SIMULTANEOUS ANALYSIS OF FLOW AND ERROR CONTROL STRATEGIES WITH CONGESTION-DEPENDENT ERRORS

by

Amarnath Mukherjee
Lawrence H. Landweber
John C. Strikwerda

Computer Sciences Technical Report #915

February 1990

Simultaneous Analysis of Flow and Error Control Strategies With Congestion-Dependent Errors

Amarnath Mukherjee
Lawrence H. Landweber
John C. Strikwerda

September 1989
[Revised February 1990]

Department of Computer Sciences
University of Wisconsin
Madison, WI 53706

[to appear in the proceedings of the ACM SIGMETRICS, Boulder, 1990]

Abstract

We investigate the performance of flow control and error control protocols and their role in controlling congestion. We address two kinds of packet errors: (a) independent errors and (b) dependent errors. The latter kind are those which are caused by congestion in the system. We consider the go-back-n and the selective repeat protocols for error recovery. The flow control strategy that we study is the sliding-window protocol where we vary the window size as the control parameter. Our performance measure is the expected time and the standard deviation of the time to transmit a large message, consisting of N packets.

The analysis of independent packet errors, with go-back-n retransmission strategy and sliding window flow control is an extension of our previous result [MLS 89] where the window was assumed not to close. To compute the expected time to transmit an N-packet message, we give sufficient conditions when we can treat the window flow control problem and the retransmission analysis separately. Many researchers have tried to develop detailed and accurate models of window flow control, and it is often very difficult to understand the effect of packet retransmissions on the result. Our result indicates that these studies are indeed valid because the two issues are quasi-independent — at least when the retransmission strategy is go-back-n.

We next develop a framework to evaluate the two retransmission strategies in presence of windows when packet errors are congestion-dependent. We find that, irrespective of retransmission strategy, the expected time as well as the standard deviation of the time to transmit N packets increases sharply if the window size is large in the face of heavy congestion. However, if the congestion level is low, the two retransmission strategies perform similarly. We conclude that flow control, and not retransmission strategy, is the important issue under congestion.

1. Introduction

We investigate the performance of flow control and error control protocols and their role in controlling congestion. We address two kinds of packet errors: (a) independent errors and (b) dependent errors. The latter kind are those which are caused by congestion in the system. We consider the go-back-n and the selective repeat protocols for error recovery. The flow control strategy that we study is the sliding-window protocol where we vary the window size as the control parameter. Our performance measure is the expected time and the standard deviation of the time to transmit a large message, consisting of N packets.

Previous studies have focussed on either flow control or error control strategies, see for example [Mor 88, MLS 89, TW 79 and Zwa 85]. The complexity of analyses has usually precluded simultaneous study of both. One of the main results in this paper shows that under some circumstances the two issues are quasi-independent. For example, to study throughput versus window size for the sliding window flow control protocol using the go-back-n error control strategy, we can study the flow control issue using window models of varying complexity and then combine these results with the term representing the cost of errors due to go-back-n.

We next develop a framework to evaluate the two retransmission strategies in presence of windows when packet errors are congestion-dependent. Earlier work on retransmission strategies, for example [MLS 89, TW 79, Zwa 85], have assumed the independence of errors. If the cause of packet errors is random noise in the communications channel, then this is a reasonable assumption. However, in most networks, such random errors are extremely infrequent as compared to packet failures due to lack of availability of buffers because of congestion [Jac 88]. This introduces complications in that the premise of independent packet failures is no longer valid. In fact, it is more likely for a failure to occur when one has already occurred than when none has occurred. In our study, we assume an error function $p(\alpha)$, where α is some relevant 'congestion information'. We then compare the two retransmission strategies for different error functions in the presence of window flow control. In particular, we show *when* an increase in window size can cause a sharp degradation in performance, and how the two retransmission strategies perform in such a case.

The rest of the paper is organized as follows. Section 2 presents the model we use in this the paper. In Section 3, we demonstrate the orthogonality property of sliding window with respect to go-back-n retransmission for exponentially distributed transmission times. A similar study with deterministic transmission times is deferred to the Appendix. In Section 4, we present the analysis of the go-back-n protocol with sliding windows and congestion-dependent failure rates. The same study is conducted for selective repeat in Section 5. Section 6 compares the two retransmission strategies with numerical examples and finally, we present the conclusions we draw in Section 7.

2. The Model

Without loss of generality, we assume that the transmission of a packet at the sender takes time $1/\lambda_1$, when it has permission to transmit. We consider two different distributions for λ_1: deterministic and exponential. At the lower levels of protocol stack that we

are interested in, we expect the coefficient of variation of λ_1 to be in between that of these two distributions. Our assumption of deterministic and exponential distributions for λ_1 should then give us optimistic and the pessimistic bounds on performance.

The performance measures of interest are the statistics of the time to transmit a large multi-packet message consisting of N packets. 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 [Tan 81]. 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 selective repeat. So, it is of interest to engineers and researchers to see if one can get away with this simple strategy.

As mentioned in the previous section, we address two kinds of packet errors: (a) independent errors and (b) dependent errors. The latter are assumed to be caused by congestion in the communication channel. We assume that if the current 'congestion state' of the system is α, then $p(\alpha)$ is the probability that a packet transmitted now will fail. In the absence of much knowledge of the background traffic in the network, we encode only the information pertaining to the current transmission activity in α, much like in [BPU 88]. Thus, for selective repeat, we assume $p(\alpha) = p(j,k)$, where j is the number of outstanding ACKs and k is the number of failures that have already taken place but not yet recovered from. For go-back-n on the other hand, all failures after the first one and before its detection are irrelevant. We therefore ignore the k-component and assume $p(\alpha) = p(j)$, where j is the number of packets with outstanding ACKs.

Let $w_{max} > w$, where w is any window size that we consider. Since $p(j,k)$ increases monotonically with both j and k, we may approximate it with an n-degree polynomial as follows: The j outstanding ACKs and k undetected failures could take away a maximum of $j+k$ buffers. In addition, k itself indicates the level of 'badness' of the congestion. Thus we may write

$$p(j,k) = p_0 + \sum_{i=1}^{n} a_i \left(\frac{j+k}{w_{max}}\right)^i + b_i \left(\frac{k}{w_{max}}\right)^i$$

where p_0 is the intrinsic failure rate of the network and the other terms are due to congestion. The higher the degree n, the sharper is the increase in $p(j,k)$ with j and k. The constants a_i and b_i are positive and are such that $0 \leq p(j,k) \leq 1$, i.e.

$$p_0 + \sum_{i=1}^{n} (a_i + b_i) \leq 1$$

3. Independent packet errors

In this section, we assume that packets fail independently of each other. Our goal is to show that sliding window flow control and go-back-n retransmission are orthogonal

issues in the sense that they can be studied independently and the results can be put back together in a simple way. Previously [MLS 89], we had seen that this result was true when the window did *not* close. We extend that result here, for the more realistic case when the window may close, even with high probability.

First, let us assume that the transmission times are exponentially distributed. Figure 3.1 shows a Generalized Stochastic Petri Net (GSPN) model [MBC 84] of a simple sliding window flow control protocol ignoring all errors and retransmissions. If the place *RdytoSend* has a token, the sender can send a packet provided the place *CreditsAvail* has a token too. The mean time to send a packet is $1/\lambda_1$. At that time one token from each of the above two places are 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 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 $1/\lambda_4$. Note that there are no errors in this model. For future reference, we shall call this Model I.

