MACA-APT: A MACA-based Adaptive Packet Train transmission protocol for Underwater Acoustic Networks

Saiful Azad*, Paolo Casari‡, Khandekar Tabin Hasan*, Michele Zorzi‡
*Department of Computer Science, American International University, Bangladesh
‡Department of Information Engineering, University of Padova, via Gradenigo 6/B, 35131 Padova, Italy
{sazadm684, tabin}@aiub.edu, {casarip, zorzi}@dei.unipd.it

ABSTRACT

In wireless communications, collision is one of the principal sources of energy wastage, which often makes collision avoidance strategies preferred for medium access control (MAC) protocols. In this paper, we propose a collision avoidance-based MAC protocol called MACA-based Adaptive Packet Train (MACA-APT), which has been designed specifically for underwater acoustic networks (UANs). The design explicitly accounts for prominent characteristics of UANs such as long propagation delays and typically high bit error rates. In particular, the former is compensated via the transmission of multiple consecutive packets to multiple different receivers; the latter, instead, is tackled by embedding a cross-layer Stop-&-Wait ARQ scheme within MACA-APT.

The performance of the proposed protocol is evaluated via simulations and compared to another MAC protocol, also based on MACA, showing that MACA-APT achieves better performance for low to intermediate packet generation rates, and equivalent performance at higher rates. Moreover, we assess the impact of the packet train size on the performance of either protocol. This result is a first step towards the design of adaptive multi-packet multi-receiver MAC protocols for underwater networks.

Categories and Subject Descriptors

C.2.0 [**Communication/Networking and Information Technology**]: General—*Data communications*; I.6.6 [**Cooperative Underwater Communications**]: Simulation and Modeling—*Simulation Output Analysis*

General Terms

Design, Measurement, Performance

Keywords

Underwater acoustic networks; MAC protocols; collision avoidance; MACA; adaptive packet train length; performance evaluation; simulation.

1. INTRODUCTION

In general, a MAC protocol based on collision avoidance prescribes the transmission of a single packet after every successful handshake [1–3]. This choice fits well in terrestrial wireless networks, where a handshake is typically completed within a relatively short time. The case of UANs is different, as typically long propagation delays (relative to a data packet transmission time) and larger contention area require longer handshake timings. Consequently, channels undergo lower utilization and the network achieves a lower throughput performance with respect to handshake-free schemes [4]. To increase the channel utilization, several MAC protocols prescribe the transmission of a packet train to a single neighbor or to multiple neighbors [5–7] within a single session, defined as the protocol operations that encompass a successfully completed preliminary handshake (if any is prescribed) and the ensuing data transmission phase. When transmitting packets to multiple receivers, all neighboring nodes that can do harm to the current transmission should refrain from accessing the channel. Eventually, non-participating nodes could be put to sleep in order to preserve energy until the ongoing session has been completed.

Although a few collision-avoidance protocols in the literature can transmit multiple consecutive packets to compensate for long propagation delays and reduce the impact of the overhead due to handshakes, the throughput-optimal number of packets that should be exchanged after a successful handshake has been comparatively less investigated. For example, one such investigation was conducted in [8] for the case of a link between two nodes with Automatic Repeat reQuest (ARQ). A similar study on packet size that achieves the optimal balance between the MAC protocol overhead and the chance of collisions was carried out in [9]. In this paper, we investigate the same issue in the case of a multiple-access network and propose our MACA-APT protocol, which improves the channel utilization by transmitting multiple consecutive packets after every successful handshake completion. It is adaptive since it does not require a fixed window size for packet transmission, but rather supports a variable window size, as described in Section 3.

Moreover, MACA-APT incorporates error-control functions in a cross-layer fashion via a S&W ARQ scheme. This function is utilized to compensate for the typically high bit error rates experienced in underwater acoustic channels [10], and therefore recover from packet transmission errors. Most of the collision avoidance protocols [5–7] designed for UANs delegate this function to other layers of the protocol stack. By concentrating the information required for both channel access and error control into the same protocol, significant savings are achieved in terms of overhead, which makes the MACA-APT more amenable to be used in UANs.

