Mobile Networks and Applications 2 (1997) 183–193

Performance of ARQ Go-Back-N protocol in Markov channels with unreliable feedback *

Michele Zorzi and Ramesh R. Rao

Center for Wireless Communications and Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407, USA

In this paper, an ARQ Go-Back-N protocol with time-out mechanism is studied. Transmissions on both the forward and the reverse channels are assumed to be subject to Markovian errors. A recently developed approach, based on renewal theory, is further extended and the steady state average number of packets in the ARQ system is evaluated. This allows us to determine analytically both throughput and transmission delay of the system. Simulation results, that confirm the analysis, are also presented. Based on the analysis, the trade-off involved in the choice of the time-out parameter is identified and discussed.

1. Introduction

Two types of error control techniques have been used extensively to enhance the reliability of data transmissions. In Forward Error Control (FEC), redundancy is introduced in order to correctly decode a corrupted packet, and in Automatic Repeat reQuest (ARQ), erroneous packets are detected and their retransmission requested [8].

With the resurgence of interest in wireless communication networks, it is important to be able to evaluate the performance of error control schemes under assumptions that accurately model the wireless communication channel. In particular, the analyses which have been done in the past, based on the assumptions of i.i.d. packet transmissions and perfectly reliable feedback, will not be applicable to situations where the errors in consecutive slots are not independent, and where the feedback information may get corrupted due to errors in the return channel. Therefore, although related papers have appeared in the past, performance evaluation of ARQ schemes over fading channels has received renewed interest.

In particular, the combined effect of dependent transmissions and erroneous feedback has been considered only by Kim and Un [7], for the basic ARQ protocols, and by Cho and Un [3], for some more elaborate protocols. In these papers, the throughput analysis is done, by using the theory of renewal processes or the flow graph technique [9]. Delay analyses are relatively easy when the delay associated with each retransmission is a constant. In such cases, the average throughput and delay are directly related, and throughput and delay analysis are essentially the same. In the context of protocols in which the time between two successive retransmissions of the same packet is difficult to express, the delay analysis does not follow easily from the throughput analysis. The Go-Back-N ARQ protocol with timer control is one such example. For such protocols, only the throughput performance is available in the literature.

In this paper, we consider the Go-Back-N ARQ protocol with timer control, whose throughput performance was recently presented in [3]. In [11,13], it is shown that the throughput analysis given in [3] provides an upper bound, and the exact performance is evaluated via a Markovian analysis. This paper presents a non-trivial generalization of the analysis in [11,13], that allows the transmission delay of the same protocol to be computed as well. It must be noted that, in order to accurately study the protocol performance, both throughput and delay are needed. Additional performance metrics, such as probability of undetected errors, are not considered here. The analytical methodology presented, although developed here for a specific protocol, is applicable to the performance study of other protocols with memory, for which standard techniques are not feasible and the flow graph approach is impractical.

In the following, the channel model (section 2) and the protocol (section 3) are described. The proposed technique, based on a Markov chain approach, is described in section 4, and is applied to the study of the protocol in section 5 (specific example) and section 6 (general case). Finally, some numerical results are discussed in section 7.

2. Channel model

We model the channel as in [3], i.e., by means of a Gilbert channel [5] in each direction. This means that the patterns of packet and feedback errors follow two independent first-order Markov models, which are adequately described by the transition matrices $M_F(x) = M_F(1)^x$ and $M_B(x) = M_B(1)^x$, with

$$M_F(x) = \begin{pmatrix} p(x) & q(x) \\ r(x) & s(x) \end{pmatrix}, \qquad M_B(x) = \begin{pmatrix} a(x) & b(x) \\ c(x) & d(x) \end{pmatrix}, \tag{1}$$

* Part of this paper has been presented at the fourth International Conference on Universal Personal Communications (ICUPC), Tokyo, Japan, 6–10 November 1995.

$$M_F(1) = \begin{pmatrix} p & q \\ r & s \end{pmatrix}, \qquad M_B(1) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad (2)$$

where $p(x) = 1 - q(x)$ ($r(x) = 1 - s(x)$) is the probability that the forward slot i is successful given that the forward slot $i - x$ was successful (unsuccessful), and similarly for the entries of $M_B(x)$, with reference to the backward channel. Note that $1/r$ and $1/c$ represent the average lengths of the bursts of errors, which are described by geometric random variables.

In the present context, unlike in most of the literature on this topic, we are interested in very rough transmission conditions, where the marginal probability of having an unsuccessful slot is fairly high (say, about 5–10%) and the length of an error burst may span a considerable number of packets. This situation, which may seem unrealistic in wired networks, is quite common in mobile radio channels, due to fading [2,4].

We assume, in the following, that the packet length is a constant, equal to one time unit. The round-trip delay from the beginning of a transmission to the reception and decoding of the corresponding feedback information is m slots. Hence if a packet transmitted in slot i is negatively acknowledged, it will be retransmitted in slot $i + m$. Positive (ACK) and negative (NAK) acknowledgements can never be confused with each other, i.e., the effect of backward errors is to map the ACK and NAK symbols to an Erasure symbol. Also, each ACK/NAK carries the identity of the last correctly received packet. This implies that a packet whose ACK is lost may be subsequently acknowledged by feedback received in the future.

3. Protocol description

In the following, we consider the Go-Back-N (GBN) ARQ protocol with timer control, as described in [3]. The receiver follows the standard rules of ARQ GBN [1], i.e., it sends an ACK for every correctly received packet. When it receives an incorrect packet, it sends a NAK instead, and discards every successive packet, until a correct copy of the negatively acknowledged packet is received.

The transmitter acts according to the following rules. It sends packets in order, as long as it receives ACKs on the backward channel. Upon reception of a NAK for packet i, it goes back and retransmits in order all packets starting from packet i. If the feedback about packet i is detected in error, it is ignored. If it was an ACK, it is possible that a future ACK/NAK will provide information that may acknowledge packet i. In fact, the ACK/NAK of packet k contains implicit acknowledgement of all packets $i < k$. However, it is possible that the lost feedback was a NAK (in which case no more feedback will be sent, and the uncertainty could last for ever), or that the feedback channel is undergoing a very long burst of errors, so that a large number of ACKs get lost. To cope with such cases, a time-out mechanism is used. This is provided by the use of a counter, which causes the transmitter to retransmit packet i after t slots (and all packet following i afterwards), if by that time it is still unknown whether or not it was correctly received. This allows the transmitter to avoid deadlock or buffer overflow. Note that the transmitter buffer needs to retain only t packets. Of course, it must be $t \geqslant m$. If $t = m$ we have the classic GBN scheme, in which if an ACK is not received at the proper time, retransmission is immediately performed.

For simplicity, the analysis is kept at the block level, and the details of specific error detection schemes are not considered. Therefore, we assume that an adequate CRC code is used, so that the undetected error rate is negligible, and do not explicitly consider the presence of overhead.

4. Analysis

4.1. Definitions

In order to precisely track the protocol evolution, one must clearly distinguish between three types of packets.

