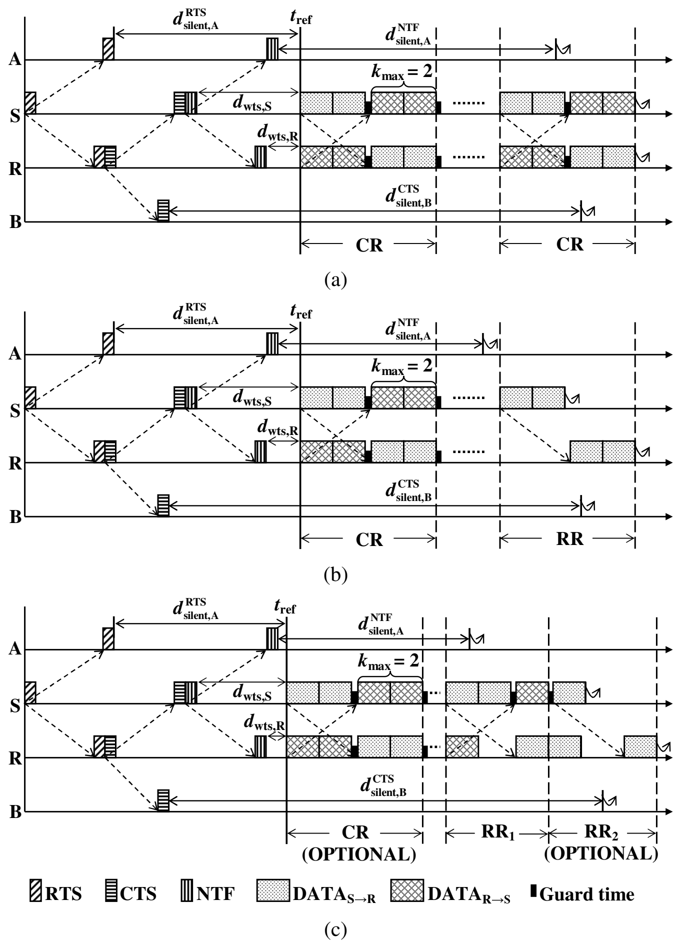

$d_{\text{silent,A}}^{\text{RTS}}$
t_{ref}
$d_{\text{silent,A}}^{\text{NTF}}$
$k_{\max} = 2$
$d_{\text{wts,S}}$
$d_{\text{wts,R}}$
$d_{\text{silent,B}}^{\text{CTS}}$
A
S
R
B
CR
RR
RR$_1$
RR$_2$
(OPTIONAL)
(a)
(b)
(c)
RTS
CTS
NTF
DATA$_{\text{S}\to\text{R}}$
DATA$_{\text{R}\to\text{S}}$
Guard time

Fig. 1. Timing diagrams of BiC-MAC: (a) Type 1 scenario, (b) Type 2 scenario, (c) Type 3 scenario. Nodes "S" and "R" refer to the initiating sender and its intended receiver, respectively; while nodes "A" and "B" are the one-hop neighbors of nodes "S" and "R", respectively. A small constant guard time can be used to accommodate any propagation delay's estimation error, as well as any slight difference in the perception of t_{ref} by both "S" and "R".

bidirectional packet transmissions, it shall announce the total number of data packets that it wishes to transmit, n_{R}, in the CTS packet by specifying both $n_{\text{relay,R}}$ and $n_{\text{new,R}}$. As long as $n_{\text{R}} > 0$, the condition is sufficient for the bidirectional data exchange. Furthermore, we also limit $n_{\text{R}} \leq n_{\text{S}}$ for simplicity. On the other hand, if the receiver node does not have any data packet in return, both $n_{\text{relay,R}}$ and $n_{\text{new,R}}$ fields are set to zero. Consequently, the handshake reduces to the conventional 3-way handshake, i.e., MACA with unidirectional packet train. Since the concurrent, bidirectional data exchange demands the participating nodes (S-R node pair) to cycle between packet transmissions and receptions, the receiver must also compute its data transmission/reception schedule. We will explain how to compute a proper schedule in the following section, but it should be noted for now that both participating nodes need to be aware of when and how many data packets can be exchanged in order to avoid transmit-receive (TX-RX) collisions. Finally, a sender (receiver) node also locally computes its expected busy duration, i.e., $d_{\text{busy,RTS}}$ ($d_{\text{busy,CTS}}$), and includes it in the broadcasted RTS (CTS) packet. Any neighboring node that overhears the control packet shall extract the embedded busy duration, and then locally compute its silent duration. Subsequently, it needs to defer initializing any transmission so as to avoid packet collisions with the ongoing transmissions.

