A Detailed Simulation Study of the UWAN-MAC Protocol for Underwater Acoustic Networks

Paolo Casari, Fabio E. Lapiccirella and **Michele Zorzi**

Department of Information Engineering — University of Padova — Via Gradenigo 6/B, I-35131 Padova, Italy
E-mail: {casarip, zorzi}@dei.unipd.it , fabioemilio.lapiccirella@gmail.com

***Abstract*— In this paper, we present a simulation study of UWAN-MAC, a recently proposed Medium Access Control (MAC) protocol for underwater acoustic networks. While the assumptions behind the protocol are interesting (coordinating the nodes' transmissions through some sort of adaptive TDMA and saving energy through sleep/awake cycles), yet they bring further challenges into the picture, *e.g.*, the need to keep nodes synchronized in an efficient way. We present a set of results that show how critical the assumptions behind the protocol are, and specifically focus on how quickly the network performance degrades in the absence of updated synchronization information. Our results show that a periodic resynchronization is necessary and allow a study of the tradeoff between the signaling overhead and the overall network performance, in terms of throughput, success ratio, asynchrony among nodes, and energy consumption.**

***Index Terms*— Acoustic telemetry and communication; Access, custody, and retrieval of data; Information management.**

I. Introduction and Related Work

DISTRIBUTED underwater sensor networks are an emerging research area, that currently stimulates an increasing number of research contributions. The interest around this subject is well justified by the additional challenges posed by wireless underwater networking with respect to terrestrial radio communications. Radio waves scatter rapidly underwater, allowing reasonable communication performance only over short distances. As an alternative to radio waves, optical technologies also enable underwater communications, but are feasible only within a limited reach, besides requiring further efforts to keep the transmitter and receiver aligned.

On the other hand, acoustic waves have been used for decades in underwater environments for different purposes, such as telemetry and sonar detection. Their application to underwater networking, instead, is rather new, as seminal contributions in the '90s proved the feasibility of underwater networking using acoustic communications. Acoustic waves bring a quite different propagation behavior into the picture [1]. First of all, they propagate at a slow speed $c \approx 1500$ m/s, which is five orders of magnitude smaller than for radio propagation in the air. The propagation speed actually changes with the depth, temperature, and salinity of the water [1]. Secondly, the attenuation incurred by acoustic waves is a function of both the covered distance and the frequency of the transmitted signal, according to the equation $A(d, f) = d^k a(f)^d$, where k describes the geometry of propagation similarly to the path loss exponent for terrestrial radio, and $a(f)$ is the absorption factor, determined by the chemical composition of water. Also, unlike in radio, the noise process at the receiver side is not white, but has a frequency-dependent power spectral density, that also depends on environmental conditions such as wind-driven waves and shipping activities. The superposition of the frequency dependence exhibited by both the attenuation and the noise yields a relationship between the available communication bandwidth and the distance of the two parties, as the bandwidth tends to shrink for increasing distance [2]. Nonetheless, properly designed acoustic systems can communicate over distances on the order of 100 km. This makes acoustic waves a suitable solution for underwater networking, as acoustic links can be required to bridge distances on the order of some kilometers.

UnderWater Acoustic Sensor Networks (UWASN) are currently at an early stage of development. Their possible future applications include water sampling, environmental monitoring, forecast of extreme weather conditions (such as a *tsunami*), support to navigation, and so forth. With the support of Autonomous Underwater Vehicles (AUVs), UWASNs may also offer an automated solution to water monitoring needs. Deploying networked underwater devices is expensive, and therefore it would be desirable that such networks offer the maximum performance for the longest time, keeping human intervention to a minimum during operation. This calls for the design of efficient communication schemes and protocols, offering reasonable data transport capabilities and high energy efficiency. Among these protocols, Underwater Wireless Acoustic Networks – Medium Access Control (UWAN-MAC) [3], [4] has been among the first to be proposed and specifically designed to work in UWASNs. It is conceived with energy saving in mind, in that nodes alternate between a sleep and an awake mode, and try to do so only when strictly necessary, *i.e.*, when they should listen to a neighboring transmission or transmit something themselves. As the main protocol operations depend on the synchronism among the schedules of the nodes to successfully establish links, a drift among them can be potentially harmful.