In Figure 3.2 we have 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 the packets from the first packet in error are retransmitted. In a real implementation of the protocol, all packets following the erroneous packet will be discarded at the receiver. In the petri-net model, we suppress their transmission altogether by providing the inhibit arc from the *failedWait* place into the *transmit* transition. Although a packet could have failed at different places in transit, we take the total probability of its failing (and its ACK failing) and lump that probability of failing as p as shown in Figure 3.2. In our numerical examples later, we assume that both data and ACK packets have the same probability of failure, p_0, so that $p = 1 - (1 - p_0)^2$. A successful packet follows the same path as in Model I. In case of a failure, a token is deposited in the place *failedWait*. This inhibits further transmission at the sender. After a timeout interval of τ, the token in restored to the *RdytoSend* place and normal transmission can begin. All the packets that would have been transmitted after the erroneous packet and before its retransmission would be retransmitted anyway under the go-back-n strategy. This justifies the inhibit arc at the *transmit* transition. Of course, in so doing, we are ignoring the loading effects of these packets at the receiver. The infrequency of these events should make this approximation reasonable.

Analysis of Model I

To study the effect of window size on throughput and round trip time of packets, we make $N \geq W$ in Figure 3.1. 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 ρ = Pr[CreditsAvail is not empty]. Then the throughput into the box is

$$\Lambda_1 = \lambda_1 \rho \tag{3.1}$$

Let $\overline{R}_I$ be the average time spent by a token in the box. Then 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 = $\overline{R}_I \lambda_1 \rho$. The expected time to transmit N packets and receive the

ACK for the last one, $E[T_{N,noErrors}]_I$, is given by

$$E[T_{N,noErrors}]_I = \overline{R}_I(\frac{N}{\overline{R}_I\lambda_1\rho}) + \overline{R}_I = \frac{N}{\lambda_1\rho} + \overline{R}_I \tag{3.2}$$

Let p be the probability of failure of a packet or its acknowledgment, and let $q = 1 - p$. Then in [MLS 89], it was shown that if the windows *never* closed then

$$E[T_{N,gbn}] = E[T_{N,noErrors}]_I + N\frac{p}{q}\tau \tag{3.3}$$

where Np/q is the expected number of errors in go-back-n and τ is the expected cost per error. This result holds even for generally distributed processing and transmitted times. Our first goal in this paper is to investigate the validity of this result when the window *does* close. We shall show that Equation 3.3 holds approximately even in this case. We also present conditions under which this relation will be exact. Note that $E[T_{N,noErrors}]_I$ is computed from an error free model. The significance of this result is that we can actually analyze sliding window flow control and go-back-n error control as two simplified separate models and put the results back together in a simple way.

Analysis of Model II

In this sub-section, we present the analysis of $E[T_{N,gbn}]$ using the more detailed model in Figure 3.2. Let

Λ_{trans}= effective throughput through transition *transmit,*
Λ_{fail} = throughput through transition *failure,*
Λ_{suc} = throughput through transition *success,* and
r = Pr[token in *failedWait*].

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

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

which simplifies to

$$\Lambda_{suc} = \frac{r}{(\tau p/q)} \tag{3.4}$$

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 :*

$$\overline{R}_{II} = \frac{\overline{N}(SuccessWait)}{\Lambda_{suc}}$$

The expected time to transmit N packets is then given by

$$E[T_{N,gbn}]_{II} = \frac{N}{\Lambda_{suc}} + \overline{R}_{II} \tag{3.5}$$

In Tables 3.1 through 3.6, we present the time to transmit 64 packets as calculated by the two models. We vary the parameters p, τ and W. We assigned measured values of $\lambda_1, \lambda_2, \lambda_3$ and λ_4 as reported in [Zwa 85]. Thus,
${\lambda_1}^{-1}$ = 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 Equations 3.2, 3.3, 3.4 and 3.5. It can be readily seen that the time predicted by extrapolating Model I (in accordance with Equation 3.3) is remarkably close to that obtained by solving Model II (cf. columns 4 and 6 in Tables 3.1 - 3.6). This is in spite of the fact that the probability of the window closing or the probability of being in the *failedWait* state are not insignificant (see columns 2 and 5). We also vary p_0 from 10^{-2} to 10^{-5}, and τ from 10 to 1000 to show that this assumption is valid for a wide range of parameter values.

Let us now consider conditions under which the two models would be equal. Comparing Model I and Model II, we see from Equations 3.1, 3.2, 3.3 and 3.5, 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

$$r = \frac{1}{1 + \frac{1}{\lambda_1 \rho h}} \tag{3.6}$$

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 orthogonality hypothesis we have $t_{good} = \frac{1}{\lambda_1 \rho}$ and $t_{bad} = \tau p/q = h$. Equation 3.6 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.

$$Pr[failedWait = 1] = \frac{t_{bad}}{t_{good} + t_{bad}}$$

This would, by itself, make perfect sense if t_{good} was somehow obtained from Model II. We are, however, calculating $t_{good} = \frac{1}{\lambda_1 \rho}$ *from Model I* and r from Model 2. 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 ρ, the probability that the window is open in the first model. The results from the petri-net analysis suggest that this is so.

If all times are deterministic, we can strengthen these results somewhat. If the window does not close then the orthogonality property (see Equation 3.3) is exact [MLS 89]. If the window does close, the error in the orthogonality approximation can be bounded by (see Appendix B)

$$\begin{aligned} E[T_{N,noErrors}] + (Np/q)(\tau - 1) &\leq E[T_{N,gbn}] \\ &\leq E[T_{N,noErrors}] + (Np/q)\tau \end{aligned} \tag{3.7}$$

4. go-back-n with sliding-window when errors are non-independent

We now investigate the more interesting case when packet errors are correlated. It has been argued persuasively by many researchers that packet losses in networks are not independent of each other. The reason for this is that in today's networks, the dominant loss of packets is due to buffer overflow at either the receiver or at some intermediate point. Characterizing the nature of this dependence remains an open research problem. In this section, we assume that the probability of a packet loss at any time is proportional to the congestion in the system. For lack of anything better, we represent 'congestion' by the number of outstanding acknowledgments. We assume no knowledge of other factors like congestion created by background network traffic, the number of receive buffers or other hardware characteristics. However, given a particular system configuration and background traffic, the congestion level should be an increasing function of the number of packets which are still in the pipeline.

We next derive the expected time to transmit N packets with sliding window 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 4.1).

The sender transmits with a mean rate λ, and the acknowledgments return with a mean rate μ. Our hypothesis is that a packet fails with probability $p(j)$ in state (i, j), where j represents the level of congestion.

Figure 4.1 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 $\lambda q(0)$ and that into f_0 is $\lambda p(0)$. Once a packet fails, we assume that it is detected after a mean time $1/\gamma$. Therefore, in Figure 4.1, we denote the rate from f_0 to (0,0) by γ. 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.

Analysis

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. We are interested in $E[T_N]$, the time to complete an N-packet transmission. However, $E[T_N]$ is just expected time to absorption into (N,0) for this Markov Process. To compute the expected time to absorption, we use the algorithm in [BRT 88]. Let η represent the vector of times spent in each of the states before absorption. Let $\mathbf{Q}$ be the transition rate matrix obtained from the original transition rate matrix by deleting the rows and columns involving the absorbing states. Finally, let $\mathbf{P(0)}$ 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 η [see BRT 88] in

$$\eta \mathbf{Q} = -\mathbf{P(0)} \tag{4.1}$$

The expected time to absorption is then given by

$$E[T_N] = \sum_{i,j} \eta_{i,j}$$

where $\eta_{i,j}$ are the individual components of η.

The solution of Equation 4.1 for the Markov Process in Figure 4.1 is especially simple. For the states (0,0) and f_0, we have

$$\eta_{0,0} = \frac{1}{\lambda q(0)} \qquad \eta_{f_0} = \frac{p(0)/q(0)}{\gamma}$$

For other states (i,j), we get

$$\begin{aligned} \eta_{i,j}\left[\lambda 1_{\{j<w, i<N\}} + \mu 1_{\{j>0\}}\right] =& \lambda q(j-1)\eta_{i-1,j-1} 1_{\{i>0, j>0\}} \\ &+ \mu \eta_{i,j+1} 1_{\{j<i, j<w\}} \\ &+ \gamma \eta_{f_i} 1_{\{j=0\}} \end{aligned} \tag{4.2}$$

$$\gamma \eta_{f_i} = \lambda p(0) \eta_{i,0} 1_{\{i<N\}} \tag{4.3}$$

where

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

Equations 4.2 and 4.3 are like '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 η_{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 Equation 4.1.