Upon receiving the CTS and validating that $n_{\text{R}} > 0$, the sender will compute its data transmission schedule as well. In our protocol, a new type of control packet is introduced, namely, the Notification (NTF) packet. The functions of an NTF packet are two-fold. Firstly, a sender can include its busy duration, $d_{\text{busy,NTF}}$, in the control packet, so that its first-hop neighbors, $\mathcal{N}_{\text{S}}$, could compute their respective silent durations. As illustrated in Fig. 1(a)–1(c), the sender's first-hop neighbors should extend their silent durations, so that the sender can successfully receive the incoming bidirectional data packets. Secondly, the information it carries synchronizes both sender and receiver to a common starting reference time, t_{ref}, such that concurrent transmissions could be performed. Note that BiC-MAC does not require global time synchronization, as the node pair merely keeps track of the offset duration. Specifically, after transmitting an NTF packet, the sender will compute, and wait for a duration of $d_{\text{wts,S}}$ (wait-to-start) before it starts to transmit its data packets. Similarly, upon receiving an NTF packet, the receiver can locally compute its mandatory waiting duration, $d_{\text{wts,R}}$, because it has the knowledge of their inter-nodal propagation delay ($\tau_{\text{S,R}}$), as well as the maximum propagation delay between the sender and its one-hop neighbors ($\tau_{\text{max,S}}$). Moreover, these waiting durations also allow the sender's most distant neighbor to completely overhear the NTF packet. The wait-to-start durations can be expressed as

$$\begin{cases} d_{\text{wts,S}} = \tau_{\text{max,S}} = \max_{j \in \mathcal{N}_{\text{S}}}(\tau_{\text{S},j}), \\ d_{\text{wts,R}} = \tau_{\text{max,S}} - \tau_{\text{S,R}}. \end{cases} \tag{1}$$

Alternatively, if every node only knows the propagation delays from its one-hop neighbors, a sender can include its $\tau_{\text{max,S}}$ in the broadcasted RTS packet for every handshake, so that the receiver can use this information subsequently. This will further relax our assumptions in Section II-A.

As mentioned, the purpose of the busy durations is to inform all neighboring nodes of the S-R node pair so that they can defer their transmissions accordingly. This is a vulnerable period where any packet transmissions from the neighboring nodes may cause harmful interference to the ongoing bidirectional transmissions. In the following, we first show how an S-R node pair can compute their respective busy durations:

$$\begin{cases} d_{\text{busy,RTS}} = T_{\text{CTS}} + 2\tau_{\text{S,R}} + T_{\text{NTF}} + \tau_{\text{max,S}}, \\ d_{\text{busy,CTS}} = \tau_{\text{S,R}} + T_{\text{NTF}} + \tau_{\text{max,S}} + d_{\text{busy,R}}^{\text{rx}}, \\ d_{\text{busy,NTF}} = \tau_{\text{max,S}} + d_{\text{busy,S}}^{\text{rx}}, \end{cases} \tag{2}$$

where $d_{\text{busy},i}^{\text{rx}}$ is the busy duration that a participating node i spends from t_{ref} until it finishes receiving its last intended data packet (the computation will be detailed in Section II-B2).