In this work, we accurately describe and analyze UWAN-MAC in order to assess the effects of such synchronization problems. We show how important it is to keep the schedules aligned and how fast the protocol performance decays under synchronization drift, and suggest a means of choosing a resynchronization interval that balances the need to keep the network working and the need not to waste resources on signaling traffic.

II. Description of UWAN-MAC

UWAN-MAC [3], [4] is among the first protocols specifically designed for underwater networks, as it takes care of both

the high propagation delays of acoustic waves and the power savings required by underwater devices. This protocol tries to avoid packet collision through setting up adaptive TDMA schedules agreed upon and shared by neighboring nodes. Schedules are used for understanding when a node should wake up to hear a transmission from a neighbor and when it is its own turn to transmit. Whenever no communications are expected, the node goes to sleep. Prior to actual packet transmissions, the network undergoes a setup phase. Both the setup and the data exchange phases are described in the following sections. More details can be found in [3], [4].

A. Setup Phase

Each node sets its duty cycle to the same value d. Since with UWAN-MAC nodes wake up specifically to listen to neighbors, the usual duty cycle definition (the ratio of the awake time over the cycle duration) does not hold here. To be more meaningful, the duty cycle is defined in the following as the ratio of a data packet transmission time, τ_D, to the total cycle time T_0. Hence, each node sets its initial cycle length to $T_0 = \tau_D/d$. No node actually goes to sleep in the setup phase. After the initial choice, each node broadcasts its own transmission instant and the cycle duration using a `SYNC` message. This way, after receiving a `SYNC` from all neighbors, a sensor is notified that the next reception of a data packet from the sender of the `SYNC` will take place exactly one cycle period after the reception instant, and thus knows when to wake up even without being aware of the propagation delay. This setup phase needs to be carried out periodically, in order to keep the schedules of the nodes synchronized. In fact, network operations as described in the following Section may lead node reception phases to be misaligned with the neighbors' transmission epochs, even in a network with static nodes. This point will be discussed later.

B. Transmission and Listen Phase

If the initialization phase is completed without errors, each node knows when to wake up to listen to one of its neighbors and when it is its turn to transmit. When a node wakes up to transmit, it sends packets composed of three fields:

- `SYNC`: allows a node to communicate changes in the cycle period length that may be necessary in order to avoid collisions (see also below)
- `MISSING`: contains a list of the nodes the present sender did not receive anything from at the specified epoch, and is used to trigger a confirmation of presence by those nodes
- `DATA`: the actual payload of the packet.

Since the initial choice of the transmission epoch is random, it may happen that two nodes exhibit overlapping transmission periods. If both nodes transmitted at this time, a collision would result. To avoid this event, each transmitter adjusts its wakeup epoch according to the following. Define $\mathcal{N}(i)$ as the set of neighbors of node i. Let X_i and T_i be the current transmit epoch and cycle period for node i, respectively. Finally, let $W_{i,k}$ be the instant when node i wakes up to listen to its neighbor k, $k \in \mathcal{N}(i)$. The new transmit epoch is chosen as follows:[1]

$$\text{if } \exists k \text{ s.t. } |X_i - W_{i,k}| + \tau_D < \tau_g \tag{1}$$

then select $X_i' \in (X_i, X_i + T_i)$
subject to:

$$|X_i' - W_{i,k}| + \tau_D \geq \tau_g \ \ \forall k\,, \tag{2}$$

where τ_D is the data transmission time, and τ_g is a suitably chosen guard time that ensures the absence of collisions and the reception of `HELLO`s when needed. For this purpose, it suffices to set τ_g to one data transmission time, τ_D, plus the maximum propagation delay in the network. The constraint in (2) means that if there is a node whose transmission instant is scheduled to less than a guard time from that of node i, then node i changes its own instant so that it is spaced at least τ_g from the wakeup epochs already scheduled to listen to any other neighbor (*i.e.*, the instants when a reception from that neighbor would begin). Note that, depending on d, the current cycle period T_i might be too short to accommodate the new transmission schedule. To solve this problem, the node can change its cycle according to $T_i' = T_i + |X_i' - X_i|$. Since our results show that nodes tend to increase their cycle period indefinitely, we propose to limit its length to be at most 20% greater than the nominal value T_0, and reset it to T_0 if it exceeds this threshold. It is worth noticing that the new wakeup epoch must be communicated to the neighbors, so that they can wake up at the right moment. This cannot be done before a data packet containing the `SYNC` field is transmitted, which can lead to misalignments between the actual transmission epoch and the instant the neighbors will wake up to listen to node i's transmission. This fact is a serious drawback of the protocol.