In Appendix A, we present a method for determining the variance of the time to absorption. The expected time to absorption falls out of that analysis as a 'byproduct'. This helped us cross-check the numbers we obtained by solving Equation 4.1.

The solution to Equations 4.2 and 4.3 corroborates our previous results. For $p(j) = p\ \forall j$, we get $\eta_{f_i} = \frac{p/q}{\gamma}\ \forall i$ (the analysis which shows that is a little cumbersome and is not the main focus of this paper). And if $w > i$, i.e. if the window does not close at the i^{th} level,

$$\sum_{j=0}^{i} \eta_{i,j} = \frac{1}{\lambda q}$$

So the expected time to transmit N packets is

$$\begin{aligned} E[T_N] &= N\left(\frac{1}{\lambda q} + \frac{p/q}{\gamma}\right) \\ &= N\left(\frac{1}{\lambda} + p/q\left(\frac{1}{\lambda} + \frac{1}{\gamma}\right)\right) \end{aligned}$$

which is also a known result [MLS 89]. We can also use the Markov process to corroborate and somewhat strengthen our previous results for independent packet error for go-back-n with windows (Section 3). In fact, it can be shown that

$$E[T_N, gbn] = E[T_N, noErrors] + O(p)$$

Again, we have omitted the detailed analysis here.

5. Selective Repeat with non-independent errors and sliding-window

In the Selective Repeat Protocol, the sender retransmits only those packets which are in error. We represent the state of a given transmission by the triplet (i, j, k) where i is the number of packets which have been successfully ACKed, j is the number of (ultimately successful) packets whose acknowledgments are outstanding and k is the number of packets which have been transmitted but will fail and their failure is not yet detected by the sender. We assume that packet losses are more predominant than bit errors. Thus in state (i, j, k), we assume that the probability of a packet failing depends on j and k and we denote this probability by $p(j, k)$. Also, let $q(j, k) = 1 - p(j, k)$. An N-packet transmission starts off in state (0,0,0) and ends in state (N,0,0). Assuming the evolution of this process as a Continuous Time Markov Process, we get the state transition diagram of Figure 5.1. To model a window of size w, we have the constraint $j + k \leq w$ for all states (i, j, k). If a new packet is transmitted from (i, j, k) (allowed only if $j + k < w$), the new state could be either $(i, j + 1, k)$ or $(i, j, k + 1)$ depending on whether or not this transmission will ultimately be successful. The corresponding rates are $\lambda q(j, k)$ and $\lambda p(j, k)$ respectively. If an acknowledgment comes back (with rate μ_{ack}) in state (i, j, k), the new state is $(i + 1, j - 1, k)$. If a failure is detected and the packet is successfully transmitted, the new state is $(i, j + 1, k - 1)$. We assume that the mean rate at which a packet error is detected in state (i, j, k) is given by $\mu_{ret}(k)$. This completes all the states to which a transition may occur from state (i, j, k). The states from which one may enter state (i, j, k) are shown in Figure 5.1 as a mirror image of the exit arcs. In the subsequent discussion, we drop the subscript ack from μ_{ack}.

One interesting property of the Markov process in Figure 5.1 is that no state may be visited more than once. To prove this formally, let us consider each of the possible exit states out of (i,j,k) separately. $(i+1,j-1,k)$ represents a state in which $i+1$ acknowledgments have already returned. We cannot ever get back from here to a state where there are only i successful acknowledgments. $(i,j+1,k)$ and $(i,j,k+1)$ represent a new transmission from state (i,j,k). A reduction from $j+1$ to j in $(i,j+1,k)$ will increase i. A reduction in $k+1$ in $(i,j,k+1)$ will increase j to $j+1$ which will in turn increase i. Finally in case of a transition to $(i,j+1,k-1)$, a new failure will increase $k-1$ to k giving $(i,j+1,k)$, but then we have seen that $(i,j+1,k)$ can never return to (i,j,k). This finally proves that state (i,j,k) can be visited at most once, i.e. the Markov process of Figure 5.1 is a directed graph with no cycles. This will help simplify the computation of the mean time to absorption, as we shall see shortly.

The rate of recovery from an error, $\mu_{ret}(k)$, satisfies the relation $\mu_{ret}(1) \leq \mu_{ret}(k) \leq k\mu_{ret}(1)$. The analogy here is to a 'First Come First Serve' scheduling of recoveries (the first inequality) and an 'Infinite Server' scheduling (the second inequality). To find the expected time to transmit N packets, we solve for η in the equation [BRT 88]:

$$\eta \mathbf{Q} = -\mathbf{P(0)} \tag{5.1}$$

where η is the vector of expected times in each of the non-absorbing states, $\mathbf{Q}$ is the generator matrix obtained by deleting the absorbing states and $\mathbf{P(0)}$ is the initial probability distribution of the non-absorbing states. The expected time to absorption then is

$$E[T_N] = \sum_{i,j,k} \eta_{(i,j,k)}$$

Let us now proceed with the solution of Equation 5.1 for the Markov process of Figure 5.1. The equation for state (i,j,k) is given by:

$$\begin{aligned} \eta_{(i,j,k)}\left[\lambda 1_{\{j+k<w\}} + \mu 1_{\{j>0\}} + \mu_{ret}(k)q(j,k)1_{\{k>0\}}\right] \\ = \eta_{(i,j,k-1)}\lambda p(j,k-1)1_{\{k\geq 1\}} \\ + \eta_{(i,j-1,k)}\lambda q(j-1,k)1_{\{j>0\}} \\ + \eta_{(i-1,j+1,k)}\mu 1_{\{i>0\}} \\ + \eta_{(i,j-1,k+1)}\mu_{ret}(k+1)q(j-1,k+1)1_{\{j>0\}} \\ \eta_{(0,0,0)} = 1/\lambda \end{aligned} \tag{5.2}$$

where

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

Equation 5.2 is like a 'flow equation', where we equate all the 'flows' into state (i,j,k) with all the 'flows' out of (i,j,k). Since each state is visited at most once, there are no cycles. Therefore if we begin with the 'root' of the directed graph and work outward, all the η's on the right hand side of Equation 5.2 will be available when required. The solution to Equation 5.1 can thus be obtained in a single pass.

In the Appendix, we present a method for determining the variance of the time to absorption. The fact that the state transition diagram is a directed graph with no cycles helps reduce the complexity of that solution too, significantly.

6. Numerical Results

In this section, we compare the relative performance of the go-back-n and the selective repeat protocols when errors are dependent on congestion. The performance measure of interest is the expected time to transmit N packets. We also investigate the standard deviation of this measure to see how much confidence we can have on the expected value. If we set $p(\alpha)$ to be degree zero (cf. Section 2) we have $p(\alpha) = p_0$, which is independent of the congestion level and is hence the intrinsic packet error rate. In most networks, $p_0 \sim 10^{-5}$. In this case, we do not expect the relative performance of go-back-n and selective repeat to be very different [MLS 89].

To get the performance figures, we need the values of λ, μ, γ and $\mu_{ret}(k)$. Let λ, the transmission rate of packets, be the same as that in the petri-net of Figure 3.1. Since μ will, in general, depend on w, we have approximated it by the inverse of the average round-trip delay of packets in Figure 3.1. This is only an approximation, because the round-trip delay we obtained from Figure 3.1 was a steady state value, whereas now we are dealing with the transient case. However, it should give a relative performance estimate for the two retransmission strategies, as the window size changes. Also, we set $\gamma = \mu_{ret}(1)$, and $\mu_{ret}(k) = k\mu_{ret}(1)$. This latter approximation may favor Selective repeat somewhat. In our experiments, we set $\gamma = \lambda/100$.