Upon overhearing a control packet and extracting the busy duration it carries, every neighbor can compute its silent duration locally. To optimize the performance, the silent duration does not need to be as long as the busy duration, because of the inter-nodal propagation delays between the participating nodes and overhearing neighbors. Each neighbor only needs to ensure that any transmission after its silent duration will not interfere with the ongoing transmissions. Hence, the respective silent durations can be computed as

$$\begin{cases} d^{\text{RTS}}_{\text{silent},i} = d_{\text{busy,RTS}} - \tau_{\text{S},i}, & i \in \mathcal{N}_{\text{S}}, \\ d^{\text{CTS}}_{\text{silent},j} = d_{\text{busy,CTS}} - 2\tau_{\text{R},j}, & j \in \mathcal{N}_{\text{R}}, \\ d^{\text{NTF}}_{\text{silent},i} = d_{\text{busy,NTF}} - 2\tau_{\text{S},i}, & i \in \mathcal{N}_{\text{S}}. \end{cases} \tag{3}$$

Note that a successful NTF packet's reception at the sender's first-hop neighbors is imperative to ensure that an uninterrupted bidirectional transmission can be performed. Hence, the $d_{\text{busy,RTS}}$ duration should be sufficiently large to cover till t_{ref}, so that the neighboring nodes' silent durations can be extended via overhearing the subsequent NTF packet, if any.

*2) **Bidirectional Concurrent Data Transmission Phase**:* As mentioned, the bidirectional transmissions in the BiC-MAC protocol can be classified into three scenarios, namely, Type 1, Type 2, and Type 3. The reason we conceive this versatile MAC framework is because it is likely that both sender and receiver may not intend to transmit the same amount of data packets. To better facilitate our explanation, we introduce the terms *Complete Round (CR)* and *Residual Round (RR)*.

Definition 1: A Complete Round (CR) is defined as the time window over which both nodes in the S-R node pair transmit the maximum allowable number of data packets, $k_{\max}$, to each other through a concurrent, bidirectional data packet transmission approach.

Definition 2: A Residual Round (RR) is defined as the time window over which either (i) only one of the nodes in a S-R node pair transmits its data packets, or, (ii) both nodes transmit to each other through a concurrent, bidirectional data packet transmission approach, but at least one of the nodes transmits less than $k_{\max}$ data packets.

To improve BiC-MAC's reliability, a small constant guard time, T_{guard}, can be inserted every time when a node switches between transmit and receive modes during the data transmission phase; this guard time will accommodate for: (i) any estimation error in the inter-nodal propagation delays, (ii) transceiver's transmit-receive turnaround time, (iii) maximum sway distance caused by underwater currents, and (iv) any slight difference in the S-R node pair's perception of starting reference time, t_{ref}. In general, the required guard time is typically very small (in the order of tens of milliseconds) compared to the data packet's transmission time. It is worth mentioning that the use of bidirectional transmissions allows BiC-MAC to correct any possible error in the inter-nodal delay estimates; a sender could examine an incoming data packet's arrival time in a CR, and use the deviation from the expected arrival time for the correction.

To enhance BiC-MAC's efficiency, the bidirectional transmissions' time-of-completion of the S-R node pair should be minimized, subject to the constraint that TX-RX collisions do not occur. Generally, the duration of $d_{\text{toc},i}$ consists of multiple rounds of d_{CR} and a d_{RR}, and can be expressed as

$$d_{\text{toc},i} = n_{\text{CR}} d_{\text{CR}} + d_{\text{RR}}, \quad i \in \{\text{S}, \text{R}\}. \tag{4}$$

In each CR, as well as any RR that has concurrent, bidirectional transmission, the condition to avoid TX-RX collisions in each round is

$$k_i T_{\text{DATA}} + T_{\text{guard}} \le \tau_{\text{S,R}}, \quad i \in \{\text{S}, \text{R}\}. \tag{5}$$

From (5), it is straightforward to compute the maximum allowable number of data packets to be sent in each CR using

$$k_{\max} = \left\lfloor \frac{\tau_{\text{S,R}} - T_{\text{guard}}}{T_{\text{DATA}}} \right\rfloor. \tag{6}$$

