ROPA: A MAC Protocol for Underwater Acoustic Networks with Reverse Opportunistic Packet Appending

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*—In most existing sender-initiated handshaking based underwater Media Access Control (MAC) protocols, only the initiating sender is allowed to transmit data packets to its intended receiver after the channel has been reserved; none of the potentially backlogged neighbors of the sender can transmit in the duration after the current handshake. Therefore, each of those neighbors must initiate their own handshakes, which incur additional overheads and potentially result in poor channel utilization. In this paper, we present a novel approach to increase the channel utilization by allowing a sender to invite its one-hop neighbors (appenders) to opportunistically transmit (append) their data packets. After the sender finishes transmitting its packets to its own receiver, it can immediately switch its role to receive the incoming appended data packets, which arrive in a packet train manner. This greatly reduces the relative proportion of time spent on control signaling. We refer to this MAC protocol as ROPA – Reverse Opportunistic Packet Appending. From our extensive simulations and comparisons with existing protocols, we show that ROPA significantly increases the channel utilization and offers performance gains in terms of throughput and delay.**

I. Introduction

Underwater acoustic networking is deemed as the enabling technology for many underwater sensing applications [1]. However, unlike terrestrial wireless communications, underwater Media Access Control (MAC) protocol design must address challenges posed by underwater acoustic communications, such as slow propagation speed and low bit rate-distance product.

Among the existing underwater MAC protocols, there is a strong focus on handshaking based protocols as they provide multi-fold benefits such as the carrying of useful information in the control packets, and the ability to alleviate hidden node problem. In the following, we briefly describe some of these protocols. In [2], Sozer *et al.* propose a handshaking based MAC protocol with Stop-and-Wait Automatic Repeat Request (ARQ) mechanism. In [3], Molins and Stojanovic propose Slotted-FAMA, which uses a 4-way (Request-to-Send (RTS)/Clear-to-Send (CTS)/DATA/ACK) handshake with a time-slotting mechanism to avoid any data collision. However, it requires clock synchronization, and long slot length. In [4], Guo *et al.* introduce a MAC protocol called APCAP, which improves channel utilization by allowing a sender to take actions for other packets (e.g., starting another handshake, transmitting data packet) while waiting for a CTS frame to return. In another attempt to improve channel utilization, Chirdchoo *et al.* [5] propose the MACA-MN, which sends multiple packets to multiple neighbors in a single 3-way handshake. This reduces the relative proportion of time spent on RTS/CTS signaling. However, due to the long duration of each handshake, the average waiting time can be very long before a node gains control of the channel to transmit. In [6], Chirdchoo *et al.* propose the RIPT protocol, which adopts a receiver-initiated approach to coordinate the packets from multiple neighbors to arrive in a packet train manner at the receiver. Although this approach also reduces the relative proportion of time spent on control signaling, the adoption of a receiver-initiated approach often demands a complex traffic prediction algorithm. In [7], we propose the MACA-U protocol; its key feature is the incorporation of several state transition rules that account for the long propagation delay in underwater. It aims to serve as a more appropriate benchmarking protocol, as previous works in underwater MAC protocols often rely on applying terrestrial MAC protocols in underwater for benchmarking.

In existing sender-initiated handshaking based MAC protocols, only the sender is allowed to transmit single or multiple packets during a successful handshake; none of the other backlogged neighbors can transmit in this handshake, even though the channel has been reserved. In this paper, we introduce the novel concept of "reverse opportunistic packet appending", for which we call the resulting MAC protocol "ROPA". Similar to some of the aforementioned protocols, it seeks to improve channel utilization by reducing the proportion of time spent on control signaling. However, it achieves this differently by allowing the initiating sender to invite its one-hop neighbors (appenders) to opportunistically transmit (append) their packets. After the initiating sender finishes transmitting its data packets to its receiver (primary data transmissions in the forward path), it can immediately switch its role to receive the incoming appended data packets from multiple neighbors (secondary data transmissions in the reverse path), which arrive in a packet train manner. This is in contrast to the conventional approach, which requires each of those backlogged neighbors to initiate a separate handshake that incurs its own overheads.

Similar to other handshaking based protocols, the ROPA utilizes information extracted from the control packets to alleviate the hidden node problem. The ROPA's framework is also very versatile; when none of the sender's neighbors has any packet to append, it can still perform its forward path's transmissions, and this reduces to MACA with packet train. On the other hand, if the sender only receives its neighbors' appending requests, but does not hear from its own intended receiver, it can still

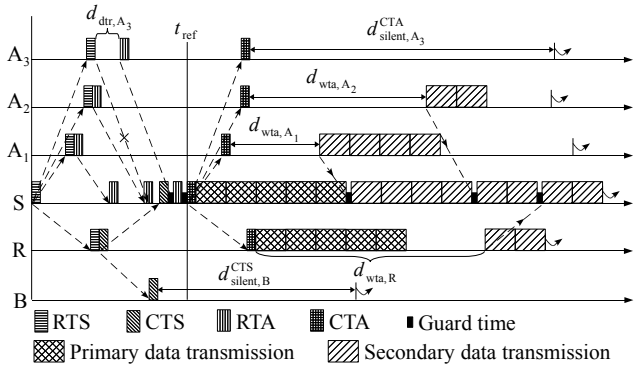


Fig. 1. Timing diagram of the ROPA protocol.

receive the reverse path's packets, and this reduces to RIPT for the current handshake. Therefore, the ROPA can be viewed as a superset of both MACA with packet train, and RIPT.

