Analysis of an automatic repeat request scheme addressing long delay channels

Leonardo Badia*, Paolo Casari†, Marco Levorato†, and Michele Zorzi†

∗ IMT Lucca Institute for Advanced Studies, piazza S. Ponziano 6, 55100 Lucca, Italy.

† Dept. of Information Engineering, University of Padova, via Gradenigo 6/B, 35131 Padova, Italy.

e-mail: {badia,casarip,levorato,zorzi}@dei.unipd.it.

Abstract

This paper proposes a variation to the classic implementation of Automatic Repeat reQuest (ARQ) which is particularly suitable for the long delay channels, as can be found, for example, in the underwater environment. The proposed technique, called Selective Repeat with a Second Replica ARQ, (SR)^2 ARQ, follows the same rationale as Selective Repeat ARQ, but, upon NACK reception, schedules two retransmissions, one taking place immediately, and the other put in a special queue to be released after further retransmissions, but before new packet transmissions. We propose an exact analysis of this technique, proving its ability of trading throughput for shorter delivery delay; thus, it is suitable for scenarios where the required data rate is not high, but a timely data delivery is very important, as is the case, for example, for underwater monitoring applications.

1 Introduction

Automatic Repeat Request (ARQ) is a fundamental error control technique widely applied in wired as well as wireless networks, which provides error recovery based on feedback messages and retransmissions. The performance of ARQ is heavily impacted by propagation delay, since the transmitter has to wait for feedback from the receiver before deciding whether or not to retransmit a packet; thus, the data delivery may be strongly delayed over channels featuring long round-trip times. ARQ realizations differ mostly in the way the interaction between the sender and the receiver is implemented. Classic ARQ schemes are Stop-and-Wait (SW), Go-Back-N (GBN), and Selective Repeat (SR), sorted by increasing throughput efficiency, as well as higher complexity cost [4].

Differently from other work [12], which considers simpler but less performing mechanisms such as SW ARQ, our analysis addresses SR ARQ, designing a variation of this scheme that behaves in a more delay-friendly manner by modulating the way packets are retransmitted in case of errors. More specifically, we enhance SR ARQ by still performing selective retransmission while at the same time introducing an additional queue, where a further duplicate of every not acknowledged packet is stored. This queue works as a fast track for packets with pending acknowledgement, that are injected on the channel more often than in a classic scheme. The rationale behind our proposal, which we call Selective Repeat with a Second Replica ARQ, or (SR)^2 ARQ for short, is that many applications require in-order packet delivery, i.e., a packet can be used at the application layer only after all preceding packets in the flow have been received correctly. Giving higher priority to retransmissions avoids continuously injecting new packets into the channel, which would have anyway to wait before being released. Our scheme, instead, trades off throughput for delivery delay, by transmitting further copies of the erroneous packets, which reduces throughput but increases the probability that a packet is correctly received before a substantial amount of new traffic is generated, thus reducing the packet delay. This is particularly useful in long delay channels, such as those found in underwater acoustic communications.

The rest of this paper is organized as follows. In Section 2 we discuss some related work, and in particular we review some important characteristics of the underwater medium and various ARQ schemes proposed in the literature. In Section 3 we describe the proposed (SR)^2 ARQ technique in detail. In Section 4 we study it through a Markov analysis of the entire transmission system and Section 5 presents numerical results. Finally, we conclude in Section 6.

2 The underwater channel and the delay performance of ARQ techniques

The underwater acoustic channel is a specific example of long transmission delay medium that has received consistently high attention during the last few years. Underwater communications can be performed by using optical and electromagnetic waves as well as acoustic pressure waves. However, electromagnetic waves tend to undergo excessive attenuation and are not suitable for oceanic reaches, that can be of the order of 10 kilometers. Optical communications support higher bit rates, yet their range is also very limited due to attenuation and optical scattering. Furthermore, they require keeping the transmitter and receiver aligned, which can be difficult due to ocean currents or node mobility. Acoustic communications, instead, are suitable for distances up to hundreds of kilometers, making them the preferred choice for both telemetry and data transfer. Nevertheless, the acoustic channel imposes significant hurdles to communications, especially in terms of propagation delay and attenuation: pressure waves propagation is 5 orders of magnitude slower than that of radio waves in the air, and incurs greater attenuation as well due both to the geometry of propagation (as happens for radio waves) and to the dispersion of acoustic pressure into heat.