To pack the transmissions as tightly as possible, the participating nodes must concurrently transmit $k_i = k_{\max}$ data packets in every CR. Thus, the S-R node pair shall expect identical CR duration for all three scenarios, i.e., $d_{\text{CR}} = \tau_{\text{S,R}} + k_{\max} T_{\text{DATA}}$. Fig. 1(a)–1(c) illustrate the example scenarios when $k_{\max} = 2$. Subsequently, the number of CRs required for the current handshake, n_{CR}, can be found using

$$n_{\text{CR}} = \lfloor \min(\alpha_{\text{S}}, \alpha_{\text{R}}) \rfloor, \quad \text{where } \alpha_i = n_i / k_{\max}. \tag{7}$$

When $n_{\text{R}} \le n_{\text{S}}$, it can be further reduced to $n_{\text{CR}} = \lfloor n_{\text{R}} / k_{\max} \rfloor$.

Next, we describe below the three distinct scenarios:

Type 1: It is characterized by the presence of at least one CR, and no RR. Here, $n_{\text{S}} = n_{\text{R}}$. Also, the time-of-completion and the busy duration can be expressed as

$$\begin{aligned} d_{\text{toc},i} &= d^{\text{rx}}_{\text{busy},i} \\ &= n_{\text{CR}} d_{\text{CR}} + (n_{\text{CR}} - 1) T_{\text{guard}}, \quad i \in \{\text{S}, \text{R}\}. \end{aligned} \tag{8}$$

Type 2: It is characterized by the presence of at least one CR, and a single RR. Here, $n_{\text{S}} > n_{\text{R}}$. In the RR, the sender shall transmit its remaining data packets unidirectionally. Similarly, the time-of-completion and the busy duration can be expressed as

$$\begin{cases} d^{\text{rx}}_{\text{busy,S}} = n_{\text{CR}}(d_{\text{CR}} + T_{\text{guard}}), \\ d_{\text{toc,S}} = d^{\text{rx}}_{\text{busy,S}} + (n_{\text{S}} - n_{\text{CR}} k_{\max}) T_{\text{DATA}}, \\ d_{\text{toc,R}} = d^{\text{rx}}_{\text{busy,R}} = d_{\text{toc,S}} + \tau_{\text{S,R}}. \end{cases} \tag{9}$$

Type 3: It is characterized by the presence of optional CRs, and a single RR. Unlike Type 2 scenario, the RR here is further divided into two sub-residual rounds – a mandatory RR_1 and an optional RR_2. In particular, the optional RR_2 is present only when a sender node has $n_{\text{S}} > (n_{\text{CR}} + 1) k_{\max}$ number of data packets to be transmitted. Notice that a receiver node always transmits $k_{\text{R}} < k_{\max}$ data packets in RR_1. As soon as the sender finishes receiving its data packets in RR_1 (after T_{guard}), it shall transmit its excess data packets unidirectionally in RR_2, if any. Here, $n_{\text{S}} > n_{\text{R}}$. Also, the time-of-completion and the busy duration can be expressed as

$$\begin{cases} d^{\text{rx}}_{\text{busy,S}} = n_{\text{CR}}(d_{\text{CR}} + T_{\text{guard}}) + (n_{\text{R}} - n_{\text{CR}} k_{\max}) T_{\text{DATA}} \\ \qquad + \tau_{\text{S,R}}, \\ d_{\text{toc,S}} = \begin{cases} d^{\text{rx}}_{\text{busy,S}} + (n_{\text{S}} - (n_{\text{CR}} + 1) k_{\max}) T_{\text{DATA}} \\ + T_{\text{guard}}, & \text{if RR}_2 \text{ is present}, \\ d^{\text{rx}}_{\text{busy,S}}, & \text{otherwise}, \end{cases} \\ d_{\text{toc,R}} = d^{\text{rx}}_{\text{busy,R}} \\ \quad = \begin{cases} d_{\text{toc,S}} + \tau_{\text{S,R}}, & \text{if RR}_2 \text{ is present}, \\ n_{\text{CR}}(d_{\text{CR}} + T_{\text{guard}}) + (n_{\text{S}} - n_{\text{CR}} k_{\max}) T_{\text{DATA}} \\ + \tau_{\text{S,R}}, & \text{otherwise}. \end{cases} \end{cases} \tag{10}$$