- *Outstanding packets*, are packets that have been transmitted but whose feedback is as yet unavailable. Note that the time-out expiration implies a decision about the transmission outcome: as a result, a packet can not be outstanding for more than t consecutive slots.
- *Packets in the system (PITS)*, are packets whose first transmission has occurred, but which are not necessarily outstanding. Note that an outstanding packet is in the system, but not all packets in the system are outstanding. Consider the following example: let the system be in the state in which a NAK for packet 1 was lost. Since the receiver will not send any other feedback information, this situation will last until the timeout associated with packet 1 expires, t slots after it was transmitted. By that time, the transmitter will have transmitted $t - 1$ more packets (2 to t) after packet 1. Upon timeout expiration, the transmitter goes back to packet 1 and retransmits it. By the time the ACK for packet 1 is expected ($m - 1$ slots later), there will be m outstanding packets (in fact, m is the minimum number of outstanding packets), i.e., packets 1 to m, whereas the number of packets in the system will still be t (i.e., packets 1 to t), until some correct receptions occur and this number decreases. Furthermore, if the number of packets in the system is greater than m, upon a successful reception (i.e., when a packet actually departs from the system) an "old" packet (one that is already in the system) will be transmitted.
- *Pending packets* are outstanding packets which have been transmitted at least m slots earlier. Note that pending packets are outstanding packets whose feedback was expected but has not been received, due to failures in the return channel. Note also that the difference between the number of outstanding packets and the number of pending packets is always m.

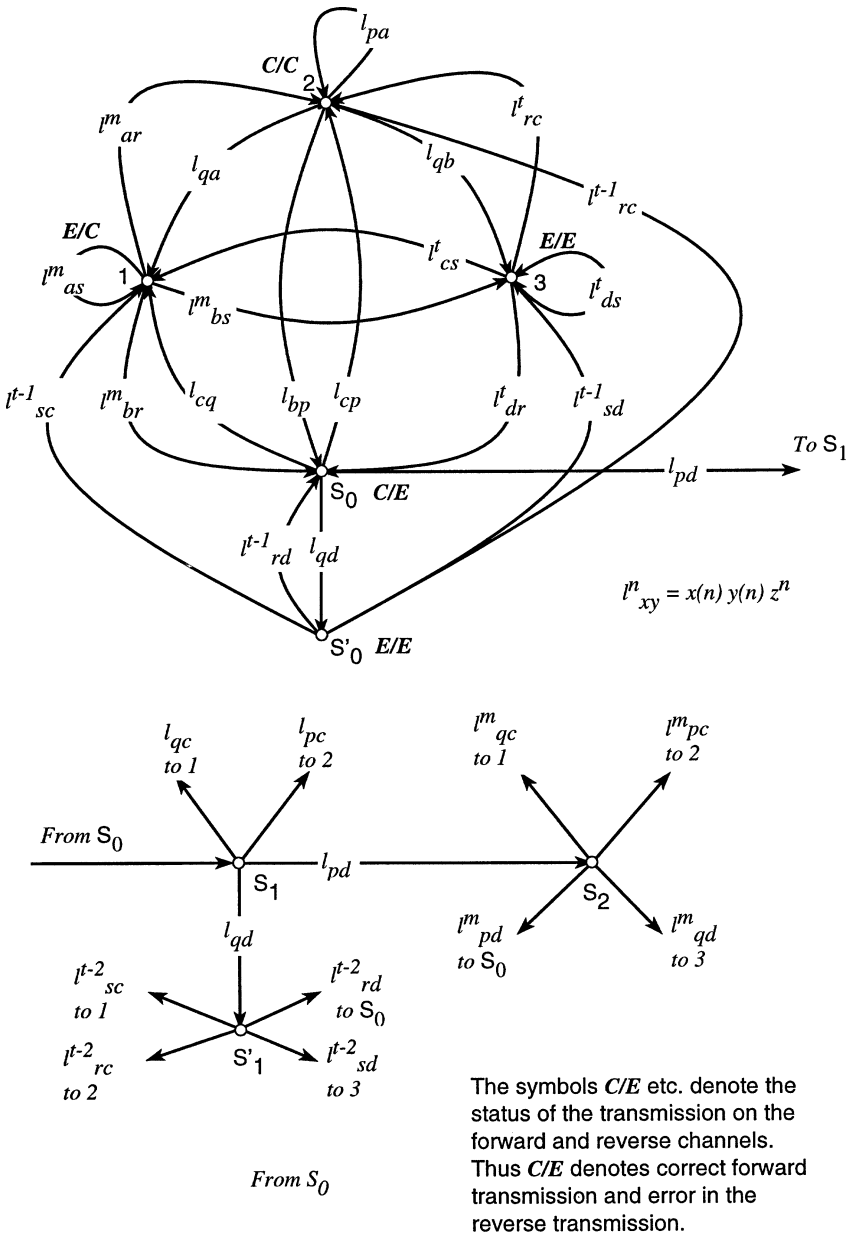


Figure 1. GBN with timer control protocol: embedded Markov chain, $m = 5$, $t = 7$.

4.2. Markov chain for the throughput analysis [11,13]

For completeness, we briefly describe the Markov approach used for the throughput evaluation of the protocol, presented in [11,13].

The error pattern is described by four states: state 1, corresponding to an erroneous packet and a correct feedback transmission (NAK); state 2, where both transmissions are correct (ACK); state 3, corresponding to an erroneous packet and erroneous feedback; and state 4, where the packet is correctly received but the ACK is lost. To take into account the protocol memory, introduced by the time-out mechanism, we need to split state 4 into a number of states, S_i, $i = 0, 1, \ldots, t - m$, and S'_i, $i = 0, 1, \ldots, t - m - 1$. This chain of states corresponds to state 4, in the sense that it is entered when a correct packet is received but its ACK gets lost, and is exited when the resulting uncertainty is resolved. All these states correspond to the uncertain situation in which some feedback information is lost. More than one of them is needed in order to keep track of the number of outstanding packets, i.e., packets which have been transmitted and not yet acknowledged, and whose time-out has not expired. For example, figure 1 reports the complete chain for the case $t - m = 2$.

When the ACK relative to packet k is lost, the system enters state S_0. From state S_0, a transition occurs to states 1 or 2, after one slot delay, if correct feedback relative to packet $k + 1$ is received. Note that this feedback information also acknowledges the outstanding packet k, whose ACK was lost. Therefore, a transition to state 2 (ACK) involves two correct receptions (the current, $k + 1$, and the outstanding, k), whereas a transition to state 1 (NAK) in-

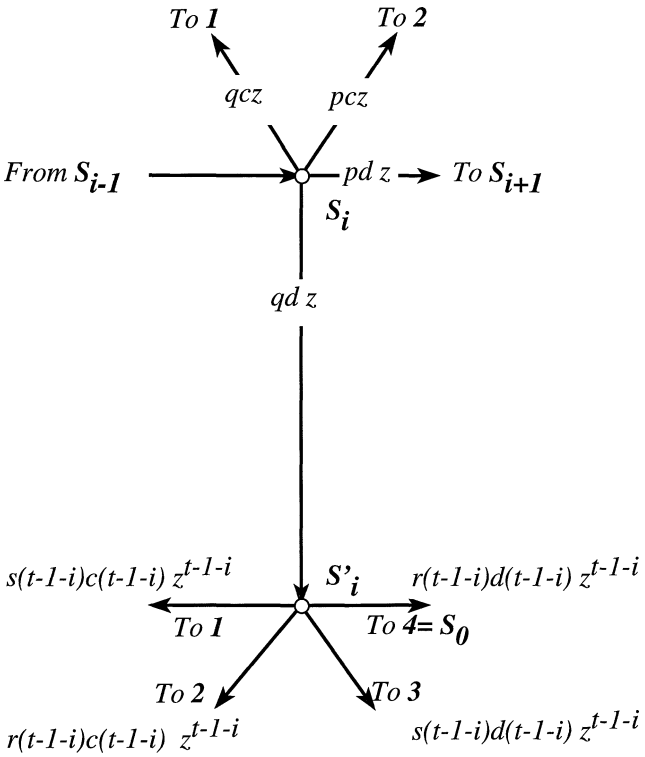


Figure 2. GBN with timer control protocol: generic stage of the embedded Markov chain.

