 TOKEN-BASED MEDIUM ACCESS CONTROL SOLUTION FOR UNDERWATER ACOUSTIC BROADCAST

Center for Intelligent Systems Research
University of Idaho
PO Box 441024, 607 Urquhart Ave., BEL W3-1
Moscow, ID 83844-1024
+1 (208) 885-6500  egbo2419@vandals.uidaho.edu

Abstract— Research into underwater acoustic communication has focused on time division multiple access (TDMA) communication protocols. Within a front-seat/backseat control architecture, effective communication is not guaranteed due to the strict clock synchronization requirement of the TDMA protocols. In the event of dropped messages during the cycle, the strict clock requirement can lead to message overlap and communication failure. In this paper, a novel, token-based medium access control (TMAC) solution for underwater acoustic broadcast is introduced. This solution is unique in the sense that it does not require that a ring be maintained for passing the token. TMAC provides a solution to the synchronization problem, ensuring effective communication within a front-seat/backseat control architecture. We expand on the structure and operation of the TMAC solution in the presence and absence of dropped messages. The solution is then applied to a mine countermeasure reference mission, and the results are compared to those generated by the TDMA protocol.

I. INTRODUCTION

In the past decade, significant advancements have been made in the development of the operational range and data throughput of acoustic underwater communication systems [1]. Underwater acoustic communication channels are characterized as multi-path, time-disperse, and rapidly fading [2]. These qualities render underwater acoustic communication error-prone, making it a challenge to exchange significant quantities of information rapidly through these channels. Efficient use of underwater acoustic channels is enabled by employment of structured communication protocols.

Acoustic underwater communication can be used to support the control of collaborative autonomous underwater vehicles (AUVs) [3]. Funded by the United States Navy, the AUV team at the University of Idaho (UI) has successfully tested several autonomous, collaborative behaviors involving a fleet of five vehicles that communicate acoustically underwater [4]. In particular, the team has combined a time division multiple access (TDMA) communication protocol with an agent communication language designed for AUV operations, AUVish-BBM, to make possible information exchange in the form of underwater acoustic broadcasts by the fleet in mine-counter measure (MCM) operations [5].

TDMA protocols divide a time interval known as a frame into time slots [6]. Each time slot is assigned to a communication source, e.g., an individual AUV in the fleet. TDMA protocols in underwater acoustic networks require strict synchronization. While this is possible in an error prone environment [7], it depends on the deployment of control architectures in fleet vehicles that support synchronization.

Recently, the UI modified its AUV control approach, adopting a front-seat/backseat paradigm [8]. Front-seat controllers interface directly with motor actuators, sensors, and navigation equipment, while the back-seat controller runs decision-making logics and manages intra- and inter-vehicle communications, delivering heading, speed, and depth commands to the front-seat controllers. This division of control can result in AUV desynchronization, such as when the front-seat and back-seat controllers employ independent processors linked via an Ethernet network. If control and communication become time independent, there is the possibility of overlap in communications, which results in signal loss. Considering the limitations of underwater acoustic communication—limited bandwidth, low propagation speed, refractive properties of the medium and wide time variation—the problem of excessive propagation delays resulting from a TDMA protocol would result in inefficient time slot allocation [6]. This inefficiency hampers communication among the AUVs and hence necessitated an alternative protocol solution.

In this paper, we discuss the new token-based medium access control (TMAC) solution that has been designed to replace the TDMA protocol in the UI AUV fleet. This TMAC solution avoids the need for synchronization and therefore underwrites successful communication among vehicles that implement the front-seat/back-seat architecture. After describing the new protocol in detail, we consider an application to simulated AUVs conducting a MCM mission.

II. LITERATURE REVIEW

Two aspects of the new TMAC communication protocol have been reported in the literature, viz., its asynchronicity and its dependence on tokens that track communication information. In [9] it was reported that asynchronous message systems are desirable over synchronous message systems due to a decrease in message interference and collision. The described asynchronous system was a novel transmission relay system for wireless networks. One consequence of experiments like these was documented in [10], where a
common clock was absent yet reliable communication was maintained asynchronously.