Finally, in order to prevent an S-R node pair from reserving the channel for an unreasonably long duration, each node is permitted to transmit at most S_{burst} data packets for each successful handshake. This also leads to shorter delays. Another issue that may arise in an actual deployment scenario is that, the sender-receiver separation might be too close at times, resulting in $\tau_{\text{S,R}} < T_{\text{DATA}} + T_{\text{guard}}$. When this occurs, it is more beneficial to perform unidirectional packet transmissions, as the communication overheads involved are much smaller.

C. RTS Attempt Triggering and Backoff Algorithm

The BiC-MAC protocol utilizes a hybrid of "batch-by-size" and "batch-by-time" strategies to determine when to trigger its RTS attempt. For the "batch-by-size", a sender accumulates at least S_{burst} data packets that are destined for a particular neighbor before triggering its RTS attempt. The composition of S_{burst} may consist of data packets from the high priority relayed traffic, as well as the low priority self-originated traffic. Note that a node can only transmit a maximum of S_{burst} data packets for every successful handshake, even if it has accumulated more than S_{burst} packets. On the other hand, the "batch-by-time" strategy demands a sender to initiate its RTS attempt when a time duration of $T_{\max}$ has already elapsed since the release from previous handshake, provided it has at least one accumulated packet. Note that a node only maintains a single global timer, instead of separate timers for different buffers. This "batch-by-time" strategy will help to reduce the packet delays when the load is low. Besides satisfying either one of the above two triggering conditions, a node also must not have been required by some other nodes to remain silent, nor currently engaged in any other handshake. Otherwise, it shall defer its RTS attempt until it can start doing so.

As mentioned earlier, a simple BEB algorithm is adopted in the BiC-MAC protocol. In the BEB algorithm, each node doubles its backoff counter, B_{cnt}, in the event of an RTS failure, with an upper bound of $B_{\max}$; on the other hand, a node resets its backoff counter to a minimum value of $B_{\min}$, upon a successful handshake. Consequently, the backoff interval, T_{bk}, of the node can be expressed as

$$T_{\text{bk}} = \textit{uniform}\{0, B_{\text{cnt}}\} \times \tau_{\max}. \tag{11}$$

Upon satisfying either one of the two RTS triggering conditions, an idle node will initialize its contention timer according to T_{bk}, and only transmit its RTS packet upon the timer expiry.

III. Simulations and Results

A. Simulation Model

We have developed our own C++ discrete event-driven network simulator. As shown in Fig. 2, our multi-hop network topology has 36 static nodes with a grid spacing of 2000 m. The maximum transmission range is 1.75 times the grid spacing. Hence, each node has exactly 8 one-hop neighbors and 16 two-hop neighbors. A wrap-around strategy is used to distribute network load evenly and eliminate boundary effects. Each node generates its data packets according to the Poisson distribution, and randomly picks one of the 16 two-hop neighbors as a destination. Every node has a half-duplex omni-directional transceiver. The acoustic propagation speed is fixed at 1500 m/s. An error-free channel is also assumed, thus all packet losses are caused by packet collisions. The transmission rate is 4800 bps and the data packet length is set to 1200 bits. For BiC-MAC's parameters, we set $T_{\text{guard}} = 10$ ms, $B_{\min} = 1$, and $B_{\max} = 32$. Also, the lengths of the RTS, CTS, and NTF packets are 104, 104, and 88 bits, respectively. A node maintains two buffers (for relayed and new packets) for each of its one-hop neighbors, where each buffer can hold 100 packets. The values of S_{burst} and $T_{\max}$ are set to 130 and 100 s, respectively, unless otherwise stated. Lastly, we do not put any upper limit on the number of retries when RTS attempts fail. To avoid any transient effect, the simulation results are collected from 2×10^4 s to 1×10^5 s.