Despite these challenges, research on acoustic communications and networking has gained momentum, and seminal works have demonstrated the possibility to signal at relatively high speed over underwater links [8, 9]. Experiments involving permanent installation of underwater nodes have also been conducted, such as the SeaWeb project [5], demonstrating the operability of underwater networks by means of simple communication protocols.

Being an established error recovery technique, ARQ over underwater links has been explored already by related work on submarine networking. Among them [12] considers three versions of SW, derives their efficiency, and finds their optimal packet size. In [6], the authors describe the ARQ mechanism employed in SeaWeb [5]: after gaining access to the channel, the transmitter employs ARQ and keeps the channel reserved until the packet has been successfully received. A multihop ARQ scheme for underwater networks is described in [14], where an explicit acknowledgment version is compared with an implicit one, whereby overhearing a packet forwarded to the next hop automatically informs the preceding hop of the correct reception. A hybrid version is also proposed that switches automatically between explicit and implicit acknowledgment, in order to improve energy consumption or latency performance. An improved multihop scheme is also presented in [7], where the authors suggest to segment packets so that the transmission time is shorter than the propagation delay. This enables each node to interlace the transmission of its own packets so that collisions with other messages are avoided. A more complex hybrid ARQ scheme employing tornado codes for recovering transmission errors over a multihop network through ARQ and forward error correction is presented in [13].

Note that [7,13,14] focus on a network scenario, whereas our approach considers a point-to-point transmission. However, techniques like the multihop packet superposition strategy pointed out in [7], can be implemented independently to support multihop, using our ARQ strategy for error control. Finally, we remark that our proposal can be viewed as a form of Hybrid ARQ (HARQ), due to the incremental redundancy of retransmissions, though achieved with a relatively simple repetition code. This makes (SR)^2 ARQ easy to implement but susceptible to improvements with more sophisticated HARQ schemes. Such an extension, which can be made following the approach of [2], is an interesting point left open for further research.

3 System description

We consider a message exchange between a transmitter and a receiver over a noisy discrete-time channel with delay significantly longer than the packet transmission time, which equals one time slot. Packets have a unique integer identifier (id) and are sent in increasing order. For simplicity, we consider independent identically distributed (iid) channel errors which occur with a fixed probability ε. We also assume a fixed round-trip time, which is not restrictive to take as integer, denoted as $m > 1$. This is the time elapsed between the transmission of a packet and the reception at the transmitter's side of the corresponding feedback message sent back by the receiver, denoting either acknowledgement (ACK) or not acknowledgement (NACK).

We assume that the transmitter is in the Heavy Traffic condition, meaning that the packet backlog is always full, and that ACK/NACK messages are error-free and non colliding with the forward transmission (e.g., they have stringent timeout and are exchanged on a different frequency). These assumptions are quite common in the literature [10], so we adopt them for the sake of analytical simplicity. Their relaxation, which would be required for the application on half-duplex underwater channels, can be an interesting future research subject. Some preliminary extension of the analysis in this sense can be found in [3, 11].