After having sent its data packet, a node waits before going back to sleep and listens to the channel. This is necessary to let nodes listed in the `MISSING` field of the data packet advertise their presence with a `HELLO` packet. In case a `HELLO` is received, the node knows that this previously missing sender is actually physically present in the neighborhood and will continue waking up at the arranged epochs to listen to its transmissions. The duration of the listen phase is set to allow `HELLO`s from possible neighbors to actually reach the data sender. If new nodes are allowed to enter the network, this phase also allows them to advertise their presence.

It can be observed that UWAN-MAC can be potentially very effective in saving energy at nodes (nodes need only wake up at certain instants), but critically relies on updated and synchronized wakeup schedules between neighbors. To study the effects of the misalignment of transmit/listen phases among communication parties is the main objective of this paper.

III. Results

A. Network Setting

The following results are obtained by simulating UWAN-MAC over a network formed of 20 nodes. The nodes are randomly deployed over a $4\ \text{km} \times 4\ \text{km}$ square area. We

[1]Note that, compared to [4], our condition has the additional term τ_D to account for the fact that the two transmissions should not overlap.

assume that the network is static, in the sense that nodes do not move from their initial position and, in addition, no nodes leave or enter the network during the simulation. Traffic is generated according to a Poisson process with rate λ pkt/min for the whole network. The nodes follow sleep/awake schedules with duty cycle d ranging from 0.002 to 0.009, depending on the specific setting and the objectives of the experiments performed. The reason why the chosen values for the duty cycles are so small is that UWAN-MAC requires a low duty cycle in order to accommodate all transmission and listening wakeup epochs at all nodes with the constraint expressed by (2). From our results, it is likely that this is not possible for duty cycles higher than 0.01.

The underwater channel model described in [2] is reproduced here. Specifically, we use it to calculate the bandwidth where all the nodes can operate and hear each other within the network area. We explicitly model the capture effect by deriving the Signal-To-Interference-and-Noise ratio (SINR) for each packet being received. This way, if two packets collide but exhibit very different received powers, it is possible that the more powerful can be received correctly. The actual signal bandwidth is 18.26 kHz wide, from 6.56 kHz to 24.82 kHz. We assume that a signaling rate of exactly 18.26 kbps is reachable in this bandwidth.

Routing is performed by computing minimum-energy forwarding paths as follows. Firstly, a neighbor discovery algorithm is run. We set a "coverage range" for each node, in the sense that all other nodes within coverage can receive the sender's transmission with an SINR exceeding a certain prescribed minimum value. Then we refined the neighbor discovery information by varying the coverage range of the nodes so that neighbors are "mutual", *i.e.*, they have each other in their neighbor list. This is particularly important for UWAN-MAC as, for example, the `MISSING` field in a data packet can trigger a `HELLO` only if the inquired node actually considers the sender as its neighbor.[2] To compute routes, we used the Bellman-Ford algorithm, and defined the link cost as the energy required to bridge the link. It should be noted that the originally proposed version of the algorithm was designed to adapt to topology-changing networks. For example, leaving or dying nodes are canceled from the neighbor list of a sender when they do not answer the `MISSING` field for a prescribed number of times. Since we test the network in a static topology and we wish to concentrate on MAC issues, we leave the neighbor list unchanged even if a neighbor results repeatedly missing.

For our evaluation, we have developed an event-driven MATLAB simulator that fully reproduces the behavior of UWAN-MAC. All results are obtained by averaging over 20 different multi-hop topologies, in order to provide sufficient statistical confidence.

[2]Neighbors may not be mutual if, for example, a node is isolated and needs to increase its transmit power (thus coverage range) in order to actually have neighbors. In our setting, the minimum number of neighbors is 3. Each of these will then need to adjust its coverage range accordingly, in order to allow bidirectional communications.

B. Simulation Results and Discussion

Our results are mainly aimed at assessing how the general performance of UWAN-MAC depends on synchronization among nodes. As mentioned in Section II, most of the protocol operations hinge on waking up receivers when a neighboring transmitter is sending data. Nevertheless, transmission wakeup instants are constrained to be at least a guard time τ_g apart from all of the neighbors' transmission epochs. If this condition is not satisfied by the initial random choices, nodes may decide not to send data and change their own transmission epoch according to (2). The neighbors can only be informed of this through a `HELLO` packet following a `MISSING` notification in a transmission. If more than one node is forced to change its schedule, it might be possible that their transmission/reception wake-up epochs do not coincide any more, and thus synchronization is no longer maintained.