2. RELATED WORK

As noted in Section 1, a traditional MACA-based protocol designed for terrestrial networks is unable to achieve a satisfactory throughput in UANs, because a single packet transmission is performed after every successfully completed handshake. There are a few MACA-based protocols already proposed in the literature

WUWNet'14, November 12 - 14 2014, Rome, Italy
Copyright 2014 ACM 978-1-4503-3277-4/14/11 ...$15.00
http://dx.doi.org/10.1145/2671490.2674580.

for underwater acoustic networks. Among them, those described in [2, 3] also transmit a single packet after every successful handshake. Due to the long propagation delay and low transmission bit rates, this approach is not efficient under water, as the contention period and handshake durations would be very long. In turn, this would mean exceedingly low channel utilization and low throughput. In order to increase the channel utilization, a MACA-based MAC protocol called MACA-MN was proposed in [5]. In MACA-MN, a node transmits a fixed-length packet train after every successful handshake. However, a node may not have a sufficient number of packets in queue to transmit a complete train, and there is no rule that allows the release of unused channel access time to other nodes. Hence, a node may waste valuable communication resources, which could be allotted to other nodes.

In [6], an extension of the MACA-MN protocol, called Reverse Opportunistic Packet Appending (ROPA), is proposed. In ROPA, when a sender completes its packet transmissions after a successful handshake, it immediately starts receiving packets. This technique reduces the amount of time spent on control signaling. A slightly different protocol called BiC-MAC [7] improves the channel utilization by employing bidirectional data packet exchanges among two communicating nodes after every handshake round.

Most protocols mentioned above assume that packet transmissions fail prominently because of collisions, and in any event delegate error control functions to other protocols. This can cause an increase of the overhead, which could be kept limited by embedding error control into the MAC layer. This is the design choice in our proposed protocol, as detailed in Section 3.

3. THE MACA-APT PROTOCOL

The following subsections detail our proposed MACA-APT protocol: Sections 3.1 and 3.2 deal with the handshaking and packet transmission scheme of MACA-APT, whereas Section 3.3 details the cross-layer ARQ technique embedded in MACA-APT.

3.1 Handshaking Technique

In general, collision avoidance is achieved by completing a handshake before any data transmissions. While it is true that this represents an overhead for the network and a possible source of collisions [9, 11], it is a cost to be paid in order to organize the communications in such a way that collisions are less likely to occur. For handshaking, MACA-based protocols generally employ Request-To-Send (`RTS`) and Clear-To-Send (`CTS`) packets: `RTS`s are transmitted to understand whether the receiving node (or nodes) are ready to receive, whereas `CTS`s inform the sender that the receiver is ready to receive packets.

Handshaking procedures become slightly more involved when a sender must contact multiple receivers within a single session. In particular, the following aspects must be taken into account: *i*) whether a sender should transmit a single `RTS` for all receivers, or rather multiple `RTS`s, one for each receiver; *ii*) how many `CTS`s should be received before a handshake is considered complete; *iii*) how to handle the case where no `CTS`s are received.

MACA-APT has been designed to solve the above issues as follows. When a node is idle and has packets to transmit, it senses the channel for a random amount of time in order to detect ongoing sessions. Any packet reception within this sensing period is interpreted as an ongoing session, and makes the sensing node enter a sleep state for the remainder of the session duration. This is made possible by piggybacking the session duration in every control and data packet, *à la* 802.11 [1]. We remark that this would be useful also to coordinate any nodes that should want to join the network.

After channel sensing, the sender moves to the `RTS` transmission state. To prepare the `RTS`, the node inspects its packet buffer to discover all receivers to whom it has packets to transmit, and compiles a list to be included in the `RTS` packet. Each packet in the buffer can be in one of the three following states: `NOT_YET_TX`, `SELECTED`, and `WAITING_FOR_ACK`. Every packet received from the upper layer is initially marked as `NOT_YET_TX`. The number of packets to be transmitted in the coming session is chosen adaptively. Unlike [5–7], a node i computes the size of the packet train it should transmit via an expression of the form $L_i = \min(Q_i, M)$, where L_i is the number of packets to be transmitted in a train, Q_i is the number of packets in the buffer of node i, M is a user-defined threshold, which can be substituted by some optimum number of packets to be sent, $\bar{L}$, if this value is known given the network topology and the channel conditions. In the following, we will look for $\bar{L}$ via simulation, and therefore play with M in order to understand the impact that this value has on the overall performance of the network. All packets selected for transmission are marked as `SELECTED`. MACA-APT estimates the duration of the upcoming session as follows:

$$T = 5\tau_{\max} + t_{\texttt{RTS}} + L_i \times t_{\texttt{DATA}} + 2\delta \qquad (1)$$

where $\tau_{\max}$ is the maximum propagation delay in the network, $t_{\texttt{RTS}}$ and $t_{\texttt{DATA}}$ are the transmission times of an `RTS` and of a `DATA` packet, respectively, L_i is the number of `DATA` packets selected for transmission and δ is a guard time. Note that the maximum propagation delay $\tau_{\max}$ is counted 5 times in (1) for the following reason. In MACA-APT the transmitter senses the channel once before the `RTS` packet transmission and once more before the `DATA` packet transmission, in order to minimize the chance of collisions. The remaining factor $3\tau_{\max}$ is a sufficiently long time to accommodate the transmission and reception of multiple `CTS`s from multiple nodes. As anticipated above, the sender includes T in the `RTS` packet as a measure of the session duration.

After receiving an `RTS`, any idle nodes not included in the list of intended receivers move to the sleep state and start a timer of duration T. This makes it possible to avoid useless idle listening, which is a source of energy wastage. Intended receivers, if idle, generate a response packet called CTS with Report Annex (`CRA`). The `CRA` is an extension of currently available `CTS` packets in MACA or MACA-based protocols. By the reception of a `CRA` packet, the `RTS` sender realizes the availability of the receiver. In `CRA` packets, the node also includes a report of the packets that have been correctly received in the previous session that involved these two nodes. This is the way the sender gets an acknowledgment for correctly delivered packets: since the process takes place at every new handshake, the information about correct deliveries is often refreshed. The details of this S&W ARQ technique are reported in Section 3.3. The `CRA` sender also includes the expected session duration, in order to inform its neighbors and avoid unwanted channel accesses. After transmitting the `CRA`, the node waits for `DATA` packet receptions by enabling a fixed length timer which is equal to the session duration. When any neighbor receives the `CRA`, it notes down the duration of the upcoming session and moves to the sleep state if it is not among the intended receivers.

As noted in Section 1, the vulnerability periods affecting a MAC protocol for UANs are long, due to the significant propagation delays. Therefore, when a node senses a channel before transmitting an `RTS` packet, it may still receive an `RTS`, a `CRA` or a `DATA` packet. In case of `CRA` and `DATA` packets, the node reads the session duration T in the packet header and moves to the sleep state in order to preserve energy. If the node receives an RTS packet from a different source right after having transmitted its own RTS packet, it drops the received RTS and keeps waiting for CRA packets. Conversely, it reads T from the received packet and moves to the sleep state. Again, when a node is waiting to receive DATA packets, if it receives an RTS where it is also an intended receiver, it drops that packet since it has already committed to another node. In case of DATA packet reception before a CRA packet transmission, the node immediately moves to the sleep state.

After receiving a CRA packet, the RTS sender stores the packet in a buffer. It keeps performing this until the timer expires. and then checks the buffer to count the number of CRA packets it receives from various receivers. If this number is lower than expected, it withholds the communication and goes to the idle state. Since the packets previously picked for transmission from its buffer (and thus marked SELECTED) are not transmitted in this case, they are marked back as NOT_YET_TX.

3.2 Data Packet Transmission

After the CRA reception timer expires, the RTS sender again starts sensing the channel for a random duration to make sure that no communication is ongoing. It only refrains from transmitting if it receives a DATA packet from another node, which in turn makes it move to the sleep state. In case the node should receive an RTS or CRA packet, it drops them, and instead keeps waiting for the expiration of the sensing timer. When the timer expires, the node moves to the data transmission state. It only transmits those DATA packets which have been chosen for transmission when the RTS was generated, and were therefore marked as SELECTED. The status of each transmitted DATA packet is advanced to WAITING_FOR_ACK after the transmission has been performed. We note that every DATA packet carries a flag that makes it possible to identify the last packet to be transmitted in the current session. Any node that receives a DATA packet marked as the last one moves to the idle state.