Token-based, wireless communication protocols have an established record of success. In [11], the token ring medium access control solution (TRMAC) is introduced. In TRMAC, by using a token as the transmission priority arbitrator, time synchronization necessary for traditional TDMA is not required. However, there is still the need to maintain a ring for passing the token. Reference [12] proposed and implemented a virtual token protocol in which a logical sequence was calculated with a clock for varying processes. The result was a token that was not passed, but maintained virtually within the memory of the individual machines.

III. IDAHO APPROACH

Researchers at the UI have developed a number of autonomous, collaborative AUV behaviors in the context of MCM missions [4]. This development has taken place on a variety of platforms, including the Autonomous Littoral Warfare Systems Evaluator-Monte Carlo (ALWSE-MC) simulation platform and a fleet of five miniature submarines modeled on those designed by Stilwell et al. [13]. The developmental strategy has been to use the simulation platform for proof of concept before the designed behaviors are ported over to the vehicles for field testing. To support this work, a MATLAB behavior module has been coded for ALWSE-MC that models vehicle and fleet dynamics, as well as the systematic interactions between vehicles required for autonomous collaboration.

Systematic interaction of the kind crucial to robust collaborative behavior requires communication. For the vehicles to act jointly in the conduct of mutual behavior they must exchange information about their current activity and the goals they are pursuing; otherwise, there is no joint behavior but merely behavior in parallel. The UI team developed an agent communication language, AUVish-BBM, and a TDMA protocol to support message exchange across the fleet [5]. AUVish-BBM is a control language designed to work on the Woods Hole Oceanographic Institute micro modem, supporting both 13-bit and 32-byte messages. The 13-bit message contains four slots that provide the following information: formation position, whether or not the communicating vehicle is the leader, what vehicles it has heard from in the last communication cycle (i.e., the “connection vector”), and what task it is currently performing (e.g., “in formation”, “inspecting”). The 32-byte message begins with 2 bytes devoted to standard header information, followed by packets of varying sizes that are designed around specific communication needs (e.g., “Report Vehicle Progress”, “Report Mine-Like Object (MLO) Location”, “Move Vehicle to Inspect a MLO”).

The TDMA communication protocol was designed to accommodate messages of different sizes in a fixed order, beginning with 13-bit messages and ending with 32-byte messages. The protocol is scalable across simulations to suit the number of vehicles in the mission; however, within a simulation, the number of slots in the protocol is fixed. The protocol includes 13-bit slots for each of the vehicles in the fleet and a spare 13-bit slot for a non-fleet communication source (e.g., a base station), along with slots for two 32-byte messages. In a typical simulation involving a fleet of five vehicles, the cycle will begin with 6 three-second slots, one for each vehicle and a spare for a non-fleet source. In each of these slots, the vehicles send out a navigation ping followed by a 13-bit message. After 18 seconds have elapsed, the first 32-byte message is sent out by the leader in a six second slot. The final slot in the protocol is also six seconds in length, and it is set aside for a second 32-byte message that is assigned by the leader. As Fig. 1 makes clear, the slots in this cycle are fixed, and successful use of the TDMA protocol requires that all communication sources have synchronized clocks. As indicated above, however, this is not a reasonable assumption to make if the communication language and protocol are to be portable from platform to platform in a front-seat/back-seat paradigm.

IV. TOKEN-BASED MEDIUM ACCESS CONTROL SOLUTION

The token-based medium access control (TMAC) solution was designed to overcome the limitations of the TDMA protocol in handling dropped messages during a communication cycle. TDMA communication will fail whenever sources of desynchronization are present in the fleet. This problem has a negative impact on performance, giving rise to problems such as duplicate mine-like object (MLO) point generation and delay in message transmission and reception.