volves one correct reception (packet k). On the other hand, if no correct feedback is received as to slot $k+1$, a transition to S'_0 or to S_1 occurs. If the lost feedback was a NAK (transition to state S'_0), no further feedback will be sent, and the time-out expiration of packet k will resolve the uncertainty. The transitions from S'_0 then involve a delay of $t-1$ slots and no rewards, except for the transition to state 2. If, on the other hand, the lost feedback was an ACK, state S_1 is entered, with two outstanding packets (k and $k+1$).

The pair of states (S_1, S'_1) is analogous to (S_0, S'_0), with the difference that there is one more outstanding unacknowledged packet (and therefore the transitions to states 1 and 2 involve two and three rewards, respectively), and that one more slot has passed, so that the transitions from S'_1 involve a delay of $t-2$ slots. In general, as depicted in figure 2, stage (S_i, S'_i), $i = 0, 1, \ldots, t-m-1$, involves a delay of $t-1-i$ slots in exiting state S'_i, and of 1 slot in exiting S_i. Moreover, there are $i+1$ and $i+2$ rewards associated to the transitions from S_i to state 1 and 2, respectively, and only one reward for the transition from S'_i to 2. Note also that the transition probabilities from S_i to S_{i+1} and S'_i do not depend on i.

For this Markov chain, state transition probabilities and the sojourn time in each state can be identified. This leads to a semi-Markov representation, with transitions weighted by different delays, as in the treatment in [6]. The number of correctly received packets (rewards) is tracked by counting rewards associated to the transitions.

Finally, the average throughput of the protocol, in packets per slot, can be expressed by [11,13]

$$\eta = \frac{\sum_{\ell=1}^{N} \pi_\ell R_\ell}{\sum_{\ell=1}^{N} \pi_\ell D_\ell}, \tag{3}$$

where R_ℓ and D_ℓ are the average reward and sojourn time associated with state ℓ, respectively, and π is the steady-state distribution of the chain.

4.3. Extended Markov chain for the delay analysis

The Markov chain approach to the delay analysis is more complex than the one outlined above. By Little's theorem [1], the average delay is given by the average number in the system divided by the average throughput. Thus, one way to compute delay is to "integrate" over a certain time the number of PITS and divide the result by the number of packets which were successfully received and acknowledged during that time. If we accumulate over an infinite interval, and ergodicity holds, then we can obtain the ensemble averages of these quantities.

Thus, to compute delay, we must keep track of sojourn times, transitions and rewards as well as the number of packets which have already been transmitted once and await retransmission. It is intuitively clear that the number of such packets will be lower-bounded by m and upper-bounded by t. Consequently, a stay of m slots in state 1 could contribute from a minimum of m^2 to a maximum of mt. Splitting each state into a number of new states, each of which is further labeled by the number of PITS, resolves this uncertainty.

As defined before, PITS are those whose first transmission has already occurred and for which an acknowledgement (explicit or implicit) has not been received. The number of packets in the system is sampled at the instant immediately before an outgoing transition from a state occurs. At that time, the information about the outcomes on the two channels (which determines the destination state) is not known. Therefore, the number of PITS is an attribute of the originating state, and does not depend on the destination state.

We also define the *cumulative number of PITS (CPITS)* associated with a transition as the sum of the number of PITS in each slot over the number of slots the system will stay in the destination. In general, this is not just the number of PITS multiplied by the time delay involved, since the number of PITS may increase as time goes by. As an example, consider the situation in which there are m packets in the system and assume that a NAK is lost. The transmitter will keep transmitting and finding no more "old" packets in the system, it will transmit new ones. Therefore, the number of PITS in the first slot after the feedback was expected will be $m+1$, then $m+2$, and so on, until the number of packets reaches t, at which time the timeout of the oldest packet in the system will expire, and the transmitter will go back to that packet and transmit it again.

In summary, for the delay analysis, the state representation must not only keep track of sojourn times, transitions and rewards but also the number of PITS. The computation of the CPITS associated with each transition requires some careful book keeping.

In order to track the protocol evolution, we observe the following methodology:

1. each state will be marked according to the number of PITS, U, immediately before feedback is expected (in the slot before the transition);
2. a transition with an associated reward will decrease U accordingly (provided that $U \geqslant m$, see 4 below);
3. a transition with no reward will keep U constant, except when U is equal to the number of outstanding packets. In this latter case, in fact, all packets in the system are outstanding, and in the next slot the transmission of a new packet is triggered, i.e., a new packet enters the system and U is increased. Note that U can increase only by one at a time, since multiple entrance is not permitted;
4. $U \geqslant m$ always: whenever a transition that could lead to $U = m - 1$ occurs, a new packet will enter the system, and U will not decrease below m;
5. each transition is marked with the transition probabilities and with the CPITS contributed by the stay in the destination state;

Note that each state has a minimum and maximum admissible value for the number of PITS. For example, state 3, which is entered after a timeout expiration, can only have $U = t$, and need not be split. The other states in general will have to be split, so that state i will give rise to a number of states, $i(U)$, where U takes all admissible values in that state. This enhanced state $i(U)$ is used in this study to track the delay, whereas i, a state of the original chain, was adequate to study the throughput. The state i will be referred to as a *superstate*.

Each state continues to have four outgoing transitions, corresponding to the four possibilities for errors on the forward and reverse channels. The destination states for transitions from states within a superstate belong to a common set of superstates. Also, the transition probabilities do not depend on the number of packets in the system, i.e., all transitions from $i(U_i)$ to $j(U_j)$ have the same probability P_{ij}, regardless of U_i and U_j (provided, of course, that the transition from $i(U_i)$ to $j(U_j)$ is admissible). The transition matrix of the state chain can therefore be found by "exploding" the individual entries of the superstate matrix into blocks, according to the rules described above.

The non-zero entries within any block are all equal and correspond to transitions between a pair of superstates. Based on this it can be seen that the protocol evolution, tracked in terms of the detailed states or in terms of the superstates, is Markovian.

Similar observations apply for the delay and reward involved in each transition, which depend only on the origin and destination superstates. On the other hand, the CPITS associated with each transition does depend on the number of PITS. This is the reason why the extended chain is needed.

A specific example will be discussed in the next section.