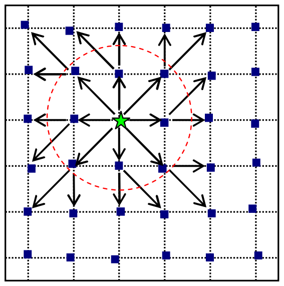


Fig. 2. The multi-hop network topology used in our simulations. Note that every other node assumes the same static routing pattern as the star node. Also, each node is allowed to randomly deviate from the grid intersection point by a maximum of 10% of the grid spacing, so as to introduce randomness.

B. Simulation Results

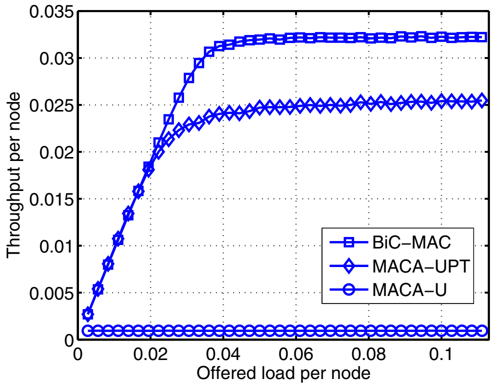
We compare BiC-MAC against two unidirectional handshaking based protocols, namely, the MACA-U [8], and MACA-U with packet train (MACA-UPT). These protocols' RTS and CTS control packets are assumed to be 88-bit long. We use the following metrics for our comparison:

- **Throughput per node**: It is defined as

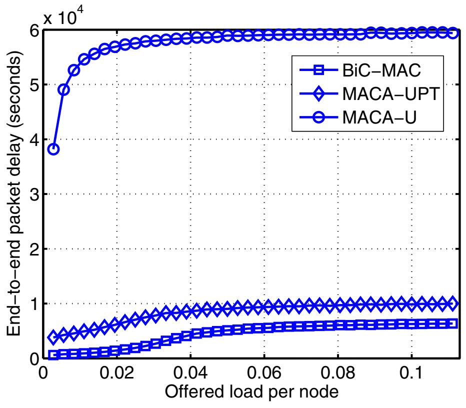
$$\gamma = \frac{1}{36}\left[\frac{\text{No. of packets received} \times \text{Packet length}}{\text{Transmission rate} \times \text{Simulation duration}}\right].$$

- **End-to-end data packet delay**: It is defined as the duration from which a data packet arrives at a source's buffer, until it is successfully received at its destination.

1) Comparison against unidirectional based MAC protocols: Fig. 3(a) and 3(b) show that both BiC-MAC and MACA-UPT protocols outperform MACA-U in terms of throughput per node, and end-to-end packet delay due to the use of packet burst mechanism. As expected, the MACA-U's throughput is unacceptably low, at around 0.001. Clearly, it is costly to transmit only a single data packet for every successful handshake in a long propagation delay environment since the proportion of time spent on the multi-way control packet exchange will become very significant. For the same reason, MACA-U has an extremely poor delay performance as well. Note that in the MACA-UPT, since there is no ACK involved, the first-hop neighbors of the sender only need to remain silent for a short duration upon overhearing an RTS packet. Thus,

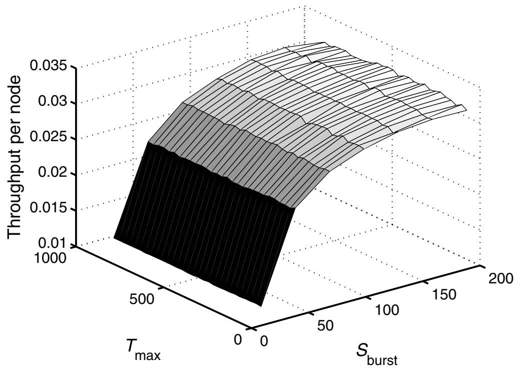


(a)

