On ARQ Strategies over Random Access Protocols in Underwater Acoustic Networks

Saiful Azad*, Paolo Casari*†, Federico Guerra*†, Michele Zorzi*†

*Department of Information Engineering, University of Padova, via Gradenigo 6/B, 35131 Padova, Italy
†Consorzio Ferrara Ricerche, via Saragat 1, 44122 Ferrara, Italy
{azad,casarip,fguerra,zorzi}@dei.unipd.it

Abstract—**In this paper, we introduce a mechanism to improve the performance of ARQ over underwater links. Our scheme aims at reproducing a Selective Repeat ARQ strategy: to do this, it sets up a form of time-division duplex link between the transmitter and its receiver, by leveraging on the propagation delay incurred by underwater sound. In fact, such delay typically allows to interlace the transmission of data and ACK packets in such a way that the two operations do not interfere or cause nodes to be deaf to the transmissions of each other.**

We consider two different versions of our protocol (in terms of channel access persistence) and compare them against ALOHA and CSMA with and without ARQ, in both static and mobile scenarios. We conclude that in multiuser networks our form of Selective Repeat ARQ outperforms other ACK-based protocols at low and intermediate traffic.

Index Terms—**Underwater acoustic networks, random access, ARQ, selective repeat, performance evaluation, WOSS.**

I. Introduction

Most Medium Access Control (MAC) protocols designed for underwater networks can employ some Automatic Repeat reQuest (ARQ) error control technique to counter packet transmission errors. Such schemes require that the receiver performs a control on incoming packets (usually in the form of a Cyclic Redundancy Check, or CRC) and informs the sender about which packets should be retransmitted. The retransmission procedure may be administered according to three different schemes, namely Stop-and-Wait (S&W), Go-Back-N (GBN), and Selective Repeat (SR). With S&W, the sender transmits a packet and waits for the corresponding confirmation, or acknowledgement (ACK), before sending the next packet. If no ACK is received within a timeout period, the corresponding packet is retransmitted. With GBN, the transmitter can send a sequence of up to N consecutive numbered packets while waiting for an ACK; the receiver sends one ACK per received packet to inform the sender about the sequence number of the last correct reception; the transmitter slides forward the transmit window according to the information in the ACK packets: in case all N packets allowed by the window have been transmitted, and no ACKs have allowed to slide the window forward in the meanwhile, the sender retransmits the packets in order, starting from the left end of the window. Finally, SR prescribes that the receiver can acknowledge out-of-order packets (a different form of feedback implies informing the transmitter only about erroneous packets, e.g., via Negative ACKs, or NACKs). At the price of a re-sequencing buffer at the receiver to cope with out-of-order receptions, the SR mechanism limits retransmissions only to erroneous packets. On the contrary, in GBN even a single error can cause the retransmission of all packets in the window, increasing the probability that correct packets are uselessly sent again.

The S&W ARQ scheme is employed (or even assumed) by most MAC protocols, because of its simplicity and of the half-duplex nature of the underwater acoustic channel. However, this choice is inherently inefficient underwater, as it requires a sender to remain idle for at least a whole Round-Trip Time (RTT) after every packet transmission. Because the propagation speed of sound underwater is low, the throughput achievable by S&W is limited [1]. For this reason, several protocols such as those proposed in [1], [2] perform multiple packet transmissions back-to-back. However, this strategy requires prolonged channel usage, which we argue to be infeasible in multiuser networks with random access, also in light of the results in Sec. III.

While the SR scheme is known to outperform S&W, its implementation in underwater networks would require to set up a duplex channel, e.g., via frequency- or time-division. In this paper, we choose time-division duplexing (TDD) because it does not require a channel on a different frequency (which typically implies a separate transceiver due to the frequency selectivity of underwater transducers). In addition, time-division inherently exploits the long underwater propagation delays, e.g., a sender can transmit while waiting for the reception of an ACK packet from its receiver, and vice-versa. However, this raises the further problem of measuring the RTT between communicating nodes, and to check whether it is long enough to support interlaced data/ACK transmissions; in turn this will require to cope with the possible mobility of nodes, which makes the RTT vary over time.

In this paper, we propose and discuss an ARQ scheme implementing the above concepts, to be used along with random access-based MAC protocols. The scheme starts in a S&W mode, where the reception of the ACK related to the first packet transmission allows to measure the RTT and, if the RTT is large enough, to switch to SR, where the sequential transmission of more packets is allowed before actually receiving an ACK. Guard times are allowed to cope with changing sound speed.