The interesting case with respect to errors is when they depend on the congestion level of the system. Therefore, we next consider $p(\alpha)$ to be of degree one, i.e., we let

$$p(j,k) = p_0 + a_1\,(j + k/w_{max}) + b_1 k/w_{max} \tag{6.1}$$

Here a_1 represents the effect of depletion of resources as the number of outstanding packets and their acknowledgments increase. A higher value of a_1 will correspond to a lower availability of buffers due to congestion. b_1, on the other hand, represents the decrease in service quality given that an error has occurred. Clearly, we expect b_1 to be much higher than a_1. This is because once an error has occurred, we are more likely to be in an acute shortage of buffers, than otherwise.

Table 6.1 tabulates the expected time to transmit N=64 packets with go-back-n and selective repeat when $a_1 = 10^{-4}$ and b_1 takes values from 0 to 0.8. The effect of b_1 is seen to be negligible in this case, even for high values of b_1. This is because a_1 is so low that it is unlikely that the $j + k$ packets will have much effect on $p(j,k)$ when $k = 0$. Since $p(j,0)$ remains low (see Equation 6.1), the likelihood of hitting a state with $k > 0$ is very low, and so the effect of b_1 is negligible for this case.

Increasing a_1 does inflate the expected time, as we can see from Table 6.2, where we have put $a_1 = 10^{-1}$. The effect is more pronounced for larger window sizes as one would expect: the larger the window size, the larger the potential for congestion, and larger the potential for error. What is interesting, and not necessarily obvious, is the sharp degradation in performance as seen in Table 6.2. This is the network equivalent of thrashing. From Table 6.1, we note that the expected time decreases at first with respect to window size but then starts increasing again, implying that there is an optimum point for the window size. In Table 6.2, that optimum is for $w = 1$. Thus the optimum point of operating the window will change for different values of a_1. We are far from being the first

to discover the potential for congestion as window size increases: Jacobson, Ramakrishnan and Jain, [Jac88, RJ88], have proposed dynamic window algorithms for the same purpose. Our contribution, however, is to quantify the effect of window size on the congestion level, and to corroborate the fact that larger windows do have a detrimental effect on performance when the network is congested (i.e., a_1 is high).

In Tables 6.3 and 6.4, we tabulate the standard deviation of the time to transmit N=64 packets for the same two values of a_1 as before. Notice that the standard deviation also gets worse with higher a_1, and this effect is again more pronounced for larger windows. A comparison of go-back-n and selective repeat shows that go-back-n performs roughly equal to selective repeat when $b_1 = 0.5$. One would normally expect selective repeat to perform better if b_1 is low, because that implies that an error does not significantly affect the 'state' of congestion. If, however, b_1 is high, transmitting more packets when an error has occurred can only worsen the congestion in the network. Clearly, we can expect go-back-n to perform better under this circumstance.

We should emphasize however, that the jury is still out because we do not yet know the exact nature of the error function.

7. Conclusions and Future Work

In this paper, we presented the analysis of the go-back-n and selective repeat retransmission strategies in conjunction with sliding-window flow control. The analysis was carried out to specifically account for packet errors which may not be independent of each other.

We discovered that when packet errors are independent, go-back-n retransmission strategy and the sliding-window flow control strategy can be treated as orthogonal issues in that the total expected time to transmit an N-packet message is approximately equal to the sum of the two separate results obtained by modeling each of them independently of the other.

We also developed a framework to evaluate the two retransmission strategies in presence of windows when errors were correlated in that back-to-back errors were more likely. The performance measures we considered were the expected time to complete an N-packet transfer and its standard deviation. We modeled the congestion dependent errors with a function $p(\alpha)$. The choice of retransmission strategy will depend on this function. We tried some alternative functions for $p(\alpha)$ to show how the two protocols compared. In particular, we saw that, irrespective of retransmission strategy, the expected time as well as the standard deviation of the time to transmit N packets increased sharply if the window size were large in the face of heavy congestion. This was the network equivalent of thrashing. We also showed the relative merits of the two retransmission strategies in this case. If the congestion level was low, (cf. a_1 small in Section 6), the two retransmission strategies performed similarly. Under heavy congestion, it all depended on the value of the probability of back-to-back errors, which we accounted for by the parameter b_1. If back-to-back error probability was high, (i.e. b_1 large), go-back-n performed better than selective repeat. For lower values of b_1, however, selective repeat was found to be better. And in both cases, the degradation due to large windows was much more pronounced, suggesting that flow control, and not retransmission strategy, is really the important issue under congestion.

We have however, not addressed other possible error characteristics, like for instance, systematic errors. Consider for example a faster sender overflowing a slower receiver's buffers in a 'systematic way', like making it drop every i^{th} packet. Selective repeat will certainly perform better in this case. However, if this is a regular occurrence, one should probably look more closely at the flow control algorithm than the error control one.

Determining the congestion function $p(\alpha)$ is at the moment an open problem. It will probably depend on details of the system architecture like the number of buffers at each point in transit, the timing characteristics of incoming and outgoing links, the background traffic, etc.

The performance of flow control strategies also needs to be investigated. We are currently investigating the performance of dynamic window flow control strategies and hope to report the results soon.

Acknowledgments

Mary Vernon made her Petri Net tools available to us, we are very grateful to her for that. We also wish to thank Vikram Adve, George Bier and Scott Leutenegger for numerous helpful discussions.

References.

[MBC 84]: Marsan, A., M.G. Balbo, and G. Conte, "A Class of Generalized Stochastic Petri Nets," *ACM Trans. on Computer Systems,* vol 2, May 1984, pp 93-122.

[BPU 88]: Bolot, J.C., B.D. Plateau and A. Udaya Shankar, "Performance analysis of transport protocols over congestive channels," Tech Report UMIACS-TR-88-22.1 CS-TR-2004.1, Department of Computer Science, University of Maryland, Revised August 1988.

[BRT 88]: Blake, J.T., A.L. Reibman and K.S. Trivedi, "Sensitivity Analysis of Reliability and Performability Measures for Multiprocessor Systems," *Proceedings of the ACM Sigmetrics 1988,* pp 177-186.

[Jac 88]: Jacobson, V., "Congestion Avoidance and Control," *Proceedings of the ACM Sigcomm 1988,* pp 314-329.

[Mor 88]: Morgan, S., "Window Flow Control on a Trunked Byte-Stream Virtual Circuit," *IEEE Trans. Comm.,* July 1988.

[MLS 89]: Mukherjee, A., L.H. Landweber and J.C. Strikwerda, "Evaluation of Retransmission Strategies in a Local Area Network Environment," *Proceedings of the ACM Sigmetrics 1989/Performance 1989,* pp 98-107.

[RJ 88]: Ramakrishnan, K.K., and R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer," *Proceedings of the ACM Sigcomm 1988,* pp 303-313.

[Tan 81]: Tanenbaum, A.S., *Computer Networks,* Prentice-Hall Inc., Englewood Cliffs, NJ 07632, 1981.

[TW 79]: Towsley, D. and J.K. Wolf, "On the Statistical Analysis of Queue Lengths and Waiting Times for Statistical Multiplexors with ARQ Retransmission Schemes," *IEEE Trans. Commun,* vol COM-27 April 1979, pp 693-702.

[Tri 82]: Trivedi, K.S., *Probability and Statistics with Reliability, Queuing, and Computer Science Applications,* Prentice-Hall Inc., Englewood Cliffs, NJ 07632, 1982.

[Zwa 85]: Zwaenepoel, W., "Protocols for Large Data Transfers on Local Networks, *Proceedings of the ACM Sigcomm 1985,* pp 22-32.