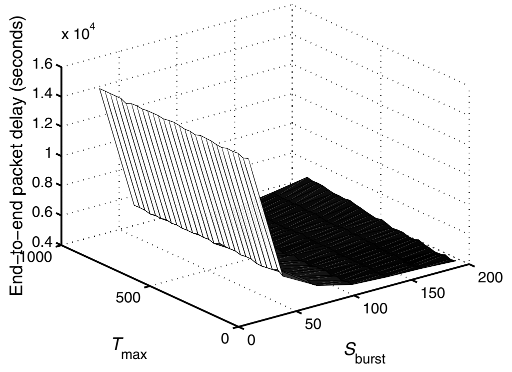


(b)

Fig. 3. (a) Throughput per node comparisons for various schemes, (b) end-to-end data packet delay comparisons for various schemes.



(a)



(b)

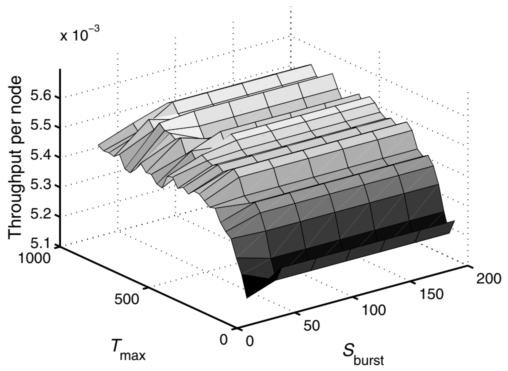
Fig. 4. Effects of varying S_{burst} and $T_{\max}$ on the BiC-MAC protocol's: (a) throughput per node when the offered load per node $= 0.0417$, (b) end-to-end data packet delay when the offered load per node $= 0.0417$.

MACA-UPT benefits from more concurrent transmissions in the neighborhood as those neighboring nodes can participate in a new handshake sooner. In contrast, BiC-MAC requires them to extend their silent durations till the end of bidirectional packet exchange; nonetheless, the bidirectional property of BiC-MAC allows it to outperform the unidirectional MACA-UPT by further increasing the throughput per node. Furthermore, BiC-MAC has lower delay than MACA-UPT across all offered load ranges. Although BiC-MAC employs a four-way handshake, as opposed to MACA-UPT's three-way handshake, the performance gain has outweighed the overhead incurred by the use of an additional NTF packet. These aforementioned results confirm that a bidirectional transmission approach is more efficient in long propagation delay environments, especially when operating at high load where the intended receiver can easily accumulate sufficient packets destined to the initiating sender. Without using this approach, the S-R node pair needs two sets of handshake to send data packets to each other, as opposed to BiC-MAC which requires only one.

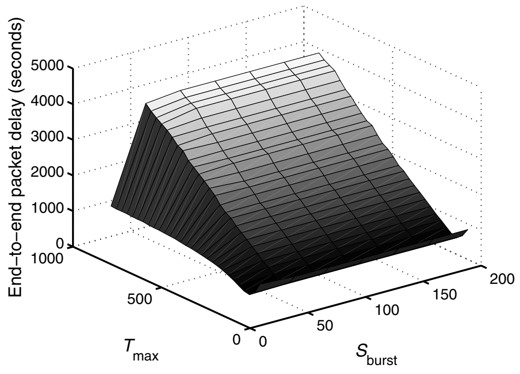
2) Effects of varying S_{burst} and T_{max}: We now study the effects of varying S_{burst} and $T_{\max}$ for two operating load regimes, i.e., offered load per node of: (i) 0.0056 (low load), and (ii) 0.0417 (high load). S_{burst} and $T_{\max}$ are varied from 10 to 190, and 10 to 850 s, respectively.

As shown in Fig. 4(a), when BiC-MAC operates at high load, its throughput increases rapidly as S_{burst} grows, but it eventually stabilizes. Similarly, its delay decreases rapidly as S_{burst} grows, but it eventually stabilizes as well, as shown in Fig. 4(b). These observations can be explained as follows. As S_{burst} gradually increases, a node can transmit more data packets for each successful handshake so that the proportion of time spent on the control packets exchange becomes less significant; thus, both throughput and delay improve accordingly. However, as S_{burst} grows further, this effect will diminish; this is because it becomes harder to meet the triggering condition of "batch-by-size", and thus the RTS attempt is more and more often triggered by the "batch-by-time" mechanism. Therefore, $T_{\max}$ must be carefully chosen as it will affect the protocol's performance when "batch-by-size" strategy loses its effects beyond a certain large S_{burst} value.