The TMAC solution has been designed to maintain the communication cycle shown in Fig. 1 while relaxing the time-dependency of the TDMA protocol that employs the same cycle. This approach has made the TMAC solution address the problem of dropped messages more effectively. We now describe in detail the structure and dynamics of the TMAC solution.

In the UI TMAC solution, every vehicle in the fleet maintains a token in its memory. Token updates are made flexible to accommodate the occurrence of dropped messages. In this token solution, the token is not passed from one vehicle
to another; rather, it is maintained on each vehicle as a continuously updated record of fleet communications. While similar in some respects to token ring protocols [14], this approach does not involve maintaining a ring of nodes for token passing.

The token is a specific area in each vehicle’s memory. In contrast with the standard token ring communication protocol, in which the token is part of the message passed from one network node to another, this token is not explicitly communicated as part of any vehicle message; rather, it remains and is updated in the vehicle’s memory. The structure of the token shown in fig. 2 is described below.

**Structure of the Token**

<table>
<thead>
<tr>
<th>Vehicle ID</th>
<th>Transmission Time</th>
<th>Message Type</th>
</tr>
</thead>
</table>

Fig. 2: The structure of a token showing three slots: Vehicle ID, Transmission time, and Message type.

**Vehicle ID:** This is the identification number of the vehicle whose turn it is to transmit the next message in the cycle. For the vehicle in possession of this token, this number indicates which AUV is expected to make the next broadcast. After a token update, a vehicle checks its identification number against that in its token as one of the criteria for claiming broadcast right.

**Transmission Time:** This indicates what time the next broadcast is expected. This time is updated differently by each vehicle in a communication cycle, based upon whether the vehicle is transmitting or receiving at that moment.

**Message Type:** This represents the type of message expected from the vehicle whose turn it is to make the next broadcast. There are three different types of messages that could be transmitted during a communication cycle: (a) the 13-bit message, (b) the leader’s 32-byte message, and (c) the spare 32-byte message assigned by the leader. The 13-bit message, the leader’s 32-byte message, and the spare 32-byte message are denoted by the numbers 1, 2, and 3, respectively. The communication cycle makes room for six 13-bit messages, one leader 32-byte message, and one spare 32-byte message. Typically, a given cycle will contain only five 13-bit messages, since that is the number of vehicles in the fleet, but the sixth slot is included to accommodate a non-fleet communication source. Depending on which vehicle is transmitting, the message type is updated differently, as we will see in the following sections.

V. **TOKEN UPDATE WITHOUT DROPPED MESSAGES**

In the TMAC solution, the token keeps track of the communication traffic generated and received by a given AUV in the fleet. The key to successful application of this protocol is the token update, i.e., the modification of the information fields in the token needed to reflect movement in the communication cycle. The updates are conducted to ensure that at any step in the communication cycle, the participating AUVs have the same expectations about what is to happen at the next step. A sufficient condition for expectation identity across vehicles—i.e., the equivalence of vehicle expectations across the fleet concerning which vehicle is next in line to transmit—is that their tokens are identical, but that is not necessary. As we will see, expectations are sensitive to whether a vehicle is transmitting or receiving at a given step in the cycle, as well as to whether any messages have been dropped. Thus, tokens can vary at a given step in the cycle while still underwriting expectation identity, although these variations will be systematic and predictable, based on the communication environment.

In what follows, we begin by considering the case in which no message is dropped and then move to the case involving a dropped message. In both cases, special attention will be paid to the differences that occur between transmitting and receiving vehicles.

A. **Transmitting a 13-bit Message**

Token update varies based on whether the vehicle at that instance of the communication cycle is transmitting or receiving a message. For a transmitting vehicle, its token is updated immediately after it sends its message to reflect which vehicle it expects to transmit next. Fig. 3 and 4 show the token in vehicle 1 before and after its 13-bit message broadcast respectively.