Appendix A

We are interested in the variance of the time to absorption for a Continuous Time Markov Process, which starts off in a designated state i. Let

R_i = the time to absorption given we are in state i

t_i = sojourn time in state i

$H_i(s)$ = Laplace transform of t_i

$F_i(s)$ = Laplace transform of R_i

B = set of non-absorbing states

p_{ij} = Probability of going from state i to j in one step in the corresponding discrete chain.

Then, by the Markov property

$$F_i(s) = H_i(s)\left[\sum_{j\in B-i} p_{ij}F_j(s)\right] \tag{A.1}$$

Taking the natural logarithm of both sides, we have

$$\ln F_i(s) = \ln\left(H_i(s)\right) + \ln\left[\sum_{j\in B-i} p_{ij}F_j(s)\right] \tag{A.2}$$

Differentiating equation A.2 and setting $s = 0$, we get,

$$E[R_i] = E[t_i] + \sum_{j\in B-i} p_{ij}E[R_j] \tag{A.3}$$

Differentiating equation A.2 a second time, setting $s = 0$, we get, after some algebra:

$$\begin{aligned} Var(R_i) = Var(T_i) &+ \sum_{j\in B-i} p_{ij}Var(j) \\ &+ \sum_{j\in B-i} p_{ij}(E[R_j])^2 - (E[R_i] - E[t_i])^2 \end{aligned} \tag{A.4}$$

which is the desired solution for the variance.

We also note that if i_0 is the initial state, then $E[R_{i_0}]$ gives the expected time to absorption for the Markov process of interest. This can be readily generalized if the initial distribution of the initial states are available.

Appendix B

Window flow control and go-back-n when all times are deterministic

We show in this appendix 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 B.1 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 is equal to the timeout, $\tau = 6$. The total time to transmit the 12 packets is 20 slots. When the roundtrip time, $r \leq w$, the window will never close. Its performance, studied in [MLS 89], was shown to obey the orthogonality property (Equation 3.3 of the Section 3). Here, we consider the case $r > w$, i.e., the window does close. The time to transmit N packets when there are no errors is now given by (see Figure B.1)

$$T_{noErrors}(N, w, r) = \begin{cases} \lceil N/w \rceil r + N \bmod w, \\ \quad \text{if } N \bmod w \neq 0; \\ (N/w)\, r + w, \text{otherwise.} \end{cases} \qquad \text{(B.1)}$$

Figure B.2 shows the timing diagram of the same transmission when packet 2 fails the first time it is transmitted. The retransmission strategy is go-back-n. Note that it takes 26 slots which is exactly $T_{noErrors} + \tau$. This is in general true, except in special cases. For example, if packet 3 were to fail instead, the timing diagram is as shown in Figure B.3. Now the total time taken is 25 slots instead of 26! Wherein lies this anomaly?

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

$$\begin{aligned} T_{noErrors}(N, w, r) + k(\tau - 1) &\leq T_{gbn}(N, w, r, \tau, k) \\ &\leq T_{noErrors}(N, w, r) + k\tau \end{aligned} \qquad (B.2)$$

Clearly, both limits can be reached. In [MLS 89], it was shown that errors in the go-back-n protocol obeyed the negative binomial distribution:

$$Pr[k \text{ errors} | N \text{ packets}] = \binom{N + k - 1}{k} p^k q^N \qquad (B.3)$$

The expectation of this distribution is Np/q. Using this and Equations B.2 and B.3, we have

$$\begin{aligned} &E[T_{noErrors}(N, w, r)] + (Np/q)(\tau - 1) \leq \\ &E[T_{gbn}(N, w, r, \tau)] \leq E[T_{noErrors}(N, w, r)] + (Np/q)\tau \end{aligned}$$

This bounds the error in the orthogonality hypothesis. For $Np \ll 1$ and $\tau \gg 1$, we see that it is a reasonably safe approximation.

