STACK ALGORITHMS FOR MULTIACCESS NETWORKS WITH ASYMMETRIC FEEDBACK

James F. Kurose, Atul Shrivastava, Don Towsley

COINS Technical Report 87-135

Stack Algorithms for Multiaccess Networks with Asymmetric Feedback[1]

JAMES F. KUROSE,[2] ATUL SHRIVASTAVA,[3] DON TOWSLEY[2]

University of Massachusetts
Amherst, Mass. 01003

Abstract

We examine the operation of stack splitting random access protocols in multiaccess networks in which individual stations may receive *asymmetric* feedback from the channel, i.e., different stations may observe different, possibly erroneous, outcomes on the channel. We propose several possible modifications to the basic stack algorithm for such environments and quantitatively examine the performance of the various alternatives. An approximate Markov chain model is developed to analytically study the time delay versus throughput performance of the various alternatives and the analytic results are validated through simulation. Representative performance results are given for the alternative stack algorithms.

[1]This work was supported in part by the National Science Foundation Grant ECS83-01771, the Office of Naval Research grant N00014-87-K-0304, and an IBM Faculty Development Award.

[2]Department of Computer and Information Science

[3]The author was with the Department of Electrical and Computer Engineering. This report is the MS thesis of this author.

Table of Contents

List of Tables

List of Figures

Chapter I

Introduction

The multiple access problem occurs whenever a geographically distributed group of users (stations) must share access to a single broadcast channel. Whenever two or more stations access the channel (transmit) at the same time, a receiver cannot distinguish any of the transmitted messages. In such a case, a collision is said to have taken place on the channel, and the collided messages must be retransmitted at a later time. Since a collision results in rescheduling the transmission of the collided messages for sometime in the future, there is a possible delay between message arrival and its successful transmission. Thus a channel access strategy is needed to

1. Guarantee successful transmission of each message generated by users in the system.

2. Make efficient use of the channel, e.g. minimize the delay experienced by each message.

The channel access strategy is also referred to as the access *protocol* or *algorithm* (also collision resolution algorithm), since it controls the way in which each user is allowed to access the channel. Various protocols have been suggested to make efficient use of the channel [13,7,1,16,14].

Most of the protocols mentioned above assume that all stations receive *perfect* feedback from the channel. That is, each station correctly observes the outcome

(i.e. zero, one or more than one message transmission) on the channel. Such an assumption may be unrealistic in a channel where, for example, an occasional empty period may be interpreted as a collision due to electrical interference in the channel or due to the presence of jamming, signal fading and capture. Although some recent works have examined the effects of different kinds of errors on these access algorithms [15,18,5,13,17,4,9], the problem is not yet fully understood [7].

A class of highly efficient multiaccess algorithms known as the tree and the stack algorithms [7,3,16] have generated a great deal of interest in the last few years. In this thesis, we look at the performance of stack algorithms in the presence of channel errors. With the brief mention in [5], previous studies addressing the problem have all assumed that when such an error does occur, *all* stations observe identical (incorrect) feedback from the channel. This work looks at the case where different stations may observe different feedback from the channel. We refer to these type of channel errors as *asymmetric* errors (since different stations may interpret the actual outcome on the channel differently).

The remainder of the thesis is structured as follows: In Chapter II, we look at the multiple access problem in detail. We also review the operation of the stack algorithm (an access protocol proposed by Tsybakov *et al.*[16]) and discuss related work. In Chapter III we present our model of the imperfect multiple access environment and give various modifications of the basic stack algorithm to transmit messages in such an environment. Chapter IV presents an approximate Markov chain model to analyze the time delay versus throughput of one specific version of the modified stack algorithm. In Chapter V, we prove results to show that the approximations used in constructing the model are appropriate. The delay versus throughput characteristics for the other versions of the modified algorithm

are presented in Chapter VI. Lastly, in Chapter VII we conclude this paper and address open problems which arise as a consequence of this work.

Chapter II

The Stack Algorithm And A Literature Review

Two possible approaches can be taken to solve the the multiple access problem described in Chapter I:

- *Controlled Access*, where the access strategy predetermines the rights to the channel at any given time such that the access to the channel is collision-free e.g. TDMA, FDMA etc.
- *Random Access*, where the access strategy is called upon to control the access to the channel whenever a conflict occurs e.g. ALOHA, CSMA Protocols [1,10]. Also known as the contention-based protocols, they operate by partitioning the stations in the network into a set of *enabled* stations (who have the right to transmit) and *disabled* stations (without transmission rights). The basis of partitioning determines the efficiency of each protocol.

