Exact statistics of ARQ packet delivery delay over Markov channels with finite round-trip delay

Michele Rossi, Leonardo Badia, Michele Zorzi
Dipartimento di Ingegneria, Università di Ferrara — via Saragat 1, 44100 Ferrara, Italy
Email: {mrossi,lbadia,mzorzi}@ing.unife.it

***Abstract*— In this paper the packet delay statistics of a fully reliable Selective-Repeat ARQ scheme is investigated. It is assumed that the sender continuously transmits packets whose error process is characterized by means of a two-state Discrete Time Markov Channel. At the receiver these packets are checked for errors and ACK/NACK messages (assumed error-free) are sent back to the sender accordingly. The feedback message is known at the transmitter m channel slots (round-trip delay) after the packet transmission started. An appropriate Markov model has been developed in order to find the exact statistics of the delays experienced by ARQ packets after their first transmission.**

I. Introduction

With the increasing development of multimedia applications in modern communication systems, effective error control techniques are required. Many applications are highly sensitive to channel impairments. To have good performance in terms of data reliability, latency, and efficient bandwidth usage, error control techniques must be carefully designed. Thus, a deep understanding of them is necessary. Usual protocol stacks perform error control at multiple levels: e.g., at the physical layer with error correction codes, at the data-link layer with ARQ techniques and at the transport layer with TCP.

In recent years, the study of ARQ error control techniques in wireless systems has enjoyed a lower popularity than, e.g., error correction coding strategies. This is mainly due to the type of application envisioned in these systems, i.e., voice and circuit-switched data, where strict delay guarantees are required. With the extension of packet data and Internet services over wireless links, the increased delay tolerance of many applications and protocols leads to a paradigm shift, where error recovery by retransmission may be more efficient than protecting all data a priori by means of costly FEC.

ARQ solutions directly interact with higher levels, by determining both delay/jitter performance and error probability of higher level packets. Their correct configuration is key in achieving the needed higher level QoS; hence, an accurate study of the delivery delay process at the ARQ level is crucial to understand the interaction between the higher level performance and the link layer retransmission process.

In ARQ, the transmitter sends packets (PDUs) consisting of payload and error detection codes. At the receiver side, based on the outcome of the error detection procedure, acknowledgment messages are sent back to the transmitter (ACK or NACK, according to the result of error detection). The sender performs packet retransmissions based on such acknowledgments. In general, ARQ protocols are variants of the following basic schemes: stop-and-wait (SW), go-back-N (GBN) and selective repeat (SR). The SR scheme is the most efficient: here packets are transmitted continuously, but only negatively acknowledged packets are retransmitted, i.e., retransmissions are selectively triggered by NACK messages.

This work has been partially supported by ERICSSON research.

When the round-trip delay goes to zero all the presented schemes become identical. In the literature [1][2], this situation is referred as *ideal SR* ARQ.

The overall PDU delay with ARQ protocol can be subdivided in three contributions. These quantities will be referred to as *queuing delay*, *transmission delay* and *re-sequencing delay*, as usually done in the literature [3]. The first is the time spent in the source buffer queue, i.e., the time between the PDU release by higher levels and the instant of its first transmission over the channel. This term depends on both the channel behavior and the PDU arrival process. The second contribution is the time between the first transmission and the correct reception of the PDU, which only depends on the channel behavior. The last delay is the time spent in the receiver re-sequencing buffer. In fact, even though the sender transmits packets in order, they can arrive out of sequence, due to random errors and consequent retransmissions. Hence, a correctly received PDU must wait in the receiver re-sequencing buffer until all the PDUs with lower identifier have been correctly received. This last term is the most complicated because it depends on errors experienced by other PDUs. In the following, the term *resolution* (e.g., of a packet) will mean correct transmission, whereas *delivery* (or equivalently, *release*) refers to the joint resolution of the considered packet as well as of all packets with a smaller id. In this paper we investigate, with an exact analysis, the statistics of the *delivery delay*, defined as the time between the first transmission of the packet and its successful release from the re-sequencing buffer, in other words, the sum of the second and third terms.