4.4. Performance of Go-Back-N

We can define an appropriate semi-Markov process, to keep track of the quantities of interest. If $X_1(\tau)$ and $X_2(\tau)$ are two reward functions, from the renewal reward process theory, we have [10]

$$\lim_{\tau\to\infty} \frac{X_1(\tau)}{X_2(\tau)} = \frac{E[X_1]}{E[X_2]}, \tag{4}$$

where $E[X_1]$, $E[X_2]$ are the average rewards earned during a renewal cycle. In the present context, we can define two "reward" functions: $R(\tau)$ keeps track of the number of "departures" (i.e., correctly received and acknowledged packets), and $C(\tau)$ accounts for the CPITS "earned" up to time τ.

As in [11,13], we can define the delay and reward matrices, which have as entries ij (with reference to the transition from state i to state j) the delay, D_{ij}, and the reward, R_{ij}, respectively, and a CPITS matrix, which has as entry ij the CPITS, C_{ij}. Also, we can find the transition probabilities between any pair of such states, which will define the Markov chain embedded in the semi-Markov process. This is done by expanding the chain as described above.

At this point, the ergodicity of the process and the theory of renewal reward processes allow us to express the average packet delay from (4) as

$$D_p = \lim_{\tau\to\infty} \frac{C(\tau)}{R(\tau)} = \frac{E[C]}{E[R]}. \tag{5}$$

Note that the throughput computation can be done by using this more complex model (which is more detailed than the one in [11,13]), through the relationship

$$\eta = \lim_{\tau\to\infty} \frac{R(\tau)}{\tau} = \frac{E[R]}{E[D]}. \tag{6}$$

As expected, the result found through (6) is numerically equal to the one given in [11,13].

The average number of packets in the system is given by

$$E[U] = \lim_{\tau\to\infty} \frac{C(\tau)}{\tau} = \frac{E[C]}{E[D]}, \tag{7}$$

which, of course, is consistent with the above and with Little's theorem. Therefore, the above formulation allows us to evaluate both throughput and delay performance by solving the same chain. If one is only interested in throughput, there is no need to solve this big chain, and the smaller one will do. For some values of the parameters, it may be that the latter is numerically feasible whereas the former is not.

The computation of the average delay relative to a state, based on the transition probability matrix, was discussed in [11,13]: it consists of averaging row i of the delay matrix with respect to the distribution of the destination, conditioned on the origin being state i. The same procedure applies to the CPITS, to find the C_i's, and to the rewards, to find R_i. Once these computations are done, one needs to solve for the steady state distribution, π, which finally yields

$$E[D_p] = \frac{E[C]}{E[R]} = \frac{\sum_{i=1}^{n} \pi_i C_i}{\sum_{i=1}^{n} \pi_i R_i}. \tag{8}$$

This is similar to the relationship we found for the throughput.

At this point, the problem is essentially solved, assuming that we are able to actually compute the matrices and solve for the stationary distribution. This is not very difficult, as long as the number of states is not too large. It is usually easy to get results for chains of a few hundred states.

5. Performance of Go-Back-N: a specific example

In order to clarify the above procedure, before dealing with the general case, we give a specific example. Let us consider the Go-Back-N protocol with timer control, where $m = 5$ and $t = 7$ slots. The embedded Markov chain transition diagram appears in figure 1 [11,13]. The branches are labeled with transition functions of the form $\ell_{xy}^{n} = x(n)y(n)z^n$, where $x(n)$ is one of the entries of the transition matrix $M_F(n)$ (and $y(n)$ is from $M_B(n)$), depending on the origin and destination of the branch. The variable z is just a placekeeper, whose exponent gives the time, in slots, the system stays in the origin before moving to the destination. Also, each transition has an associated reward. The transition functions and the reward matrix contain all the necessary information about the possible transitions and their probabilities, rewards and delays, as discussed in [11,13]. The transition functions are found by inspection from figure 1, and correspond to the following transition matrix:

$$\boldsymbol{P} = \begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{1S_0} & 0 & 0 & 0 & 0 \\ P_{21} & P_{22} & P_{23} & P_{2S_0} & 0 & 0 & 0 & 0 \\ P_{31} & P_{32} & P_{33} & P_{3S_0} & 0 & 0 & 0 & 0 \\ P_{S_0 1} & P_{S_0 2} & 0 & 0 & P_{S_0 S_1} & 0 & P_{S_0 S_0'} & 0 \\ P_{S_1 1} & P_{S_1 2} & 0 & 0 & 0 & P_{S_1 S_2} & 0 & P_{S_1 S_1'} \\ P_{S_2 1} & P_{S_2 2} & P_{S_2 3} & P_{S_2 S_0} & 0 & 0 & 0 & 0 \\ P_{S_0' 1} & P_{S_0' 2} & P_{S_0' 3} & P_{S_0' S_0} & 0 & 0 & 0 & 0 \\ P_{S_1' 1} & P_{S_1' 2} & P_{S_1' 3} & P_{S_1' S_0} & 0 & 0 & 0 & 0 \end{pmatrix} \tag{9}$$

The delay and reward matrices are given by [11]

$$\boldsymbol{D} = \begin{pmatrix} m & m & m & m & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ t & t & t & t & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ m & m & m & m & 0 & 0 & 0 & 0 \\ t-1 & t-1 & t-1 & t-1 & 0 & 0 & 0 & 0 \\ t-2 & t-2 & t-2 & t-2 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{10}$$

$$\boldsymbol{R} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{11}$$

In the above, the states are ordered as follows:

$$(1 \quad 2 \quad 3 \quad S_0 \quad S_1 \quad S_2 \quad S_0' \quad S_1'). \tag{12}$$

The number of states needed to adequately represent the channel conditions and the number of outstanding packet was found to be $4 + 2(t - m) = 8$ in [11]. We will split each of these states according to the possible values of the number of PITS, U, allowed in it.

- State 1 allows all values of U between m and t.
- State 2 does not allow $U = t$. In fact, if 2 is reached from any state with $U = t$, U will then decrease to $U \leqslant t - 1$. This is because a reward is associated with the transition, and because no new packets may enter the system, since there will be always old packets to be retransmitted (unless all pending packets are acknowledged, in which case the destination will have $U = m < t$ anyway).
- State 3 allows only $U = t$, since it always involves the timeout expiration. This is true for states S_0', S_1', S_2 as well.
- As to states S_i, note the following. State S_0 is reached when an ACK is not received. As a result, in the slot during which the system stays in S_0, another packet will be transmitted. This means that the minimum number of packets in the system just before leaving S_0 is $m+1$, whereas the maximum is t. For the same reason, the number of packets in state S_1, which is equal to that in state S_0 (from which S_1 is entered) plus one, must be at least $m + 2$, i.e., it has to be t.

The complete state vector is therefore given as follows (the same order of the entries will be used in the matrix representations):

$$\begin{aligned} &\big(1(5) \;\; 1(6) \;\; 1(7) \;\; 2(5) \;\; 2(6) \;\; 3(7) \\ &S_0(6) \; S_0(7) \; S_1(7) \; S_2(7) \; S_0'(7) \; S_1'(7)\big). \end{aligned} \tag{13}$$

5.1. Expanded transition diagram

The composite information, including transition probabilities, transition times, rewards and CPITS, can be adequately described by a semi-Markov model with 12 states. The transition matrix, which is found by exploding the entries of (9) into blocks when passing from (12) to (13), is given in table 1, where the block structure of the matrix has been highlighted.