The remainder of this paper is organized as follows. In Section II, we describe the ROPA in detail. Next, we describe our simulations and results in Section III. Finally, we present our conclusion and future work in Section IV.

II. Protocol Design

A. General Assumptions

We consider static multi-hop acoustic networks where each node is equipped with a half-duplex underwater acoustic modem. Note that these "static" nodes may still have some limited movement as they are typically anchored to the seabed, and subjected to sway movements caused by underwater currents. We also assume that every node knows its estimated propagation delay from each of its first-hop neighbors, and the estimated inter-nodal propagation delays between its first-hop and second-hop neighbors. During network deployment and initialization phase, the inter-nodal propagation delays could be estimated via round-trip time measurements of control packets, and disseminated to the neighbors. Note that any estimation errors and the fluctuations caused by sway movements are typically very small compared to the propagation delays. In addition, our protocol can accommodate such errors or fluctuations via the use of guard times, which will be explained later.

B. How the ROPA Protocol Works

Fig. 1 shows ROPA's timing diagram. Here, nodes "S" and "R" refer to the initiating sender and its intended receiver, respectively. Nodes "A_1", "A_2", and "A_3" are one-hop neighbors of node "S", and can potentially become the appenders for this handshake. Node "B" is a one-hop neighbor of node "R" only, and is hidden from node "S". The notations used to explain the protocol are given in Table I.

1) Channel Reservation Phase: The broadcasted RTS packet serves two purposes. First, it informs the receiver about the request to perform primary data transmissions, so that the receiver will try to reserve the floor around its neighborhood if it can indeed accept the data packets. Second, it is used for polling the potential appenders whether they have any data packets to append. When a neighboring node has packets to append, it shall request for permission by using a Request-to-Append (RTA) packet. As can be seen in Fig. 1, the RTA packets might result in RTA-RTA or RTA-CTS collisions at the sender if each potential appender were to simply respond with its RTA packet immediately. For instance, this can happen when two or more potential appenders have similar distances from the sender (e.g., nodes "A_2" and "A_3"). Fortunately, this can easily be avoided as the sender can compute a collision-free RTA requests' schedule, so that the appenders know when to transmit their RTA packets (to be described in Section II-C1). In the RTS packet, the sender specifies four types of information, namely, the total number of primary data packets that it wishes to transmit ($n_{\text{S}}^{\text{pri}}$), the maximum allowable number of secondary data packets that can be appended for the current handshake ($S_{\text{max}}^{\text{sec}}$), its expected busy duration ($d_{\text{busy,RTS}}$), and the collision-free RTA requests' schedule it has computed. The busy duration will be used by all overhearing nodes to compute their silent durations. For the collision-free schedule, the RTS packet only needs to explicitly specify the defer-to-request duration ($d_{\text{dtr},i}$) of an appender node i if it is not allowed to transmit its RTA packet immediately after receiving the RTS packet.

TABLE I
Notations Used for Explaining the ROPA 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 $x \in \{\text{RTS}, \text{CTS}, \text{RTA}, \text{CTA}\}$
$\tau_{i,j}$	Propagation delay between node i and node j
$\tau_{\text{S}}(k)$	Propagation delay between a sender and its appender that has been assigned the order k
T_{max}	Time threshold for triggering an RTS attempt
S_{burst}	Threshold number of accumulated data packets for triggering an RTS attempt
$S_{\text{max}}^{\text{pri}}$	Maximum allowable number of primary data packets that can be sent in a single handshake
$S_{\text{max}}^{\text{sec}}$	Maximum allowable number of secondary data packets that can be appended in a single handshake
T_{guard}	Guard time to accommodate any error in the inter-nodal propagation delay's estimation
$d_{\text{dtr},i}$	Defer-to-Request; duration that node i must wait before starting to transmit its RTA packet to avoid collision
$d_{\text{wta},i}$	Wait-to-Append; duration that node i must wait before starting to transmit its secondary data packets
$d_{\text{silent},i}^{x}$	Duration that node i must remain silent after overhearing a type x control packet, where $x \in \{\text{RTS}, \text{CTS}, \text{RTA}, \text{CTA}\}$
$d_{\text{busy},i}$	Busy duration information that is included in a type i control packet, where $i \in \{\text{RTS}, \text{CTS}, \text{RTA}, \text{CTA}\}$
$d_{\text{rel,S}}$	Duration that a sender spends from t_{ref} till it releases from the current handshake
$t_{\text{RTS}}^{\text{RX}}(k)$	Time instant at which an appender of order k finishes overhearing the RTS packet
$t_{\text{RTA}}^{\text{TX}}(k)$	Time instant at which an appender of order k starts transmitting its RTA packet
t_{ref}	Time instant at which sender starts sending CTA packet
$n_{\text{S}}^{\text{pri}}$	Number of primary data packets (both relayed and new) that sender S intends to transmit in current handshake
$n_{\text{relay},i}^{\text{sec}}$	Number of relayed packets that node i intends to append in current handshake
$n_{\text{new},i}^{\text{sec}}$	Number of new packets that node i intends to append in current handshake
$\mathcal{N}_i$	Set of node i's one-hop neighboring nodes