Note that the inclusion of the session duration in every packet increases the overhead bits spent to organize the network traffic. However, this information is very helpful for two reasons, namely i) it makes it possible for a joining node to easily detect the ongoing communications and their duration, and ii) if a node wakes up too early during an ongoing session (e.g., due to a drifting internal clock) it can read the remaining session duration from any DATA packet it receives from its neighbors and thereby move back to the sleep state again.

3.3 Cross-Layer based S&W ARQ

To compensate for the packet transmission errors that likely occur in UANs, while keeping the additional overhead to a minimum, we embedded a S&W error control protocol in MACA-APT in a cross-layer fashion. Among other things, this makes it possible to avoid the potential ACK storms that may occur in a MAC protocol that targets multiple packet transmissions to multiple receivers. Sending reports on previous successful transmissions is the purpose of the CRA packets introduced in Section 3.1.

When a node generates a CRA packet, it digs into its reception buffer to discover all packets for which an acknowledgment has not yet been delivered to the current RTS sender. As it would require too many bits to selectively report the sequence number of all packets to be acknowledged, a node only includes the lowest and highest sequence numbers and a bitmap in the packet. An example of such bitmap is shown in Fig. 1 where every bit refers to a packet. The lowest sequence number represents the first bit of the map and the highest represents the last bit. A 1 means that

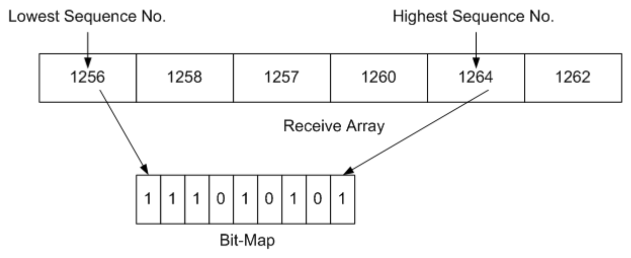

Figure 1: An example of acknowledgment bitmap generation based on the receiver buffer The first entry in the bitmap corresponds to the packet with sequence number 1256, whereas the last entry corresponds to the last received packet (1264).

the corresponding packet has been received successfully, whereas a 0 means it is not received or has been dropped because of errors. After sending the CRA, the node releases the buffer and delivers the acknowledged packets in order to the application layer (in Fig. 1, packets 1256 to 1258). When an RTS sender receives a CRA packet, it utilizes the highest and the lowest sequence numbers mentioned in the packet, as well as the bitmap, to discover all successfully delivered packets. It then deletes all the delivered packets from the buffer, whereas any erroneous packets are marked back as NOT_YET_TX, so that they can be transmitted at a later attempt. Packets can be retransmitted up to a user-defined maximum number of times.

4. SIMULATION SCENARIOS

The performance of the proposed MACA-APT protocol is evaluated via a simulation campaign and by considering various aspects which influence the performance of the protocol. All simulations are carried out using the DESERT Underwater framework [12, 13], which can realistically reproduce complex interactions between multiple nodes over a shared communication medium. The World Ocean Simulation System (WOSS) [11, 13] has also been used in order to accurately reproduce the behavior of underwater acoustic channels via the Bellhop ray tracing software [14], to which WOSS automatically feeds environmental data based on the configuration of the network simulation.

For simulation, we chose an area in the Mediterranean Sea, near the Pianosa Island, Italy, which is set at $(42.590°N, 10.125°E)$. The network covers an area of 1500 m $\times$ 1500 m. Every node is placed randomly within the area and one meter above the seabed at its own location, as acquired from the bathymetry database [15]. The Binary Phase Shift Keying (BPSK) modulation technique is utilized by all the nodes at a bit rate of $R_b = 4800$ bps. The length of DATA packets is fixed to 125 Bytes, and the size of the other packets is dynamically managed depending on the information to be included in them. We perform simulations for several values of the data generation rates per node, e.g., $\lambda' = 2$ bps up to $\lambda' = 100$ bps of payload information per generating node per destination. The traffic is generated randomly according to a Poisson process. At the beginning of each simulation run, the number of destinations per node is chosen uniformly at random between two and the maximum number of potential receivers $N - 1$, where N is the number of network nodes. The results of the simulations are averaged over 25 runs. Most of the simulations in this paper are conducted using 5 nodes unless otherwise mentioned.