Other papers considered TDD as a means of supporting

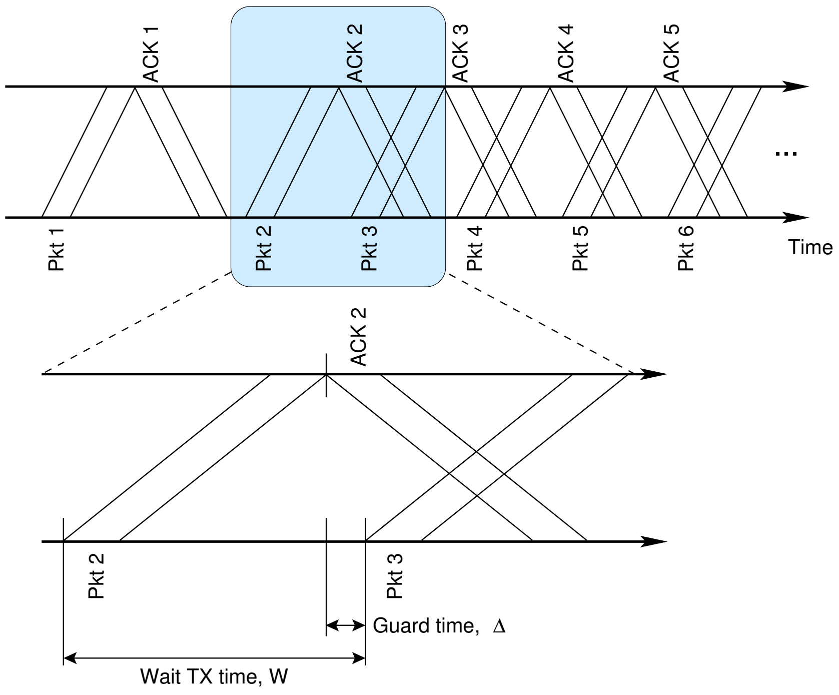

Figure 1. Messaging pattern of the Underwater Selective Repeat (USR) ARQ technique.

multiple packet transmissions throughout a single RTT. In [3], the authors proposed a technique where RTT is divided equally among two nodes. However, this technique requires perfect time synchronization between the transmitter and the receiver, which is difficult to achieve even in static networks. Moreover, the technique in [3] cannot be extended to larger networks. Another TDD-like technique, namely juggling-like-stop-and-wait (JSW), is proposed in [4]. With JSW, transmitters inject a fixed number of data packets in the network (according to a pre-calculated window size) and wait for the ACK/NACK packet. The transmitter always expects feedback related to previously transmitted data packets. However, in real networks, a node may as well transmit when it has only one packet to send, and should therefore expect the related ACK/NACK right thereafter. In this case, a plain S&W technique would be more suitable. Moreover, if the nodes are randomly placed in an area, it is suboptimal to employ the same window size for all nodes, since some may be located close to their receivers, and multiple packet transmissions in a single RTT may cause the transmitter to be deaf to the receiver's ACKs, and vice-versa. The authors in [5] devise a technique for improving the performance of SR ARQ over long delay channels: however, they design the protocol for a single-link, and do not report results on the performance of the scheme in multiuser scenarios.

The technique we propose in this paper adapts to the distance between the nodes and to the number of packets to be sent to the same destination: if the distance between the sender and the receiver is less than some threshold (which is specified in more detail in Sec. II) or if a sender has only one packet to transmit, the sender resorts to plain S&W; otherwise, it employs our proposed SR strategy. In addition, our protocol does not require any time synchronization, and it can be implemented in any network with any number of nodes (unlike, e.g., [3]).

II. Underwater Selective Repeat (USR)

In plain S&W ARQ, a sender waits for ACK packets after every data transmission. If no ACK is received within a given timeout period, the same packet is retransmitted. Call τ the propagation delay between the sender and its receiver, T_D the data packet transmission time, and T_A the ACK transmission time. In our proposed Underwater Selective Repeat (USR) technique (sketched for reference in Fig. 1) if the propagation delay τ is longer than the time required to receive one data packet and one ACK, i.e., $\tau > T_D + T_A$, the transmitter sends another packet within the same RTT, instead of waiting for the ACK. In order to avoid receiving an ACK while transmitting a data packet (which would result in deafness to the ACK), the sender waits for a fixed time W before sending the next packet. This is marked as Wait_tx_time in Fig. 1. The waiting time is calculated as $W = \tau + \Delta$, where Δ is a guard time required to compensate for changes in the RTT during the transmission (e.g., due to changes in the instantaneous sound speed). The interlacing of data transmission and ACK reception at the sender allows a significant performance improvement while keeping error control very simple.