Upon overhearing an RTS packet, each potential appender i checks whether it has any data packets in return. If it does not, it simply extracts the RTS's busy duration to locally compute its silent duration, $d^{\text{RTS}}_{\text{silent},i}$; if it does, it prepares its RTA packet with the 3-tuple, $[n^{\text{sec}}_{\text{relay},i}, n^{\text{sec}}_{\text{new},i}, d_{\text{busy,RTA}}]$, which specifies the number of relayed and new packets that it wishes to append, as well as its expected busy duration. Note that $n^{\text{sec}}_{\text{relay},i} + n^{\text{sec}}_{\text{new},i} \leq S^{\text{sec}}_{\text{max}}$. The duration $d_{\text{busy,RTA}}$ is the expected time taken before it could receive the sender's Clear-to-Append (CTA) packet, which carries the decision on whether its packet appending request has been granted. The purpose of $d_{\text{busy,RTA}}$ is to allow other neighboring nodes that overhear the RTA packet to compute their respective silent durations, so that they could avoid interfering with the CTA reception. Before responding with the RTA, an appender needs to look up the collision-free schedule given by the RTS packet. In the case where the schedule does not require it to defer its RTA transmission, it can do so immediately. Otherwise, it defers its RTA transmission by the amount of time indicated within the schedule. Note, however, that the appender could only transmit an RTA packet provided it is currently not involved in any other handshake, and is also not required by any other node to remain silent.

As for the receiver, upon receiving the RTS packet, it checks whether it has any data packets in return, before responding with a CTS packet. Unlike the RTA response, the receiver always transmits its CTS packet without any deferment. Note, however, that the receiver can only transmit a CTS response provided it is currently not involved in any other handshake, and is also not required to remain silent. When the receiver is interested in appending packets, it specifies in the CTS packet the number of relayed and new packets that it wishes to append, $n^{\text{sec}}_{\text{relay,R}}$ and $n^{\text{sec}}_{\text{new,R}}$, respectively. Note that $n^{\text{sec}}_{\text{relay,R}} + n^{\text{sec}}_{\text{new,R}} \leq S^{\text{sec}}_{\text{max}}$ as well. For the case where the receiver does not have any packet to append, the $n^{\text{sec}}_{\text{relay,R}}$ and $n^{\text{sec}}_{\text{new,R}}$ fields are set to zero. Regardless of which is the case, the CTS always carries a busy duration field, $d_{\text{busy,CTS}}$, which allows all overhearing nodes to compute their respective silent durations.

Unlike the original MACA protocol where a sender waits unproductively for the CTS, ROPA's sender can utilize this time window to collect some of the RTA requests, and collect any remaining RTA requests after receiving the CTS. After broadcasting an RTS packet, a sender waits until a reference time (t_{ref}) that is sufficiently large to accommodate even its most distant one-hop neighbor's incoming RTA packet. Having acquired all incoming requests, the sender allocates its available secondary data slots, $S^{\text{sec}}_{\text{max}}$, using a simple strategy that prioritizes all relay packets over new packets (to be detailed in Section II-C2). Then, a CTA packet is broadcasted to inform those granted appenders when and how many packets they should append. For each of the granted appenders, the CTA packet carries the following 4-tuple, [node ID, $n_{\text{relay},i}$, $n_{\text{new},i}$, $d_{\text{wta},i}$], where $n_{\text{relay},i}$ and $n_{\text{new},i}$ are the number of relay and new packets that appender node i can transmit, respectively, and $d_{\text{wta},i}$ is the duration that node i must wait before it can start to append its packets. Similar to the other control packets, the CTA also contains a busy duration field, $d_{\text{busy,CTA}}$. Immediately after transmitting the CTA packet, the sender starts transmitting its data packets to the receiver.