In standard SR ARQ, under these same conditions, the transmitter continuously transmits packets, which are either fresh ones (upon reception of an ACK message) or selective retransmissions (triggered by a NACK). Thus, the transmitter needs to store transmitted packets which are not yet acknowledged in a buffer in order to be able to selectively retransmit them in case a NACK is received. Similarly, also the receiver requires a buffer, called the re-sequencing buffer, where to store packets which have been correctly received but can not be released to the upper layers. We assume the *in-order release* condition, i.e., a packet can be processed by upper layers only when it is correctly received and also all packets with lower id have been correctly received. The time interval elapsed between the first transmission of a packet and its subsequent release to the upper layers is called *delivery delay*, which is the delay term on which we focus in this paper. Even though other delay terms can be considered as well (see [1, 3] for related detailed investigations), we believe that characterizing the delivery delay under the Heavy Traffic assumptions is sufficiently representative as it adequately characterizes many scenarios of interest taking place in underwater networks. In particular, as argued in [11], this condition is appropriate when considering continuous data streaming, e.g., coming from surveillance sensors. Finally, observe that, as done by other related papers [3, 10], we compute the delivery delay at the receiver's side and we neglect the constant term corresponding to the propagation delay (approximately equal to $m/2$). E.g., if a packet is released already at its first transmission, the delivery delay is zero.

Our proposed (SR)^2 ARQ mechanism involves the addition of another buffer, where packets are processed on a first-come-first-serve basis, at the transmitter's side. This additional buffer has a maximum size of m packets and keeps a replica of every not acknowledged packet, for which we call it *replica queue*. A replica is just an exact duplicate of the packet, which is sent over the channel as a normal retransmission, but with a different timing, as explained next.

The (SR)^2 ARQ scheme proceeds as follows. At the beginning, the replica queue is empty, and remains such as long as the feedback consists of ACK messages, in which case new packets are transmitted continuously. Upon reception of a NACK, the transmission of fresh packets is paused, and a retransmission of the not acknowledged packet takes place, as per the standard SR ARQ. At the same time, a replica of the same packet is inserted in the replica queue. Thus, the packet which is revealed to be in error is duplicated, and two retransmission takes place. The former is called *immediate retransmission* and always happens in the slot after the reception of the NACK. The latter corresponds to the retransmission of the copy placed in the replica queue, which can be transmitted only after an ACK is received, as a NACK always triggers instead an immediate retransmission of the related packet. Though preempted by further immediate retransmissions, the extraction of the head of the replica queue and its transmission have priority over the transmission of new packets. This means that if the transmitter receives two NACKs related to packets with id 1 and 2, followed by two ACKs, it will first perform an immediate retransmission of 1 (while at the same time inserting a copy of 1 in the replica queue) then an immediate retransmission of 2, which has priority over the transmission of the replica of 1, finally followed by the transmissions of the replicas, first of 1, then of 2. Note that the mechanism does not simply duplicate retransmissions but enables their interleaving. Thus, even though this is unnoticed in the present paper, where the errors are iid, our proposal is also beneficial for correlated channels, where errors affect adjacent slots.

However, to prevent the double retransmission mechanism described above from causing an explosive behavior of the number of replicas present in the system, a further modification of the ARQ scheme is required. To this end, we impose that the immediate retransmission, if received incorrectly, causes the receiver to answer with a *quiet NACK* (qNACK). Such a packet does not trigger any retransmission but instead enables a different packet to be sent (which is a replica, if the queue is non empty, otherwise a fresh packet). Thus, the first erroneous transmission of a packet is followed by two retransmissions (the immediate retransmission and the replica), and if an ACK is received for either of them, the packet can be put in the re-sequencing packet (or resolved, if all packets with lower id are correct as well). Otherwise, i.e., if both retransmissions are not acknowledged, the feedback of the immediate retransmission is a qNACK, whereas the feedback of the replica is a real NACK, thus triggering another immediate retransmission and putting another copy in the replica queue. In this way, the stability of the system is preserved, having no more than two copies of the same packet scheduled for retransmission at the same time.

