A Proof of Quasi-Independence Of Sliding Window Flow Control and Go-Back-N Error Recovery Under Independent Packet Errors*

Amarnath Mukherjee
GIT–CC–94/28

Last Revised: June 26, 1995

Abstract

A quasi-independence result holds for the go-back-n automatic repeat request (ARQ) protocol and the sliding window flow control protocol if packet errors are independent. The result is independent of the magnitude of the packet error probability or the cost of an error. A parallel result for the selective repeat ARQ protocol, however, does not appear to hold.

Keywords: Error Recovery Protocols, ARQ, Go-back-N, Window Flow Control.

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

*To be published in Computer Networks and ISDN Systems. This research was supported in part by the National Science Foundation under grant NCR91-16117. Parts of this paper were presented as evidence of quasi-independence in Sigmetrics 1990[7]. The proof of quasi-independence (Section 4) is new.
1 Introduction

The objective of this paper is to show that under certain conditions, the performance of the go-back-n error recovery protocol and the sliding window flow control protocol are quasi-independent. Specifically, if $E[T_{gbn}(N, w, p)]$ is the expected time to transmit an $N$ packet message with retransmissions using a fixed send window of size $w$, $E[T_{noErrors}(N, w)]$ is the expected time to transmit the $N$ packets over an error free channel with same window size, $E[T_{errCost}(N, p)]$ is the expected cumulative cost of retransmissions, and $p$ is the probability of error in either the data packet or its acknowledgment, then under the assumption of independent packet errors,

$$E[T_{gbn}(N, w, p)] = E[T_{noErrors}(N, w)] + E[T_{errCost}(N, p)] + o(p), \quad (1)$$

where $o(p)/p \to 0$ as $p \to 0$. Notice that $E[T_{errCost}(N, p)]$ in (1) is independent of the window size $w$. Figure 1 shows a few examples of the accuracy of the quasi-independence assumption.

Previous analytical studies have focussed on either flow control strategy or error control strategy in isolation (e.g., [2, 3, 5, 8, 6, 10]). The complexity of analyses has usually precluded a simultaneous study of both. Quasi-independence asserts that sliding-window flow control (with a fixed window size) and go-back-n error recovery are orthogonal when packet errors are independent. The result is of course provably valid only for the models studied.

(1) is easy to show when the window does not close (see for example [6]). For the case when the window does close, however, no such proof was known, although we have observed it empirically in many of our measurements and simulations over the past several years. The purpose of this paper is to give a proof for this observation.

The result is valid only if packet errors are independent. An example scenario where this might be true is a mobile host environment where communication is achieved through packet radio or Infra Red, and the dominant cause of packet errors is white noise in the medium. A scenario where this assumption does not apply is where the channel is relatively error free and most packet errors are caused by buffer overflows at intermediate switches.

The quasi-independence property does not, however, hold for the selective repeat protocol, even for independent packet errors. In particular, if $E[T_{sr}(N, w, p)]$ denotes the expected time to transmit $N$ packets with a window size $w$ using selective repeat, then

$$E[T_{sr}(N, w, p)] = E[T_{noErrors}(N, w)] + f(N, w, p, \tau), \quad (2)$$

where $f(N, w, p, \tau)$ is the expected cost due to errors using selective repeat with a window size $w$ and time-out $\tau$ (fixed or variable), and $E[T_{noErrors}(N, w)]$ is as defined earlier for go-back-n. Based on simulation experiments, our conjecture is that $f(N, w, p, \tau)$ is of the form $\frac{Np}{q} \tau(w, p) + o(p)$, where $q = 1 - p$. However, we do not have an exact form for $\tau(w, p)$, except for trivial special cases. Hence, we do not address selective repeat in this paper.

For the go-back-n protocol, however, quasi-independence holds for all values of $p$ and all values of $w$, no matter how large or small. This is shown first through a Petri-Net example
Each set of barplots in these two figures shows the time to transmit $N = 64$ packets using (I) the quasi-independence assumption (the left bar, marked I), and (II) a detailed model for go-back-n (the right bar, marked II). Model I computes $E[T_{\text{newErrors}}(N, w)]$ from a simple window flow control model with no errors and adds $E[T_{\text{errCost}}(N, p)]$ to it, resulting in the right hand side of Equation (1). Model II computes $E[T_{\text{gbn}}(N, w, p)]$ from a more detailed go-back-n model. The models that were used to compute these numbers are described in Section 3.

In part (a), the contribution of the error-free component dominates the error-component. In part (b), the reverse is true: the contribution of the error component dominates over the error-free component. In both cases, however, their sum add up to that obtained by Model II. $p_0$, $w$ and $\tau$ are parameters of the model. $p_0$ and $\tau$ control the contribution of the error component, $w$, the window size, controls the contribution of the error-free component. While the two models yield similar numbers for go-back-n, it can be shown that they do not do so for selective-repeat.