According to the rules defined in the previous section, a transition involving some reward will lead to a state whose

Table 1
Transition matrix of the extended Markov chain, $m = 5$, $t = 7$.

	1(5)	1(6)	1(7)	2(5)	2(6)	3(7)	S_0(6)	S_0(7)	S_1(7)	S_2(7)	S'_0(7)	S'_1(7)
1(5)	P_{11}	0	0	P_{12}	0	P_{13}	P_{1S_0}	0	0	0	0	0
1(6)	0	P_{11}	0	P_{12}	0	P_{13}	P_{1S_0}	0	0	0	0	0
1(7)	0	0	P_{11}	0	P_{12}	P_{13}	0	P_{1S_0}	0	0	0	0
2(5)	P_{21}	0	0	P_{22}	0	P_{23}	P_{2S_0}	0	0	0	0	0
2(6)	0	P_{21}	0	P_{22}	0	P_{23}	P_{2S_0}	0	0	0	0	0
3(7)	0	0	P_{31}	0	P_{32}	P_{33}	0	P_{3S_0}	0	0	0	0
S_0(6)	P_{S_01}	0	0	P_{S_02}	0	0	0	0	$P_{S_0S_1}$	0	$P_{S_0S'_0}$	0
S_0(7)	0	P_{S_01}	0	P_{S_02}	0	0	0	0	$P_{S_0S_1}$	0	$P_{S_0S'_0}$	0
S_1(7)	P_{S_11}	0	0	P_{S_12}	0	0	0	0	0	$P_{S_1S_2}$	0	$P_{S_1S'_1}$
S_2(7)	0	0	P_{S_21}	0	P_{S_22}	P_{S_23}	0	$P_{S_2S_0}$	0	0	0	0
S'_0(7)	0	0	$P_{S'_01}$	0	$P_{S'_02}$	$P_{S'_03}$	0	$P_{S'_0S_0}$	0	0	0	0
S'_1(7)	0	0	$P_{S'_11}$	0	$P_{S'_12}$	$P_{S'_13}$	0	$P_{S'_1S_0}$	0	0	0	0

Table 2
CPITS matrix of the extended Markov chain, $m = 5$, $t = 7$.

	1(5)	1(6)	1(7)	2(5)	2(6)	3(7)	S_0(6)	S_0(7)	S_1(7)	S_2(7)	S'_0(7)	S'_1(7)
1(5)	25	0	0	5	0	48	6	0	0	0	0	0
1(6)	0	30	0	5	0	48	6	0	0	0	0	0
1(7)	0	0	35	0	6	49	0	7	0	0	0	0
2(5)	25	0	0	5	0	48	6	0	0	0	0	0
2(6)	0	30	0	5	0	48	6	0	0	0	0	0
3(7)	0	0	35	0	6	49	0	7	0	0	0	0
S_0(6)	25	0	0	5	0	0	0	0	7	0	42	0
S_0(7)	0	30	0	5	0	0	0	0	7	0	42	0
S_1(7)	25	0	0	5	0	0	0	0	0	35	0	35
S_2(7)	0	0	35	0	6	49	0	7	0	0	0	0
S'_0(7)	0	0	35	0	6	49	0	7	0	0	0	0
S'_1(7)	0	0	35	0	6	49	0	7	0	0	0	0

U decreases accordingly, under the constraint that $U \geqslant m = 5$. Therefore, from any state $i(t)$ all transitions due to the reception of an ACK and involving one reward will lead to state $2(t-1)$. Similarly, those from $i(t-1)$ will lead to $2(t-2) = 2(m)$, and those from $i(m)$ to $2(m)$, since the number of packets cannot be smaller than m. Note, however, that transitions with multiple rewards can occur, for example, from $S_0(t)$ to $2(m)$ (two rewards) and from $S_1(t)$ to $2(m)$ (three rewards). In the latter case, all pending packets are acknowledged, and a new packet enters the system, so that the label of the destination is $m = t - 3 + 1$. On the other hand, transitions entering state $1(U)$ will come from states with U packets (i.e., through transitions without any reward), except for the case in which this state is reached from $S_0(U+1)$ or $S_1(U+2)$ and some reward due to the implicit acknowledgement of outstanding packets is earned. All transitions from $3(t)$ will lead to nodes with maximum number of packets (i.e., $1(t)$, $2(t-1)$, $3(t)$ and $S_0(t)$), as happens for transitions exiting $S'_0(t)$, $S'_1(t)$ and $S_2(t)$.

As to CPITS, note that transitions not corresponding to the timeout expiration will lead to a state where the number of packets remains constant during the stay. The number of slots of stay is also a constant depending only on the state, and therefore for these transitions it is easy to find the C_{ij}'s. For example, a transition into node $1(U)$ will involve U packets for m successive slots, and therefore will have a corresponding $C_{ij} = mU$. On the other hand, transitions to states $2(U)$, $S_0(U)$ and $S_1(U)$ involve U packets for one slot, and therefore the CPITS is just U.

When the system arrives at state $S_2(t)$, the timeout expires, and the next feedback is expected m slots later, during which period the number of PITS will remain equal to t, giving rise to a CPITS of mt slots. The same happens upon entering state $S'_1(t)$, whereas in state $S'_0(t)$ there will be $t-1$ slots to the next expected feedback, and therefore we have a CPITS of $t(t-1)$.

Finally, consider transitions to state $3(t)$. Those coming from all states with $U = t$ involve t packets in the system on the first slot of stay in $3(t)$ and in all the following, so that the CPITS will be t^2. On the other hand, a transition from a state for which $U = m$ will correspond to the following situation: in the first slot of stay in $3(t)$ we have $m+1$ PITS, in the following $m+2 = t$, and from then it will remain constant for the following $t-m$ slots. Also, if the previous state had $U = t-1$, in the first slot of stay in $3(t)$ the number of PITS will not increase, since there is still an "old" packet to be transmitted before a new packet

enters the system in the next slot. In both cases, the CPITS involved is given by $t-1$ packets in the first slot, and t in the following $t-1$, i.e., t^2-1.

Based on the above considerations, the CPITS matrix for the process with $m=5$ and $t=7$ is given in table 2 (the order corresponds to the vector (13)).

6. Performance of Go-Back-N ARQ: general case

The above formulation can be generalized. First note that the size of the state space depends only on the difference $t-m$, not on the individual values of the two parameters. Therefore, when $t-m=2$, the same model as before applies. When $t-m>2$, however, the state space needs to be enlarged.

6.1. Number of states

The rules listed in section 4.2 about transitions and state labeling still hold. It is fairly easy to verify that the admissible value of U in any superstate must lie between the bounds given in table 3. Table 3 also reports the number of states into which a superstate is to be split, given by $\max U - \min U + 1$. The value of the index i in the table runs from 0 to $t-m-1$.

The total number of states required is obtained by summing the last column of table 3, for all values of the index i, to obtain $(t-m+6)(t-m+1)/2$. As anticipated, the number of states grows as $(t-m)^2$, since an increase in $t-m$ has the double effect of increasing the number of superstates and the number of states into which a superstate is to be split. To convey a sense of the complexity, we give some values of the number of states vs. $t-m$ in table 4.