Also note that there is no need to explicitly differentiate quiet and real NACKs, that is, the packet structure can be kept unchanged, as both terminals can tell quiet NACK apart, as they are the second, the fourth, and in general every even-numbered NACK received for the same packet. Also note that, while a qNACK does not enable the resolution of any packet, it is however, for what concerns the evolution of the ARQ process, identical to an ACK. Even though this may appear counterintuitive, observe that indeed sending qNACK as a feedback for an immediate retransmission will not trigger any further retransmission; however, also a replica of the same packet is pending, which will determine whether the packet is correctly received or not. Even if the feedback of the immediate transmission is instead an ACK, the transmitter does not wait for it before transmitting the replica.[1] Thus, the feedback of an immediate retransmission is irrelevant to the transmitter, as regardless of its being either a qNACK or an ACK, the system evolves in the same manner.

Finally, another property which is worth noting is that the transmission of a fresh packet is possible only when the replica queue is empty, since, as long as there are packets in the replica queue, they have priority over fresh packets.

Anticipating the results which will be shown in the following, we can thus say that the (SR)^2 ARQ mechanism reduces the delivery delay because of two reasons. On the one hand, adaptively increasing the number of retransmissions improves the probability of correcting packets, which directly affects the delivery delay. On the other hand, it

[1]There is an exception, involving the situation where the replica queue is full, which will be discussed in more detail in the next section. However, as will be shown, there is no need to differentiate them for the transmitter, which is able to distinguish the two cases from other information.

also reduces the number of pending packets as, instead of transmitting fresh packets, replicas are transmitted first; that is, instead of potentially introducing even more erroneous packets, the ones which are not yet acknowledged are addressed first with more strength.

4 Analysis of the (SR)^2 ARQ scheme

The system described in the previous section can be analyzed by means of a proper Markov chain. The memory of the system required to track its evolution consists of two terms: the state of the replica queue, denoted in the following by $\mathbf{a}$, and the state of the last m transmitted packets, described by the m-sized binary vector $\mathbf{b} = (b_i)$, with $i = 1, \dots, m$. Note that the latter is also required when studying the SR ARQ, as shown in [10], where the same notation is used, whereas the former is added due to the introduction of the replica queue in the (SR)^2 ARQ scheme.

The binary values b_i can be either 0 or 1; following [10], 0 corresponds to an ACK and 1 to a NACK. However, the (SR)^2 ARQ also includes the qNACK feedback, but this, as said, is equivalent to an ACK. Thus, the qNACK is also represented by $b_i = 0$. We can represent the replica queue as a list of L binary elements, i.e., $\mathbf{a} = (a_i)$, with $i = 1, \dots, L$, where $L \leq m$. To describe the elements a_i we need instead the following preliminary definition of *fake replica*.

A replica in the queue is conventionally said to be a fake (as opposed to a *real* one) if the corresponding immediate transmission is acknowledged. Since this is sufficient to acknowledge the packet, the transmission of the replica is useless. Note that, looking at the system from an external standpoint, one is immediately able to tell whether the replica is fake or real already at the time of its insertion in the replica queue. In fact, its fake/real status depends on the outcome of the corresponding immediate retransmission which, as the name says, takes place in the same time slot when the replica is generated. As one already knows whether the immediate retransmission will succeed, similarly knows whether the replica is a fake as well. However, the transmitter is not informed about this until it receives the feedback packet. Thus, it may well happen that the fake replica is transmitted before the ACK arrives at the transmitter's side. The transmission of fake replicas is the main reason of the throughput degradation which affects (SR)^2 ARQ, which is also the price for the decreased delivery delay. Thus, the binary digit a_i, with $i = 1, \dots, L$ is 0 if the ith element of the queue is a fake replica, and 1 if it is a real one. Note that $\mathbf{a}$, being represented with at most m bits, can thus take $2^{m+1} - 1$ possible values.