Figure 1: Comparison of the expected time to transmit $N = 64$ packets as computed from two different models.
in Section 3 and then proved in Sections 4 and 5. The paper is organized as follows. Section 2 presents the protocol definitions and system model. Section 3 illustrates quasi-independence through an example. Section 4 gives the proof of the main result of the paper. It is assumed here that transmission times and other delays are exponentially distributed. Section 5 bounds the error due to a quasi-independence assumption for the case when the transmission times and other delays are deterministic. Section 6 presents our conclusions.

2 Preliminaries

2.1 Protocol Definitions

We assume that the sender has a window of size $w$. This is the upper limit on the number of packets that it is allowed to transmit without waiting for an acknowledgment. The sliding window protocol, in conjunction with the go-back-n and selective-repeat retransmission strategies, works as follows. When a packet successfully reaches a receiver, it is always ACKed if it is in-sequence. An error is detected at the sender by either a timer interrupt or by a NACK from the receiver. At this point, if the sender backs up to the first packet in error and restarts the transmission, the strategy is referred to as go-back-n [9]. If, on the other hand, the sender retransmits only that packet which is in error, the strategy is called selective-repeat. The state machine of go-back-n is simpler than that of selective repeat. Also, the selective repeat protocol may require a large receive buffer to cache packets which are received correctly, but out of order. While go-back-n can operate with one receive buffer only, it may be appropriate to allocate additional buffers for re-sequencing out-of-order packets if necessary.

2.2 System and Error Models

We assume that the mean transmission time of a packet at the sender is $1/\lambda$ when its window size allows it to transmit a packet. When the window is not open, however, the sender is inhibited from transmitting. The analysis will consider two cases: (i) the transmission time and other delays are all exponentially distributed, and (ii) they are all deterministic. At the lower levels of protocol stack where the protocol is usually implemented, the coefficient of variation of the transmission time is likely to be less than one but greater than zero. The analysis attempts to capture these two extreme cases.

The performance measure of interest is the expected time to transmit a large multi-packet message consisting of $N$ packets.

3 An Illustration of Quasi-Independence

3.1 Petri-Net Models

In this section, we present an illustration of the quasi-independence property of window flow control protocol and the go-back-n error recovery protocol when packet errors are independent. In [6], this result was shown to hold when the window did not close, i.e., it was shown that

$$E[T_{gln}(N, w = \infty, p)] = E[T_{noErrors}(N, w = \infty)] + \frac{Np}{q}r,$$  \hspace{1cm} (3)
Figure 2: Simple sliding window flow control: Model I

where

- $E[T_{gbn}(N, w = \infty, p)]$ denotes the expected time to transmit $N$ packets in the presence of errors using go-back-$n$,
- $E[T_{noErrors}(N, w = \infty)]$ denotes the time it would take to transmit $N$ packets in an error-free channel,
- $\tau$ is the average cost of an error using go-back-$n$, and
- $(Np/q)\tau$ was the (computed) extra cost due to errors, using go-back-$n$.

Observe that $\tau$ is independent of $w$, so the expected cost due to errors [second term on the right hand side of (3)] was not dependent on the window size. In this section, we investigate with a Petri-Net example, the accuracy of this result when the window may close, even with high probability and both $p$ and $\tau$ may be small or large.

Figure 2 shows a Generalized Stochastic Petri Net (GSPN) model [4] of a simple sliding window flow control protocol ignoring all errors and retransmissions. This model will be used to compute $E[T_{noErrors}(N, w)]$, corresponding to the first term on the right hand side of (3).

The protocol, or more accurately, its Petri-Net representation, works as follows. If the place RdytoSend has a token, the sender can send a packet provided the place CreditsAvail has a token as well. The mean time\(^1\) to send a packet is $1/\lambda_1$. When a packet is sent, one token from each of the above two places is removed; one is added to the place WaitAck where the sender waits for an acknowledgment. Another is added to the place CreditsUsed which is subsequently used

\(^1\)On Notation: Since there are several timed transitions in this model, and we need to distinguish them, the subscripts $\lambda_i \ i \in \{1, 2, 3, 4\}$ are introduced. Hopefully, this will not confuse the reader. The $\lambda$ defined earlier as the mean send rate when the sender's window is open is the same as $\lambda_1$ in the Petri-Net models.
by the receiver of the data. The transition *RecvData* can fire when the receiver has a token in *RdytoRecv* and a token is available in *CreditsUsed*. Upon receipt of the data, the receiver sends an acknowledgment packet which takes a mean time of $1/\lambda_4$. For analytical tractability, we assume here and in future that all packets are separately acknowledged. Notice that there are no errors in this model. For future reference, we shall call this Model I.