Several studies have been performed on the delay performance of the SR protocol over a wireless channel [1][3][4][5][6][7]. In [4], queuing theory is used by Konheim to evaluate the different kinds of delay affecting the PDU transmission, in the situation of a finite round-trip time. In this work a static channel is considered, and the developed model allows only to estimate average values of the delays. In [5] an alternative approach for the same problem considering a Bernoulli arrival process is proposed. Rosberg and Shacham in [6] and Rosberg and Sidi in [7] analyzed in detail re-sequencing delay and re-sequencing buffer occupancy at the transmitter and at the transmitter and the receiver jointly, respectively, but again in the case of static channel. The time varying channel was investigated for the first time by Fantacci in [1], by means of queuing theory. However, here the re-sequencing delay is not studied and only average values for the other delays can be quantified, with a lower bound on them with respect to the situation of a finite round-trip delay. Finally, in [3] Kim and Krunz accounted for a time varying channel and a finite round-trip delay, by developing the analysis for all the ARQ delay contributions. However, several approximations are introduced, as for example the hypothesis of *ideal SR* is used for the queuing delay evaluation, so that

only approximate mean values can be quantified.

In this paper we study the delay performance of a fully reliable SR ARQ scheme, considering both time varying channel and finite round-trip time, and also the effect of bursty channel errors is taken into account. Some assumptions are made to simplify the formal description, however they do not affect the generality of the results, since they can be relaxed if needed. Previous studies are greatly extended, since an exact analysis is presented for the *delivery delay* statistics: this is an instrument that allows, for example, to write closed mathematical expressions for quantities related to the ARQ delay, and to give not just approximate mean values, that in certain cases could be misleading, but the complete statistic description, so that the performance can be exactly evaluated.

The remaining part of the paper is organized as follows: in Section II the ARQ policy and the channel model are described, in Section III the exact analysis of the *delivery delay* is reported. In Section IV results are reported and finally, in Section V, some conclusions are given.

II. Model for ARQ Processes

We consider a pair of nodes, that communicate data packets through a noisy wireless link and use a fully reliable Link Layer protocol (unlimited retransmission attempts) to counteract channel impairments. Data packets (ARQ PDUs) and ACKs/NACKs flow in forward and backward direction, respectively: it is not restrictive to consider error-free ACKs and NACKs. Moreover, we assume that both transmitter and receiver have unlimited buffer size and they adopt the following Selective-Repeat ARQ protocol (a generalization of the protocol described in [8]) at the Link Layer.

The sender continuously transmits new PDUs from its buffer in increasing numerical order as long as ACKs are received. The time is slotted and the slot time corresponds to a single PDU transmission. After each PDU reception, the receiver checks for packet errors and replies with an ACK/NACK accordingly. If m is the round-trip delay the sender receives the ACK/NACK message for each packet after the transmission of $m-1$ subsequent PDUs, new or retransmitted. In the literature [9], m is also commonly referred as the *ARQ window size*. In case of NACK, the corresponding PDU is retransmitted m slots after the previous transmission, else a new PDU is sent. The wireless channel is characterized by means of a two-state Discrete Time Markov Chain (DTMC), and let denote the states as 0 and 1. We can define the related transition probability matrix $\mathbf{P}$ and the corresponding i-step transition probability matrix $\mathbf{P}(i)$, as follows:

$$\mathbf{P} = \begin{pmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix}, \quad \mathbf{P}(i) = \mathbf{P}^i = \begin{pmatrix} p_{00}(i) & p_{01}(i) \\ p_{10}(i) & p_{11}(i) \end{pmatrix}$$

The steady-state channel error probability is $\varepsilon = \frac{p_{01}}{p_{10}+p_{01}}$, while the average error burst length is given by $b = 1/p_{10}$. We model the errors in the channel with the hypothesis that transmissions during state 1 are always erroneous, whereas state 0 is error free. This is a reasonable assumption in many cases [2] and the model we propose can be extended to account for a higher order Markov Chain. A more complicated channel model only makes the analytical study cumbersome.