There is one special case when the fake replica can be detected, which is when the feedback of the immediate retransmission arrives at the transmitter before than the replica is sent over the channel; if an ACK arrives, the transmitter knows that the replica is fake. Thus, in this case, a qNACK is not equivalent to an ACK. However, this special case happens only when the m previous transmissions were in error, since otherwise the replica would have been transmitted before the arrive of the feedback; thus the replica queue is full. Hence, in the case of full replica queue, i.e., when its length L is equal to m, a qNACK is distinguishable from an ACK by checking the value of a_1.

The procedure to analyze the Markov chain is outlined as follows, along the lines of [2] and [3]. First, the steady-state probability $\pi(\mathbf{a}, \mathbf{b})$ is derived for any state $(\mathbf{a}, \mathbf{b})$. This already enables the evaluation of the throughput as the sum of the $\pi(\mathbf{a}, \mathbf{b})$s for which the receiver acknowledges a packet, discarding those where a fake replica is sent. Then, we scale the steady-state probabilities under the condition that a new packet is transmitted for the first time. At this point, the arrival process is virtually turned off and the "depletion time" of the system is evaluated. In fact, subsequent arrivals do not affect the release of the packet of interest [10], which happens when all previous packets are acknowledged, i.e., when $\mathbf{b}$ is the all-zero vector and the replica queue $\mathbf{a}$ is empty or only contains fake replicas.

Denoting with $\mathcal{S}$ the set of all possible states, the steady-state probabilities can be derived by adding the condition $\sum_{(\mathbf{a},\mathbf{b}) \in \mathcal{S}} \pi(\mathbf{a}, \mathbf{b}) = 1$ to the balance equations, which in turn can be obtained by the transition probabilities of the system. Any state $(\mathbf{a}, \mathbf{b})$ has at most two transitions, corresponding to the events of channel error, with probability ε, and correct transmission, with probability $1 - \varepsilon$. Thus, state $(\mathbf{a}, \mathbf{b})$ has transitions to the pair of states as per the table below, where $\xrightarrow{p}$ denotes a transition with probability p.

1.	if $b_1 = 1$	$\xrightarrow{1-\varepsilon}$	$(\mathbf{a}^{\hookleftarrow 0}, \mathbf{b}^{\leftarrowtail 0})$
2.		$\xrightarrow{\varepsilon}$	$(\mathbf{a}^{\hookleftarrow 1}, \mathbf{b}^{\leftarrowtail 0})$
3.	elseif $L{=}m \& a_1{=}0$	$\xrightarrow{1-\varepsilon}$	$(\mathbf{a}^{\hookrightarrow\hookrightarrow}, \mathbf{b}^{\leftarrowtail 0})$
4.		$\xrightarrow{\varepsilon}$	$(\mathbf{a}^{\hookrightarrow\hookrightarrow}, \mathbf{b}^{\leftarrowtail a_2})$
5.	elseif $L > 0$	$\xrightarrow{1-\varepsilon}$	$(\mathbf{a}^{\hookrightarrow}, \mathbf{b}^{\leftarrowtail 0})$
6.		$\xrightarrow{\varepsilon}$	$(\mathbf{a}^{\hookrightarrow}, \mathbf{b}^{\leftarrowtail a_1})$
7.	else	$\xrightarrow{1-\varepsilon}$	$(\emptyset, \mathbf{b}^{\leftarrowtail 0})$
8.		$\xrightarrow{\varepsilon}$	$(\emptyset, \mathbf{b}^{\leftarrowtail 1})$