In Section II we recalled that a suitable solution (yet possibly burdensome) is to go through a `SYNC` message exchange again after a number of cycles, say every tenth cycle. Our first evaluation refers to this case. Figures 1 and 2 depict the throughput (defined as the number of packets per second that successfully reach their destination) and the transmission success ratio of UWAN-MAC, with resynchronization every 10 cycles, as a function of the packet generation rate per node λ and for different values of the duty cycle d. If resynchronization is performed frequently enough (*e.g.*, every 10 cycles as we did), UWAN-MAC shows a stable behavior, and increasing the duty cycle has a beneficial effect on throughput. Yet, recall that a higher duty cycle decreases the degrees of freedom of the protocol, by imposing shorter cycles, and making it more difficult to accommodate all receive and transmit wakeup epochs while still satisfying (2). As the nodes try to adapt their transmission schedules in a more constrained, high duty cycle condition, schedule misalignments and collisions may take place, which in turn lowers the success ratio, as shown in Figure 2. This is actually an important drawback for the protocol. The number of transmissions performed at higher duty cycle increases, so that the actual network throughput increases as well despite the lower success ratio. Yet, more energy is wasted on packet losses due to collisions. In underwater environments, where transmissions are responsible for the biggest share of the power expense, collisions should be avoided [5] or at least limited [6]. This argument is further supported by Figure 3, that shows the ratio of the energy wasted on collisions to the total energy consumption. Figure 3 confirms that increasing the duty cycle makes the node spend more energy uselessly, up to 65% at the highest value of the duty cycle (the one yielding the highest throughput in our experiments). The previous results suggest that the duty cycle, that is a controllable parameter, can be tuned in order to achieve a higher throughput at the price of a higher energy consumption. Figure 4 plots the tradeoff between throughput and energy wasted due to collisions, where each curve corresponds to one traffic value and is spanned left to right by increasing the duty cycle d. At low traffic, the curve is almost flat. This means that the most convenient choice is to keep d as low as possible, since this does not significantly affect the network performance. Conversely, for higher traffic, increasing the duty cycle causes a proportional increase of

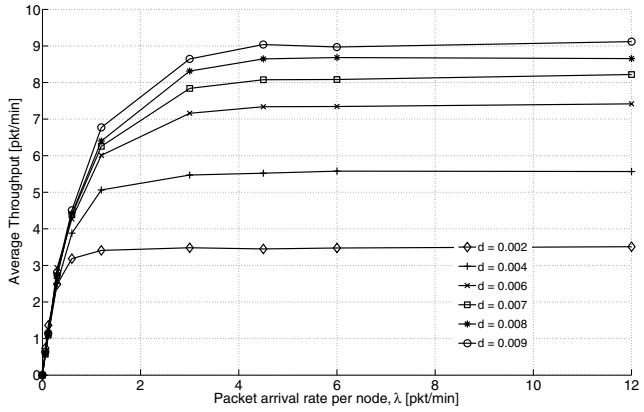

Fig. 1. Throughput vs. traffic for varying d, with resynchronization every 10 cycles.

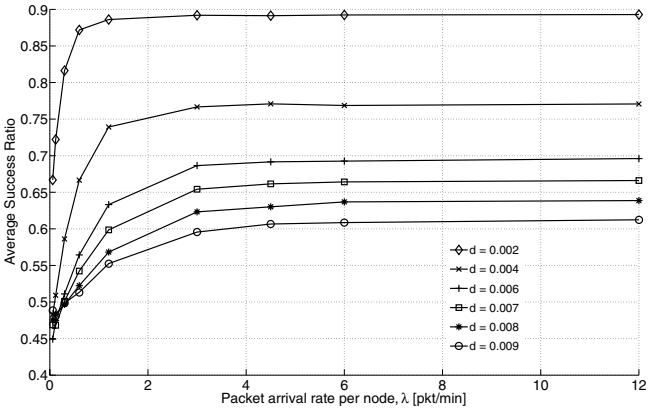

Fig. 2. Success ratio vs. traffic for varying d, with resynchronization every 10 cycles.