**Token: Vehicle 1**

| Vehicle ID = 1 | Transmission Time = T₁ | Message Type = 1 |

In Fig. 3, the token of vehicle 1 is shown before its 13-bit message is transmitted. Its vehicle number corresponds to the vehicle ID in its token. Hence, it transmits its 13-bit message when the transmission time is T₁ seconds.

Upon transmitting its 13-bit message, the token in vehicle 1 is updated, as shown in fig. 4. The vehicle ID in the token changes to 2, indicating that vehicle 2 is next in line to transmit a message. Vehicle 2 is expected to transmit a 13-bit message, as indicated by message type 1, at time T₁+3 seconds. Every vehicle undergoes this update process after transmitting its message.
The only variation will occur when the last 13-bit message is broadcast. Typically, the transmitting vehicle will be vehicle 5. Since it is the last vehicle in the fleet to transmit a 13-bit message and since it does not expect any communication source to fill the sixth slot it adds six seconds to its transmission time. This leaves space in the cycle open for a non-fleet transmission, should that be forthcoming, and readies the vehicle for the first 32-byte message. This message will be transmitted by the leader.

Fig. 5 shows the updated token in vehicle 5 after it has transmitted its last 13-bit message. The ID in the token indicates that vehicle 5 expects the vehicle 1—the fleet leader in this case—to transmit the first 32-byte message, as indicated by the message type. This leader message is expected 6 seconds after T, the time when vehicle 5 broadcast its 13-bit message. The 6 seconds makes room for receipt of vehicle 5’s message and an extra 13-bit message slot in the communication cycle.

B. Receiving a 13-bit Message
When a vehicle receives a message, it updates its token differently due to the times involved. When transmitting a message, a vehicle knows immediately that its token should be updated and it proceeds to do so. Updates cued by receipt of messages must wait until those messages have been received, and this takes time. The TMAC solution assumes that the 13-bit and 32-byte messages take 1 second to be recognized by fleet vehicles. Thus, by T+1 seconds, the transmitting vehicle will have updated its token, but a receiving vehicle will only have begun to receive the message and it will not have updated its token. This disagreement in the tokens is not a problem since it simply reflects their different perspectives.

Once the receiving vehicle receives the message, it can update its token. Since token identity implies expectation identity, the objective is to structure this update so that the updated token in the waiting vehicle is identical to the updated token in the transmitting vehicle. Given that it has taken 1 second to receive the message, it will add to the receipt time R a total equal to the time added by the transmitting vehicle minus 1 second. Thus, if upon receipt of a 13-bit message, the receiving vehicle (viz., vehicle 2) expects another 13-bit message, it will add 2 seconds to the message receipt time R, indicating that it expects to receive the next 13-bit message at R+2 seconds. This is displayed in fig. 6.

If it receives the last 13-bit message in the cycle from vehicle 5, then it will expect the next message to be the 32-byte message from the leader; in this case, it updates the transmission time field to R+5 seconds. In fig. 7, the token indicates that the leader’s 32-byte message (i.e., message type 2) is expected from vehicle 1 at R+5 seconds.

C. Transmitting and Receiving a 32-byte Message
A 32-byte message is transmitted in a cycle only after the last vehicle has transmitted its 13-bit message. This last vehicle usually will be vehicle 5 in a fleet of 5 vehicles, as illustrated in Figure 5 above. At the appropriate time, T+6 seconds relative to the token in vehicle 5, vehicle 1 transmits its 32-byte message and updates its token. Its message type is updated depending on whether or not it assigns a spare 32-byte message for use within that communication cycle. The leader could either assign the spare 32-byte message to itself or to another vehicle in the fleet. If it assigns the spare 32-byte message to a vehicle in the fleet, say vehicle 3, vehicle 1 will update its token as shown in figure 8 below.
VI. Token Update With Dropped Messages

Due to the nature of the acoustic environment, acoustic transmissions often fail [7]. In an effort to address this problem, the TMAC protocol has been bolstered to enable AUVs to respond to multiple independent scenarios involving dropped messages without having communications fail. It should be noted that the impact of dropped messages on token updates only affects receiving vehicles—a transmitting vehicle will not drop its own message, and so will update its token correctly during the process of transmitting its message, as described above.