We note that our technique works like plain S&W whenever $\tau \leq T_D + T_A$, since the data packet transmission would otherwise collide with ACK reception. S&W is also resorted to when there is only one packet to transmit in the queue. If $\tau > T_D + T_A$, it utilizes the SR technique. We also note that τ

has to be estimated, in order to understand whether plain S&W or SR is to be employed. To do this, the transmitter waits for an ACK after the first packet transmission, as in plain S&W. The ACK of the first packet helps the sender determine the distance of the receiver, hence whether or not it can send multiple packets in one RTT. If so, after transmitting one packet, the sender will check the queue to determine whether there are more packets to transmit. In case there are in fact multiple packets for the same destination, the sender will transmit again after a time equal to W. Otherwise, the sender just waits for the ACK. Note that the highest number of packets a node can transmit in one RTT using this technique is two.

The USR technique is simple and can be implemented over most MAC protocols. In what follows, we will consider USR coupled with the CSMA-ALOHA version employed in [6],[1] where the sensing phase is very short, and serves only the purpose to avoid superimposing a node's signal to other signals currently propagating in the same area, which would result in likely collisions at the receivers of those signals. The sensing phase is of random duration in order to avoid synchronization phenomena among different nodes. This opens the way to two different versions of USR: in USR-v1, a transmitter senses the channel only once, i.e., before starting the first packet transmission. Then the USR technique is applied as long as there are packets to send to the same node, or until any ACK is lost. This results in the transmission pattern shown at the top of Fig. 1. In the latter case, the node employs a standard binary exponential backoff to schedule a later attempt. When backoff is over, it senses the channel again and starts transmitting. In USR-v2, instead, sensing is repeated every other packet (after the ACK has been received or after the ACK timeout has expired). This is expected to make the protocol more robust to collisions. Like in USR-v1, the sender waits for a random backoff time before rescheduling another transmission attempt.

In case the nodes are mobile, the round-trip times are subject to change. This makes USR-v1 unsuitable, since its transmission scheme may sooner or later lead the transmitter to be deaf to ACKs (or the receiver to be deaf to data packets). On the contrary, USR-v2 overcomes this problem by repeating channel access procedures at regular intervals, and is thus more suitable for mobile scenarios. USR-v2 can also adapt its behavior depending on the movement pattern of the communicating nodes as follows.

Whenever a node A receives an ACK from another node B, it stores the information about the distance d_{AB} between them and the reception time t_1. Since underwater mobiles usually move at constant speed, it is reasonable to assume that each node knows the movement speed v. If, at some later time t_2, A has to transmit more packets to B, it can make a worst-case estimate of the current distance d'_{AB} using the stored information, according to the following relationship:

$$d'_{AB} = d_{AB} - (t_2 - t_1) \cdot v \cdot N_M, \qquad (1)$$

[1]USR can actually work on top of any MAC protocol: we choose CSMA-ALOHA for simplicity and easy comparison among the results in this and in previous papers, such as [7].

where N_M counts if the sender and receiver are both static ($N_M = 0$), if one of the two moves ($N_M = 1$), or if they are both mobile ($N_M = 2$), respectively. This is a worst-case estimate because we are assuming that nodes are always moving towards each other, and will eventually have to drop SR to turn to S&W. However, with this approach, the nodes do not have to estimate their position or their velocity vector.

III. Simulation Results

A. Scenario Description

The performance evaluation that follows has the objective to highlight the pros and cons of the USR scheme, as well as to compare it to other mechanisms employing random access or carrier sensing, both with and without error control. In particular, we consider a plain version of the ALOHA protocol [8], where nodes send packets directly as they are generated, a slotted version thereof, where nodes are synchronized, time is slotted, and transmissions can take place only at the beginning of a slot. In addition, we consider the form of Carrier-Sense Multiple Access (CSMA-ALOHA) employed in [6], shortly described in Sec. II. ALOHA, Slotted ALOHA and CSMA-ALOHA are considered both with and without S&W ARQ.