Figure 3 shows the GSPN model of the same sliding window protocol, but this time it includes the go-back-n retransmission strategy. In case of an error, all packets from the first packet in error are retransmitted. In the Petri-Net model, we suppress transmission of those packets which follow the erroneous one by using an inhibit arc from the *failedWait* place into the *transmit* transition. In a real implementation, these packets would actually have been transmitted (and then retransmitted following a timeout or a negative acknowledgment). The inhibit arc would be an approximation if these additional packets were to increase the error rate by increasing the level of congestion in the network. However, since this paper is on independent packet errors, that is not of concern.

A successful packet follows the same path as in Model I. In case of an error, however, a token is deposited in the place *failedWait*. This inhibits further transmission at the sender. After a timeout interval of $\tau$, the token in restored to the *RdytoSend* place and normal transmission begins.

A packet error could occur at different points in transit. Let the aggregate probability of error (of the packet or its acknowledgement) be $p$. If the probability of error in data and acknowledgment packets are $p_d$ and $p_a$, respectively, then $p = 1 - (1 - p_d)(1 - p_a)$. In the numerical examples that follow, we assume that both data and ACK packets have the same probability of failure, $p_0$, so that $p = 1 - (1 - p_0)^2$. This is, however, not necessary for quasi-independence to hold (see Section 4).
3.2 Analysis of the Petri-Nets

3.2.1 Analysis of Model I

To study the effect of window size on throughput and round-trip time of packets, we assume that the number of packets to be sent, $N$, is at least equal to the window size $w$, see Figure 2. This ensures that the sender always has a packet to send, and its transmission is delayed only if the window closes. Let $\overline{N}$ be the average number of tokens in the place WaitAck. Let $\rho = \Pr[\text{Credits\;Avail\;is\;not\;empty}]$, which is the probability that the window is not closed. Then the throughput into the box marked with dashed lines is

$$\Lambda_1 = \lambda_1 \rho. \tag{4}$$

Let $\overline{R_I}$ be the average time spent by a token in the box. By Little’s law, we have $\overline{R_I} = \overline{N}/\lambda_1 \rho$, which implies that the expected number of packets initiated by the sender per round-trip time is $\overline{R_I} \lambda_1 \rho$. The expected time to transmit $N$ packets and receive the ACK for the last one, $E[T_{\text{no\;Errors}}(N, w)]$, is given by

$$E[T_{\text{no\;Errors}}(N, w)] = \overline{R_I} (\frac{N}{\overline{R_I} \lambda_1 \rho}) + \overline{R_I} = \frac{N}{\lambda_1 \rho} + \overline{R_I}. \tag{5}$$

Since $\overline{N}$ and $\rho$ can be computed using a Petri Net analyzer and $\overline{R_I}$ can be computed from $\overline{N}$, $E[T_{\text{no\;Errors}}(N, w)]$ is easily obtained.

As before, let $p$ be the aggregate probability of failure of a packet or its acknowledgment, and let $q = 1 - p$. It was shown in [6] that, if the windows never closed then (3) holds even for generally distributed processing and transmission times. We shall show next that (3) holds approximately even when the window may close.

3.2.2 Analysis of Model II

Consider the analysis of the go-back-n protocol with independent packet errors as modeled by the Petri-Net in Figure 3. Let

- $\Lambda_{\text{trans}}$ = effective throughput through transition transmit,
- $\Lambda_{\text{fail}}$ = throughput through transition failure,
- $\Lambda_{\text{suc}}$ = throughput through transition success, and
- $r = \Pr[\text{token in failedWait}]$.

Then, applying Little’s Law to the box surrounding failedWait, we get $\Lambda_{\text{fail}} = r/\tau$, since $\tau$ is the expected time spent in the place failedWait. Now, noting that $\Lambda_{\text{fail}} = p \Lambda_{\text{trans}}$, we have

$$\Lambda_{\text{suc}} = q \Lambda_{\text{trans}} = (q/p) \Lambda_{\text{fail}},$$

which simplifies to

$$\Lambda_{\text{suc}} = \frac{r}{(q/p) \Lambda_{\text{fail}}}. \tag{6}$$

The average cycle time of a token in the successful path is obtained by applying Little’s Law to the box around the place SuccessWait :

$$\overline{R}_{II} = \frac{\overline{N}(\text{SuccessWait})}{\Lambda_{\text{suc}}}.$$
The expected time to transmit \( N \) packets is then given by

\[
E[T_{gh}(N, w, p)]_{II} = \frac{N}{\Lambda_{suc}} + \overline{R}_{II}.
\]    