The traffic model mainly affects the *queuing delay*, that is out of the scope of our analysis, and the delivery delay only slightly depends on it. So, it is reasonable to consider a simple model for the arrival process, although our analysis can be again extended with a more complicated one if necessary. Hence, we suppose that once a PDU is correctly transmitted, a new one is always present in the source buffer. This model is referred in the literature [3] as *Heavy Traffic* condition, and describes exactly a continuous packet source. Thus, it holds for example for a TCP file transfer (FTP-like session or video/audio continuous data streaming): reliable ARQ almost completely avoids TCP timeouts (when the channel error rate is not too large) and the TCP level, after filling the bandwidth-delay product, behaves as a continuous packet source (the TCP window size is not decreasing because error recovery is never triggered). Should the *Heavy Traffic* assumption not be verified, the delivery delay computed with it can be seen as an upper bound (worst case analysis). An evaluation relaxing this hypothesis would still be possible, by following an approach as in [7].

III. Computation of the Delivery Delay Statistics

We compute the delay statistics for a single PDU transmitted using Selective Repeat ARQ. We do this by tracking the successful delivery of the PDU of interest (called *tagged PDU*), as well as all previous PDUs. Some remarks about the notation used for the rest of the paper are presented here. The slot in which the tagged PDU is transmitted for the first time will be indicated as slot $t = m$. The m-sized window from slot 1 to slot m will be called *fundamental window*, due to the important role it plays in the analysis. For the sake of simplicity, we will finally consider the delivery of the tagged PDU as complete when all PDU with smaller id have been correctly *transmitted*. This is not completely appropriate, since the release time is evaluated at the receiver. However, the delay between these two instants is a constant t_c (known a priori and approximately equal to $m/2$), that is the sum of the path delay and the processing delay: in other words, since this term does not affect the analysis, we will not write it to avoid unnecessarily long expressions. Thus, in the following we will study the statistics of $P_d[k]$, defined as the probability that the delivery delay equals k slots plus the constant term t_c. For example, $P_d[0]$ is the probability that the tagged PDU is released at the instant of its reception, i.e., the first transmission attempt is successful and the re-sequencing delay is zero.

Proposition 1: The channel state at time $t = 0$ is $i = 0$, i.e., the corresponding PDU is correctly received.

Proof: This follows immediately from the fact that at time $t = m$ the *first* transmission of the tagged PDU occurs. If the transmission at time 0 would have been erroneous, a retransmission would have been scheduled instead. □

Proposition 2: The tagged PDU can be delivered in order if and only if ($\Leftrightarrow$) all the PDUs transmitted in the fundamental window are resolved.

Proof: Since PDUs identifier are assigned in increasing order, the PDUs contained in the fundamental window are surely characterized by a smaller id than the one assigned to the tagged packet. Hence, in the slot in which the tagged PDU is released all these PDUs must have been resolved. This proves the $\Rightarrow$ condition. In the following, we show that the resolution of the fundamental window is also sufficient ($\Leftarrow$), i.e., no other PDUs can affect the release of the tagged packet. This can be shown considering the slot of the *first* transmission of each PDU. If it is between 1 and m, these PDUs are obviously taken into account in the fundamental window. PDUs transmitted for the first time at $t < 0$ can be either acknowledged or retransmitted m slots later. If they are acknowledged before time m, then they are all resolved when

the tagged PDU is transmitted for the first time and do not affect its delivery. For this reason they can be neglected. The only PDUs we need to consider are the ones that are still unacknowledged at time m, these PDUs are retransmitted in the fundamental window. Finally, PDUs transmitted for the first time in slot $t > m$ have a larger id than the tagged PDU, thus they do not affect the delivery delay statistics. In conclusion, the only PDUs that can block the delivery of the tagged packet are transmitted in the fundamental window that is for this reason sufficient. □

To evaluate the resolution of the fundamental window we formulate this algorithm, in which the slots in position $t \geq 1$ are marked as follows:

1) Every slot $t \geq 1$ begins unmarked. Let $t = 1$.
2) If there are m consecutive marked slots, starting from t, the procedure ends. Else, increase t until an unmarked slot is encountered.
3) If in t the transmission is successful, mark with the label *resolved* every slot in position $\kappa m + t$, with κ integer, $\kappa \geq 0$. Increase t by 1 and go to step 2.
4) Else, in t an erroneous transmission occurs. In this case, mark only slot t with the label *unresolved*, increase t by 1 and go to step 2.