6.2. Possible transitions

- For any state $i(U_i)$ and any superstate j, such that the two superstates are connected in the transition diagram obtained by generalizing figure 1, there is exactly one value of U_j such that a transition from $i(U_i)$ to $j(U_j)$ is possible. This is due to the fact that the number of packets in the destination is only a function of the origin state and of the channel outcomes. As already observed, a transition with some reward in the superstate chain will decrease the number of packets accordingly, with the constraint that U can never go below m. On the other hand, transitions that do not involve rewards will lead to the state with the same number of packets or, if this state does not exist, to the state which has the smallest number of packets in that superstate.
  For example, for $U>m$, transitions exiting $1(U)$ can lead to $1(U)$ itself, $2(U-1)$ (reward), $3(t)$ or $S_0(U)$. On the other hand, for $U=m$, the destinations in superstates 2 and S_0 will be $2(m)$ and $S_0(m+1)$. The same is true for transitions exiting states in superstate 2. Transitions from state $3(t)$ or from states $S'_i(t)$, $i=0,\ldots,t-m-1$, and $S_{t-m}(t)$ always lead to states with the maximum number of packets, i.e., $1(t)$, $2(t-1)$, $3(t)$, $S_0(t)$.
- As to state $S_i(U)$, $i=0,\ldots,t-m-1$, $U=m+i+1,\ldots,t$, note that $i+1$ gives the number of pending packets, i.e., already transmitted packets (not including the current one) which can potentially be acknowledged by the currently expected feedback information. Note that the difference between the number of outstanding packets and the number of pending packets is m: when there are m outstanding packets, there is none pending (the current one is not counted). Therefore, a transition from $S_i(U)$ to superstate 1 involves a reward of $i+1$ packets, which will therefore be removed from the system, leading to state $1(U-i-1)$. The same argument applies to transitions to the superstate 2, which involve $i+2$ rewards, and therefore lead to $2(U-i-2)$ (with the usual exception that $S_i(m+i+1)$ goes to $2(m)$). Note that if the number of packets in the system is larger than the number of outstanding packets, a transition from superstate S_i to S_{i+1} (the expected ACK is not received) will keep the number of PITS constant, since the next transmission will actually be the retransmission of an old packet. On the other hand, if all packets in the system are outstanding (i.e., $U=m+i+1$), a new packet enters the system and is transmitted for the first time. Therefore, we have transitions from $S_i(U)$ to $S_{i+1}(U)$, $U=m+i+2,\ldots,t$, and from $S_i(m+i+1)$ to $S_{i+1}(m+i+2)$ (note, in fact, that state $S_{i+1}(m+i+1)$ does not exist). Finally, in the event a NAK is lost, a transition will occur from $S_i(U)$ to $S'_i(t)$, for all possible values of U.

Table 3
Maximum and minimum admitted values of the number of PITS, U, in a superstate, and corresponding number of states in which it is to be split.

Superstate	min U	max U	Number of states
1	m	t	$t-m+1$
2	m	$t-1$	$t-m$
3	t	t	1
S_i	$m+i+1$	t	$t-m-i$
S_{t-m}	t	t	1
S'_i	t	t	1

Table 4
Number of states of the extended Markov chain vs. the amount of time-diversity, $t-m$.

$t-m$	0	1	2	3	4	5	6	10	20
States	3	7	12	18	25	33	42	88	273

6.3. Transition probabilities, delays and rewards

Note that the transition probabilities and the time delays and rewards involved depend only on the superstates. In fact, the delays are even independent of the destination. Therefore, they can be found from the corresponding values of the chain in figure 1 (or its extension).

6.4. Cumulative number of PITS

To complete the approach described above, we need to find the matrix of CPITS. For all transitions in which the number of packets in the system does not vary during the stay in the destination (i.e., all transitions entering superstates 1, 2 and S_i, $i = 0, \ldots, t-m$), the CPITS is the product of the number of PITS of the destination and the number of slots the system stays in it. Therefore, a transition entering state 1(U) will involve a CPITS of Um slots; similarly, we find U for 2(U) and $S_i(U)$, $i = 0, \ldots, t-m-1$, and tm for $S_{t-m}(t)$.

The derivation of the CPITS involved in transitions corresponding to timeout expiration is a little more elaborate since the number of packets in the system is in general not constant during the stay in the destination. In order to elaborate further, we must divide the time of stay in any state into three periods, some of which may be of zero length, depending on the states. Let p and $U \geqslant p$ be the numbers of outstanding packets and of PITS corresponding to the origin state, respectively.

- If a NAK is lost, the transmitter will transmit old packets in the following $U-p$ slots. After that, it will start transmitting new ones, for the following $t-U$ slots. Finally, when the number of packets in the system reaches t, no further increase is possible, and the numer of PITS remains constant thereafter. Thus, we have a first period, of length $U-p$, during which the number of packets in the system remains equal to U (contributing a CPITS of $U(U-p)$); a second period, of length $t-U$, when the number of PITS increases by 1 in every slot; and a third period, with number of PITS equal to t. Note that the sum of the durations of the first two periods is $t-p$ slots.
- If the destination is state 3(t), which implies $p = m$, i.e., no pending packets, the system will remain in it for exactly t slots, and therefore the length of the third period and the contributed CPITS are m and tm, respectively. If the destination is $S'_i(t)$, $i = 0, \ldots, t-m-1$, the corresponding time of stay is $t-i-1$ slots, and the above quantities are again given by $p-i-1 = m$ and tm, respectively.
- Also, when the destination is state 3(t) the length of the first period will be $(U-m)$, with CPITS $U(U-m)$. When the destination is $S'_i(t)$, $i = 0, \ldots, t-m-1$, we have $p = m+i+1$, and the corresponding length of the first period is $(U-m-i-1)$, with CPITS $U(U-m-i-1)$.
- The middle period, during which the number of PITS varies, results in a contribution to the CPITS equal to

$$\sum_{k=1}^{t-U}(U+k) = U(t-U) + \frac{(t-U)(t-U+1)}{2}. \quad (14)$$

Therefore, the total CPITS involved in a transition due to a lost NAK is

$$U(U-m) + U(t-U) + \frac{(t-U)(t-U+1)}{2} + tm \quad (15)$$

for transitions to state 3(t), and

$$U(U-m-i-1) + U(t-U) + \frac{(t-U)(t-U+1)}{2} + tm \quad (16)$$

for transitions to $S'_i(t)$, $i = 0, \ldots, t-m-1$. Note that the notation could be unified by labeling state 3(t) as $S'_{-1}(t)$.

7. Results

Based on the analysis presented above, some numerical results are presented in this section. Note that there are four parameters which affect the throughput performance, viz., the round trip delay, $m-1$, the time-out period, t, the marginal block error rate, ε, and the average burst length, $1/r$. We assume that the forward and backward channels exhibit the same statistics (i.e., $M_F(x) = M_B(x)$).

In figures 3 and 4, the average throughput, η, and the average packet delay, D_p, are plotted vs. the average block error rate, ε, for two values or r. Throughput is expressed in packets per slot, and therefore tends to 1 as $\varepsilon \to 0$, although the throughput efficiency can never be equal to 100% due to the presence of the CRC bits. As expected, an increase in ε produces a worse performance; also, note that, for the same value of ε, long bursts (i.e., small r) yield better performance, since the errors tend to be clustered, and therefore are less harmful. It can be argued from Information theoretic considerations that i.i.d. errors are the most difficult ones to cope with. Similar considerations suggest that it ought to be possible to exploit the memory inherent in correlated errors to improve system performance.