This is due to the fact that while still using the internal clocks of the vehicles involved in the cycle, the TMAC protocol is more dependent on the time that a message is received. Thus, even if a vehicle’s clock is reporting an incorrect time, the vehicle will still be able to function by using the time at which messages are received as reference, as described in the previous section. It was assumed that the clocks of independent vehicles would be able to maintain synchronization with a margin of error less than 1 second, especially at the ranges used with the UI fleet in simulations.

Assuming a speed of sound between 1400-1500 m/s, and because the vehicles are approximately 40 yards apart, and in the most extreme case the furthest vehicles are no more than 400 to 500 yards away from each other, the transit time for an acoustic message in the ALWSE-MC environment is less than a second. Thus, seeing as the transit times assumed in the ALWSE-MC simulation environment were 1 second for 13-bit messages and 5 seconds for 32-byte messages, the simulation includes more response lag than would actually exist in a field test.

Therefore, if the clock of a given vehicle was out of sync with the clocks of the other vehicles, because of the short message transit delay, it proves inconsequential because the TMAC protocol relies not on the clock per se, but on the time of message receipt; as vehicles are receiving more messages than they are transmitting, synchronization of a vehicle within the TMAC cycle is possible even if that particular vehicle’s clock is reporting a different time than the other vehicles.

A. Dropped 13-bit Messages

For a dropped 13-bit message, update of the token occurs in the following fashion. First, a receiving vehicle waits to receive a message at the transmission time indicated in its token. If the vehicle is expecting to receive a 13-bit message, it will begin waiting at the transmission time indicated in the token for three seconds; if in that period it does not receive the message, it will deem the message dropped. If the vehicle expected to receive the message at R, it will deem the message dropped at R+3 seconds. At this time, it will do one of two things. If the transmitting vehicle’s number is any but the highest in the formation, the TMAC logics dictate that vehicle ID in the waiting vehicle’s token be updated by incrementing the current number by 1. The expected transmission time of the message is updated by adding one second to the current time rendered by the vehicle’s internal clock and storing this value into the token. The new value will be R+4 seconds. The expected message type contained within the token does not change, since we are not yet at the last 13-bit slot. Figure 10 illustrates this transition for a vehicle other than vehicle 2.

The addition of one second may seem to desynchronize the communication cycle, but it actually helps avoid message overlap, which renders the affected messages uninterpretable.
vehicle at any time during the window, not just at the beginning of the window. Because the vehicles are all running independent clocks, there is a possibility of synchronization loss, resulting in the delay in the receipt or transmission of a particular acoustic message. If after update, the token of a receiving vehicle indicates expected transmission from a vehicle other than itself, the additional second leaves open a bit of additional time should the message be merely delayed while also giving the next expected message time to arrive. This time could prove crucial if that message is also delayed, as this keeps the dropped message clock from starting earlier than it should for the next step in the cycle. If the updated token indicates that the receiving vehicle is next to transmit, the additional time helps it accommodate the delayed appearance of the previous message without risking overlap.

If the token shows that the transmitting vehicle’s number is the highest in the formation, then the dropped message is the last 13-bit message from a fleet source in the current cycle. To accommodate this, and to ensure transition from 13-bit to 32-byte messages, the waiting vehicle’s logics dictate that the token be updated to expect the leader’s 32-byte command message from the leader 7 seconds after the last expected receipt time, which is determined by the waiting vehicle’s internal clock. If the waiting vehicle expected the message at R seconds, then at R+3 seconds it will update its token to expect the leader’s 32-byte message at R+7 seconds. This allows time for the last 13-bit message and for the spare 13-bit window, plus an additional second to avoid overlap with delayed messages, as before. This update utilizes the waiting vehicle’s memory of the last known leader, which is noted regularly throughout a communications cycle. Finally, the expected message type contained within the token is changed to a 32-byte message. This process is illustrated in figure 11.