We compare MACA-APT against another MACA-based proto-

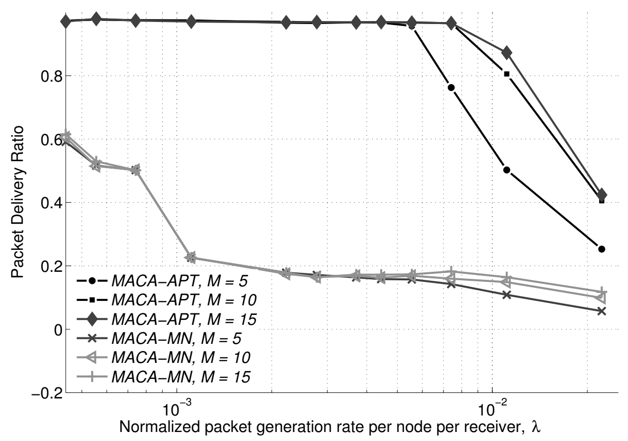

Figure 2: Packet Delivery Ratio vs. normalized packet generation rate per node, λ, for a network of $N = 5$ nodes.

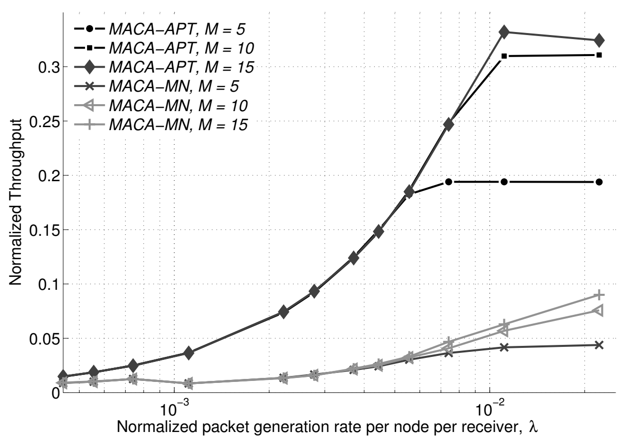

Figure 3: Normalized Throughput vs. normalized packet generation rate per node, λ, for a network of $N = 5$ nodes.

col named MACA-MN [5], which is the most similar to MACA-APT among the protocols in the literature. An overview of this protocol is given below in Subsection 4.1

4.1 MACA-MN protocol

Similar to our proposed MACA-APT protocol, the MACA-MN can also transmit multiple packets to multiple neighbors during a single session. In MACA-MN, a node triggers an RTS in one of the following two cases: i) if it has not transmitted an RTS for a given time duration $T_{\max}$; and ii) if it has an adequate number of packets (M_{train}) in the buffer. A sender generates an RTS packet, which includes the ID of all neighbors to which it wishes to transmit DATA packets. It then broadcasts the packet locally and starts a waiting timer to listen to the replies from its neighbors. When an intended destination receives an RTS packet and is in the idle state, it replies with a CTS packet. The RTS sender buffers a received CTS packet which is intended for it during the listening period. After the timer expires, the sender moves to the data transmission state and sends the DATA packet to those neighbors from which it received a reply.

4.2 Simulation Results

We divide our simulation campaign into two parts. In the first part, we compare our proposed protocol with a modified version of the MACA-MN protocol which is discussed in Subsection 4.1; in the second part, we investigate the performance of MACA-APT by changing the maximum number of packets that a node can send in the same train. We note that the original MACA-MN protocol does not embed any error correction technique. In order to achieve a more fair comparison, we modified MACA-MN by including a S&W ARQ scheme in it as well. Figs. 2 and 3 show the packet delivery ratio (defined as the ratio of the number of packets correctly delivered to their intended destination to the number of generated packets) and the normalized throughput (defined as the number of packet correctly delivered per packet transmission time), respectively, as a function of $\lambda = \lambda'/R_b$, where R_b is the transmission bit rate.

From Figs. 2 and 3, it can be observed that for any value of λ MACA-APT outperforms MACA-MN. There are a couple of significant reasons for these performance differences. In case of data transmission, MACA-APT adopts a conservative approach whereas MACA-MN adopts a greedy approach: MACA-APT prescribes that lack of reception of an expected CRA (even from only one among multiple neighbors), should make the sender refrain from data transmission and schedule a later attempt. Conversely, MACA-MN requires that a node transmits packets to all neighbors that correctly answered with a CTS. This greedy data transmission approach introduces more collisions in the network, and results in lower packet delivery ratio as well as lower normalized throughput. Moreover, as opposed to MACA-MN, MACA-APT is opportunistic since it transmits DATA packets utilizing an adaptive train size, depending also on the number of packets available in the buffers of the senders at the time an RTS is generated.