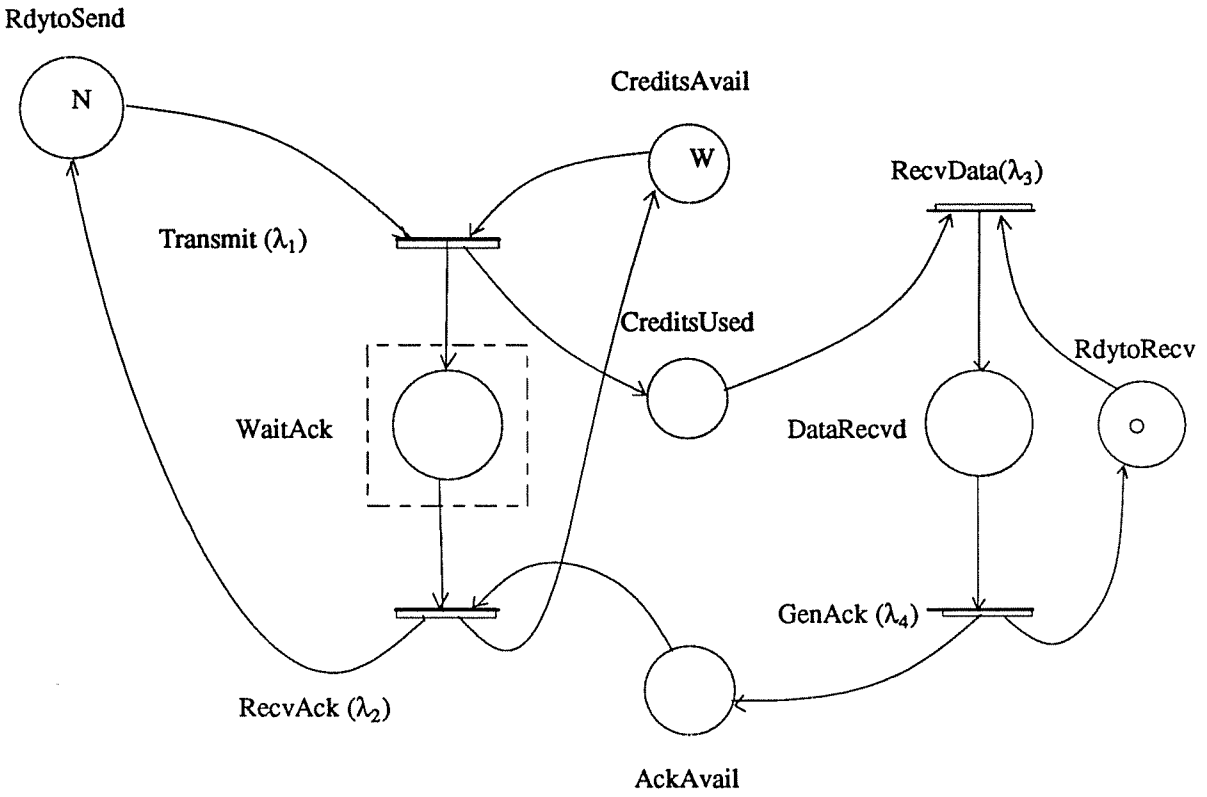

Fig 3.1: Simple sliding window flow control: Model I

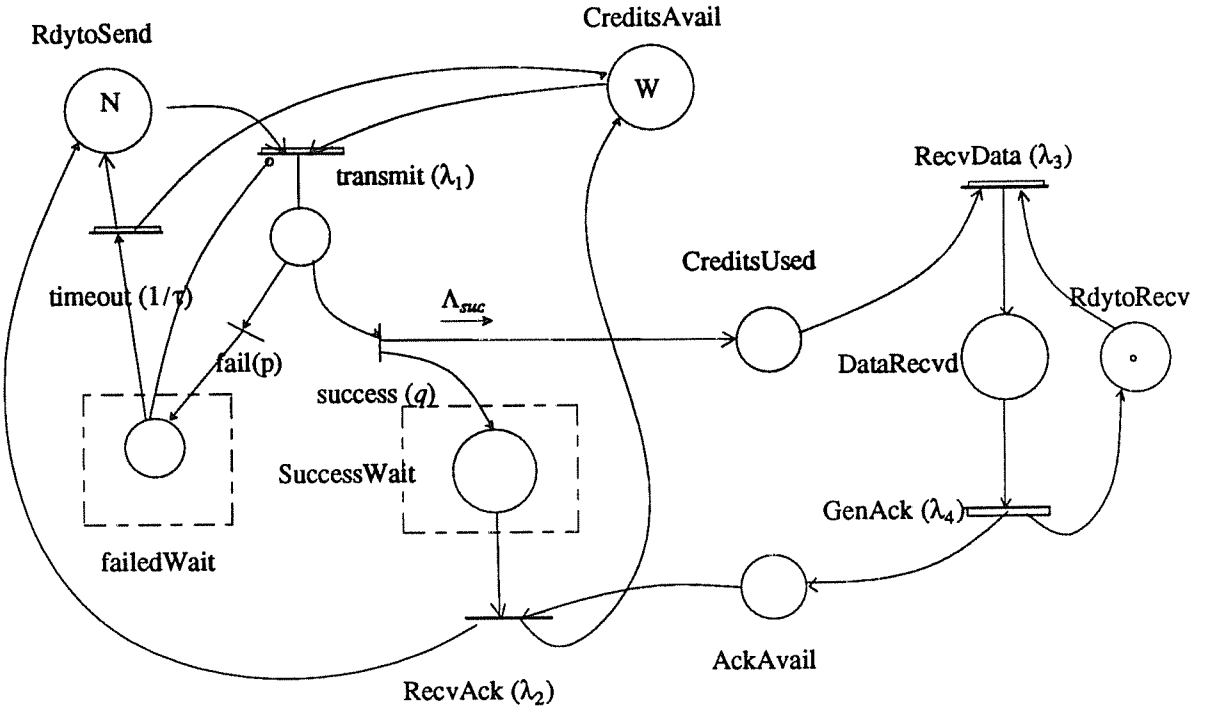

Fig 3.2: Sliding window flow control with go-back-n retransmission: Model II

Model I				Model II	
W	$P[W=0]_I$	$E[T_N]_I$	$E[T_N]_I+\frac{Np}{q}\tau$	$P[failedWait=0]$	$E[T_N]_{II}$
1	0.4450	251.9	264.9	0.951	267.7
2	0.2337	183.5	196.5	0.933	198.6
4	0.0863	155.3	168.3	0.922	170.1
8	0.0172	145.9	158.9	0.917	160.9
16	0.0009	144.2	157.2	0.916	159.5

Table 3.1: N=64, $p_0=10^{-2}$, $\tau=10$

Model I				Model II	
W	$P[W=0]_I$	$E[T_N]_I$	$E[T_N]_I+\frac{Np}{q}\tau$	$P[failedWait=0]$	$E[T_N]_{II}$
1	0.4450	251.9	381.9	0.6607	384.7
2	0.2337	183.5	313.5	0.5851	315.5
4	0.0863	155.3	285.3	0.5420	287.0
8	0.0172	145.9	275.8	0.5249	277.8
16	0.0009	144.2	274.2	0.5217	276.4

Table 3.2: N=64, $p_0=10^{-2}$, $\tau=100$

Model I				Model II	
W	$P\,[W=0]_I$	$E\,[T_N]_I$	$E\,[T_N]_I+\frac{Np}{q}\tau$	$P\,[failedWait=0]$	$E\,[T_N]_{II}$
1	0.4450	251.9	1551.4	0.163	1554.2
2	0.2337	183.5	1483.0	0.123	1485.0
4	0.0863	155.3	1454.8	0.105	1456.5
8	0.0172	145.9	1445.3	0.099	1447.3
16	0.0009	144.2	1443.7	0.098	1445.9

Table 3.3: N=64, $p_0=10^{-2}$, τ=1000

Model I				Model II	
W	$P\,[W=0]_I$	$E\,[T_N]_I$	$E\,[T_N]_I+\frac{Np}{q}\tau$	$P\,[failedWait=0]$	$E\,[T_N]_{II}$
1	0.4450	251.9	380.1	0.661	380.3
2	0.2337	183.5	311.7	0.585	311.8
4	0.0863	155.3	283.5	0.542	283.6
8	0.0172	145.9	274.1	0.524	274.2
16	0.0009	144.2	272.4	0.520	272.6

Table 3.4: N=64, $p_0=10^{-3}$, τ=1000

Model I				Model II	
W	$P\,[W=0]_I$	$E\,[T_N]_I$	$E\,[T_N]_I+\frac{Np}{q}\tau$	$P\,[failedWait=0]$	$E\,[T_N]_{II}$
1	0.4450	251.9	264.7	0.951	264.7
2	0.2337	183.5	196.3	0.934	196.3
4	0.0863	155.3	168.1	0.922	168.1
8	0.0172	145.9	158.7	0.916	158.7
16	0.0009	144.2	157.0	0.915	157.0

Table 3.5: N=64, $p_0=10^{-4}$, τ=1000

Model I				Model II	
W	$P\,[W=0]_I$	$E\,[T_N]_I$	$E\,[T_N]_I+\frac{Np}{q}\tau$	$P\,[failedWait=0]$	$E\,[T_N]_{II}$
1	0.4450	251.9	253.2	0.994	253.2
2	0.2337	183.5	184.8	0.992	184.8
4	0.0863	155.3	156.6	0.991	156.6
8	0.0172	145.9	147.1	0.991	147.1
16	0.0009	144.2	145.5	0.990	145.5