As mentioned, every control packet carries a busy duration field, which will be used by all overhearing nodes to calculate their respective silent durations. They are computed as follows:

$$\begin{cases} d_{\text{busy,RTS}} = t_{\text{ref}} - T_{\text{RTS}}, \\ d_{\text{busy,CTS}} = 2\tau_{\text{S,R}} + T_{\text{CTA}} + n^{\text{pri}}_{\text{S}} T_{\text{DATA}} \\ \qquad\qquad + [t_{\text{ref}} - (2\tau_{\text{S,R}} + T_{\text{RTS}} + T_{\text{CTS}})], \\ d_{\text{busy,RTA}} = 2\tau_{\text{S},i} + T_{\text{CTA}} \\ \qquad\qquad + [t_{\text{ref}} - (2\tau_{\text{S},i} + T_{\text{RTS}} + d_{\text{dtr},i} + T_{\text{RTA}})], \\ d_{\text{busy,CTA}} = d_{\text{rel,S}} - T_{\text{CTA}}, \end{cases} \tag{1}$$

where $i \in \mathcal{N}_{\text{S}} \setminus \{\text{R}\}$, and $d_{\text{rel,S}}$ is the duration that a sender spends from t_{ref} till it releases from the current handshake ($d_{\text{rel,S}}$ can be computed using (6)). The respective silent durations are:

$$\begin{cases} d^{\text{RTS}}_{\text{silent},i} = d_{\text{busy,RTS}} + T_{\text{CTA}}, & i \in \mathcal{N}_{\text{S}}, \\ d^{\text{CTS}}_{\text{silent},j} = d_{\text{busy,CTS}} - 2\tau_{\text{R},j}, & j \in \mathcal{N}_{\text{R}}, \\ d^{\text{RTA}}_{\text{silent},k} = d_{\text{busy,RTA}} - 2\tau_{i,k}, & i \in \mathcal{N}_{\text{S}} \setminus \{\text{R}\}, k \in \mathcal{N}_i, \\ d^{\text{CTA}}_{\text{silent},i} = d_{\text{busy,CTA}} - 2\tau_{\text{S},i}, & i \in \mathcal{N}_{\text{S}}. \end{cases} \tag{2}$$

Notice that the silent duration (except $d^{\text{RTS}}_{\text{silent},i}$) computed by an overhearing neighbor can be shorter than the busy duration contained within the control packet; this is because of inter-nodal propagation delays. As for $d^{\text{RTS}}_{\text{silent},i}$, it should be sufficiently large so that all potential appenders can fully receive the CTA packet.

2) Data Packet Transmission Phase: After a sender finishes transmitting $n^{\text{pri}}_{\text{S}}$ primary data packets to its intended receiver, it immediately switches its role to anticipate the incoming appended packets. As for the potential appenders, each of them checks whether its previous request has been approved by looking up the grant decision in the overheard CTA packet. A "non-granted" appender i (e.g., node "A_3" in Fig. 1) shall remain silent for a duration of $d^{\text{CTA}}_{\text{silent},i}$. In contrast, each granted appender i can start to transmit its granted number of secondary data packets after the specified wait-to-append duration, $d_{\text{wta},i}$, has passed. After transmitting its share of appended packets, it shall remain silent if necessary, so that it does not disrupt the sender's reception of packets from other appenders. For this purpose, each granted appender will locally compute its handshake release duration (relative to the time at which it finishes receiving the CTA) as $d^{\text{CTA}}_{\text{silent},i}$.

A point to note is that, the secondary transmissions from an appender might potentially interfere with the packet receptions of its neighbors. To reduce the likelihood of excessive data collisions, each appender node pays close attention to all overheard control packets before it begins its packet appending, and will abort the appending if necessary. For instance, a granted appender shall remain silent if it has previously overheard any xRTS, xCTS, or xCTA packets, where "x" implies that the control packet is destined to others.

The ROPA protocol has three operation modes. In mode 1, a sender receives the CTS reply, and at least one RTA response; it thus contains both primary and secondary transmissions. In mode 2, a sender only receives the CTS reply, and thus it

1 Sort $\tau_{\mathrm{S},i}$ in ascending order for all $i \in \mathcal{N}_{\mathrm{S}} \setminus \{\mathrm{R}\}$.
2 $t^{\mathrm{TX}}_{\mathrm{RTA}}(0) \leftarrow 0; t^{\mathrm{RX}}_{\mathrm{RTS}}(0) \leftarrow 0; \tau_{\mathrm{S}}(0) \leftarrow 0;$
3 **for** $k \leftarrow 1$ **to** $|\mathcal{N}_{\mathrm{S}} \setminus \{\mathrm{R}\}|$
4 $\quad t^{\mathrm{TX}}_{\mathrm{RTA}}(k) \leftarrow \max(t^{\mathrm{RX}}_{\mathrm{RTS}}(k), (t^{\mathrm{TX}}_{\mathrm{RTA}}(k-1) + \tau_{\mathrm{s}}(k-1) - \tau_{\mathrm{s}}(k) + T_{\mathrm{RTA}} + T_{\mathrm{guard}}));$
5 $\quad$ **if** $[t^{\mathrm{TX}}_{\mathrm{RTA}}(k) + \tau_{\mathrm{s}}(k),\ t^{\mathrm{TX}}_{\mathrm{RTA}}(k) + \tau_{\mathrm{s}}(k) + T_{\mathrm{RTA}}] \bigcap [2\tau_{\mathrm{S,R}} + T_{\mathrm{RTS}} - T_{\mathrm{guard}},\ 2\tau_{\mathrm{S,R}} + T_{\mathrm{RTS}} + T_{\mathrm{CTS}} + T_{\mathrm{guard}}] \neq \emptyset$
6 $\quad\quad t^{\mathrm{TX}}_{\mathrm{RTA}}(k) \leftarrow 2\tau_{\mathrm{S,R}} + T_{\mathrm{RTS}} + T_{\mathrm{CTS}} - \tau_{\mathrm{s}}(k) + T_{\mathrm{guard}};$
7 $\quad$ **end if**
8 $\quad d_{\mathrm{dtr}}(k) \leftarrow t^{\mathrm{TX}}_{\mathrm{RTA}}(k) - (\tau_{\mathrm{s}}(k) + T_{\mathrm{RTS}});$
9 **end for**