In this chapter we review past work that has been done in the area of random access communication. Since the field of random access communication is so broad, we restrict ourselves to the past work in Tree and Stack based algorithms [16,3,7], and various error models proposed and analysed for this class of algorithms.

Ever since Abramson studied the ALOHA protocol in 1971 [1], much work has been done in the field of random access communications. Researchers have designed new protocols to improve the maximum throughput or delay characteristics of already existing protocols. Others have looked into relaxing assumptions upon which

previous protocols were based. Refer to [11] for a survey of the general problem of multiaccess communication and random access strategies.

In the following description of multiaccess systems, the following definitions will be useful:

Active Stations: Set of stations which have a message to send.

Enabled Set: Set of active stations which have transmission rights to the channel.

Disabled Set: Set of active stations which do not have transmission rights to the channel.

Slotted Channels: When all users in the system are synchronized (externally) such that each user knows exactly when a transmission starts and when it ends. This small period of synchronization is known as a *slot.* Such channels are referred to as Slotted channels.

Unslotted Channels: When no such external synchronization controls the beginning and end of any transmission in a system, the channels are referred to as Unslotted channels.

Conflict of multiplicity n: The number of users (n), that transmit in a time slot.

Collision Resolution Interval (CRI): The time (or number of slots) between the first collision between two or more messages and the time until all messages involved in that collision are successfully transmitted is referred to as a CRI. An n -fold CRI refers to the length of time needed to resolve an initial conflict of multiplicity n. Note that a CRI with 0 or 1 messages is degenerate of length 1 slot.

Blocked or Continuous Entry: Consider an n-fold CRI, ($n \geq 2$), starting at time t. If stations which become active after t must wait till the end of CRI to transmit for the first time, the entry to the channel is *blocked.* If a station can transmit

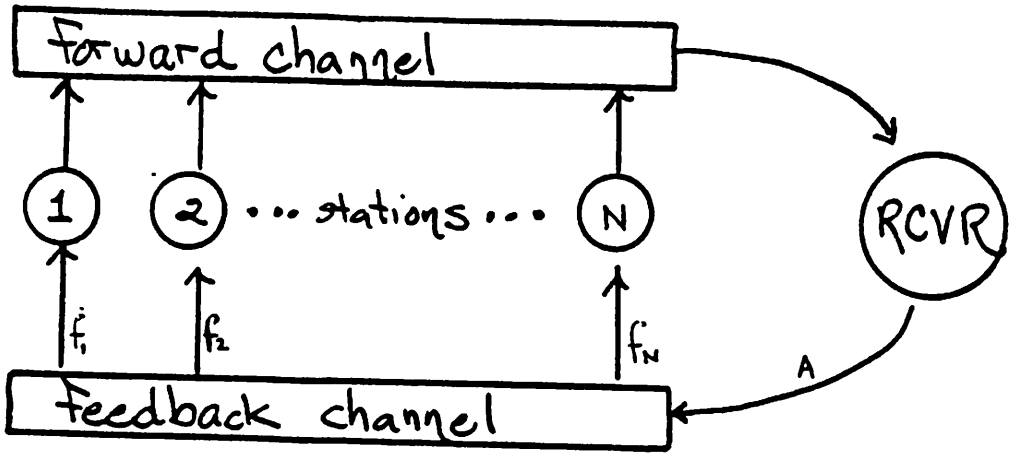

Figure 2.1: Multiple Access System

immediately after it becomes active the channel entry is *continuous.*

A multiaccess system in which all stations transmit to a common receiver can be viewed as shown in Figure 2.1. The system has a forward channel (the actual messages are transmitted on this channel) and a feedback channel (the feedback messages are relayed back on this channel). Feedback messages are transmitted by the common receiver at the end of a transmission to inform users of the outcome of a previous transmission. These two channels alongwith the stations and a common receiver completes the model for such a multiaccess system. The feedback channel is also sometimes referred to as the reverse channel.