Table 3.6: N=64, $p_0=10^{-5}$, τ=1000

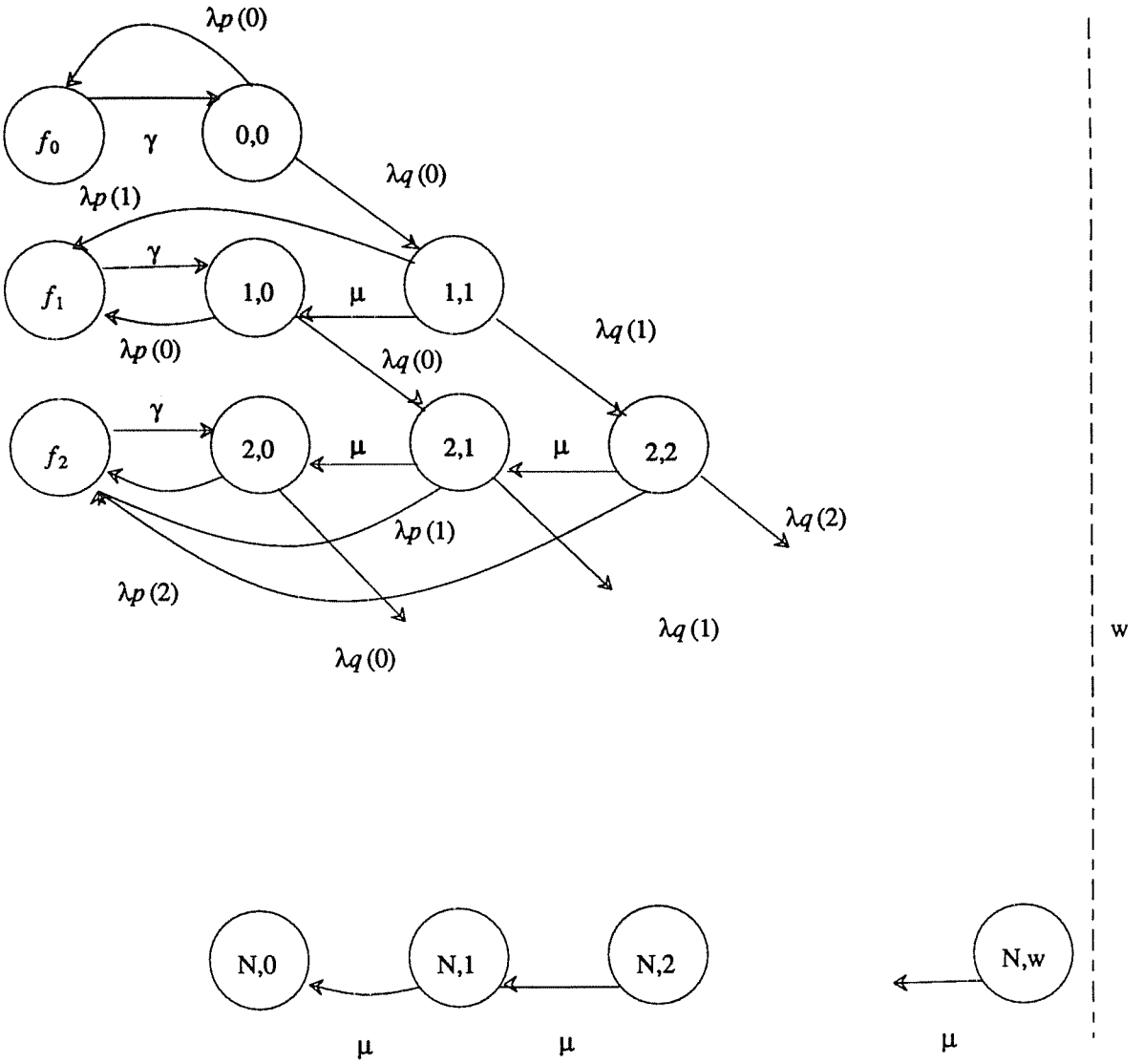

Fig 4.1: go-back-n with non-independent errors and windows

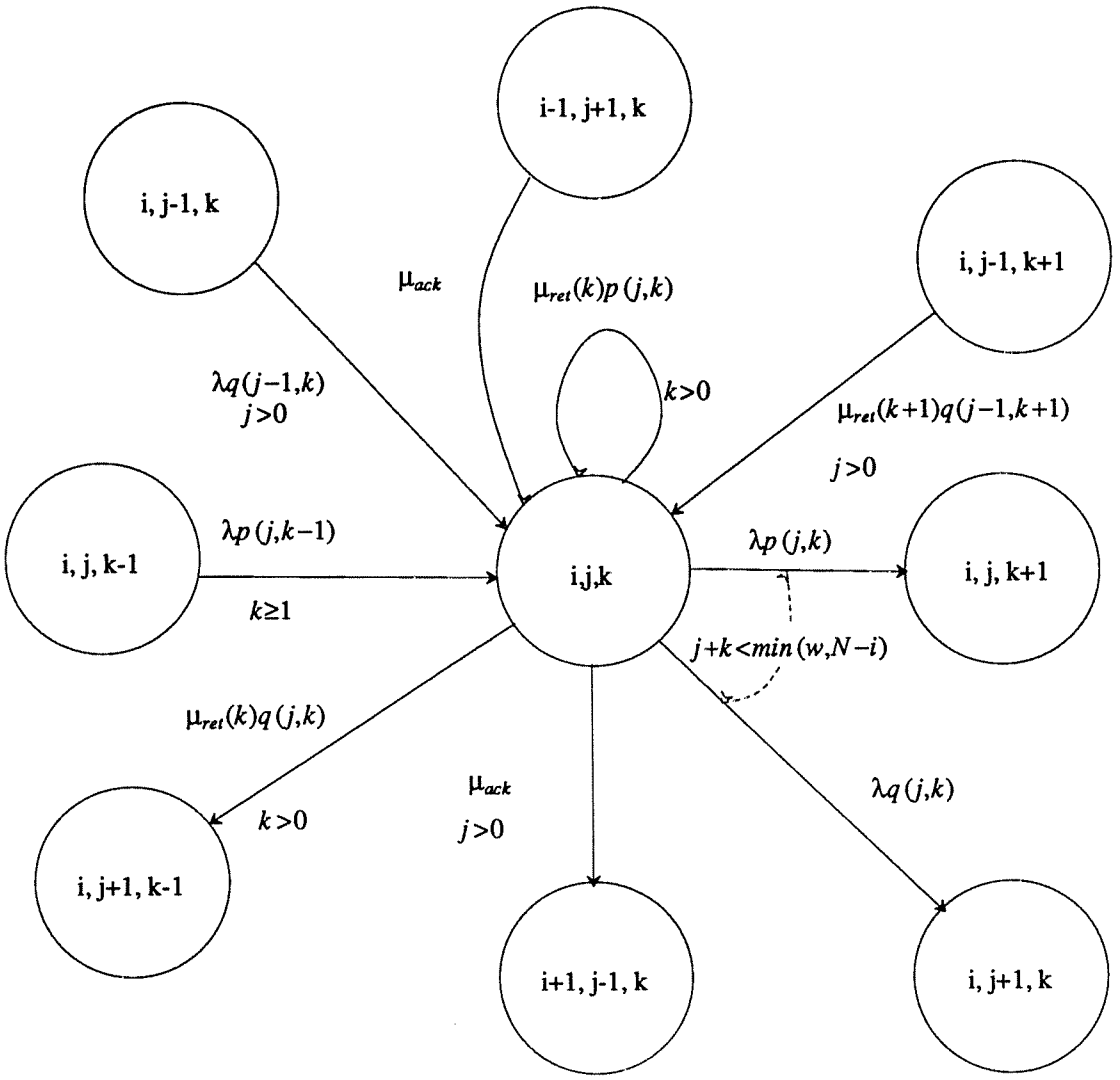

Fig 5.1: Selective Repeat state transition diagram

W	$E[T_{N,gbn}]$	$E[T_{N,sr}]$ b_1=0.0	$E[T_{N,sr}]$ b_1=0.1	$E[T_{N,sr}]$ b_1=0.5	$E[T_{N,sr}]$ b_1=0.8
1	200.8	200.8	200.8	200.8	200.8
2	191.5	191.4	191.4	191.5	191.5
4	187.1	186.9	187.0	187.0	187.1
8	293.5	293.2	293.3	293.4	293.5
16	341.9	341.5	341.6	341.8	342.0

Table 6.1: Expected time to transmit N=64 packets. $a_1=10^{-4}$, $w_{max}=20$.

W	$E[T_{N,gbn}]$	$E[T_{N,sr}]$ b_1=0.0	$E[T_{N,sr}]$ b_1=0.1	$E[T_{N,sr}]$ b_1=0.5	$E[T_{N,sr}]$ b_1=0.8
1	200.8	200.8	200.8	200.8	200.8
2	222.4	213.5	215.3	223.0	229.3
4	293.2	257.9	264.1	295.2	326.1
8	591.7	457.9	471.4	544.3	629.4
16	722.7	589.3	608.6	716.2	878.6

Table 6.2: Expected time to transmit N=64 packets. $a_1=10^{-1}$, $w_{max}=20$.

W	$\sigma[T_{N,gbn}]$	$\sigma[T_{N,sr}]$ b_1=0.0	$\sigma[T_{N,sr}]$ b_1=0.1	$\sigma[T_{N,sr}]$ b_1=0.5	$\sigma[T_{N,sr}]$ b_1=0.8
1	20.5	20.5	20.5	20.6	20.7
2	20.0	19.3	19.5	20.3	21.0
4	21.1	20.1	20.3	21.6	23.2
8	38.4	37.2	37.4	38.5	40.1
16	45.5	43.9	44.1	45.4	47.4

Table 6.3: Standard deviation of the time to transmit N=64 packets. $a_1=10^{-4}$, $w_{max}=20$.