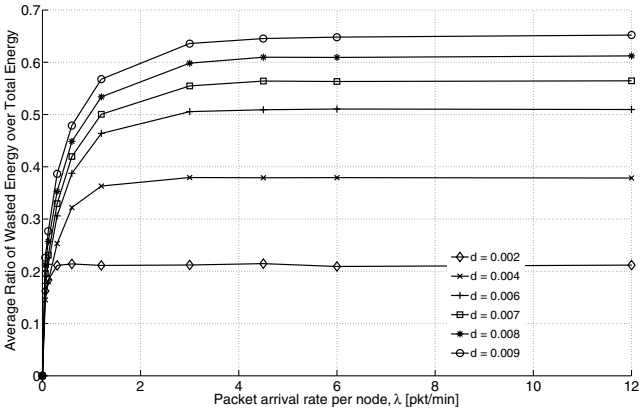

Fig. 3. Energy wasted due to collisions vs. traffic for varying d, with resynchronization every 10 cycles.

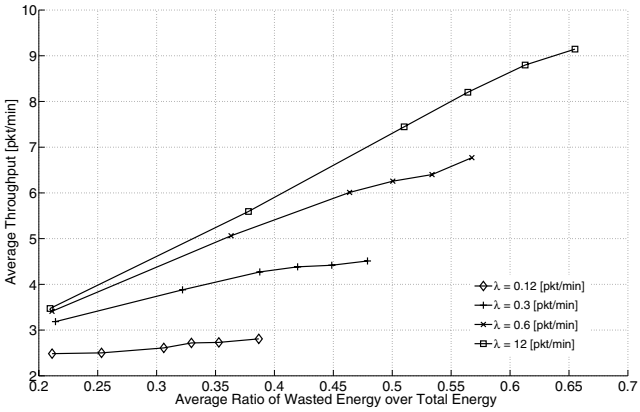

Fig. 4. Throughput vs. energy wasted due to collisions with resynchronization every 10 cycles for different traffic values.

both the throughput and the energy wasted on collisions, and allowing a tradeoff between throughput (hence data transport capabilities) and energy consumption (hence communication efficiency and network lifetime).

All previous results were drawn in the case where resynchronization takes place once every 10 cycles. It could be argued that this is a very high rate, as every tenth cycle yields no throughput. To give some insight into the importance of a sufficiently frequent resynchronization, Figure 5 shows a superposition of the instantaneous throughput and the average success ratio per cycle as a function of time. There we also depict the *asynchrony factor*, defined as the fraction of nodes that did not schedule their wake up epochs correctly for the upcoming cycle, causing, for example, unnecessary wakeups to hear transmitters that would not send data, or vice-versa, sleeping when a neighbor is actually awake to transmit. We set the duty cycle to 0.002 for this experiment and consider the case $\lambda = 18$, which corresponds to the saturation condition. Figure 5 suggests that the throughput experiences a steep drop if synchronization is not refreshed regularly. The reason behind this is the increasingly higher asynchrony among the nodes' schedules, the correspondingly greater number of failures in listening to transmissions, and hence the lower success ratio. In order to keep the nodes sufficiently synchronized, the `SYNC` message exchange should be performed before the asynchrony factor gets too high (no more than 10–20%) and the success ratio limits throughput. 10 cycles are a good choice in this case. This is also confirmed by Figure 6, that shows the same results of Figure 5, but resynchronizing nodes every 10 cycles. This keeps the asynchrony under control (usually below 10% within a cycle, but never more than 20%), and a better throughput is granted by a high enough success ratio.

Nevertheless, the effectiveness of the resynchronization rate is affected by the duty cycle. In fact, higher duty cycles tend to worsen the transmission performance more quickly, because they impose stricter constraints on the schedules of the nodes. To support this claim, we set the duty cycle to 0.004 and repeat the previous experiments. The results are shown in Figures 7 and 8. Doubling the duty cycle halves the cycle time for all nodes and makes it more difficult to arrange the schedules. Overall, this causes a worse asynchrony, that spreads much faster and causes a severe throughput decrease, until receptions are only sporadically correct. Resynchronizing only every 10 cycles helps, but is still an insufficient effort. From Figure 8 we can observe that even if the throughput is improved as opposed to the case without resynchronization, yet the success ratio does not go beyond 0.8, and the asynchrony typically oscillates between 15% and 30%. A shorter interval between `SYNC` exchanges, such as 3 to 5 cycles, could help in this case, but would represent a serious limitation for the network, as it