Fig. 2. Algorithm for scheduling collision-free RTA requests.

performs the primary transmissions alone, without having to broadcast a CTA packet; this reduces to MACA with packet train. In mode 3, a sender does not receive the CTS, but receives at least one RTA response; the transmission pattern becomes purely receiver-initiated, and thus reduces to the RIPT protocol.

A key mechanism in ROPA for improving channel utilization in underwater acoustic network is that, multiple appenders can be scheduled to transmit with partial overlap in time, such that the appended packets arrive at sender S in a packet train manner without overlapping. This is not possible in terrestrial wireless network where the propagation delay is too short to allow any concurrent transmission. Since ROPA relies on the internodal propagation delay estimates to work, it is essential to protect against any estimation errors; this is achieved via the insertion of a small guard time, T_{guard}, between packet bursts arriving at sender S from different appenders (see Fig. 1). A reasonable value for T_{guard} would be in the range of tens of milliseconds.

C. Algorithms in the ROPA Protocol

1) Algorithm for Scheduling Collision-free RTA Requests: Fig. 2 shows how a sender can construct its collision-free RTA requests' schedule. In line 1, the sender sorts the internodal propagation delays between itself and all its one-hop neighbors (excluding the receiver) in ascending order. From lines 3 to 9, it calculates the transmit time and the defer-to-request duration for each potential appender. Specifically, in line 4, the transmit time of the k^{th}-order RTA is tentatively set to either immediately after receiving the RTS if it will not collide with a prior potential appender's RTA, or deferred to the earliest time instant when it can do so. Since the RTA must also avoid collision with the receiver's CTS, line 5 checks this condition, and defers the transmit time further, if necessary, in line 6. Finally, in line 8, we calculate the defer-to-request duration for the k^{th}-order potential appender, $d_{\mathrm{dtr}}(k)$.

Recall that t_{ref} is the reference time instant at which a sender starts sending the CTA packet. It can be calculated once we have scheduled all the transmit times of the potential appenders:

$$t_{\mathrm{ref}} = \begin{cases} t^{\mathrm{TX}}_{\mathrm{RTA}}(n) + \tau_{\mathrm{S}}(n) + T_{\mathrm{RTA}} + T_{\mathrm{guard}}, & \text{if RTA is last,} \\ 2\tau_{\mathrm{S,R}} + T_{\mathrm{RTS}} + T_{\mathrm{CTS}} + T_{\mathrm{guard}}, & \text{if CTS is last,} \end{cases} \tag{3}$$

where $n = |\mathcal{N}_{\mathrm{S}} \setminus \{\mathrm{R}\}|$.

2) Algorithm for Assigning Secondary Data Slots: Before we explain the algorithm, let us define two regions within a sender's first-hop neighborhood, which we call regions R_1 and R_2. Here, R_1 is defined as the region in which a node can only communicate directly with the sender, but not the receiver. In contrast, R_2 is defined as the region in which a node can communicate directly with both the sender and the receiver.

We shall now discuss how a sender allocates its secondary data slots to the potential appenders. Consider a scenario where the sender receives N appending requests, and each request is organized in the form [node ID, $n^{\mathrm{sec}}_{\mathrm{relay},i}$, $n^{\mathrm{sec}}_{\mathrm{new},i}$, region]. The sender first assigns a random priority to these N entries so that an ordered list is created. It then cycles through each entry to compute the corresponding node's wait-to-append duration. To avoid any transmit-receive (TX-RX) collision at the sender, the appended packets must only arrive after it finishes transmitting its primary packets. Furthermore, the appended packets from different appenders must not result in any RX-RX collision. These are achieved by ensuring that

$$\begin{aligned} &2\tau_{\mathrm{S}}(k-1) + d_{\mathrm{wta}}(k-1) + n^{\mathrm{sec}}(k-1)T_{\mathrm{DATA}} + T_{\mathrm{guard}} \\ &\leq 2\tau_{\mathrm{S}}(k) + d_{\mathrm{wta}}(k), \end{aligned} \tag{4}$$