The symbols used in the table are explained as follows. Notation $\mathbf{a}^{\hookleftarrow x}$ means that binary digit x is inserted in the queue $\mathbf{a}$, whereas $\mathbf{a}^{\hookrightarrow}$ means that the first element of the queue is removed. The symbol $\mathbf{b}^{\leftarrowtail x}$ means instead that binary digit x is inserted into $\mathbf{b}$ with a cyclical shift of the vector, i.e., $\mathbf{b}^{\leftarrowtail x} = (b_2, b_3, \dots, b_m, x)$. The expressions 1–8 can be motivated by the following observations. First of all, note that a cyclical shift of the vector $\mathbf{b}$ is always present. This is a general property of the feedback vector in ARQ systems, which is investigated in more detail in [10]. Transitions 1 and 2 describe the case where a NACK (of the real kind, not a quiet one) arrives at the transmitter. Thus, an immediate retransmission is performed, which, depending on the error probabilities, generates either an ACK or a qNACK from the receiver. However, both cases are described by $b_m = 0$, which is why the destination state is always $\mathbf{b}^{\leftarrow 0}$. However, there is a differentiation in that the transmitter also inserts a replica in the queue $\mathbf{a}$, which is a fake in case the immediate transmission is correctly received. Transitions 5 and 6 describe instead the arrival of an ACK or a qNACK at the transmitter when the replica queue is non-empty. In such a case, the head packet of the replica queue is extracted and transmitted, and $\mathbf{b}$ is updated with a cyclical shift according to the feedback related to this transmission, which can be either ACK or NACK (the NACK is always real and not quiet as the packet is a replica). It is worth noting that 6 inserts a_1 in the cyclical shift to account for the case where the transmitted replica is a fake (i.e., $a_1 = 0$), as in this case the answer is ACK anyway, not due to the channel status, but to the previous reception of a correct immediate retransmission. Analogous reasonings explain 3 and 4, where the special case of full replica queue with a fake in head position is considered. Thus, the fake replica is discarded so that a_2 is transmitted instead and two packets are removed from $\mathbf{a}$. Finally, 7 and 8 describe instead the transmission of a new packet. In this case, the replica queue is empty and stays so also for the next slot.

After the $\pi(\mathbf{a}, \mathbf{b})$ have been derived, they can be used, as discussed previously, to evaluate the throughput and, after a proper re-scaling, the delivery delay. For the latter, we need to evaluate the steady-state probability conditioned to the event of transmitting a new packet. Recall that a fresh packet can be transmitted only if $\mathbf{a} = \emptyset$, as argued in the previous section. Thus, we calculate probabilities $\pi'(\mathbf{a}, \mathbf{b})$ as

$$\pi'(\mathbf{a}, \mathbf{b}) = \begin{cases} 0 & \text{if } \mathbf{a} \neq \emptyset \\ \dfrac{\pi(\emptyset, \mathbf{b})}{\sum\limits_{(\emptyset,\mathbf{x}) \in \mathcal{S}} \pi(\emptyset, \mathbf{x})} & \text{if } \mathbf{a} = \emptyset \end{cases} \qquad (1)$$

which is the probability that a packet of interest is transmitted for the first time when the system state is $(\mathbf{a}, \mathbf{b})$.

Finally, the transmission of new packets is "turned off" and the first passage time to any of the system states where neither $\mathbf{a}$ nor $\mathbf{b}$ contain any 1 is evaluated, which gives the delivery delay. The suspension of the new packet transmissions is due to the property, discussed, e.g., in [3], that they do not affect the delivery delay of the packet of interest. To implement this, we can simply replace condition 8 with the following 8′

$$8'. \qquad \xrightarrow{\varepsilon} \quad (\emptyset, \mathbf{b}^{\leftarrow 0})$$

and let the system evolve as per transitions 1–8′.

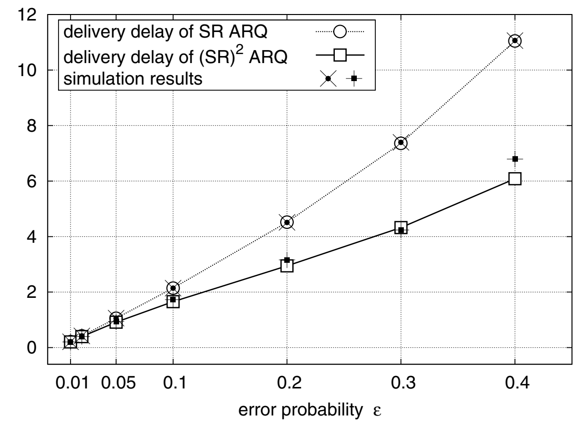

Figure 1. Delay comparison of SR ARQ and (SR)^2 ARQ for *m*=6 as a function of the error probability.