The feedback messages received by stations at the end of each slot are:

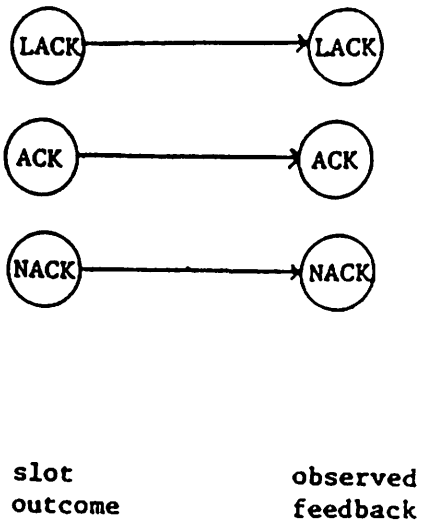

Figure 2.2: Channel Model: No Error (Perfect)

LACK when no station transmits in a time slot. (Channel *idle*). This corresponds to the first transition in Figure 2.2.

ACK when exactly one station transmits in a time slot. (Channel *success*). This corresponds to the second transition in Figure 2.2.

NACK when two or more stations transmit in a time slot. (Channel *collision*). This corresponds to the third transition in Figure 2.2.

This feedback scheme is known as *ternary* feedback since three possible feedback messages is be received by all stations in the system.

Limited or Continuous Sensing: Those multiaccess protocols which require stations to sense the channel (listen to feedback from the channel) only when they are active are known as *limited* sensing protocols. If stations must sense the channel at all times, then the protocol is a *continuous* sensing protocol. It is important to note that continuous sensing is needed when stations must track the channel history (track the end of a CRI) to determine their access rights as a part of the

collision resolution algorithm. Continuous sensing is undesirable due to the extra overhead involved when adding new stations in the system or when previously inactive stations become active.

Noiseless Channel: The channel is modeled as being 'perfect' i.e. the output from the channel is same as the input to the channel. (No Errors in either the forward or feedback channel).

Noisy Channel: This type of a channel can make idle and success slots appear as collision slots. Such an event may take place probabilistically due to electrical interference, noise etc.

Capture: In a channel with the capture property, a collision slot may appear to be a success slot due to the difference in transmission levels of the colliding signals.

Erasure: This is a property of the channel due to which a success or collision slot may be interpreted as idle.

Symmetric Feedback: When all active stations get the *same* feedback message at the end of a slot, the feedback mechanism is symmetric. If a feedback error occurs, it will thus be symmetric, i.e., the error would be introduced at point A of the channel in Figure 2.1.

Asymmetric Feedback: When each active station either gets the correct feedback message from the receiver or its feedback message is corrupted *independent* of the feedback message to other active stations, the feedback process is asymmetric, i.e., an error takes place independently on one or more f_i arrows in Figure 2.1.

2.1 An Overview of Splitting Random Access Algorithms

Capetanakis's Tree algorithm [3], was the first group random access algorithm known to have better maximal throughput over the ALOHA type protocols. This algorithm makes use of a slotted channel, has continuous sensing and blocked entry with ternary feedback. It has better delay characteristics over the previously known TDMA protocols for low arrival rates, and was stable for arrival rates up to 0.43 packets/slot [1], assuming a Poisson message arrival process to the system. The tree algorithm determines the *enabled* and *disabled* sets by considering all active stations as leaves of a binary tree. All active stations form the enabled set at the start of a CRI. After a collision, the stations in the enabled set flip a coin and probabilistically (and independently) stay in the enabled set or join the disabled set. After an idle or a success slot a new enabled set is formed by those stations which joined the disabled set after the previous collision.

In 1978, Gallager [7] proposed an algorithm in which the the message arrival time at each station is used as the basis of partitioning stations into enabled and disabled sets. In this algorithm, stations whose message arrival times fall within a time window are enabled. This algorithm is a blocked entry algorithm with continuous sensing in a slotted channel and ternary feedback and is known to have the highest throughput of all known group random access algorithms, with a maximum throughput of 0.487 packets/slot, with Poisson arrivals into the system.

Although the tree and time window protocols mentioned above are extremely efficient, they have the following undesirable properties

- Blocked entry for new arrivals during a CRI.

[1] The modified ALOHA system has a maximal throughput of 0.368 packets/slot [14]

- Continuous sensing of the channel by all stations.