All protocols have been implemented using the ns2-Miracle framework [9]. Simulations are run using the World Ocean Simulation System (WOSS) package [6], which is employed to output realistic acoustic propagation results via the Bellhop ray tracing software [10]: WOSS queries ns2-Miracle for node positions, queries oceanographic databases for the data related to these positions, and feeds the retrieved environmental information (in terms of sound speed profile throughout the watercolumn, bottom bathymetry and sediments and related geoacoustic parameters) to Bellhop. A random displacement is added to the sound speed profile in order to generate different propagation behaviors. These behaviors are varied over time according to a user-tunable granularity, which has been set to $10\,\mathrm{s}$ here. With respect to the first version of WOSS described in [6] we have also added an option to generate time-varying surface waves profiles. New surface wave realizations are generated with the same granularity reported above.

The location chosen for simulation is a square area of side $1058\,\mathrm{m}$ in the Mediterranean Sea, whose upper-left corner is placed at $43.0217°N$, $9.3658°E$. All nodes are randomly distributed within the area. Network operations are assumed to be taking place in July, hence the corresponding SSP is retrieved from the oceanographic databases.

The nodes communicate via a Binary Phase Shift Keying (BPSK) modulation technique at a bit rate of $4800\,\mathrm{bps}$. The size of the data packet and the ACK packet are fixed to $L_D = 125\,\mathrm{Bytes}$ and $L_A = 10\,\mathrm{Bytes}$, respectively. The maximum number of retransmissions is 5. The simulations have been performed for various average data generation rates per node, λ, from $8\,\mathrm{bps}$ to $200\,\mathrm{bps}$ unless otherwise mentioned. The traffic is generated randomly according to a Poisson process. We analyze the performance of the protocols in two different scenarios: i) a static network, where each node randomly generates data for any other node, and ii) a mobile

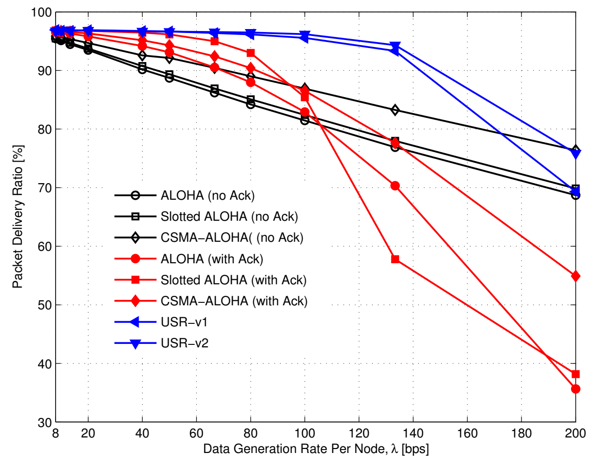

Figure 2. Packet delivery ratio as a function of traffic for all protocols, 5 nodes.

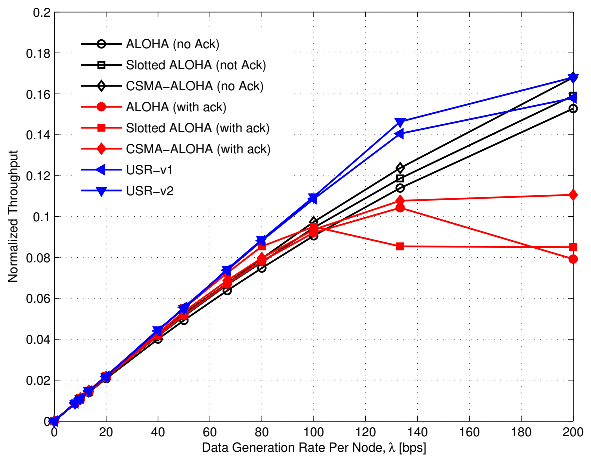

Figure 3. Normalized throughput as a function of traffic for all protocols, 5 nodes.

network, where 5 nodes are allowed to move freely around the area according to a Gauss-Markov mobility model [11] (the depth of each node is fixed throughout a simulation run, but is different for different nodes). In this scenario, we have all nodes randomly choose another node to be their fixed destination throughout the simulation run (the choice is randomized across different runs).

In order to stress the protocols under comparison, all nodes are put within the coverage area of each other, where the coverage range is defined as the distance at which the average MAC-level packet delivery rate drops below a given threshold, fixed to 0.9 in the present evaluation. As the number of nodes increases, collisions become more likely, and will allow to identify which protocol strikes the best balance between the average throughput experienced by the average transmitter-receiver pair, the increased delivery ratio allowed by ARQ techniques, and the increased probability that transmissions and retransmissions collide with one another.

In the simulations, we compare our USR protocol against plain ALOHA, slotted ALOHA, and CSMA-ALOHA, and for all three competitors we consider both a version with ACKs and one without ACKs. S&W ARQ is assumed for the ACK versions. We analyze the performance of the protocols in terms of Packet Delivery Ratio (PDR) (defined as the ratio between the number of packets correctly received by their intended destination and the overall number of packets sent), and normalized throughput (defined as the number of packets delivered in the network per packet transmission time).