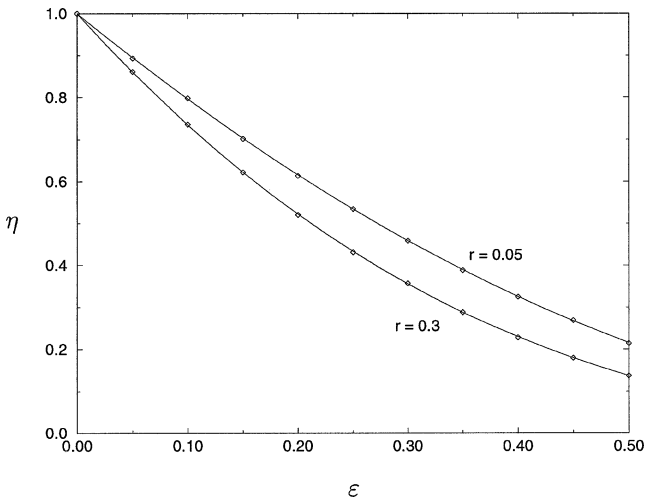


Figure 3. Average throughput, η, vs. block error rate, ε, for $r = 0.3$ and $r = 0.05$, $m = 5$, $t = 7$ ($\diamond$: simulation).

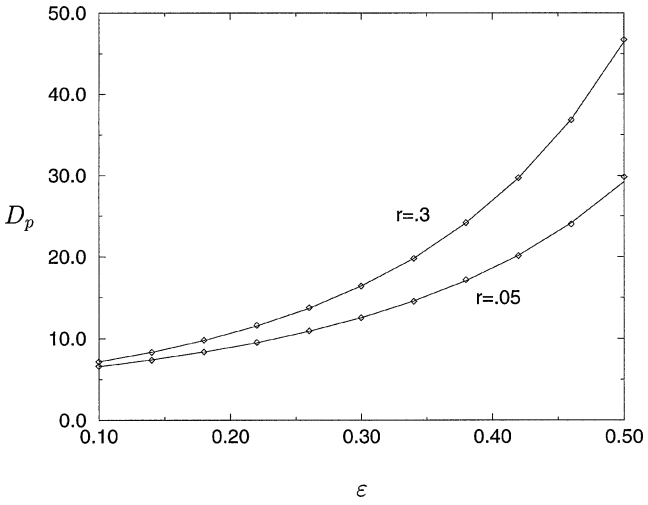


Figure 4. Average packet delay, D_p, vs. block error rate, ε, for $r = 0.3$ and $r = 0.05$, $m = 5$, $t = 7$ ($\diamond$: simulation).

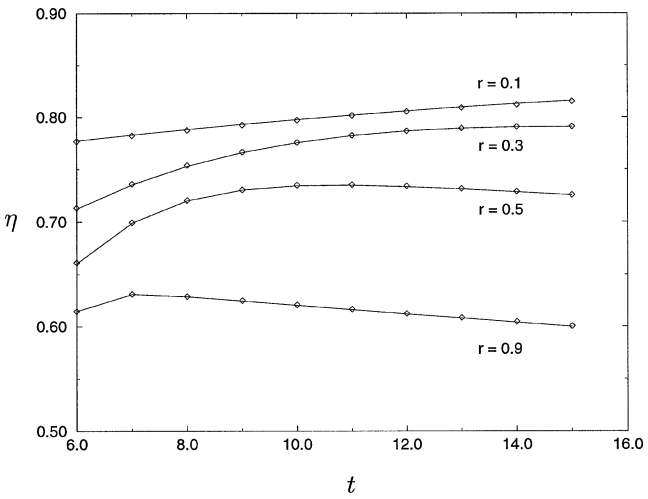


Figure 5. Average throughput, η, vs. time-out period, t, for $\varepsilon = 0.1$ and $r = 0.1$, 0.3, 0.5 and $1 - \varepsilon$ (iid errors); $m = 5$ ($\diamond$: simulation).

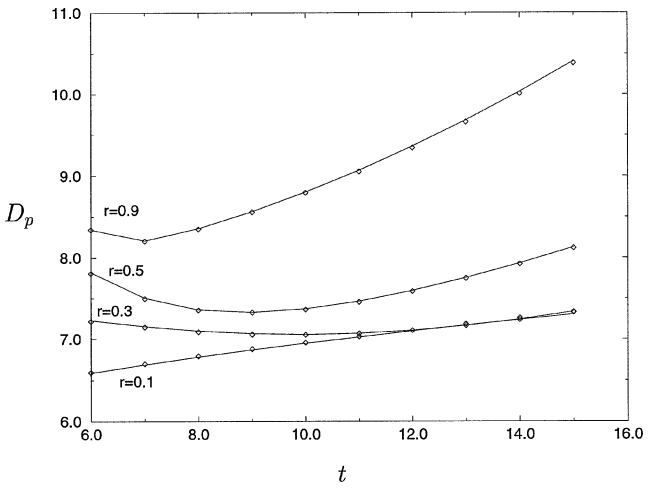


Figure 6. Average packet delay, D_p, vs. time-out period, t, for $\varepsilon = 0.1$ and $r = 0.1$, 0.3, 0.5 and $1 - \varepsilon$ (iid errors); $m = 5$ ($\diamond$: simulation).

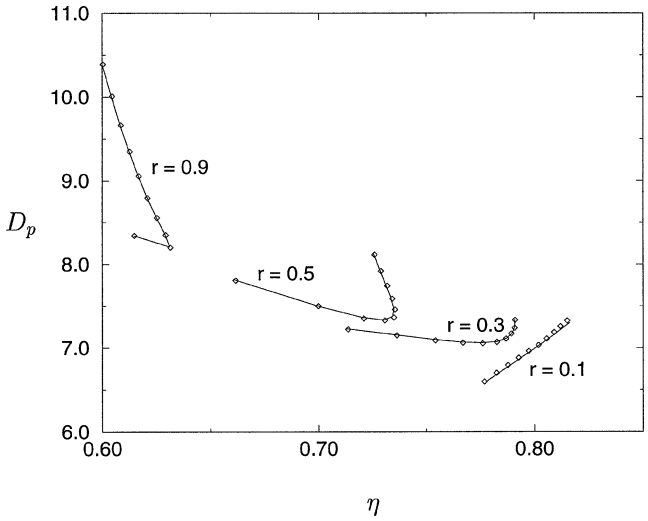


Figure 7. Average packet delay, D_p, vs. throughput, η, for $\varepsilon = 0.1$ and $r = 0.1$, 0.3, 0.5 and $1-\varepsilon$ (iid errors); $m = 5$, $t = 6$ to 15 ($\diamond$: simulation).