(7)

### 3.2.3 Comparison of the Two Models

In Figures 1 and 4, we show the time to transmit \( N = 64 \) packets as calculated by the two models. We vary the parameters \( p, \tau \) and \( w \). The values of \( \lambda_1, \lambda_2, \lambda_3 \) and \( \lambda_4 \) are assigned according to those reported in [11], i.e.,

- \( \lambda_1^{-1} \) = time to copy a data packet from the sending host’s memory onto the wire = 2.17 msec;
- \( \lambda_2^{-1} \) = time to copy an acknowledgment packet from the wire into the sending host’s memory = 0.17 msec;
- \( \lambda_3^{-1} \) = time to copy the data packet from the wire into the receiving host’s memory = 1.35 msec; and
- \( \lambda_4^{-1} \) = time to copy an acknowledgment packet from the receiving host onto the wire = 0.22 msec.

The time to complete an \( N \)-packet transmission is obtained by first solving the two GSPN models and then using their outputs as inputs to (3), (5), (6) and (7). We observe that the time predicted by extrapolating Model I (in accordance with (3)) is remarkably close to that obtained by solving Model II. (See Figures 1 and 4.) This is true whether the error-free component dominates over the component that is due to error in Model I, or vice versa, or when both components are comparable in magnitude.

In Figures 5(a)-(b), we study the sensitivity of the result further. For instance, Figure 5(a) shows the probability that the window equals to zero for different experiments using Model I. (All experiments in Figures 1 and 4 use the same error free model because the latter is independent of \( \tau \) and \( p_0 \).) We observe that the probability of window-closing varies over a significant range, from 0.0009 to 0.4450. Similarly, the probability of not being in the failedWait state in Model II is shown in 5(b). It shows the proportion of time spent in transmitting a useful packet to being in an erroneous state in the exact model (Model II). This too varies over a wide range, indicating that the quasi-independence assumption is fairly robust.

Let us now consider conditions under which the two models would in fact be equal. Comparing Model I and Model II, we see from (3), (5), (6) and (7), that the two models yield (asymptotically) identical results if

\[
\frac{1}{\Lambda_1} + \frac{\tau p}{q} = \frac{1}{\Lambda_{suc}}.
\]

For convenience, let us denote \( h = \tau p/q \). Then for the previous condition to hold, we require that

\[
\frac{1}{\lambda_1 \rho} + h = \frac{h}{r},
\]

or

\[
r = \frac{1}{1 + \lambda_4 \rho h}.
\]    

(8)
Figure 4: Comparison of the expected time to transmit $N = 64$ packets as obtained from Models I and II. We observe that the Model I and Model II results are approximately equal for all values of $p_0$, $w$, and $\tau$ considered. In (a), $p_0 = 10^{-2}$, and $\tau = 100$. This results in comparable contributions from the error-free component and the error component as seen from the Model I bar plots. In (b), (c) and (d), $\tau$ is fixed at 1000 and $p_0$ is varied from $10^{-3}$ through $10^{-5}$, resulting in progressively decreasing contribution of the error-component.
Figure 5: Sensitivity analysis of quasi-independence. These two plots show that (a) the probability of window-closing in Model I and (b) the probability of not being in the failedWait state in Model II vary over a wide range. Quasi-independence holds, nevertheless.
Now, if the expected useful time spent per packet is \( t_{good} \) and the wasted time is \( t_{bad} \), then from Model I and our quasi-independence hypothesis we have \( t_{good} = \frac{1}{\lambda p} \) and \( t_{bad} = \tau p/q = h \). (8) says that the expected times derived from the two models will be equivalent if

\[
    r = \frac{1}{1 + \frac{t_{good}}{t_{bad}}},
\]

i.e.,

\[
    \Pr[\text{failed Wait} = 1] = \frac{t_{bad}}{t_{good} + t_{bad}}
\]

This would, by itself, make perfect sense if \( t_{good} \) was somehow obtained from Model II. We are, however, calculating \( t_{good} = \frac{1}{\lambda p} \) from Model I and \( r \) from Model II. The two models will be close if the probability that the window is open given that we are not in the midst of handling an error in the second model, is close to \( p \), the probability that the window is open in the first model. The results from the Petri-Net analysis suggest that this is so.