B. Static scenario

Figs. 2 and 3 show the average Packet Delivery Ratio (PDR) and the normalized throughput in a static network of 5 nodes. The first observation inferred from the pictures is that our USR protocols consistently perform comparable to or better than other ACK-based protocols. In particular, we recall that USR-v1 is more "selfish," and has every transmitter continue sending packets to the same receiver insofar as there are any in the queue; on the contrary USR-v2 has every transmitter send two packets, and then perform channel access again, before other packets can be sent. (Recall that in this case no initial S&W phase is carried out for RTT assessment, as the estimate from the previous transmission is assumed to hold, and variations to be accommodated by the guard times of the protocol.) The stronger persistence of USR-v1, while allowing a negligibly higher throughput at low traffic, shows worse performance at high traffic, when more users access the channel (all in a random fashion) and therefore the probability of experiencing collisions is higher. From this first evaluation we can therefore conclude that USR-v2 is better than -v1, which is also confirmed by the following observations. For the same reasons, we argue that USR-v2 would also perform well against back-to-back packet transmissions as those proposed in [1], [2], especially when the latter are employed on top of the ALOHA scheme.

Figures 4 and 5 detail the same metrics for a network of 10 nodes. Again, when λ is small, ACK based protocols perform better than non-ACK based protocols, since they have the capability of recovering errors by retransmission. However, retransmissions may induce collisions in the network at higher traffic, which worsens the performance of ACK-based protocols. In any event, USR-v1 and -v2 are still the best among all ACK-based protocols before congestion begins to hamper successful transmissions. Among the non-ACK based protocols, ALOHA and slotted ALOHA perform quite similarly, a result in line with those in [12]. We also note that, since CSMA-ALOHA senses the channel before transmitting a packet, it is less subject to collisions: in fact, the ratio between the maximum propagation delay and the packet transmission time in our scenario is still bearable, and makes CSMA work sufficiently well. The above statements are also true for the ACK-based versions of ALOHA, slotted ALOHA and CSMA-ALOHA.

In the following figures we present the throughput variation

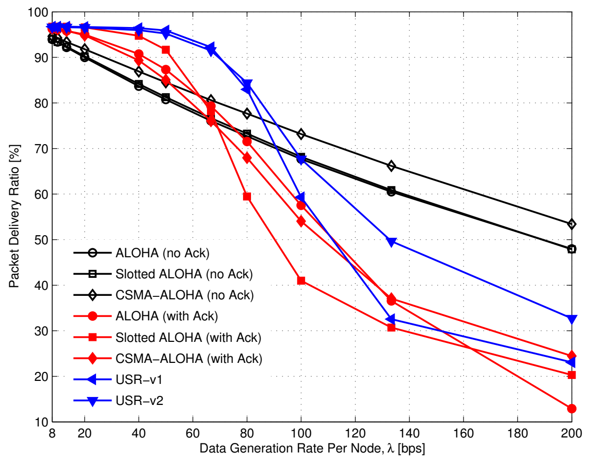

Figure 4. Packet delivery ratio as a function of traffic for all protocols, 10 nodes.

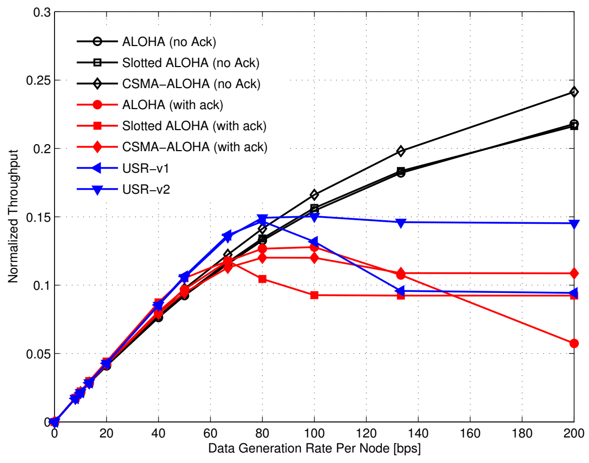

Figure 5. Normalized throughput as a function of traffic for all protocols, 10 nodes.