A continuous entry, limited sensing algorithm was proposed by Tsybakov and Vvedenskaya [16], known as the *'Stack'* algorithm. We look at the algorithm in detail in the following discussion. We shall also consider some inherent properties of the stack algorithm in ensuing chapters.

Consider a synchronous multiaccess network environment in which all messages are of fixed size and the slot length equals the transmission time for each message. For simplicity we assume that each station has at most one message to transmit at any given time. We continue to assume that the channel is 'perfect' with the properties mentioned above. The outcome of each slot is assumed to be available immediately at the end of the slot, although the problem of delayed feedback can also be easily handled [13,12].

The algorithm operates by partitioning the active stations into groups using a recursive binary search technique. The algorithm can be easily visualized by assuming that each station i maintains a stack, in which the topmost element is labeled 0 and current stack depth at station i is given by $stack\_depth_i$. As we shall see, a station with a message to send may be thought of as having its message at level $stack\_depth_i$ and can only transmit this message when $stack\_depth_i = 0$, i.e., when its stack is of depth zero and hence its message is at the top of its stack.

If station i is inactive, i.e. it has no message to send, it need not participate in the algorithm. If a message to be transmitted is generated at station i during some time slot T_j, then $stack\_depth_i$ is set equal to zero and station i begins to execute the stack algorithm described below at the beginning of time slot T_{j+1}.

BASIC STACK ALGORITHM

Each active station, i, executes the following steps:

1. /* At the *beginning* of each slot */
   if ($stack\_depth_i = 0$) /* message on top of stack */
   then transmit the message
2. /* At the *end* of each slot */
   if ($stack\_depth_i = 0$) /* station i in *enabled* set */
   if ACK feedback then stop executing the algorithm since station i's message has just been successfully transmitted.
   if NACK feedback then with probability p
   set $stack\_depth_i = stack\_depth_i + 1$.
   otherwise leave $stack\_depth_i$ unchanged.
   else /*($stack\_depth_i \geq 1$), station i in *disabled* set */
   if LACK or ACK feedback then set $stack\_depth_i = stack\_depth_i - 1$
   if NACK set $stack\_depth_i = stack\_depth_i + 1$

Figure 2.3 shows the operation of the stack algorithm for a three node network. The state of the stack at each of the three stations is shown at the beginning of the time slots T_j through T_{j+7}, and the filled circle represents the presence of a message in a stack element. Note that after the collision during slot T_{j+1}, station 1 has probabilistically assigned its message to stack level 1 while the message at station 2 remains at the top of the stack. Also note that station 2 leaves the system after having successfully transmitted its message in slot T_{j+2}. Also , since station 2 has a new arrival during slot T_{j+4}, it joins the algorithm in the following slot.

We will later find it convenient to use the notion of a *composite* stack associated