W	$\sigma[T_{N,gbn}]$	$\sigma[T_{N,sr}]$ b_1=0.0	$\sigma[T_{N,sr}]$ b_1=0.1	$\sigma[T_{N,sr}]$ b_1=0.5	$\sigma[T_{N,sr}]$ b_1=0.8
1	20.5	20.5	20.6	20.6	20.7
2	116.4	97.3	103.0	127.1	147.1
4	212.4	166.7	176.2	224.0	271.9
8	342.1	231.6	241.5	294.2	359.6
16	380.1	259.4	267.8	314.6	417.8

Table 6.4: Standard deviation of the time to transmit N=64 packets. $a_1=10^{-1}$, $w_{max}=20$.

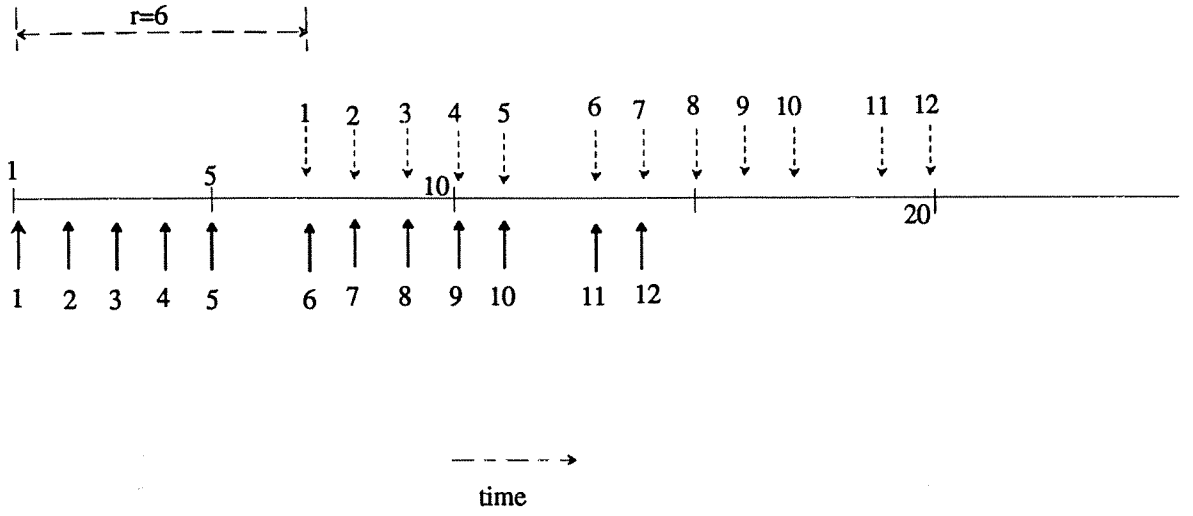

Fig B.1. Timing diagram of sliding window protocol with *deterministic* transmission times when there are *no errors*. The window size, w = 5, the roundtrip time, r = 6 and per packet transmission time is 1. Solid arrows are data, dashed arrows are ACKs. The total time to transmit N = 12 packets and receive all acknowledgments is 20.

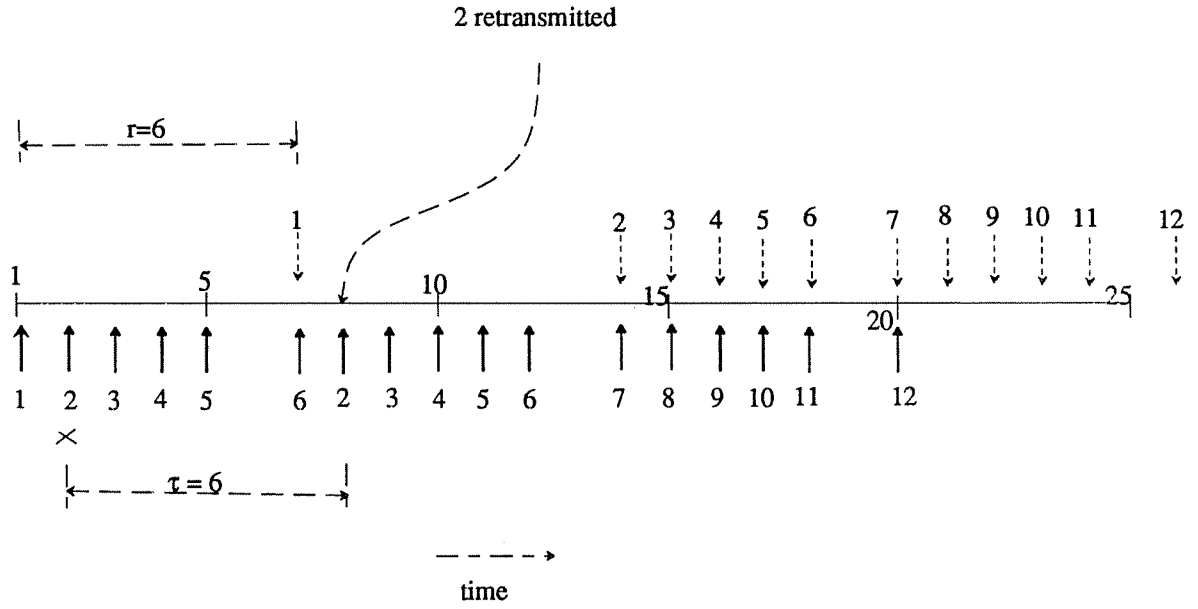

Fig B.2. The timing diagram of the same transmission when Packet 2 fails. The time to detect the error τ=6. Note that the total time to transmit 12 packets successfully now is $26 = 20 + \tau$.

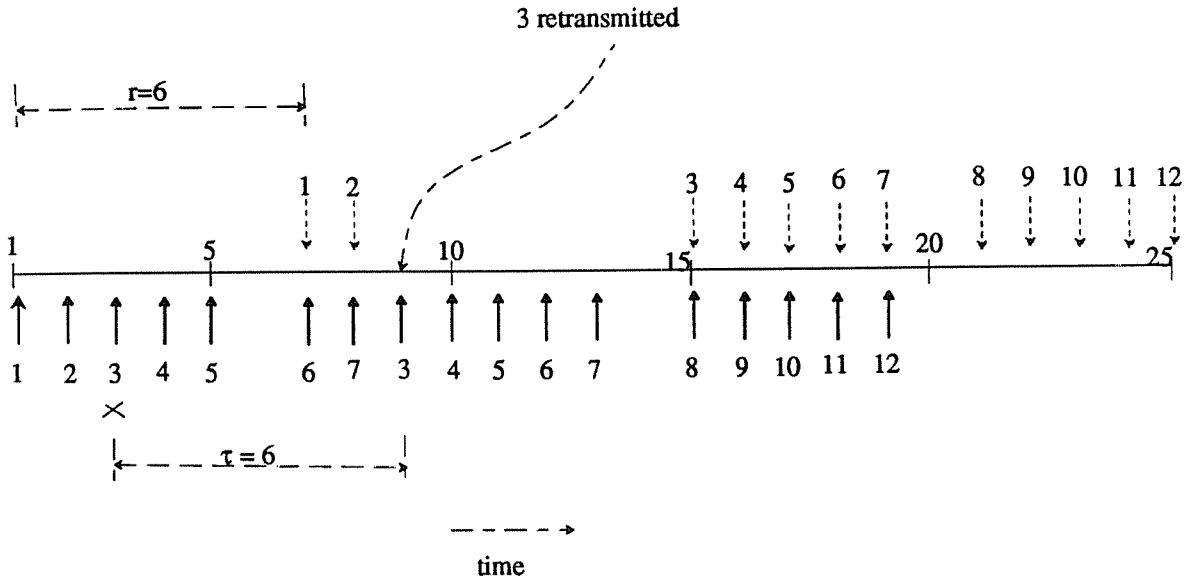

Fig B.3. The timing diagram of the above transmission when Packet 3 fails. The time to detect the error is the same as in Figure 4.2. Unlike Figure 4.2, the total time to transmit the 12 packets is now 25 and not 26. This is because the *remaining* number of packets at the begining of the retransmission of Packet 3 is 10, and 10 mod w = 0. This causes one slot to be 'gained' from 26.