4 Proof of Quasi-Independence for Exponentially Distributed Times

4.1 Markov Model

In this section, we prove the quasi-independence property when all times are exponentially distributed (as in the GSPN example in the previous section). The expected time to transmit \( N \) packets with sliding window is derived using a Continuous Time Markov Process. The state of the system consists of a pair of tuples \((i,j)\) where \( i \) is the number of packets that will not require retransmission and \( j \) is the number of these \( i \) packets whose acknowledgments are still outstanding. Clearly \( j \leq w \), if the window size is \( w \). In addition, we have the states \( f_i \) corresponding to the states where an error occurs after \( i \) packets have been successfully transmitted (see Figure 6).

The sender transmits with a mean rate \( \lambda \), and the acknowledgments return with a mean rate \( \mu \). Let the packet error probability be \( p \), and let \( q = 1 - p \).

Figure 6 shows the state transition diagram of the ensuing Markov Process. The initial state is \((0,0)\). When a packet is transmitted there can be two possible next states. If the transmission is going to be successful (ultimately), we designate the next state as \((1,1)\). Else, the packet will fail and the next state is \( f_0 \). The rate into \((1,1)\) is \( \lambda q \) and that into \( f_0 \) is \( \lambda p \). Once a packet fails, we assume that it is detected after a mean time \( \tau = 1/\gamma \). Therefore, in Figure 6, we denote the rate from \( f_0 \) to \((0,0)\) by \( \gamma \). The rest of the arcs in the figure follow a similar argument. Note that for all \( j \), a failure transition from \((i,j)\) is into \( f_i \) and the recovery arc from \( f_i \) is only into \((i,0)\). This is a property of the go-back-n protocol: all the packets which are transmitted before a failure are represented by \( i \). By the time the sender detects the failure of packet \( i + 1 \) and acts upon it, the outstanding acknowledgments of all packets up to packet \( i \) must have returned to the sender for it to consider packet \( i + 1 \) as the first failure and the point of beginning a retransmission.
Figure 6: Go-Back-N with Window Flow Control: State Transition Diagram.
4.2 Analysis of the model

We next consider the transient analysis of this Markov Process. We set \((0,0)\) as the initial state and \((N,0)\) as the final state and are interested in computing \(E \left[ T_{gbn}(N, w, p) \right] \), the time to complete an N-packet transmission. Here, \(E \left[ T_{gbn}(N, w, p) \right] \) is the expected time to absorption into \((N,0)\) for this Markov Process. To compute the expected time to absorption, we use the algorithm in [1]. Let \(\eta\) represent the vector\(^2\) of times spent in each of the states before absorption. Let \(Q\) be the transition rate matrix obtained from the original transition rate matrix by deleting the rows and columns involving the absorbing states. Also, let \(P(0)\) be the initial probability distribution of the non-absorbing states. Then the mean time spent in each state before absorption can be computed by solving for \(\eta\) (see [1]) in

\[ \eta Q = -P(0). \]  

The expected time to absorption is then given by

\[ E \left[ T_{gbn}(N, w, p) \right] = \sum_{i,j} \eta_{i,j}, \]

where \(\eta_{i,j}\) are the individual components of \(\eta\).

The solution of (9) for the Markov Process in Figure 6 is especially simple. For the states \((0,0)\) and \(f_0\), we have

\[ \eta_{0,0} = \frac{1}{\lambda q}, \quad \text{and} \quad \eta_{f_0} = \frac{p/q}{\gamma}, \]

For other states \((i,j)\), (9) yields

\[ \eta_{i,j} \left[ \lambda 1_{\{j<w, i<N\}} + \mu 1_{\{j>0\}} \right] = \lambda \eta_{i-1,j-1} 1_{\{i>0,j>0\}} + \mu \eta_{i,j+1} 1_{\{i<j<i+w\}} + \gamma \eta_{i+1,j=0}, \]

and

\[ \gamma \eta_{f_i} = \lambda \eta_{i,0} 1_{\{i<N\}}, \]

where

\[ 1_{\{C\}} = \begin{cases} 1, & \text{if } C = \text{true}; \\ 0, & \text{otherwise.} \end{cases} \]

(11) and (12) are similar to ‘flow equations,’ where we equate all the ‘flows’ into state \((i,j)\) with all the ‘flows’ out of \((i,j)\).

It turns out that for all states \((i,j), j > 0\) in level \(i\), we have all the values needed to compute \(\eta_{i,j}\), if we index through \(j\) from its highest possible value in state \(i\) downwards. Once these values are available, \(\eta_{i,0}\) and \(\eta_{f_i}\) are given in terms of each other and the other known values. This is a considerable simplification over using a general Gaussian elimination algorithm to solve (9).