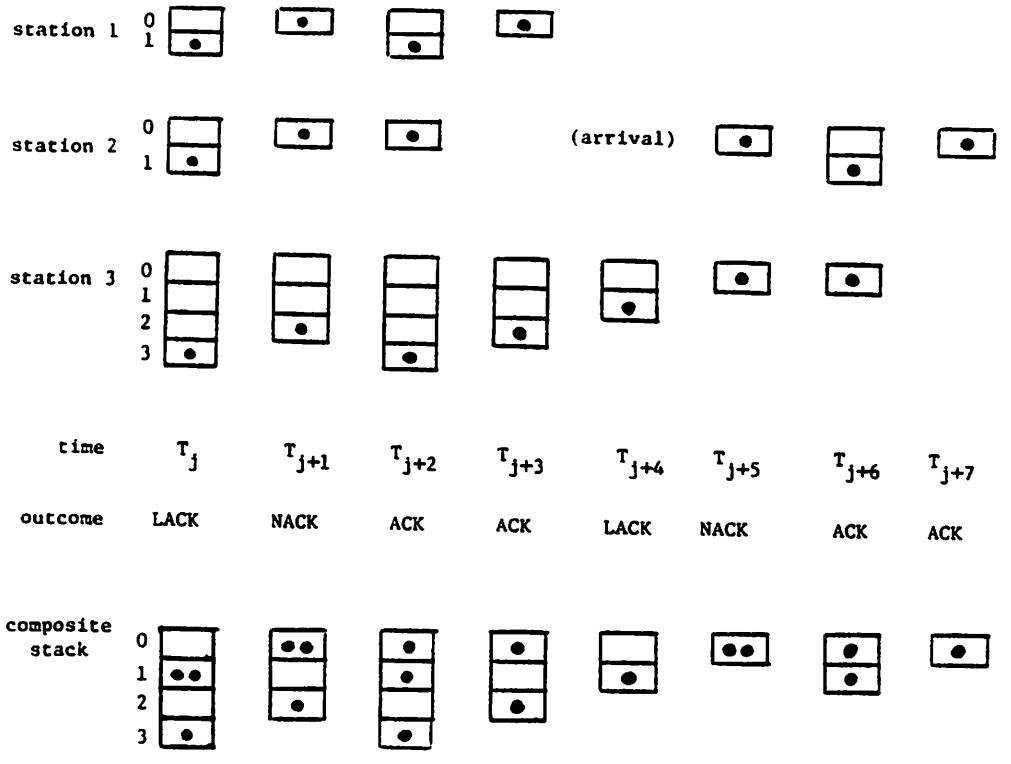

Figure 2.3: Operation of the stack algorithm

with the set of individual stacks during each time slot. By definition, the kth element of composite stack contains the messages at level k in each of the individual stacks in all stations in the network. It should be noted, however that the algorithm described above is fully decentralized, and no such composite stack actually exists in the network. However, it will be extremely useful for analysis of the stack protocol [6,16,8].

The Stack algorithm described above has a maximum throughput of 0.36017 packets/slot with a Poisson arrival process [6]. In addition, its simple state information (just a decrement/increment counter associated with $stack\_depth_i$) makes the algorithm particularly attractive and also makes it easy to implement.

In their analysis Fayolle *et al.* [6] studied the following characteristics of the Stack Algorithm:

- The distribution of the *length of a CRI* starting with a n-fold collision.
- The *delay* (number of slots) experienced by a packet before it is successfully transmitted.
- The distribution of the *multiplicity of conflict* in a time slot.

They also noticed a high variance for packet delays due to the non-FIFO nature of the algorithm. We use above results to help prove some interesting properties of the stack algorithm in Chapter IV.

Georgiadis *et al.* [8] provide a method for throughput-delay characteristics of the stack algorithm. In their analysis, they considered the interval between those points in time when the top of the composite stack was formed and the end of the corresponding CRI. These points in time were defined as *renewal instants*. Thus

the whole time axis could be seen as a succession of of such renewal instants. We borrow their results to corroborate our analytic model in Chapter IV.

Now let us relax the assumption that channel feedback is 'perfect'. A study of different type of channel errors can be found in [13]. Massey also gives feedback error conditions in which these algorithms cannot guarantee successful transmission of every message.

In [18], the operation of the stack algorithm was studied under symmetric channel errors in a 'noisy' channel. The arrival rates for which the system was stable was studied as a function of the error probabilities.

A similar approach was used by [13,9] to analyze the tree algorithms in 'noisy' channels. These works were extended to channels with Capture and Erasure properties by Cidon and Sidi [4]. In [13,9,4] the effect of channel errors on protocol throughput was derived. Results show that the maximum arrival rates for which these algorithms were stable, decrease proportionally as the rate of the noise and erasure errors increase, but the algorithms are stable for higher arrival rates in presence of capture. All of the above studies are done for *symmetric* channel errors only.

Except for a brief mention of the problem of *asymmetric* feedback in [5], all previous studies of channel errors have been restricted to the case of symmetric feedback errors, in which all active stations observe the same (incorrect) feedback.

In the remaining chapters, we address the problem of asymmetric feedback in multiaccess channels, and investigate how the basic stack algorithm can be modified to transmit messages in such conditions.

Chapter III

Model Formulation And Protocols

3.1 Model Formulation

We consider a communications system serving a population of bursty users which communicate over a multi-access broadcast channel. The system is specified as follows:

1. Each station in the system transmits messages of fixed length.
2. The channel is slotted, where slot length equals the transmission time of a message.
3. The system serves a large (essentially infinite) population in which the number of users active at any time need not be constant.
4. Each station has at most one message to transmit at any given time. This means that each arrival means a new station becomes active. In this sense, the terms station and message are synonymous and hence we use these terms interchangeably.
5. The overall arrival process of messages (to all stations in the network) is a time invariant Poisson process with parameter λ.
6. The outcome of each slot (feedback from the channel) is available to each active station at the end of the current slot.