where $k = \{1, 2, \ldots, N\}$, $d_{\mathrm{wta}}(k)$ is the wait-to-append duration for the k^{th}-order node, and $n^{\mathrm{sec}}(k-1)$ is the number of secondary slots granted to the $(k-1)^{\mathrm{th}}$-order node. Note that $\tau_{\mathrm{S}}(0) = d_{\mathrm{wta}}(0) = 0$ and $n^{\mathrm{sec}}(0) = n^{\mathrm{pri}}_{\mathrm{S}}$. More importantly, when the propagation delay between the k^{th}-order node and the sender is sufficiently large, such that $2\tau_{\mathrm{S}}(k) > 2\tau_{\mathrm{S}}(k-1) + d_{\mathrm{wta}}(k-1) + n^{\mathrm{sec}}(k-1)T_{\mathrm{DATA}} + T_{\mathrm{guard}}$, there will be a gap in the appended packet train received by the sender when the k^{th}-order node's packets arrive (even if $d_{\mathrm{wta}}(k) = 0$), which reduces channel utilization and throughput. For this reason, our algorithm always seeks to avoid this inefficiency by de-prioritizing the current entry to the last, and proceeds to evaluate the next entry. For an entry where $2\tau_{\mathrm{S}}(k) \leq 2\tau_{\mathrm{S}}(k-1) + d_{\mathrm{wta}}(k-1) + n^{\mathrm{sec}}(k-1)T_{\mathrm{DATA}} + T_{\mathrm{guard}}$, there is no gap between appended packet bursts from different appenders (apart from a small guard time). In this case, $d_{\mathrm{wta}}(k)$ is computed from (4) by making its LHS equal to its RHS.

If a k^{th}-order potential appender (say, node j) is located in R_2, it must be further ensured that the secondary transmissions will not interfere with the receiver's reception of the primary packets. Hence, the sender should evaluate whether the previously computed $d_{\mathrm{wta}}(k)$ satisfies the following inequality

$$\tau_{\mathrm{S,R}} + n^{\mathrm{pri}}_{\mathrm{S}} T_{\mathrm{DATA}} + T_{\mathrm{guard}} \leq \tau_{\mathrm{S}}(k) + d_{\mathrm{wta}}(k) + \tau_{\mathrm{R},j}. \tag{5}$$

If it does not, the sender will also de-prioritize the current entry, and consider the next entry. As for the case where the receiver also intends to append packets, its $d_{\mathrm{wta}}(k)$ must be sufficiently large to accommodate its primary packet's receptions, i.e., $d_{\mathrm{wta}}(k) > n^{\mathrm{pri}}_{\mathrm{S}} T_{\mathrm{DATA}} + T_{\mathrm{guard}}$.

After obtaining a valid $d_{\mathrm{wta}}(k)$, the sender proceeds to allocate data slots from its quota of $S^{\mathrm{sec}}_{\mathrm{max}}$ to this k^{th}-order node using a strategy that prioritizes all relay packets over new packets. This is because the relay packets have already consumed valuable channel resources, and it would be wasteful if they were to be discarded due to potential buffer overflows. After the slot allocations, the sender repeats the aforementioned

procedures for the $(k+1)^{\text{th}}$-order node. The stopping condition of the algorithm is either: (i) when all $S_{\max}^{\text{sec}}$ slots are exhausted, or (ii) when slot allocations have been completed for all N entries. Finally, let $\mathcal{A}_{\text{G}}$ denote the set of granted appenders; the sender's handshake release duration, $d_{\text{rel,S}}$, is

$$d_{\text{rel,S}} = T_{\text{CTA}} + 2\tau_{\text{S}}(i^*) + d_{\text{wta}}(i^*) + n^{\text{sec}}(i^*)T_{\text{DATA}}, \qquad (6)$$

where $i^* = \arg\max_{i\in\mathcal{A}_{\text{G}}}[2\tau_{\text{S}}(i) + d_{\text{wta}}(i) + n^{\text{sec}}(i)T_{\text{DATA}}]$.

D. RTS Attempt Triggering and Backoff Algorithms

A hybrid of "batch-by-size" and "batch-by-time" strategies is used in the ROPA protocol to determine when to trigger an RTS attempt. Specifically, an RTS attempt can be triggered either when a node has not triggered its RTS attempt for a duration of $T_{\max}$ from the time it last releases from handshake, or when it has accumulated at least S_{burst} data packets that are destined for a particular neighbor. Note that even when the above is satisfied, an RTS attempt will only occur if the node is not currently residing in a silent state, or engaging in any other handshake; otherwise, it must defer its RTS attempt until these constraints no longer hold. Also, note that we have assumed $S_{\max}^{\text{pri}} = S_{\text{burst}}$; therefore, a node can only transmit at most S_{burst} packets for each handshake.