The solution to the system of equations in (11) and (12) agrees with the results in [6], when the window size is infinite. To see this, consider \(w > i\), i.e., the window does not close at the \(i\)th level. Then

\[ \sum_{j=0}^{i} \eta_{i,j} = \frac{1}{\lambda q}. \]

\(^2\)The individual components of \(\eta\) are \((\eta_\alpha)\), where \(\alpha = (i,j)\) represent the states of the Markov process in Figure 6. For the purposes of the matrix multiplication in (9), \(\eta\) is to be interpreted as a one dimensional vector, subscripted by the \(\alpha\)'s.
The expected time to transmit \( N \) packets is, therefore,

\[
E[T_{gbn}(N, w = \infty, p)] = N \left( \frac{1}{\lambda q} + \frac{p/q}{\gamma} \right)
\]

\[
= N \left( \frac{1}{\lambda} + \frac{p}{q} \left( \frac{1}{\lambda} + \frac{1}{\gamma} \right) \right),
\]

(13)

which was shown in [6].

### 4.3 Proof of Quasi-Independence

We begin this investigation with the solution to (9) for the Markov Chain in Figure 6. For states \((0, 0)\) and \(f_0\) we have:

\[-\eta_{0,0} \lambda + \gamma \eta_{f_0} = -1 \tag{14}\]

and

\[\eta_{0,0} \lambda p - \gamma \eta_{f_0} = 0 \tag{15}\]

These may be obtained from Figure 6 by inspection. From (14) and (15),

\[\eta_{0,0} = \frac{1}{\lambda q} \tag{16}\]

and

\[\eta_{f_0} = \frac{p/q}{\gamma} \tag{17}\]

For states \((1,*)\) we have:

\[
\lambda q \eta_{0,0} - (\lambda + \mu) \eta_{1,1} = 0 \tag{18}
\]

\[\gamma \eta_{f_1} - \lambda \eta_{1,0} + \mu \eta_{1,1} = 0 \tag{19}\]

\[-\gamma \eta_{f_1} + \lambda p (\eta_{1,0} + \eta_{1,1}) = 0 \tag{20}\]

From (16) and (18), we have

\[\eta_{1,1} = \frac{1}{\lambda + \mu} \tag{21}\]

and from (19) (20) and (21), we have

\[\eta_{f_1} = \frac{p/q}{\gamma}, \quad \text{and} \quad \eta_{1,0} + \eta_{1,1} = \frac{1}{\lambda q}\]

Proceeding this way, one may verify that as long as \(w > m\), i.e., the window size is greater than the current row in Figure 6, the following is true:

\[\sum_{j=0}^{m} \eta_{m,j} = \frac{1}{\lambda q}, \quad \text{and} \quad \eta_{f_m} = \frac{\lambda p}{\gamma} \sum_{j=0}^{m} \eta_{m,j} = \frac{p/q}{\gamma}\]

We next investigate \(\eta_{m,j}\) for different values of \(m\) for the cases \(p = 0\) and \(p > 0\).

Case \(p = 0\), \(m < w\):
\[
\sum_{j=0}^{m} \eta_{m,j} = \sum_{j=0}^{m-1} \eta_{m-1,j} = 1/\lambda
\]  \hspace{1cm} (22)

Proof: : Trivial.  ■

Case \( p = 0, \ m \geq w \):

\[
\sum_{j=1}^{w-1} (\lambda + \mu) \eta_{m,j} + \mu \eta_{m,w} + \lambda \eta_{m,0} = \lambda \sum_{j=0}^{w-1} \eta_{m-1,j} + \mu \sum_{j=1}^{w} \eta_{m,j}
\]

\[
\Rightarrow \sum_{j=0}^{w-1} \eta_{m,j} = \sum_{j=0}^{w-1} \eta_{m-1,j} = 1/\lambda
\]  \hspace{1cm} (23)

The last equality follows from taking \( m = w \) and (22) and then progressively increasing \( m \).

Case \( p > 0, \ m < w \):

\[
\sum_{j=0}^{m-1} \eta_{m,j} = \sum_{j=0}^{m-1} \eta_{m-1,j} = \frac{1}{\lambda q}
\]  \hspace{1cm} (24)

\[
\eta_{f_m} = \frac{\lambda p}{\gamma} \sum_{j=0}^{m} \eta_{m,j} = \frac{p/q}{\gamma}
\]  \hspace{1cm} (25)

Case \( p > 0, \ m \geq w \):

\[
\sum_{j=1}^{w-1} (\lambda + \mu) \eta_{m,j} + \mu \eta_{m,w} + \lambda \eta_{m,0} = \lambda q \sum_{j=0}^{w-1} \eta_{m-1,j} + \mu \sum_{j=1}^{w} \eta_{m,j} + \gamma \eta_{f_m}
\]  \hspace{1cm} (26)