Figure 5 shows the throughput, η, vs. the time-out period, t. It is clearly seen that, for given m, ε and r, the choice of the time-out is a trade-off, and there exists an optimum value, t_{opt}. This was to be expected, since t should be short in order not to waste too many slots in the event of an uncertainty to be resolved by time-out expiration, but, on the other hand, a larger t enhances the time diversity feature. Similarly, figure 6 shows D_p vs. the time-out parameter, t. When t is sufficiently large, further increase in its value results in larger delay. Nonetheless, for large r, corresponding to poorer performance, the length of a burst of errors tends to be small. As a result, time diversity can be exploited, and the beneficial effect of a limited increase of t outweighs the potential increase in delay, so that the overall performance turns out to be better. It should be noted that the optimizations of throughput and delay lead to different values of t_{opt}: a more effective way of studying the throughput-delay trade-off is given next.

The analysis here presented makes it possible to compute both throughput and delay for a given set of values of ε, r, m and t. In particular, it is possible to choose one of them as a parameter, and to plot D_p vs. η as the parameter varies. The results obtained allow us to simultaneously study how a parameter affects the performance, in terms of throughput and delay. As an example, in figure 7 we have plotted D_p vs. η for $m = 5$, $\varepsilon = 0.1$, and some values of r. The curves are parameterized by the time-out period, t. Depending on the value of r, the throughput-delay curve exhibits a variety of interesting characteristics.

The curve can be monotonically increasing, as happens for $r = 0.1$. In this regimen increases in throughput come at the expense of delay. Somewhat more surprisingly, it is possible for the curve to fold back on itself. This implies that two values of delay may be associated with a common value of throughput. Of the two, clearly the one with the lower delay is more desirable. Each of the two delay values of course corresponds to a different value of the time out

period t. Consequently, the precise operating point can be controlled and there is no danger of straying from the desired point of operation to the undesired one. We also find that for some r, the lower branch of the curve has a negative slope. This means that both throughput and delay can be enhanced at the same time. This is in contrast to the first regimen described, where increases in throughput occur at the expense of delay. Therefore, we are able to explicitly identify the value of t that will make it possible to exploit the burstiness of the errors.

The analysis of the performance vs. the value of the time-out has not been addressed completely in previous contributions dealing with this type of channel [2–4]. However, such a study is important, since t is the one design parameter which can be easily altered. In fact, for a given protocol, there is some (but not very much) flexibility in choosing the data rate, packet and frame size (which affect the parameters ε and r), whereas the parameter t can be more easily chosen based on the above analysis. Also, if the dependence of t_{opt} on r and ε is known, its adaptive adjustment can be a feasible solution to be implemented in non-stationary channels. On the other hand, as already mentioned, it must be observed that t jointly affects the throughput and delay performance, and this makes the optimization more difficult. In fact, in the presence of stringent delay limitations, one might be forced to accept a considerably suboptimal throughput performance.

Application of the Markov model to transmission on a radio fading channel was studied in [14,15], and results for the performance of GBN on that channel appeared in [11,12,15]. The combined throughput-delay analysis, never presented so far in the literature for the case of dependent errors and unreliable feedback, is a powerful tool for the performance study and for the design of ARQ systems.

8. Conclusions

In this paper, we study a Go-Back-N ARQ scheme which exploits time-diversity to recover from feedback errors. A dependent structure for the error processes on the two channels is assumed, and modeled as a Markov chain. This allows us to compute exactly the throughput and delay performance. The methodology enables us to study the protocol performance as function of the average channel error rate and of the amount of time diversity. The combined throughput-delay analysis, never presented so far in the literature for the case of dependent errors and unreliable feedback, is key to the design of ARQ systems.

Further directions of this research include extensions to more elaborate ARQ protocols, e.g., Selective Repeat, and possibly to hybrid techniques.

References

[1] D. Bertsekas and R. Gallager, *Data Networks* (Prentice-Hall, 1992).

[2] L.F. Chang, Throughput estimation of ARQ protocols for a Rayleigh fading channel using fade- and interfade-duration statistics, IEEE Trans. Vehicular Technol. 40 (February 1991) 223–229.

[3] Y.J. Cho and C.K. Un, Performance analysis of ARQ error controls under Markovian block error pattern, IEEE Trans. Commun. 42 (February–April 1994) 2051–2061.

[4] J.C.-J. Chuang, Comparison of two ARQ protocols in a Rayleigh fading channel, IEEE Trans. Vehicular Technol. 39 (November 1990) 367–373.

[5] E.N. Gilbert, Capacity of a burst-noise channel, Bell System Tech. Journal 39 (September 1960) 1253–1266.

[6] R.A. Howard, *Dynamic Probabilistic Systems* (Wiley, New York, 1971).

[7] S.R. Kim and C.K. Un, Throughput analysis for two ARQ schemes using combined transition matrix, IEEE Trans. Commun. 40 (November 1992) 1679–1683.

[8] S. Lin et al., Automatic-repeat-request error control schemes, IEEE Commun. Mag. 22(12) (December 1984) 5–17.

[9] D.L. Lu and J.F. Chang, Analysis of ARQ protocols via signal flow graphs, IEEE Trans. Commun. 37 (March 1989) 245–251.

[10] S.H. Ross, *Stochastic Processes* (Wiley, New York, 1983).

[11] M. Zorzi and R.R. Rao, Throughput analysis of ARQ Go-Back-N protocol in Markov channels with unreliable feedback, in: *Proc. IEEE ICC '95* (June 1995) pp. 1232–1237.

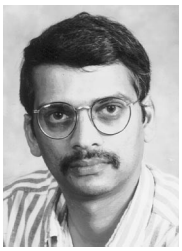
[12] M. Zorzi and R.R. Rao, Performance of ARQ Go-Back-N protocol in Markov channels with unreliable feedback: Delay analysis, in: *Proc. IEEE ICUPC '95* (November 1995) pp. 481–485.

[13] M. Zorzi and R.R. Rao, On the use of renewal theory in the analysis of ARQ protocols, IEEE Trans. Commun. 44 (September 1996) 1077–1081.

[14] M. Zorzi, R.R. Rao and L.B. Milstein, On the accuracy of a first-order Markov model for data block transmission on fading channels, in: *Proc. IEEE ICUPC '95* (November 1995) pp. 211–215.

[15] M. Zorzi, R.R. Rao and L.B. Milstein, ARQ error control for fading mobile radio channels, IEEE Trans. Vehicular Technol. 46 (May 1997) 445–455.

Michele Zorzi was born in Venice, Italy, in 1966. He received the Laurea Degree and the Ph.D. in electrical engineering from the University of Padova, Italy, in 1990 and 1994, respectively. During the academic year 1992/93, he was on leave at the University of California, San Diego (UCSD), attending graduate courses and doing research on multiple access in mobile radio networks. In 1993, he joined the faculty of the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy. He is currently with the Center for Wireless Communications at UCSD. His present research interests involve performance evaluation in mobile communications systems, and random access in mobile radio networks. Dr. Zorzi is a member of IEEE and AEI.
E-mail: zorzi@ece.ucsd.edu



Ramesh R. Rao was born in Sindri, India, in 1958. He received his Honours Bachelor's degree in electrical and electronics engineering from the University of Madras in 1980. He did his graduate work at the University of Maryland, College Park, Maryland, receiving the MS degree in 1982 and the Ph.D. degree in 1984. Since then he has been on the faculty of the Department of Electrical and Computer Engineering at the University of California, San Diego. His research interests include architectures, protocols and performance analysis of computer and communication networks.
E-mail: rrao@ucsd.edu