<table>
<thead>
<tr>
<th>Original Token showing expected 13-bit Message from Vehicle 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Updated Token after dropped 13-bit Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle 1</td>
</tr>
</tbody>
</table>

Fig. 11: Token update for a dropped 13-bit message from the vehicle with the last vehicle number. In the scenario described, vehicle 5 is the expected transmitting vehicle. When the 13-bit message from vehicle 5 is dropped, the intended recipient vehicle updates its token to reflect the fact that the leader is expected to broadcast a 32-byte message next; the 7 second window shown here is utilized to make room for the non-fleet 13-bit slot and to ensure no overlapping messages.

B. Dropped 32-byte Messages

For a dropped 32-byte message from either the leader or some other transmitting vehicle, a procedure similar to the one used for 13-bit messages is followed. First, the waiting vehicle recognizes that the message has been dropped once the 6-second window dedicated to a 32-byte message has expired. The vehicle accomplishes this via its internal clock. Once the waiting vehicle realizes this, the token is updated in one of two ways. If the dropped message was a command message from the leader, then the token is updated so that the next transmission expected is a spare 32-byte message from the leader. The expected transmission time within the token is updated by adding 1 second to the current clock time. Since we are dealing with a vehicle that dropped the leader’s 32-byte message, we know that the vehicle in question is not the leader. However, it may have a need to use the 32-byte message, and failure to preclude it from using the spare could lead to multiple vehicles attempting to use the second slot—the one that was assigned the message and those that are acting as if they were assigned the message because they need to use it. Updating the token in this fashion ensures that no vehicle will attempt to use the message unless it has received and processed the assignment from the leader; in addition, it prepares the vehicle to receive a 32-byte message—it will expect it from the leader but can receive it from any source—while also providing a 1 second buffer against overlap. This is important because of the length the 32-byte message and the fact that another 13-bit message is on its way. By expanding the 32-byte slots in the token as described, there is no chance that a vehicle could move past the spare slot too quickly, missing the second 32-byte message, and begin transmitting a 13-bit message that overlaps the spare message.

If the dropped message was a 32-byte spare message from one of the other vehicles, then the waiting vehicle updates its token to reflect that the next message ought to be a 13-bit message from the vehicle with the first vehicle number in the fleet. Thus, the waiting vehicle updates its token so that it expects a 13-bit message from the first fleet member after a 1 second delay.

Figs. 12, 13, and 14 display various updating scenarios involving dropped messages. They follow different vehicles in the fleet as they transmit and either receive or drop messages. One advantage of the TMAC protocol is its capacity for resynchronization. As we noted above, identical expectations may not correspond to identical tokens; however, if token difference arises because of message failure, the protocol has the resilience necessary to resynchronize tokens and bring expectations back into alignment. (See Appendix for Figures 13 and 14.)

Referencing the procedures for dealing with dropped messages shows that the TMAC protocol is robust enough to deal with dropped messages and sources of desynchronization. Because of its dependence on the time of receipt of a message, the number of opportunities to resynchronize is larger than the number of opportunities for a source of desynchronization to destabilize communications. For example, with five vehicles, each vehicle has the opportunity to transmit at least once every 30 seconds; the remaining time is spent receiving messages from other vehicles, thus ensuring that synchronization is maintained throughout all of the communications cycle.
members. Even a vehicle whose clock is reporting an incorrect time is capable of staying in sync with the other vehicles in the formation, albeit with a bias. As long as vehicle transmission windows are maintained by received messages, the TMAC protocol ensures that the communications cycle will continue.

VII. APPLICATION

The TMAC solution has been employed in a MCM reference mission being implemented at the UI. The effectiveness of the TDMA protocol is undermined by desynchronization between cycle members. This desynchronization can stem from two sources. First, because the vehicles are employing the front-seat/backseat paradigm previously described, each vehicle has decision-making logics running separately from vehicle controllers. Because of this separation, it is difficult to assume that the clocks of the front-seat will be synchronized with the clocks of the back-seat, resulting in questionable performance when the TDMA cycle is applied.