But

\[
\gamma \eta_{f_m} = \lambda p \sum_{j=0}^{w-1} \eta_{m,j}
\]  \hspace{1cm} (27)

Putting this in (26), taking it to the left hand side, and canceling out the \( \lambda q \) factor, we get

\[
\sum_{j=0}^{w-1} \eta_{m,j} = \sum_{j=0}^{w-1} \eta_{m-1,j} = \frac{1}{\lambda q}
\]  \hspace{1cm} (28)

As before, the last equality follows from taking \( m = w \) and (24) and then progressively increasing \( m \). From (27) and (28), we have

\[
\eta_{f_m} = \frac{p/q}{\gamma}
\]  \hspace{1cm} (29)
Let us denote \( \eta_{m,j}(p) \) and \( \eta_{f,m}(p) \) as the values of \( \eta_{m,j} \) and \( \eta_{f,m} \) when the error probability is \( p \). Then for \( p > 0 \), we have from (23) and (28),

\[
\eta_{f,m}(p) + \sum_{j=0}^{w} \eta_{m,j}(p) = \frac{1}{\lambda(1 - p)} + \eta_{m,w}(p) + \frac{p/q}{\gamma},
\]

\[
\eta_{f,m}(0) + \sum_{j=0}^{w} \eta_{m,j}(0) = \frac{1}{\lambda} + \eta_{m,w}(0).
\]  

(30)  

(31)

Let the left hand side of (30) and (31) be denoted as \( T_m(p) \) and \( T_m(0) \) respectively. Then

\[
T_m(p) - T_m(0) = \frac{p/q}{\lambda} + (\eta_{m,w}(p) - \eta_{m,w}(0)) + \frac{p/q}{\gamma}
\]

\[
= \frac{p}{q} \left( \frac{1}{\lambda} + \frac{1}{\gamma} \right) + p \eta_{m,w}'(0) + o(p)
\]

\[
= \frac{p}{q} \left( \frac{1}{\lambda} + \frac{1}{\gamma} + q \eta_{m,w}'(0) \right) + o(p).
\]  

(32)

Now, let \( S_N(p) = \sum_{m=0}^{N-1} T_m(p) \). Then, summing over \( m \) in (32), we get

\[
S_N(p) \to S_N(0) + \sum_{m=0}^{N-1} \frac{p}{q} \left( \frac{1}{\lambda} + \frac{1}{\gamma} + q \eta_{m,w}'(0) \right)
\]

\[
= \sum_{m=0}^{N-1} \frac{p}{q} \left( \frac{1}{\lambda} + \frac{1}{\gamma} + q \eta_{m,w}'(0) \right),
\]

(33)

where \( \eta_{m,w}'(0) \) is defined as zero for \( m < w \). In practice, the timeout \( 1/\gamma \) will dominate over \( \eta_{m,w}'(0) \) and \( 1/\lambda \), so the effect of the latter two is negligible on \( S_N(p) \). Notice also that \( S_N(p) \) is equal to \( E[T_{gbn}(N,w,p)] \).

This analysis corroborates and extends the Petri Net analysis results presented earlier. We summarize the result in the following proposition:

**Proposition 1** The expected time to transmit \( N \) packets under go-back-\( n \) error recovery and window flow control when errors are independent is given by:

\[
E[T_{gbn}(N,w,p)] = S_N(p) = S_N(0) + \sum_{m=0}^{N-1} \frac{p}{q} \left( \frac{1}{\lambda} + \frac{1}{\gamma} + q \eta_{m,w}'(0) \right) + o(p).
\]

(34)

5 Proof of Quasi-Independence for Deterministic Times

We show in this section that for deterministic transmission times and delays, sliding window flow control and go-back-\( n \) retransmission strategies are quasi-independent. In particular, we bound the possible error resulting from the use of an error-free component and an error-related-component.

Figure 7(a) shows the timing diagram of a typical sequence of packet transmissions when there are no errors. All times shown have been normalized to a packet transmission time or a slot. We assume in the subsequent analysis that all times are integral multiples of a slot. Here \( N \), the total number of packets is 12, the window, \( w = 5 \), and roundtrip time, \( r = 6 \). The total time to transmit 12 packets for this case is 20 slots. When roundtrip
time, \( r \leq w \), the window does not close. Its performance, studied in [6], was shown to obey the quasi-independence property (Equation (3) in Section 3). Here, we consider the case \( r > w \), i.e., when the window does close. The time to transmit \( N \) packets when there are no errors is now given by (see Figure 7(a))

\[
T_{\text{noErrors}}(N, w, r) = \begin{cases} 
\lfloor N/w \rfloor r + N \mod w, & \text{if } N \mod w \neq 0; \\
(N/w)r + w, & \text{otherwise}.
\end{cases}
\] (34)

Figure 7(b) shows the timing diagram if Packet 2 is in error the first time it is transmitted, but the recovery is successful and all other packet transmissions are successful. We observe that it takes 26 slots which in this case is \( T_{\text{noErrors}} + \tau \). This is in general true, except in special cases. For example, if Packet 3 were to be in error instead, the timing diagram will be as shown in Figure 7(c). We observe that the total time taken is now 25 slots instead of 26.