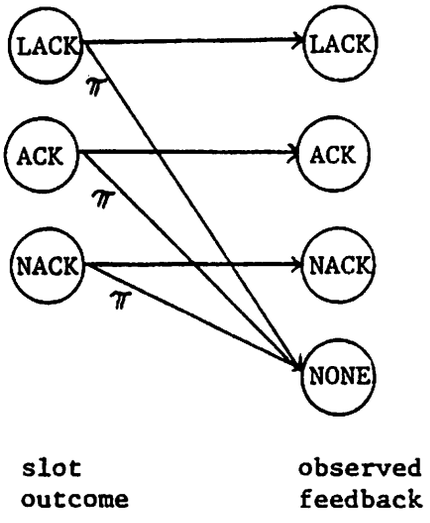

Figure 3.1: Channel Model: with Errors

7. Each active station *independently* receives one of the following feedback messages at the end of a time slot, Figure 3.1

   LACK if no station transmits in a time slot (Channel *idle*).

   ACK if one station transmits in a time slot (Channel *success*).

   NACK if two or more stations transmit in a time slot (Channel *collision*).

   NONE if the station is not able to distinguish the feedback message as any of the above.

   It is important to note that each station independently sees, either the correct outcome of a slot (LACK, ACK or NACK) or receives no feedback at all (NONE) with probability π.[1]

Different probabilities for interpreting a LACK, ACK or NACK signal as NONE signal can be easily incorporated in the model, but for simplicity we assume that these three error probabilities are identical.

[1] $\pi = 0$ reduces to 'perfect' channel case.

With this model we are now ready to handle an important feature of the feedback process: the *asymmetric* and *independent* nature of the errors taking place in the feedback channel. Since each active station sees different feedback and is not aware of the actual feedback observed by some other station, its actions are not necessarily the same from as those of other stations.

3.2 Modified Stack Algorithms

Given the error process in Figure 3.1, there are many ways in which an active station might respond when it receives a NONE feedback signal. Here we present the various options that a station can choose from when a NONE feedback signal is received. A detailed algorithmic description of the different schemes is presented below.

We name all the different algorithms by a two letter description. The first letter describes the action of stations at level 0 when they receive a NONE signal, whereas the second letter describes the action taken by a station on receiving a NONE signal when it's message is at level one or below on the stack, i.e., it has its $stack\_depth_i \geq 1$.

In algorithm PN (PersistNack), if station i transmits a message (thus its $stack\_depth_i = 0$), and observes a NONE feedback signal, it *persists* by maintaining its $stack\_depth_i = 0$ and hence retransmits in the following slot. Any other station (with $stack\ depth_i \geq 1$) on observing a NONE feedback signal, treats it as a NACK signal and thus sets $stack\_depth_i = stack\_depth_i + 1$.

SCHEME I: MODIFIED STACK ALGORITHM PN

Each active station, i, executes the following steps:

1. /* At the *beginning* of each slot */

   if ($stack\_depth_i = 0$) then transmit the message

2. /* At the *end* of each slot */

   if ($stack\_depth_i = 0$)

   if ACK feedback then stop executing the algorithm since station i's message has just been successfully transmitted.

   if NACK feedback then with probability p

   set $stack\_depth_i = stack\_depth_i + 1$.

   otherwise leave $stack\_depth_i$ unchanged.

   if NONE feedback, keep $stack\_depth_i$ unchanged.

   else /*($stack\_depth_i \geq 1$)*/

   if LACK or ACK feedback then set $stack\_depth_i = stack\_depth_i - 1$

   if NACK or NONE set $stack\_depth_i = stack\_depth_i + 1$

Several other approaches can be adopted:

- **Algorithm PL:** All stations with $stack\_depth_i = 0$ and observing a NONE signal persist. Other active stations observing a NONE signal treat it as an idle slot (LACK) (or equivalently as an ACK) and set $stack\_depth_i = stack\_depth_i - 1$.

- **Algorithm PP:** All stations observing a NONE signal persist and leave their stacks unchanged.

- **Algorithm NN:** All stations observing a NONE signal treat it as a collision (NACK). Station with $stack\_depth_i = 0$ set $stack\_depth_i = stack\_depth_i + 1$ with probability p, while other stations unconditionally increment $stack\_depth_i$ i.e. $stack\_depth_i = stack\_depth_i + 1$.