For example, suppose $m = 3$. Suppose a good channel state at $t = 1$ is followed by a burst of four erroneous slots and then the channel is again good for three slots, the algorithm gives: 1=resolved, 2=unresolved, 3=unresolved, 4=resolved (despite the channel error, it was previously marked), 5=unresolved, 6=resolved, 7=resolved, 8=resolved. After slot 8, the algorithm ends as slots $9 \div 11$ are resolved.

Additionally, observe that also every slot $t \geq 9$ is marked. It is straightforward to prove that this is always true, i.e., the ending condition of the algorithm at step 2, is equivalent to say that after a slot with a sufficiently high position every other slot is marked as resolved. Next propositions explain how this can be useful to evaluate the delivery delay.

Proposition 3: Consider a sequence of slots from $t-m+1$ to t, with $t \geq m$. Every slot of this sequence can be associated, by means of a one–to–one correspondence, to a different PDU transmitted in the fundamental window, so that the label assigned by the above algorithm applies both to the slot and the status of the related PDU at time t.

Proof: Formally, we define a function $x \to g_t(x)$ (slot $\to$ PDU), that relates the generic slot in position x to a PDU $g_t(x)$ transmitted in the fundamental window. We have to show that it is always possible to do this with the right correspondence between the label of each marked slots and the status of the related PDU at time t. This can be proven by induction. For $t = m$ the statement is true for the correspondence $g_m(x) = x$. In this case we relate each slot of the fundamental window with the PDU transmitted in it, and the satisfaction of the further conditions is trivial. Now, suppose that the statement holds for t, for which the correspondence $g_t(\cdot)$ is defined. We can define the correspondence $g_{t+1}(\cdot)$ for $t + 1$ by letting $g_{t+1}(x) = g_t(x)$ for $t - m + 1 < x \leq t$ and $g_{t+1}(t+1) = g_t(t-m+1)$. It is also straightforward to verify that the correspondence between slot label and PDU resolution is correct. In fact, for every slot x, $t - m + 1 < x \leq t$, the condition holds for the inductive hypothesis, whereas slot $t+1$ represents the resolved/unresolved status for the same PDU of the fundamental window as slot $t - m + 1$. □

Proposition 4: The tagged PDU is released in the last slot of the first m-sized group of only resolved slots.

Proof: Let us call f the last slot of this group of PDUs. At slot f the tagged PDU is surely delivered, as every PDU transmitted in the fundamental window has been correctly received: this comes from Proposition 3 applied to an m-sized window of only resolved slots. We prove *per absurdum* that the tagged PDU can not be released before this slot. Suppose that the tagged packet is resolved in slot $t < f$: thus, since f is the *last* slot of the *first* m-sized group of resolved PDUs, the window from slot $t - m + 1$ to slot t contains at least one unresolved slot. Proposition 3 implies then that at slot t at least one PDU transmitted in the fundamental window has not been correctly received and Proposition 2 states that in this case the delivery is not ended. □

Thus, according to Proposition 2, the memory of the system is the state of m PDUs of the fundamental window and the channel state. Suppose that we are looking at slot t as the current one. Proposition 3 suggests also that we can keep track of the past memory by using the resolved/unresolved status of the $m - 1$ most recent past slots, i.e., slots $t - m + 1, t - m + 2, \ldots, t - 1$. Therefore, this information can be carried with a binary variable for each slot: we assign $b_k = 1$ if slot $t - m + 1 + k$ is still unresolved, and $b_k = 0$ otherwise, for $k = 0, 1, \ldots, m-2$. This string of bits can also be represented by the integer $i = \sum_{k=0}^{m-2} b_k 2^k$. In addition, we need to specify the status of the current slot, i.e., slot t. In this case we also need to track the channel state, which is necessary to determine the future evolution of successful transmissions. (Note that this is not necessary for past slots, since the Markovian nature of the channel evolution allows to neglect the channel state in slots $s < t$ once the channel state in t is known.) For the current slot, three situations are possible: the channel is good, which implies that the slot is resolved (if it is not resolved already, the good channel state makes it resolved now); the channel is bad and the slot is resolved (in a previous transmission); the channel is bad and the slot is still unresolved. These three possibilities will be denoted by 0, 1 and 2, respectively, and the associated variable will be denoted by u.