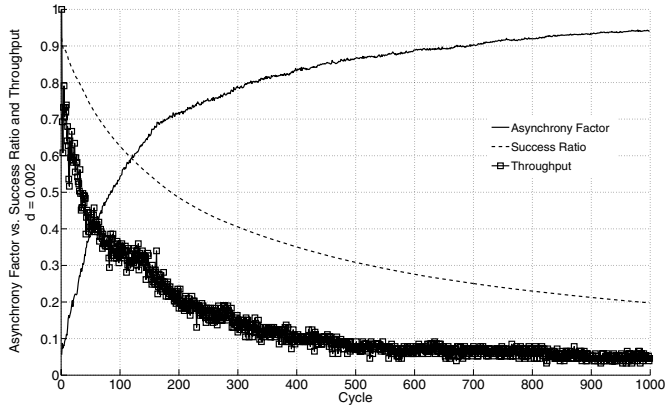

Fig. 5. Throughput, average success ratio and asynchrony factor per cycle without resynchronization, $d=0.002$, $\lambda=18$.

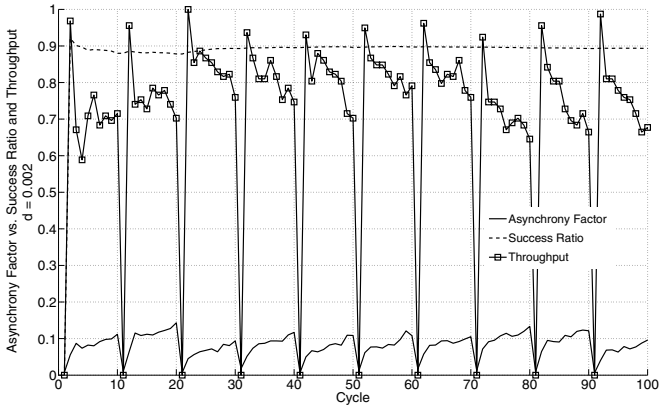

Fig. 6. Throughput, average success ratio and asynchrony factor per cycle with resynchronization, $d=0.002$, $\lambda=18$.

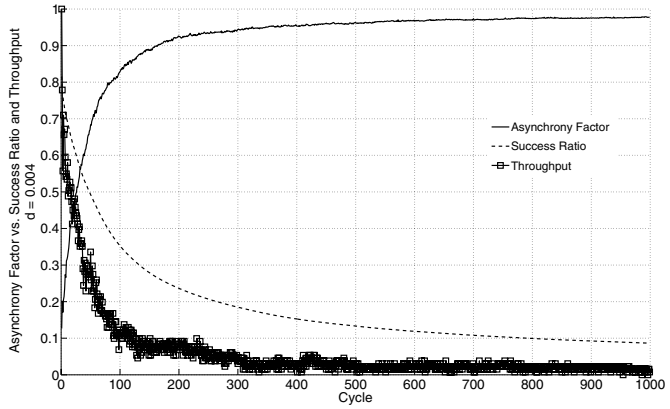

Fig. 7. Throughput, average success ratio and asynchrony factor per cycle without resynchronization, $d=0.004$, $\lambda=18$.

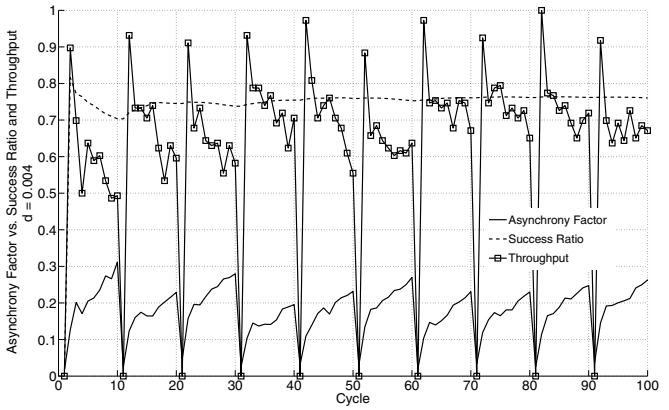

Fig. 8. Throughput, average success ratio and asynchrony factor per cycle with resynchronization, $d=0.004$, $\lambda=18$.

would dedicate a lot of resources to keeping the schedules updated, thereby sacrificing throughput. Yet, given the rapid drift of the synchronization, this is the only way to keep UWAN-MAC running with higher duty cycles. Whether or not to perform these protocol adjustments is a design choice and largely depends on the application to support. On one hand, long network operations would require duty cycles even shorter than this one, in order to limit the synchronization drift and require only infrequent exchanges of `SYNC` messages. On the other hand, higher duty cycles may be chosen for time-critical operations that require a prompt data delivery, thus providing greater throughput at the price of higher energy wasted on collisions.