We now turn to investigating the performance of MACA-APT by changing the maximum number of packets that a node can send in the same train. We consider a single link between $N = 2$ nodes as well as a network of $N = 5$ nodes. The node locations are set so that the network is fully connected, in order to maximize the stress on the MAC protocol. Since our objective is to investigate the optimum number of consecutive packet transmissions, we consider several values for the packet train size, M, from 1 to 20.

In Fig. 4, we report the PDR against λ for different values of M and N. We observe that the choice of $M = 5$ or higher results in the highest PDR (close to 1) for $N = 2$. On the other hand, M does not have any impact on the network performance at lower packet generation rates, for $N = 5$. However, with increasing traffic, higher values of M result in higher PDR except when $M = 20$. This performance difference is due to the higher number of contenders competing for channel access. In particular, for higher values of M, a node transmits more packets during each channel access, on average. In turn, other nodes wait longer for their chance to transmit. When a session ends, all the waiting nodes start their own attempt, and the intense contention that results leads to a high probability of collisions among MACA-APT's RTS/CRA signaling messages. Hence, a node may fail to complete the handshake, or multiple nodes may erroneously access the channel. In the former case, the channel is underused, whereas in the latter case, the packet trains transmitted by different nodes are likely to collide. If the value chosen for M is too small, on the other hand, the channel becomes underused as well, because the number of packets transmitted per channel access is too small. In the scenario presented here, choosing $M = 5$ leads to the optimum tradeoff between these two undesirable working points.

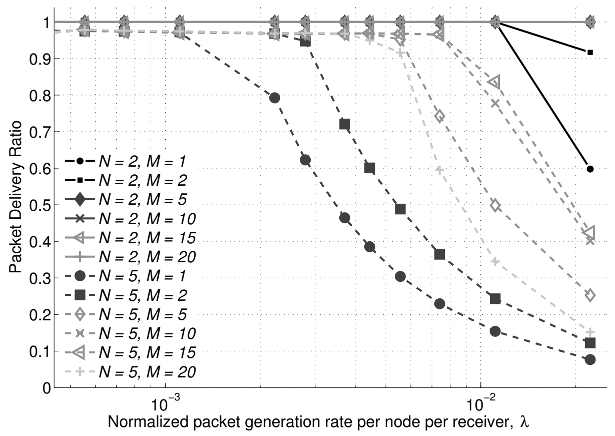

Figure 4: Packet Delivery Ratio for MACA-APT vs. normalized packet generation rate per node, λ, for different number of nodes N and different packet train size M.

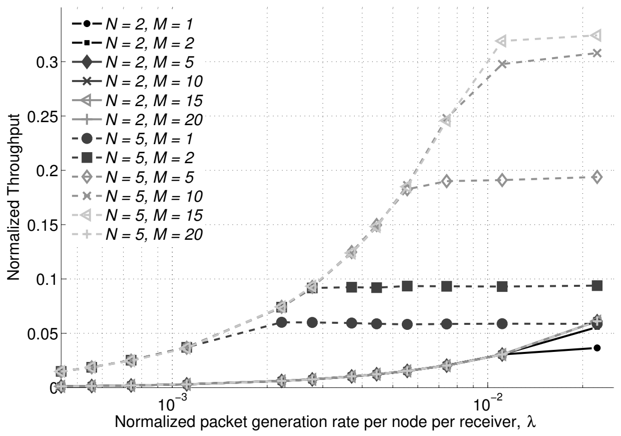

Figure 5: Normalized Throughput for MACA-APT vs. normalized packet generation rate per node, λ, for different number of nodes N and different packet train size M.

From Fig. 5, we further observe that for increasing values of M up to $M = 15$, the throughput also increases. When $M = 20$, the throughput drops for the same reason already mentioned for the PDR in Fig 4. The general behavior of the curves is to increase for increasing values of λ and finally level to a value that depends on N and M. For the values considered in this set of simulations, the highest normalized throughput (about 0.35) is achieved when $N = 5$ and $M = 15$.