5 Numerical results

We implemented both (SR)^2 ARQ and classic SR ARQ (solved in [10]) operating over a link with $m = 6$ and varying error probability. This choice of m has been made only to simplify the computational complexity, but preliminary experiments done for larger round-trip times, which would be more interesting for underwater channels, confirm the results shown in this section. We also obtained simulation results to validate the analysis, where the average delivery delay was computed for an indefinitely long queue of packets, which were affected by errors with probability ε; in this case, they were retransmitted according to the specific ARQ scheme. We stress that the exact analysis is able to give an entire characterization from the statistical standpoint, instead of deriving average values a posteriori.

Fig. 1 shows the delivery delay as a function of ε, for both SR ARQ and (SR)^2 ARQ. Simulation results are plotted for comparison. As visible, the proposed (SR)^2 ARQ significantly improves the delay term. This gain is only marginal when the error probability is low, where however the delivery delay is low anyway, whereas it is significant (about 40% decrease) for high ε.

The (SR)^2 ARQ pays a price for this improvement, as it also achieves lower throughput, due to the transmission of fake replicas. Fig. 2 compares the throughput achieved by SR ARQ and (SR)^2 ARQ, again confirmed by simulation results. The throughput of plain SR ARQ is simply $1 - \varepsilon$, where the one of (SR)^2 ARQ is about 13% lower. Note also that the decrease tends to be relatively lower as ε increases. This is justified by the observation that when the error probability is very high, most of the replicas are real ones and not fakes. Finally, Fig. 3 reports the probability and the complementary cumulative distribution function (ccdf) of the delivery delay for the case $\varepsilon = 0.3$, again comparing

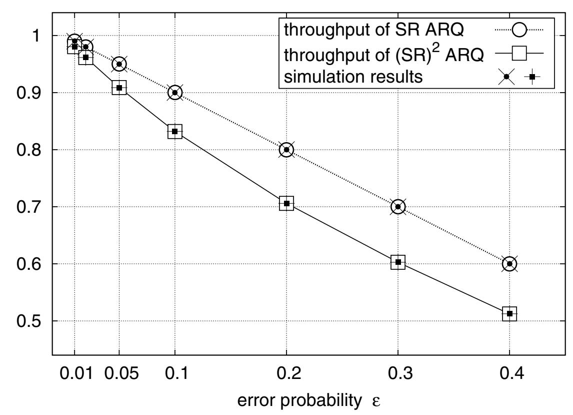

Figure 2. Throughput comparison of SR ARQ and (SR)^2 ARQ for *m*=6 as a function of the error probability.

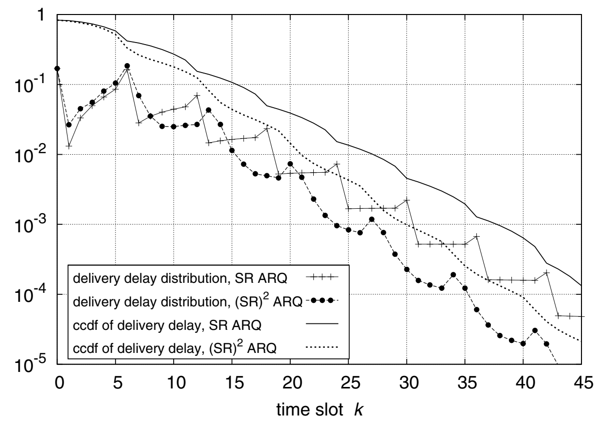

Figure 3. Probability and complementary cumulative distribution of the delay for *m*=6 and ε=0.3.