Consider now the random process $X(t) = (i(t), u(t))$ which jointly tracks slot-by-slot the Markov channel evolution and the status of the m latest slots. This process is a Markov chain, as proved before. In order to determine the possible transitions $X(t) \to X(t + 1) = (i', u')$ and the corresponding transition probabilities, suppose at time t the bitmap $i(t)$ is $(b_0, b_1, \ldots, b_{m-2})$, where the most significant bit b_{m-2} denotes the status of the most recent among the past slots. At time $t + 1$ this bitmap is clocked one position into the past, i.e., $i'(t + 1) = (b'_0, b'_1, \ldots, b'_{m-3}, b'_{m-2}) = (b_1, b_2, \ldots, b_{m-2}, f(u))$, where $f(u) = 1$ if $u = 2$ (current slot at time t was still unresolved), and $f(u) = 0$ if $u = 0, 1$. (More compactly, in this case $f(u) = \lfloor u/2 \rfloor$.)

Regarding the value of $u' = u(t + 1)$, note the following. If at time t $b_0 = 0$, the corresponding slot has already been resolved, and therefore $u' = 0$ or 1 according to the channel state at time $t + 1$. On the other hand, if $b_0 = 1$, the slot is still unresolved at time t, hence we have $u' = 0$, if the channel at time $t + 1$ is good (slot is resolved at this time), and $u' = 2$ otherwise (slot remains unresolved). There are only two possible destinations for $X(t + 1)$, given $X(t)$, since the shift of the bitmap is deterministic and the only random variable is the channel state which can assume two values. More precisely, the transition probabilities are given as follows:

- if i is even (i.e., $b_0 = 0$), then

$$P[X(t+1) = (i', u') | X(t) = (i, u)] = \quad (1)$$
$$= \begin{cases} p_{xv} & \text{if } i' = \lfloor \frac{i}{2} \rfloor + \lfloor \frac{u}{2} \rfloor 2^{m-2}, \\ & x = \lceil \frac{u}{2} \rceil, u' = v, v = 0, 1 \\ 0 & \text{otherwise} \end{cases}$$

- if i is odd (i.e., $b_0 = 1$), then

$$P[X(t+1) = (i', u') | X(t) = (i, u)] = \quad (2)$$
$$= \begin{cases} p_{xv} & \text{if } i' = \lfloor \frac{i}{2} \rfloor + \lfloor \frac{u}{2} \rfloor 2^{m-2}, \\ & x = \lceil \frac{u}{2} \rceil, u' = 2v, v = 0, 1 \\ 0 & \text{otherwise} \end{cases}$$

where the use of $u' = 2v$ in the latter case means that a good channel $v = 0$ leads to $u' = 0$ whereas a bad channel $v = 1$ leads to $u' = 2$, i.e., the situation of bad channel and unresolved slot. According to the above rule, the transition probability matrix can be built, with only two non-zero entries per row.

In order to find the delay statistics, we proceed as follows. Let $\boldsymbol{\pi} = [\boldsymbol{\pi}_0 \ \boldsymbol{\pi}_1 \ \cdots \boldsymbol{\pi}_K]$ be a $1 \times K$ vector whose $K = 3 \cdot 2^{m-1}$ entries represent the probabilities that the system starts in a given state. $\boldsymbol{\pi}$ is computed as follows:

- if u is even (0, 2):

$$\boldsymbol{\pi}_{(i,u)} = p_{0 b_0} \left[\prod_{j=1}^{m-2} p_{b_{j-1} b_j} \right] p_{b_{m-2} \frac{u}{2}} \quad (3)$$

- if u is odd (1):

$$\boldsymbol{\pi}_{(i,u)} = 0 \quad (4)$$

Let $\boldsymbol{e}_0$ be a column vector of all zeros except for the entries corresponding to states $(0, 0)$ and $(0, 1)$, that are equal to 1. If $\boldsymbol{T}$ is the transition matrix of the Markov chain $X(t)$, we determine:

$$\mathcal{P}_c[k] = \boldsymbol{\pi} \boldsymbol{T}^k \boldsymbol{e}_0, \quad k \geq 0 \,. \quad (5)$$