As a last set of results showing the importance of keeping the network synchronized, we show in Figure 9 the success ratio decrease as a function of the asynchrony factor, for $d = 0.002$ and 0.004. As synchronization drifts, the average success ratio tends to drop. In addition, a higher duty cycle causes worse performance at even lower asynchrony. Given that a periodic `SYNC` exchange is needed to keep the protocol working, these results, along with those in Figures such as 5-6 and 7-8, can help design the right resynchronization interval. For example, looking at Figure 9, we might choose not to let the success ratio drop more than 5% below its initial value, which means, for example, allowing for a maximum asynchrony of 0.1 for $d = 0.002$ and of 0.2 for $d = 0.004$. Then we look at Figures 5 and 7 and conservatively choose to resynchronize every 10 cycles for $d = 0.002$ and every 5 cycles for $d = 0.004$. In the example above, we chose to preserve the success ratio, because failing transmissions yields a high cost in terms of wasted energy, but similar considerations may apply to the throughput or to any other metric as measured in the simulations, depending on the specific design needs

IV. Conclusions

In this paper, we have performed an analysis of UWAN-MAC, a recently proposed protocol that is specifically designed to be applied to UnderWater Acoustic Sensor Networks (UWASNs), aiming at showing its advantages and drawbacks. More specifically, we have focused on the tradeoff between the duty cycle of the nodes, the throughput of the network, the success ratio and the energy consumption. Since UWAN-MAC strongly relies on the synchronization between the nodes' adaptive TDMA schedules, we have highlighted how fast network performance drops for increasing synchronization drift, and confirmed periodic exchange of synchronization messages to be a good solution for the protocol. Through our results, we suggested that this decrease be measured so as to

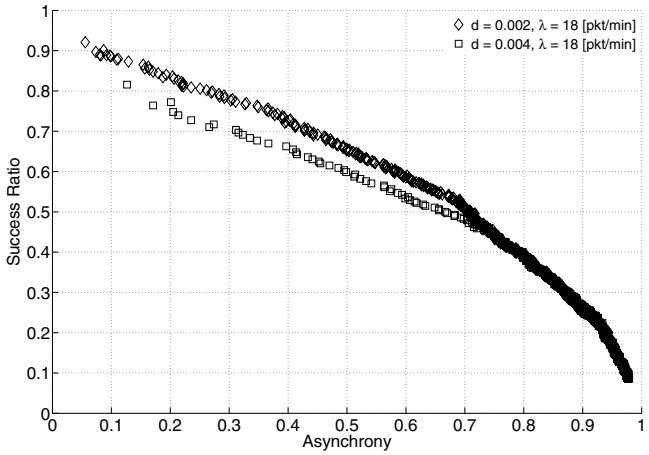

Fig. 9. Average success ratio vs. asynchrony factor per cycle without resynchronization, $d = 0.002, 0.004$, $\lambda = 18$.

set a suitable resynchronization interval. This interval should be short enough not to let asynchrony grow too much, while being suitably long not to impose too much signaling burden on the network.

Future directions of this work include the comparison of UWAN-MAC to other MAC protocols for UWASNs and an analysis of its suitability under different application requirements and with different routing and broadcasting protocols.

V. Related Work

The study of networking protocols for underwater sensor networks is a relatively new topic. Currently, some works have pointed out the different pros and cons of Time-, Frequency- and Code-Division Multiple Access schemes [7], [8], also taking clustering into account. ALOHA and a collision avoidance scheme have also been compared in [9]. While the schemes proposed in those papers are quite general, the design of efficient protocols for underwater Medium Access Control (MAC) protocols is still in its infancy. In the following we briefly review the main approaches found in the literature.