SR ARQ and (SR)^2 ARQ. It is shown that the (SR)^2 ARQ has in particular a significantly higher probability of delivery already in the first slots, which confirms the discussion made at the end of Section 3 about its ability of improving the correction capability for erroneous pending packets, while at the same time decreasing the number of erroneous packets unnecessarily sent over the channel.

To sum up, these sample results show the ability of our proposed technique to improve the delivery delay performance, at the price of a throughput decrease which is, however, limited. This makes (SR)^2 ARQ suitable when the channel has high error probabilities and long round-trip time, for scenarios without high requirements in term of throughput but where the delivery delay must be kept low, which is the typical case of the transmission of signalling flows coming from underwater monitoring stations.

6 Conclusions and Future Work

Based on a Markov approach, we analyzed an ARQ scheme which improves the delivery delay at the price of a decreased throughput.

Simple possible extensions to the present paper include the analysis of correlated channel cases, and/or the extension to HARQ techniques. As a further development, we are currently investigating half-duplex channels, and the evaluation of the queueing delay depending of the arrival process.

References

[1] L. Badia. On the impact of correlated arrivals and errors on ARQ delay terms. *IEEE Trans. Commun.*, 57(2):334–338, Feb. 2009.

[2] L. Badia, M. Levorato, and M. Zorzi. Markov analysis of selective repeat type II hybrid ARQ using block codes. *IEEE Trans. Commun.*, 56(9):1434–1441, Sept. 2008.

[3] L. Badia, M. Rossi, and M. Zorzi. SR ARQ packet delay statistics on Markov channels in the presence of variable arrival rate. *IEEE Trans. Wireless Commun.*, 5(7):1639–1644, July 2006.

[4] H. O. Burton and D. Sullivan. Errors and error control. *Proceedings of the IEEE*, 60(11):1293–1301, Nov. 1972.

[5] J. Rice *et al.* Evolution of Seaweb underwater acoustic networking. In *Proc. of MTS/IEEE Oceans*, pages 2007–2017, Providence, RI, Sept. 2000.

[6] J. Rice *et al.* Performance of undersea acoustic networking using RTS/CTS handshaking and ARQ retransmission. In *Proc. of MTS/IEEE Oceans*, pages 2083–2086, Honolulu, HI, Nov. 2001.

[7] J.-W. Lee *et al.* An improved ARQ scheme in underwater acoustic sensor networks. In *Proc. MTS/IEEE Oceans*, pages 1–5, Kobe, Japan, Apr. 2008.

[8] M. Stojanovic. Recent advances in high-speed underwater acoustic communications. *IEEE J. Ocean. Eng.*, 21:125–136, Apr. 1996.

[9] J. G. Proakis, E. M. Sozer, J. A. Rice, and M. Stojanovic. Shallow water acoustic networks. *IEEE Commun. Mag.*, 39:114–119, Nov. 2001.

[10] M. Rossi, L. Badia, and M. Zorzi. Exact statistics of ARQ packet delivery delay over Markov channels with finite round-trip delay. *IEEE Trans. Wireless Commun.*, 4(4):1858–1868, July 2005.

[11] M. Rossi, L. Badia, and M. Zorzi. SR ARQ delay statistics on N-state Markov channels with non-instantaneous feedback. *IEEE Trans. Wireless Commun.*, 5(6):1526–1536, June 2006.

[12] M. Stojanovic. Optimization of a data link protocol for an underwater acoustic channel. In *Proc. IEEE/OES Oceans*, pages 68–73, Brest, France, June 2005.

[13] M. Stojanovic. An FEC-based reliable data transport protocol for underwater sensor networks. In *Proc. IEEE ICCCN*, pages 747–753, Honolulu, HI, Aug. 2007.

[14] H.-P. Tan, W. K. G. Seah, and L. Doyle. A multi-hop arq protocol for underwater acoustic networks. In *Proc. IEEE/OES Oceans*, pages 1–6, Aberdeen, Scotland, June 2007.