The distribution $\mathcal{P}_c[k]$ is the probability that the delivery delay is greater than or equal to k. Finally, $P_d[k]$ is determined as:

$$P_d[0] = \mathcal{P}_c[0], \ \ P_d[k] = \mathcal{P}_c[k] - \mathcal{P}_c[k-1] \ \ \forall k > 0 \,. \quad (6)$$

IV. Results

The delivery delay statistics $P_d[k]$ has been computed according to the above analysis, for various values of the channel error probability ε and the channel burstiness b. To test the accuracy, we used a simulator in which we simply implemented the transmission of packets with a SR ARQ scheme applied to the same scenario; thus, we empirically measured the delivery delay statistics, instead of deriving them from the exact analysis.

In Figure 1 we evaluate $P_d[k]$ and compare the case of independent (*iid*) channel with different values of the correlation b. In any case, the shape of the delivery delay statistics presents a step-wise behavior with a consistent decreasing gap after every position κm, κ integer. Moreover, when errors are independent, $P_d[k]$ is almost constant within a given m-sized window, whereas in the correlated case it presents an increasing behavior with the maximum placed at the end of the round. This means that the transmissions of the tagged PDU, which occur in position κm, are always a bottleneck that blocks the resolution of the entire window. In fact, only after the slot κm every PDU that was still unresolved at $(\kappa - 1)m$ has had another retransmission. That is, when two or more PDUs (possibly including the tagged one) block the delivery,

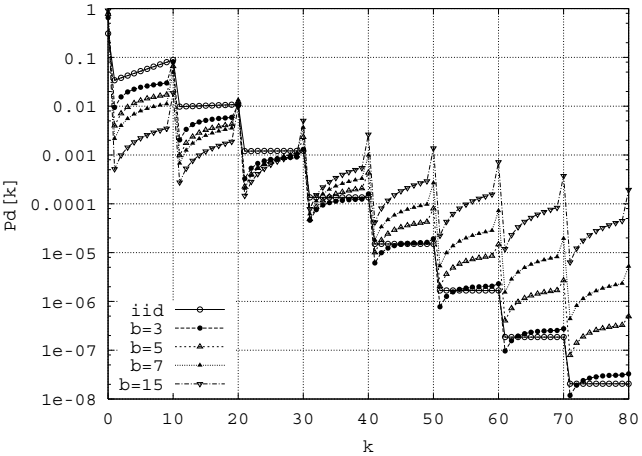

Fig. 1. $P_d[k]$, *iid* channel vs. $b = 3, 5, 7, 15$, with $m = 10$, $\varepsilon = 0.1$.

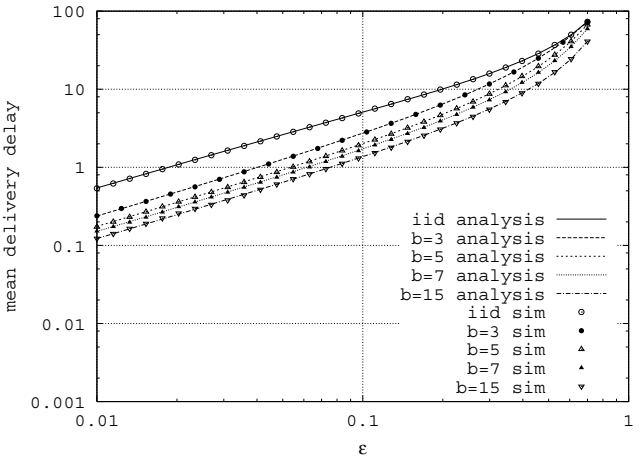

Fig. 2. Mean delivery delay as a function of ε.

the most restrictive condition is the correct transmission of the PDU with highest position in the fundamental window. Being the tagged PDU transmitted at time m, it is more likely to be the most restrictive one.