Slotted FAMA [5], for example, strives to save energy by avoiding collisions through handshaking and carrier sensing. All nodes in the network are synchronized to a slotted TDMA schedule. Each slot is long enough to accommodate the maximum propagation time in the network. Transmitters are allowed to initiate a transmission only at the beginning of a new slot. The usual 4-way handshake is employed, with a Request-To-Send (RTS) issued by the transmitter, followed by a Clear-To-Send (CTS) from the receiver, by the actual DATA transmission and by some feedback from the receiver in the form of an ACK/NACK packet. Channel sensing is performed prior to every RTS transmission, and lasts two slots (to cover one round-trip time). PCAP [10] pursues similar objectives by making the duration of a handshake fixed, thus predictable. This is achieved by having the receiver wait before sending a CTS, so that the transmitter hears the CTS exactly after one maximum round-trip time. With this solution, all the neighbors know when transmissions will take place, and can schedule theirs accordingly. Moreover, a node can perform other system- or application-required tasks during protocol idle times. The MAC protocol proposed in [6] works in the absence of a shared synchronization. The protocol is based on an RTS/CTS exchange, with an additional waiting time before sending data that allows to directly postpone the transmissions if a concurrent handshake is detected nearby. In addition, the protocol helps avoid collisions by using warning messages: if a receiver, after sending a CTS, hears an RTS from a second node, it sends out a message to the transmitter asking it to refrain from sending data. This protocol does not avoid collisions completely but tries to limit them, ultimately outperforming protocols such as Slotted FAMA, that require more listening time to ensure a proper channel access. In [11] the efficient management of idle sensor time is addressed, also exploiting the lower power consumption required to receive an acoustic signal, as opposed to transmission. Tone–Lohi [12] also exploits this fact and proposes to avoid collisions by sending very short busy tones to signal that the channel is being used.

Acknowledgment

This work is supported in part by NOAA's Sea Grant College Program, Project no. NA060AR4170019.

References

[1] R. Urick, *Principles of Underwater Sound*. McGraw-Hill, 1983.
[2] M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," in *Proc. ACM WUWNet*, Los Angeles, CA, Sept. 2006, pp. 41–47.
[3] M. K. Park and V. Rodoplu, "An Energy–efficient MAC protocol for underwater wireless acoustic networks," in *Proc. IEEE/OES Oceans*, Brest, France, June 2005, pp. 1198–1203.
[4] ——, "UWAN-MAC: an energy-efficient MAC protocol for underwater acoustic wireless networks," *IEEE J. Oceanic Eng.*, 2007, to appear. [Online]. Available: http://www.ece.ucsb.edu/rodoplu/Pubs/ParkRodoplu_UWANMAC.pdf
[5] M. Molins and M. Stojanovic, "Slotted FAMA: a MAC Protocol for underwater acoustic networks," in *Proc. IEEE Oceans*, Singapore, Sept. 2006.
[6] B. Peleato and M. Stojanovic, "A MAC protocol for ad hoc underwater acoustic sensor networks," in *Proc. ACM WUWNet*, Los Angeles, CA, Sept. 2006, pp. 113–115.
[7] E. M. Sozer, M. Stojanovic, and J. G. Proakis, "Underwater acoustic networks," *IEEE J. Oceanic Eng.*, vol. 25, no. 1, pp. 72–83, Jan. 2000.
[8] P. Casari, S. Marella, and M. Zorzi, "A comparison of multiple access techniques in clustered underwater acoustic networks," in *Proc. IEEE/OES Oceans*, Aberdeen, Scotland, June 2007.
[9] Jose dos Santos Coelho, "Underwater acoustic networks: evaluation of the impact of media access control on latency in a delay constrained network," Master's thesis, Naval Postgraduate School, Monterey, CA, Mar. 2005.
[10] X. Guo, M. Frater, and M. Ryan, "A propagation-delay-tolerant collision avoidance protocol for underwater acoustic sensor networks," in *Proc. IEEE Oceans*, Singapore, Sept. 2006.
[11] A. F. Harris III, M. Stojanovic and M. Zorzi, "When underwater acoustic nodes should sleep with one eye open: idle–time power management in underwater sensor networks," in *Proc. ACM WUWNet*, Los Angeles, CA, Sept. 2006, pp. 105–108.
[12] A. Syed, W. Ye, and J. Heidemann, "Medium Access Control for Underwater Acoustic Networks," in *Proc. ACM WUWNet*, Los Angeles, CA, Sept. 2006, work-in-progress paper.