5. CONCLUSIONS

In this paper, we proposed a new MACA-based Adaptive Packet Train (MACA-APT) protocol for administering channel access in UANs. MACA-APT transmits a train of packets of adaptive size to help compensate the protocol overhead and the typically long propagation delays incurred in UANs. MACA-APT also embeds a cross-layer S&W ARQ scheme. This technique causes a minimal increase to the protocol overhead thanks to a modified `CTS` packet, called `CRA`, which includes a delivery report to notify the sender about correctly received packets. We conducted a simulation campaign to evaluate MACA-APT and compare its performance against that of a similar competing protocol. We finally demonstrate the impact of the packet train size on the performance of MACA-APT. The latter results can be used as a starting point to tune the performance of any other MACA-based protocol employing packet train transmissions.

Acknowledgements

This work has been partially supported by the US Office of Naval Research under Grant no. N62909-14-1-N127.

6. REFERENCES

[1] IEEE 802 LAN/MAN Standards Committee, "Wireless LAN Medium Access Control MAC and Physical Layer (PHY) Specifications," IEEE Standard 802.11, 1999.

[2] H.-H. Ng, W.-S. Soh, and M. Motani, "MACA-U: A media access protocol for underwater acoustic networks," in *Proc. IEEE Globecom*, New Orleans, Louisiana, USA, Dec. 2008.

[3] W. Lin, E. Cheng, and F. Yuan, "A MACA-based MAC protocol for underwater acoustic sensor networks," *Academy Publisher Journal of Communications*, vol. 6, no. 2, pp. 179–184, Dec. 2011.

[4] F. Guerra, P. Casari, and M. Zorzi, "A performance comparison of MAC protocols for underwater networks using a realistic channel simulator," in *Proc. MTS/IEEE Oceans*, Biloxi, MS, Oct. 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 Spring*, Singapore, May 2008.

[6] H.-H. Ng, W.-S. Soh, and M. Motani, "ROPA: A MAC Protocol for Underwater Acoustic Networks with Reverse Opportunistic Packet Appending," in *Proc. IEEE WCNC*, Sydney, Australia, Apr. 2010.

[7] ——, "BiC-MAC: Bidirectional-Concurrent MAC Protocol with Packet Bursting for Underwater Acoustic Networks," in *Proc. MTS/IEEE OCEANS*, Seattle, WA, Sep. 2010.

[8] M. Stojanovic, "Optimization of a data link protocol for an underwater acoustic channel," in *Proc. IEEE Oceans*, Brest, France, Jun. 2005, pp. 68–73.

[9] S. Basagni, C. Petrioli, R. Petroccia, and M. Stojanovic, "Optimized packet size selection in underwater wireless sensor network communications," *IEEE J. Ocean. Eng.*, vol. 37, no. 3, pp. 321–337, Jul. 2012.

[10] S. Azad, P. Casari, and Michele Zorzi, "The underwater selective repeat error control protocol for multiuser acoustic networks: Design and parameter optimization," *IEEE Trans. Wireless Commun.*, vol. 12, no. 10, pp. 4866–4877, Oct. 2013.

[11] F. Guerra, P. Casari, and M. Zorzi, "World Ocean Simulation System (WOSS): a simulation tool for underwater networks with realistic propagation modeling," in *Proc. ACM WUWNet 2009*, Berkeley, CA, Nov. 2009.

[12] R. Masiero, S. Azad, F. Favaro, M. Petrani, G. Toso, F. Guerra, P. Casari, and M. Zorzi, "DESERT Underwater: an ns–MIRACLE-based framework to DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols," in *Proc. MTS/IEEE OCEANS*, Yeosu, South Korea, May 2012.

[13] P. Casari, C. Tapparello, I. Calabrese, F. Favaro, G. Toso, S. Azad, R. Masiero, and M. Zorzi, "Open-source suites for the underwater networking community: WOSS and DESERT Underwater," *IEEE Network, special issue on "Open Source for Networking: Development and Experimentation"*, vol. 28, no. 5, pp. 38–46, Sep. 2014.

[14] M. Porter *et al.*, "Bellhop code." [Online]. Available: http://oalib.hlsresearch.com/Rays/index.html

[15] "General bathymetric chart of the oceans." [Online]. Available: www.gebco.net