Second, each vehicle is independent of the other vehicles, which implies a decentralized clock arrangement. Multiple clocks operating within the same communications cycle can be a source of desynchronization if one or more of the clocks begins to drift out of alignment with the others. This problem is linked to the strict time synchronization requirement of the TDMA.

The logics of the MATLAB behavior module developed for the ALWSE-MC simulation environment use a random number generator to determine that a message is successfully received. If the message is received, it is processed by logics that dictate how the vehicle updates its token in memory, as described above. If it is lost, the logics offset the transmission time in the vehicle’s token by a predetermined number of seconds, depending on the type of message that should have been received.

In order to test the TMAC solution, it was applied to a fleet of three vehicles performing a MCM mission with an implied reacquire and identify (RID) search pattern. The mission was designed and simulated in the ALWSE-MC environment. One vehicle (viz., the “swimmer”) functioned as the fleet leader, searching a given operation area and identifying MLOs. It then communicated the coordinates of these MLOs to the remaining vehicles (viz., the “trailers”), which then inspected the MLOs by performing the implied RID search patterns. In this simulation, both trailers were tasked with the responsibility of carrying out a thorough inspection of a number of MLOs. In this mission, the TMAC solution supported robust communication among the vehicles, enabling the trailers to have a good average battery at the end of the mission. The results obtained showed that the TMAC solution improved the fleet’s ability to handle dropped messages while sustaining the performance level experienced in previous simulations involving the TDMA. This suggests that the TMAC protocol is at least as efficient as the TDMA protocol, while also handling dropped messages more effectively and enabling the UI AUV fleet to adopt a front-seat/back-seat control paradigm.

VIII. FUTURE WORK

The UI research team plans to focus effort on further development of the TMAC communications protocol. A mathematical model comparing vehicle resynchronization to message drop rate would be beneficial. Other analysis could involve investigating the resilience of the TMAC in the event of unforeseen circumstances, or a quantitative analysis of communication throughput as a function of dropped message probability. Improving the flexibility of the TMAC protocol to allow it to accommodating a greater degree of desynchronized vehicle communications would increase the applicability of the protocol. For example, enabling the TMAC protocol to support the dynamic addition and removal of communication sources would cause it to be more robust. Collapsing or expanding communications windows as necessary to accommodate vehicles exiting or entering the cycle would tighten the cycle structure and improve efficiency.

Further work into expansion of the TMAC cycle would involve the inclusion of a non-fleet communications source into the communications cycle. Allowing a non-fleet member to participate in the cycle intermittently at a specified time, or to simply break into the cycle in an emergency, would provide significant benefits to the behaviors and logics of the formation.
APPENDIX

Fig. 13: Dropped 13-bit message scenario. In this scenario, vehicle 1 acting as the formation leader, and vehicle 2 fails to receive a 13-bit message from it. Vehicle 2 re-synchronizes by identifying that a message was dropped, and then updating its token accordingly; it is re-synchronized with the other vehicles by $t = 11$ seconds.

Fig. 14: Dropped spare 32-byte message scenario. In this scenario, vehicle 1 acts as the formation leader and gives vehicle 3 permission to broadcast a spare 32-byte message. Vehicle 5 fails to receive this message, and therefore defaults to expecting a spare 32-byte message from vehicle 1. Vehicle 5 is eventually re-synchronized with the other members of the formation after waiting the spare 32-byte message from vehicle 3, and all three vehicles are completely re-synchronized by $T = 203$ seconds.
ACKNOWLEDGMENT

The authors acknowledge the Office of Naval Research, which has supported the work under the award “Cooperative Autonomous Underwater Vehicle used to Search Large Ocean Areas for Mines” No. N00014-08-1-0276

REFERENCES