An interesting value is $P_d[0]$, which corresponds to a successful first transmission of the tagged packet being the $m - 1$ previous transmissions correct. From Fig. 1 it can be observed that $P_d[0]$ in the bursty case is higher, due to the greater probability to have a whole window of correct slots when errors occur in bursts. Also the values for high k are considerably different, because of the increase of the probability of high delivery delays due to correlated channels.

Let us discuss the variations of b. For a given ε, for increasing b also the probability to encounter a long sequence of slots without errors increases. This is the reason why $P_d[0]$ increases as b increases. Note that a significant increase is visible even for $b = 3$, i.e., when the channel burstiness is small. Moreover, the slope of each curve decreases with increasing b, i.e., the larger the burst, the more likely that the starting window will be resolved after a large delay. This means that, on average, a number of slots equal to b are needed for the channel to be restored into the good state, and from here a further round for the starting window to be resolved. Similar observations can be made for Fig. 2. Here, the mean delivery delay is reported as a function of ε by varying b, and also simulation points are plotted for comparison.

These considerations allow us to conclude that the inde-

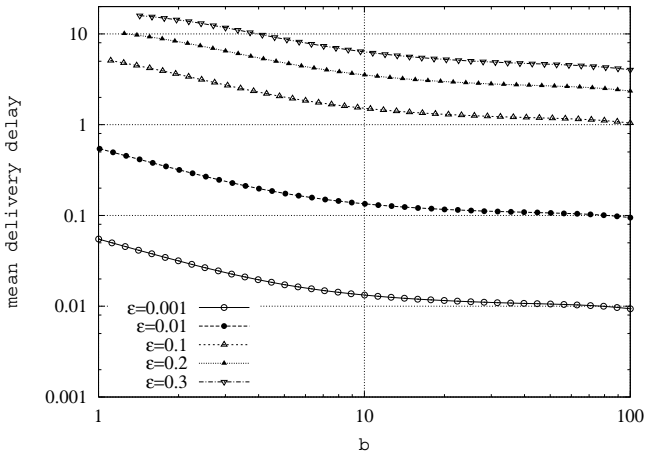

Fig. 3. Mean delivery delay as a function of b.

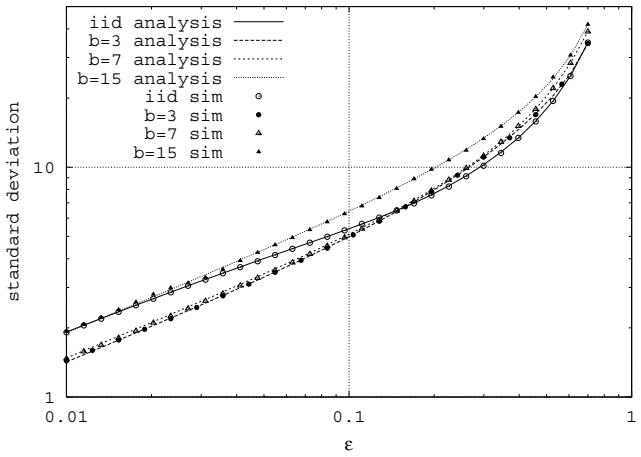

Fig. 4. Delivery delay standard deviation as a function of ε.

pendent case, in general, can not be used to derive a good approximation under many aspects, i.e., the knowledge of the average channel error probability does not suffice to obtain a good delay statistics estimate.

In Fig. 3 the mean delivery delay is reported against the error burstiness b by varying ε. The first value of b on the leftmost part of the graph corresponds to the *iid* case ($b = 1/(1-\varepsilon)$). For each ε, an increasing b always leads to a lower value for the mean delivery delay. The *iid* case is the one characterized by the highest mean delivery delay under all channel conditions.

Fig. 4 reports the delivery delay standard deviation, simulation points are reported for comparison. Unlike for the mean delivery time, this metric in the *iid* case can not be interpreted as a bound. In fact, its role with respect to the correlated case depends on both b and ε. Moreover, its behavior is clearly different from that of the other curves.