As shown in Fig. 5(a), when BiC-MAC operates at low load, its throughput and delay remain almost consistent across all S_{burst} range. This is because for the low packet arrival rate, it is very hard to meet the "batch-by-size" triggering condition, especially when S_{burst} is set to a large value; therefore, the RTS attempt is predominantly triggered by "batch-by-time", and varying S_{burst} has little effect on the performance. Fig. 5(b) shows that there is an optimal point for which $T_{\max}$ can



(a)



(b)

Fig. 5. Effects of varying S_{burst} and T_{max} on the BiC-MAC protocol's: (a) throughput per node when the offered load per node $= 0.0056$, (b) end-to-end data packet delay when the offered load per node $= 0.0056$.

minimize the packet delay. When T_{max} is too small, a node triggers its RTS attempt too often, and this intense contention will cause more CTS failures; thus, more retransmissions are required and larger delay is expected. Moreover, a node does not have enough opportunity to accumulate more data packets before its RTS triggering, and this causes a large communication overhead for each successful handshake. In contrast, when T_{max} is too large, the inter-RTS triggering time becomes larger than necessary, and thus the delay becomes larger as well.

IV. Conclusion

In this paper, we have presented a novel approach of using concurrent, bidirectional packet bursts exchange to boost the underwater acoustic channel utilization. We then propose the BiC-MAC protocol, which is designed with a versatile MAC framework to support all three possible modes of bidirectional transmissions. Although the handshake is essentially sender-triggered, an intended receiver can exploit that opportunity to initiate bidirectional data exchange if it has packets in return. Via simulation results, we show that BiC-MAC greatly outperforms the normal unidirectional handshaking based protocols in terms of both throughput and delay. Our future work includes the study of an adaptive packet bursting mechanism, as well as a theoretical throughput analysis.

References

[1] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater Acoustic Sensor Networks: Research Challenges," *Ad Hoc Networks (Elsevier)*, 2005.

[2] E. Sozer, M. Stojanovic, and J. Proakis, "Underwater Acoustic Networks," *IEEE Journal of Oceanic Eng.*, Jan. 2000.

[3] M. Molins and M. Stojanovic, "Slotted FAMA – A MAC Protocol for Underwater Acoustic Networks," in *Proc. MTS/IEEE OCEANS'06*, 2006.

[4] B. Peleato and M. Stojanovic, "Distance Aware Collision Avoidance Protocol for Ad-hoc Underwater Acoustic Sensor Networks," *IEEE Communications Letters*, vol. 11, no. 12, pp. 1025-1027, Dec. 2007.

[5] X. Guo, M. R. Frater, and M. J. Ryan, "Design of a Propagation-Delay-Tolerant MAC Protocol for Underwater Acoustic Sensor Networks," *IEEE Journal of Oceanic Eng*, Apr. 2009.

[6] S. Shahabudeen, M. Chitre, and M. Motani, "A Multi-Channel MAC Protocol for AUV Networks," in *Proc. MTS/IEEE OCEANS'07*, June 2007.

[7] N. Chirdchoo, W. S. Soh, and K. C. Chua, "MACA-MN: A MACA-based MAC Protocol for Underwater Acoustic Networks with Packet Train for Multiple Neighbors," in *Proc. IEEE VTC'08*, 2008.

[8] H. H. Ng, W. S. Soh, and M. Motani, "MACA-U: A Media Access Protocol for Underwater Acoustic Networks," in *Proc. IEEE GLOBECOM'08*, Dec. 2008.

[9] ——, "ROPA: A MAC Protocol for Underwater Acoustic Networks with Reverse Opportunistic Packet Appending," in *Proc. IEEE WCNC'10*, Apr. 2010.