The ROPA protocol does not adopt the Binary Exponential Backoff (BEB) algorithm, where a sender doubles its backoff counter in the event of an RTS failure. When an RTS fails, the ROPA's sender may still be able to receive appended packets. Note that the probability of having no appender is very low, especially when operating at high load. Hence, by the time the sender finishes receiving all appended packets, the receiver may already be free; if the BEB were used, the sender's backoff window becomes unnecessarily large, and leads to lower throughput. Therefore, the ROPA's backoff interval, T_{bk}, is taken from

$$T_{\text{bk}} = \textit{uniform}(0, B_{\text{win}}) \times \tau_{\max}, \qquad (7)$$

where B_{win} is a constant backoff window, and $\tau_{\max}$ is the maximum propagation delay.

III. Simulations and Results

A. Simulation Model

We have developed our own C++ discrete event-driven network simulator. Our multi-hop topology, shown in Fig. 3, has 36 static nodes with a grid spacing of 2000 m. Each node deviates from the grid intersection point following a uniform distribution by a maximum of 10% of the grid spacing. The maximum transmission range is 1.75 times the grid spacing, or 3500 m. Each node has 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 assumed to be fixed at 1500 m/s. An error-free channel is also assumed,

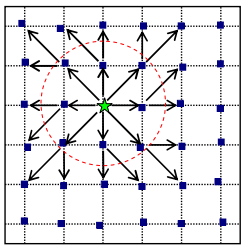


Fig. 3. The multi-hop network topology used. Note that every other node assumes the same static routing pattern as the node marked as a star.

thus all packet losses are caused by packet collisions. The transmission rate is 4800 bps and the data packet length is 1200 bits, unless stated otherwise. For ROPA's parameters, we set $T_{\text{guard}} = 10$ ms, $B_{\text{cnt}} = 30$, and $T_{\max} = 100$ s. Also, the lengths of the RTS, CTS, RTA, and CTA packets are 336, 104, 104, and 456 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 $S_{\max}^{\text{pri}}$ and $S_{\max}^{\text{sec}}$ are both 110 unless otherwise stated. To avoid transient effect, the results are collected from 2×10^4 s to 1×10^5 s.

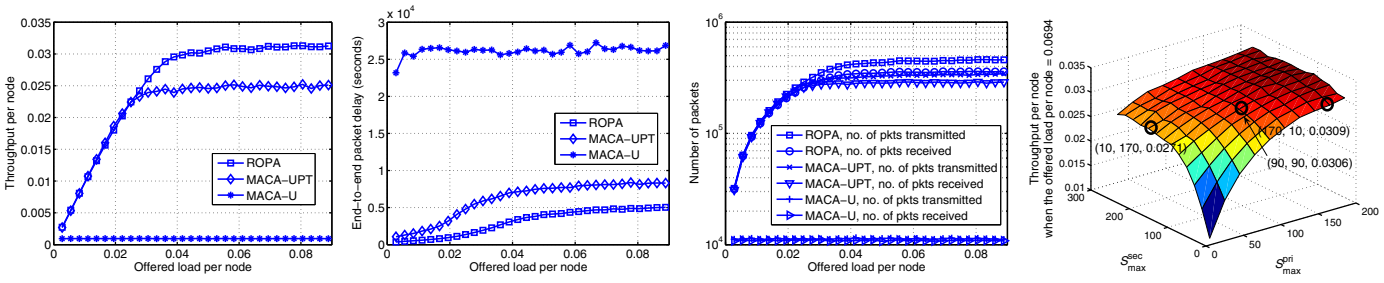
B. Simulation Results

1) Performance of ROPA in Multi-hop Networks: We compare ROPA's performance against that of two handshaking based protocols, namely, MACA-U [7], and MACA-U with packet train (MACA-UPT). These protocols' RTS and CTS control packets are assumed to be 88-bit long. We also define a performance metric, namely, *throughput per node*, as follows:

$$\gamma = \frac{1}{36}\left[\frac{\text{No. of packets received} \times \text{Packet length}}{\text{Transmission rate} \times \text{Simulation duration}}\right]. \qquad (8)$$

The above is a more meaningful metric because, in a multi-hop setting, the system's aggregate throughput could grow with the network size due to concurrent transmissions in multiple regions that are sufficiently far apart from each other.