The cumulative complementary distribution of the delivery delay statistics, $ccdf[x]$, is plotted in Figure 5, where $ccdf[x]$ is the probability that the delivery delay exceeds x slots, formally:

$$ccdf[x] = 1 - \sum_{k=0}^{x} P_d[k] \tag{7}$$

We report the complementary distribution by varying b in Fig. 5. It is clear that the *iid* case is not a suitable model when errors are correlated (also when the correlation is low, e.g., $b = 3$). In particular, in the correlated case, the tagged packet has a higher probability to be delivered in the first round (slots 0 through m). Once again, it is clear from the Figure that in this range *iid* and correlated cases differ significantly. Even after a full round ($x \geq m$) the curves do not match. For instance, $ccdf[x = m = 10]$ in the *iid* case is almost twice that in the correlated case with $b = 15$.

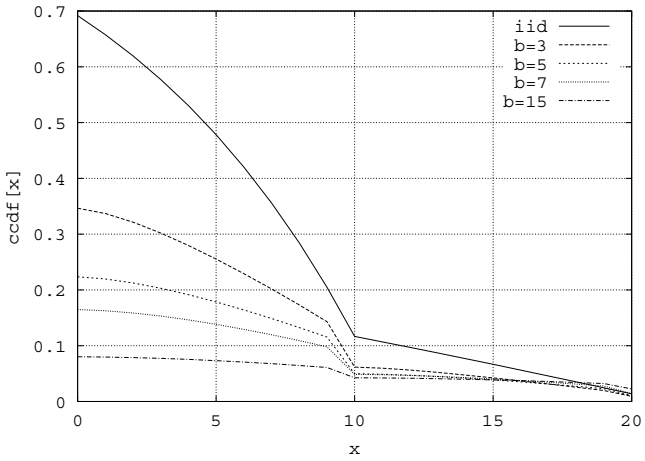

Fig. 5.Cumulative complementary delivery delay distribution, $\varepsilon = 0.1$, $m = 10$.

V. Conclusions

In this paper we studied the delivery delay performance of a Selective Repeat ARQ scheme over a two-state Discrete Time Markov Chain. We obtained the exact statistics of the delivery delay process regarding a single ARQ packet. Main characteristics of the statistics have been compared for several values of the channel error probability and error correlation, and simulation results confirm the goodness of the analysis. The only drawback of the exact analysis is that its complexity grows exponentially with the round trip delay, so approximations with lower complexity can be the goal of future research.

References

[1] R. Fantacci, "Queueing Analysis of the Selective Repeat Automatic Repeat Request Protocol for Wireless Packet Networks," *IEEE Trans. Veh. Technol.*, vol. 45, no. 2, pp. 258–264, May 1996.

[2] M. Zorzi, R. R. Rao, and L. Milstein, "Error statistics in data transmission over fading channels," *IEEE Trans. Commun.*, vol. 46, no. 11, pp. 1468–1477, Nov. 1998.

[3] J. G. Kim and M. M. Krunz, "Delay Analysis of Selective Repeat ARQ for a markovian Source over a Wireless Channel," *IEEE Trans. Veh. Technol.*, vol. 49, no. 5, pp. 1968–1981, Sept. 2000.

[4] A. G. Konheim, "A Queueing Analysis of Two ARQ Protocols," *IEEE Trans. Commun.*, vol. 28, pp. 1004–1014, 1980.

[5] M. E. Anagnostou and E. N. Protonotarios, "Performance Analysis of the Selective-Repeat ARQ Protocol," *IEEE Trans. Commun.*, vol. 34, no. 2, pp. 127–135, Feb. 1986.

[6] Z. Rosberg and M. Shacham, "Resequencing Delay and Buffer Occupancy under the Selective Repeat ARQ," *IEEE Trans. on Inf. Theory*, vol. 35, no. 1, pp. 166–173, Jan. 1989.

[7] Z. Rosberg and M. Sidi, "Selective-Repeat ARQ: the Joint Distribution of the Transmitter and the Receiver Resequencing Buffer Occupancies," *IEEE Trans. Commun.*, vol. 38, no. 9, pp. 1430–1438, Sept. 1990.

[8] S. Lin, D. Costello, and M. Miller, "Automatic-repeat-request error control schemes," *IEEE Commun. Mag.*, vol. 22, no. 12, pp. 5–17, Dec. 1984.

[9] D. Bertsekas and R. Gallager, *Data Network*, 2nd ed. Prentice Hall, 1992.