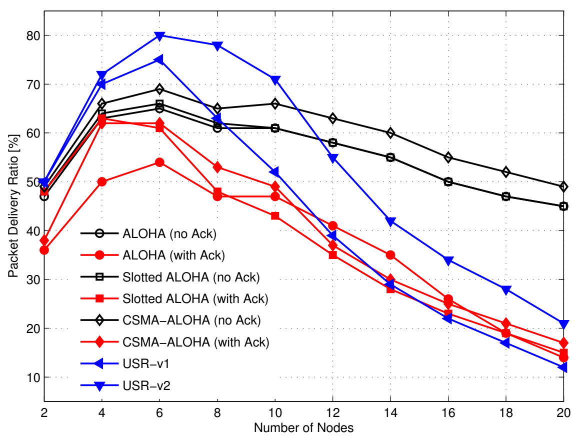

Figure 6. Packet delivery ratio as a function of the number of nodes in the network for all protocols, $\lambda = 100\,\text{bps}$ per node.

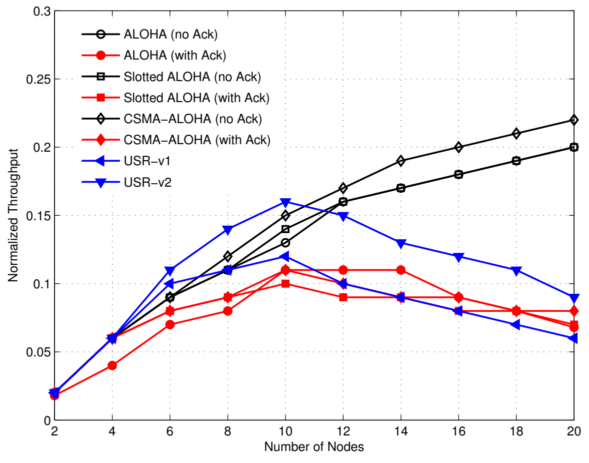

Figure 7. Normalized throughput as a function of the number of nodes in the network for all protocols, $\lambda = 100\,\text{bps}$ per node.

as a function of the number of nodes in the network for fixed traffic generation rate $\lambda = 100\,\text{bps}$ per node. In particular, Fig. 6 details the packet delivery ratio, whereas Fig. 7 shows the normalized throughput. We start from Fig. 6 to observe that when the network is very sparse, e.g., with only 2 nodes, there are only two cases where an error can occur, i.e., i) when the signal-to-noise ratio is too low to support the transmission, and ii) when a node is reached by a signal during its own transmission, and is thus deaf to the incoming packet. Since the traffic generation rate has been set to a low value for this evaluation, the deafness event is unlikely to take place, whereas it can happen that some realizations of the node placements turn into a low-quality channel. On average, this results in a packet delivery ratio of 0.5 for a network of 2 nodes, which increases to a maximum between 6 and 8 nodes, and then begins to decrease again due to the higher amount of traffic injected in the network, and the subsequently increased chance of collisions.

While the general behavior described above is still valid for all protocols, we observe that again USR-v2 tops all ACK-based protocols. Non-ACK-based versions experience a lower packet delivery ratio for a low number of nodes, but then outperform ACK-based ones, as the absence of ACK messages eliminates one source of vulnerability (we recall that a correctly received packet whose corresponding ACK will not reach the sender will be assumed to have been lost and therefore retransmitted). Similar observations hold for throughput (Fig. 7), which is better for non-ACK protocols if the number of nodes is higher than 10. In addition, throughput keeps increasing, which suggests that the higher number of nodes, and the corresponding more intense traffic, still outweighs the lower delivery ratio.

The results in Fig. 6, as well as the comparison between Figs. 2 and 4, suggest that the performance of the USR

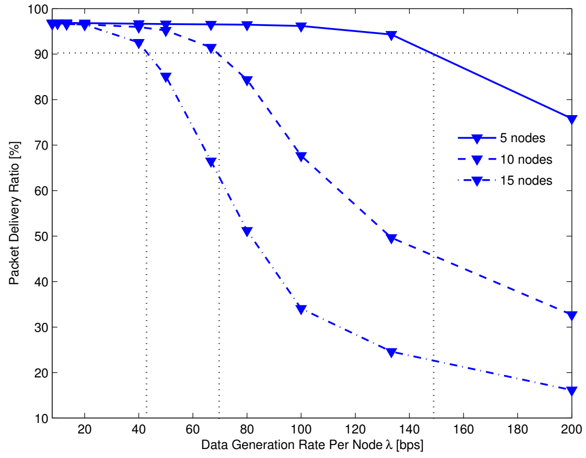

Figure 8. Packet delivery ratio of USR-v2 as a function of traffic for a different number of nodes in the network. The dotted lines show the data generation rate at which the ratio falls below 90%, and highlight that this rate is inversely proportional to the number of nodes.