The reason for this difference is the dichotomy in (34). When Packet 3 is in error, there are 10 remaining packets that need to be transmitted, and 10 \( \mod w \) is zero (because the window size, \( w = 5 \)). However, when Packet 2 is in error, there are 11 packets remaining at the beginning of the retransmission, and 11 \( \mod w \) is not zero. In general, if the remaining number of packets are not exactly divisible by \( w \), a cost of \( \tau \) is incurred as one would expect. However, if a packet error is at a point where the remaining number of packets is exactly divisible by \( w \), the additional cost of the failure is \( \tau - 1 \) instead of \( \tau \). Therefore, for any sequence of \( k \) errors, \( T_{\text{gbn}}(N, w, r, \tau, k) \), the total time to transmit the \( N \) packets is bounded by

\[
T_{\text{noErrors}}(N; w, r) + k(\tau - 1) \leq T_{\text{gbn}}(N, w, r, \tau, k) \leq T_{\text{noErrors}}(N, w, r) + k\tau
\] (35)

Clearly, both limits can be reached. In [6], it was shown that errors in the go-back-n protocol obeyed the negative binomial distribution:

\[
Pr[k \text{ errors}\mid N \text{ packets}] = \binom{N + k - 1}{k}p^k q^N
\] (36)

The expectation of this distribution is \( Np/q \). Using this and (34) and (35), we have

\[
E[T_{\text{noErrors}}(N, w, r)] + (Np/q) (\tau - 1) \leq E[T_{\text{gbn}}(N, w, r, \tau)] \leq E[T_{\text{noErrors}}(N, w, r)] + (Np/q)\tau.
\] (37)

(37) bounds the error in the orthogonality hypothesis. For \( Np \ll 1 \) and \( \tau \gg 1 \), it is reasonably accurate to address error control and flow control separately.

6 Concluding Remarks

Go-back-n error recovery protocol and sliding-window flow control protocol are quasi-independent when packet errors are independent, and transmission times and delays in the system are either all exponentially distributed or all deterministic. Quasi-independence refers to the fact that the expected time to transmit an \( N \)-packet message can be approximately decomposed into sum of two separate components: one obtained by modeling window flow control in an error free channel, and the other obtained by modeling go-back-n in the absence of windows.
Figure 7: Timing diagram of sliding window flow control with deterministic transmission times. Window size, $w = 5$, round-trip time, $r = 6$, per packet transmission time is 1, time to detect an error, $\tau = 6$, and total number of packets, $N = 12$. Arrows pointing upwards indicate data transmission, arrows pointing downwards represent ACKs. (a) Case: No Errors. The time to transmit all packets and receive the acknowledgments is 20. (b) Case: Packet 2 is in error. The time to transmit all packets successfully and receive ACKs is 26, which is equal to $20 + \tau$. (c) Case: Packet 3 is in error. The time to transmit all packets successfully for this case is 25, not 26. This is because at the beginning of retransmission of Packet 3, the remaining number of packets is 10, and $10 \mod w = 0$. We observe that this causes one slot to be ‘gained’ from 26.
A similar straightforward result does not exist for selective repeat. This is because the cost per error in selective repeat is dependent on the window size even for independent packet errors. If the window is large enough to be always open for example, the cost of each error is the time to retransmit an erroneous packet. However, if the window were to close, the cost is a function of the packet in the window that was in error, the window size and the round-trip delay. It is possible to analyze this protocol with windows. However, no simple approximation corresponding to that of go-back-n is known.

A question has been asked as to whether the result generalizes for rate-based flow control strategies. Since there is no unique rate based flow control strategy, it is a difficult question to answer. If transmission rates are dependent on congestion, then most likely, so are packet errors, in which case the quasi-independence will almost certainly not hold. (For example, see numerical examples for go-back-n with congestion-dependent errors in [7].) However, if packet errors, transmission times and other system delays were independent, generally distributed random variables, and there were no windows, then (3) will hold[6, Section 6]. We do not expect these assumptions to hold in practice for systems with rate-based flow control.

7 Acknowledgments

We wish to thank Larry Landweber and John Strikwerda for many stimulating discussions.

References