Fig. 4(a) and Fig. 4(b) show that ROPA outperforms MACA-UPT and MACA-U in terms of both throughput per node, and end-to-end packet delay. As expected, MACA-U is very inefficient because the sender only transmits a single packet for every successful handshake, and thus a large proportion of time is spent on exchanging control packets. In contrast, the MACA-UPT offers great improvement by transmitting multiple packets for each successful handshake. Note that in MACA-UPT, since there is no ACK involved, the first-hop neighbors of the sender only need to remain silent for a short duration after overhearing an xRTS to accommodate the subsequent CTS. Thus, they can participate in a new handshake sooner, and benefit from more concurrent transmissions in the neighborhood. The ROPA, on the other hand, requires them to remain silent till the end of secondary transmissions; nevertheless, it still outperforms MACA-UPT because multiple appenders can opportunistically append their packets at the end of the sender's primary transmissions, which greatly reduces the proportion of time spent on the control signaling overheads. From Fig. 4(c), we see that even though ROPA transmits more packets for each successful handshake, it still maintains a high success ratio. This shows



(a) (b) (c) (d)

Fig. 4. (a) Throughput per node comparisons for various schemes, (b) End-to-end packet delay comparisons for various schemes, (c) The comparisons of the number of packets transmitted, and received for various schemes, (d) Effects of varying $S_{\max}^{\text{pri}}$ and $S_{\max}^{\text{sec}}$ on ROPA's throughput per node.

that ROPA is efficient at alleviating the packet collisions caused by the hidden node problem; specifically, an appender will abort its packet appending when it overhears certain control packets that indicate its transmissions are likely to interfere with other ongoing transmissions in the neighborhood.

2) Effects of Varying ROPA's parameters: Fig. 4(d) shows ROPA's throughput for different $S_{\max}^{\text{pri}}$ and $S_{\max}^{\text{sec}}$ when the offered load per node is 0.0694 (large enough to saturate throughput). The throughput increases rapidly, and stabilizes as $S_{\max}^{\text{pri}}$ and $S_{\max}^{\text{sec}}$ are increased from 10 to 190 and from 10 to 250, respectively. Note that when the total train size is constant (e.g., when $(S_{\max}^{\text{pri}}, S_{\max}^{\text{sec}})$ is (170,10), (90,90), or (10,170)), a larger $S_{\max}^{\text{pri}}$ offers better performance. This can be explained as follows. When a receiver's neighbors overhear its CTS, they have sufficiently long silent durations till the end of its primary packets' reception. In contrast, an appender's neighbors only remain silent for a short duration upon overhearing its RTA (up to the CTA's reception at the appender), and they may participate in other handshakes after that. Hence, the secondary transmissions from an appender might cause interference to its neighbors' reception of packets subsequently, which reduces throughput. Although ROPA has a mechanism to alleviate excessive packet collisions in which an appender may abort its packet appending, its previously assigned slots will be wasted. These explain why a larger $S_{\max}^{\text{pri}}$ offers better performance.

We have also studied the effects of varying $T_{\max}$, but we do not include the details here due to space constraint. It is found that varying $T_{\max}$ only affects ROPA's delay at low load; this is intuitive since a node now needs to wait longer to accumulate S_{burst} packets, thus making the "batch-by-time" mechanism mainly responsible for RTS triggering. In contrast, the "batch-by-size" mechanism is dominant at high load, making S_{burst} more important in determining ROPA's throughput and delay.

3) Performance of ROPA in Single-hop Networks: Here, we evaluate ROPA's performance in single-hop networks and compare it with several other protocols that Guo *et al.* [4] have compared their APCAP against. We follow the same setup where 20 nodes are randomly deployed in a 4500×4500 m^2 area. The data rate is 2400 bps, and the packet length is 8000 bits. A sender's data packet can be destined to any other node with equal probability. For ROPA, we set both $S_{\max}^{\text{pri}}$ and $S_{\max}^{\text{sec}}$ to 70. The results are averaged over five random topologies. From Fig. 5, we see that ROPA has the highest

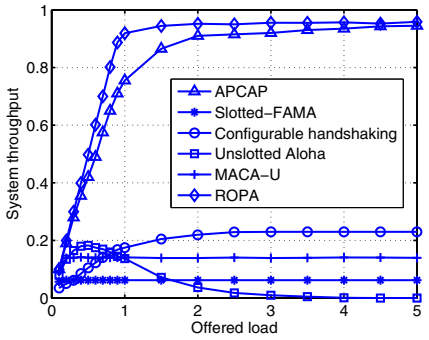


Fig. 5. Throughput comparisons against other selected MAC protocols in [4].

throughput, and outperforms APCAP across all offered loads. It is also interesting to note that MACA-U outperforms Slotted-FAMA across all loads, as well as the configurable handshaking protocol in the low load region.

IV. Conclusion

We have presented the ROPA protocol, which utilizes a reverse opportunistic packet appending approach to significantly increase channel utilization. Via simulation results, we show that ROPA offers performance gains in terms of both throughput and delay. Based on our findings, we contend that ROPA is a suitable candidate for a network with limited mobility, where each node has a large number of neighbors. For our future work, the following directions shall be investigated: (i) an adaptive train mechanism; (ii) protocol evaluation using a propagation loss model; and (iii) energy expenditure of the ROPA protocol.

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] 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.

[5] 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.

[6] ——, "RIPT: A Receiver-initiated Reservation-based Protocol for Underwater Acoustic Networks," *IEEE JSAC, Special Issue on Underwater Wireless Communications and Networks*, Dec. 2008.

[7] 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.