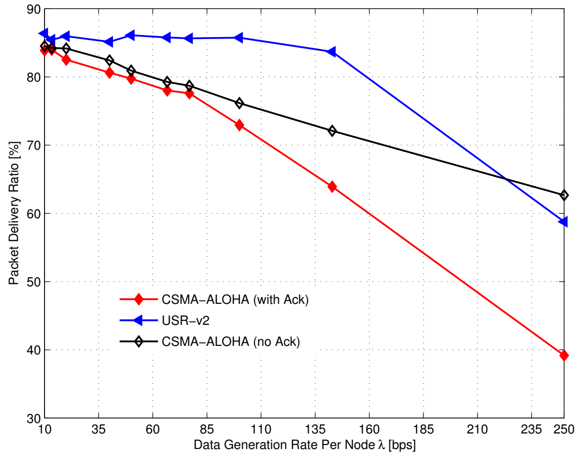

Figure 9. Packet delivery ratio as a function of traffic for all protocols in a mobile scenario, 5 nodes.

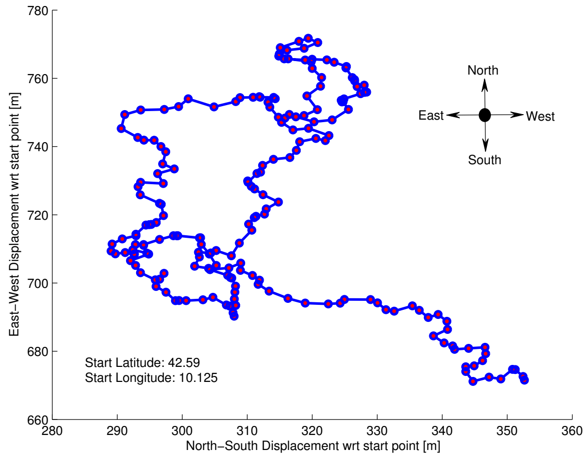

Figure 10. Sample Gauss-Markov mobility pattern with correlation parameter $\alpha = 0.5$. The red-filled blue circles correspond to the locations where the mobile randomly picks a new velocity vector (α-correlated with the previous one).

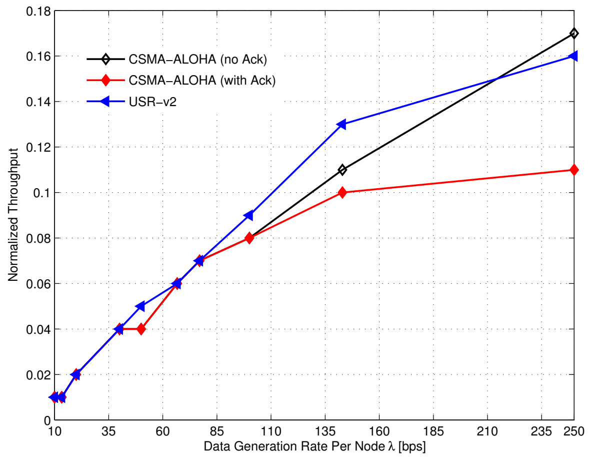

Figure 11. Throughput as a function of traffic for all protocols in a mobile scenario, 5 nodes.

protocols is almost inversely proportional to the overall amount of traffic generated in the network (i.e., to the generation rate times the number of nodes). This relationship offers a handy rule-of-thumb for designing the network. For example, consider Fig. 8, where we focus on the USR-v2 protocol and plot its performance as a function of traffic for a network of 5, 10 and 15 nodes. Assume that we want to constrain the packet delivery ratio to be above a typical threshold of 90%: then if we want to make the network twice as dense, the average generation rate per node which will still satisfy the constraint will be half as large.

C. Mobile scenario

We conclude our study by considering a mobile network with 5 nodes. Each node moves through the network area at fixed depth, but varies its latitude and longitude according to a Gauss-Markov mobility model of fixed correlation parameter $\alpha = 0.5$. We recall that according to this model, each node periodically chooses a new random velocity vector, which will be then maintained for a given time (also random), after which a new vector is generated. The correlation parameter α describes how similar the new vector is to the previous one: $\alpha = 0$ means a fully uncorrelated selection, whereas $\alpha = 1$ results in linear movement. The nodes are bounced off the border of the network area in case they happen to cross it. This mobility model results in such trajectories as the one depicted in Fig. 10, where the blue circles highlight at which locations the node randomly generates a new velocity vector. We assume that 10 mobiles are cruising in the same network area as above. The depth of each mobile is initially set to a random value and held throughout every simulation run.

Figs. 9 and 11 show a comparison of packet delivery ratio and throughput in a mobile network of 10 nodes. We limit this evaluation to the protocols that performed better in the previous study for static networks, i.e., USR-v2 and CSMA-ALOHA, in both its no-ACK version (which ranked first for throughput at high traffic) and its ACK version, which is a direct competitor for USR-v2. We observe that in CSMA-ALOHA, the usage of the ACK packet tends to cause a further source of interference and therefore a net decrease of throughput. On the contrary, USR-v2 allows packet transmissions to be sped up thanks to its improved packet and ACK exchange scheme; paired up with its mild channel access persistence, this fact allows better performance and a consistently higher delivery ratio. We note that even USR-v2 never reaches 100% delivery in Fig. 9, because the number of retransmissions allowed in case of errors is limited to 5. However, the improvement over both CSMA-ALOHA versions is still notable, as only the no-ACK version of the latter outperforms USR-v2 at high traffic, i.e., when the number of packets and corresponding ACKs in the network begins to cause significant interference, and the traffic generation rate is high enough to make deafness phenomena likely.

Similar indications are provided by the normalized throughput in Fig. 11, where we observe that all protocols initially show the same performance; as traffic increases, however, the curves of CSMA-ALOHA with ACKs and USR-v2 fork as the latter offers 20% more throughput until traffic is too high and the ranking of the two protocols switches.

IV. Conclusions

In this paper, we proposed a mechanism for improving the performance of MAC protocols with ARQ in underwater acoustic networks. We presented a simple scheme which estimates the round-trip time, and uses this estimate to arrange transmissions in a time division duplexing fashion. We compared a persistent version of this scheme (continuously transmitting packets until there are no more for the same destination) and a non-persistent version (performing carrier sensing once every two transmitted packets). We showed that the second (USR-v2) achieves better delivery ratio and throughput performance, and consistently outperforms all other ACK-based protocols.

We studied the performance of the protocol in mobile networks (where the initial estimation of the round-trip time gets progressively outdated due to node movement) and showed that a worst-case design of the way the protocol copes with mobility still guarantees satisfactory performance.

Acknowledgment

This work has been supported in part by the Italian Institute of Technology within the Project SEED framework (NAUTILUS project), and by Whitehead Alenia Sistemi Subacquei, Livorno, Italy, and the European Defence Agency, under the RACUN project.

References

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

[2] 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. of IEEE VTC Spring*, Singapore, May 2008.

[3] Y. Xiao, Ed., *Underwater Acoustic Sensor Networks*. Auerbach publications, 2008, ch. MAC Protocol Design for Underwater Networks: Challenges and New Directions, by V. Rodoplu and A. A. Gohari.

[4] M. Gao, W.-S. Soh, and M. Tao, “A Transmission Scheme for Continuous ARQ Protocols over Underwater Acoustic Channels,” in *Proc. of IEEE ICC*, Dresden, Germany, Jun. 2009.

[5] L. Badia, P. Casari, M. Levorato, and M. Zorzi, “Analysis of an automatic repeat request scheme addressing long delay channels,” in *Proc. of WAINA*, Bradford, UK, May 2009.

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

[7] ——, “A performance comparison of MAC protocols for underwater networks using a realistic channel simulator,” in *Proc. of MTS/IEEE Oceans*, Biloxi, MS, Oct. 2009, to appear.

[8] L. G. Roberts, “ALOHA packet system with and without slots and capture,” *ACM SigComm Computer Communication Review*, vol. 5, no. 2, pp. 28–42, 1975.

[9] N. Baldo, F. Maguolo, M. Miozzo, M. Rossi, and M. Zorzi, “NS2-MIRACLE: a modular framework for multi-technology and cross-layer support in network simulator 2,” in *Proc. of ACM NSTools*, Nantes, France, Oct. 2007.

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

[11] B. Liang and Z. J. Haas, “Predictive distance-based mobility management for PCS networks,” in *Proc. of IEEE INFOCOM*, New York, NY, Mar. 1999, pp. 1377–1384.

[12] A. Syed, W. Ye, B. Krishnamachari, and J. Heidemann, “Understanding spatio-temporal uncertainty in medium access with ALOHA protocols,” in *Proc. of ACM WUWNet*, Montréal, Canada, Sep. 2007.