- **Algorithm NL:** All stations with $stack\_depth_i = 0$ and observing a NONE signal treat it as a collision (NACK) and hence set $stack\_depth_i = stack\_depth_i + 1$ with probability p. Other active stations observing a NONE signal treat it as an idle slot (LACK) and set $stack\_depth_i = stack\_depth_i - 1$.

- **Algorithm NP:** All stations with $stack\_depth_i = 0$ and observing a NONE signal treat it as a collision (NACK) and hence set $stack\_depth_i = stack\_depth_i + 1$ with probability p. Other active stations observing a NONE signal leave their stacks unchanged.

It may be noted that in all the schemes mentioned above, a station at the top of the stack ($stack\_depth_i = 0$) must either persist or treat the NONE signal as a NACK, since there is no guarantee that its message was successfully transmitted in the previous slot. To prevent a duplicate message being passed on to the receiver, a message sequence numbering (such as that typically used in any data link protocol) can be used.

An algorithmic description of the five schemes mentioned above can be found in Appendix A.

In Figure 3.2, we demonstrate the operation of the PN algorithm. We start out with the same scenario as in the example of the previous chapter. Thus we have three active stations in the system starting at time T_j. Note that although the event in slot T_{j+2} is a success, station 1 receives NONE as feedback and persists (as a result of PN Policy) at level 0, and retransmits its message in the following slot. In slot T_{j+6}, station 3 on receiving NONE as feedback increments its stack depth, although the slot was idle. Again the concept of a composite stack is very important to visualize the state of the system. We shall use the composite stack to develop an analytic model in Chapter IV.

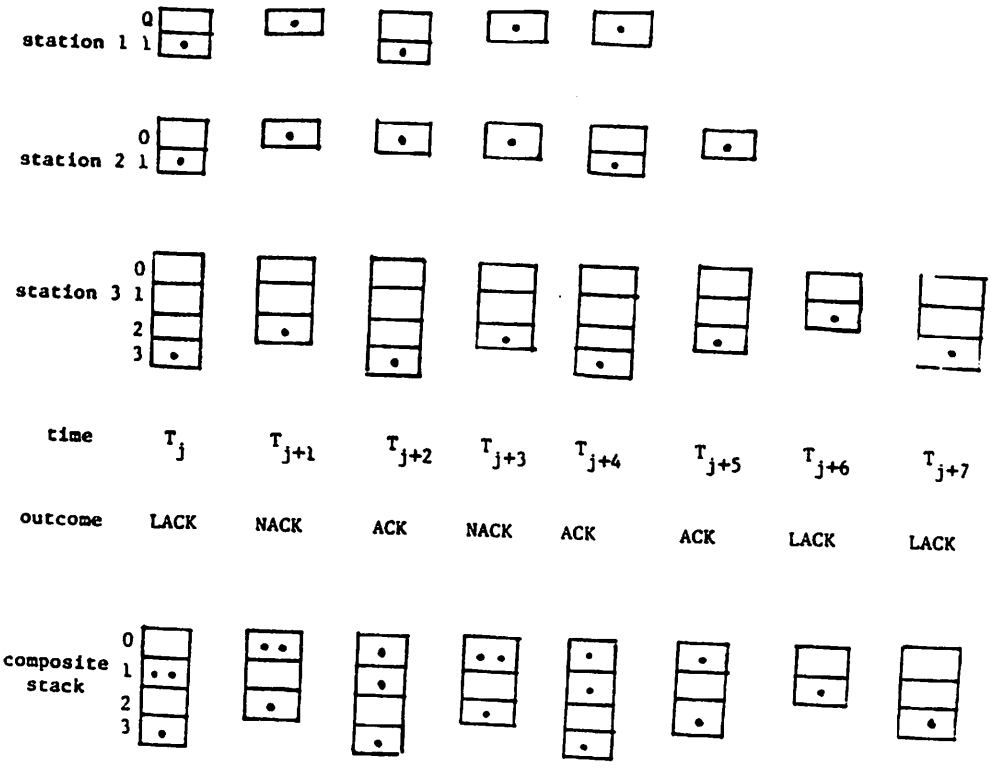

Figure 3.2: Operation of Algorithm PN

In the next chapter we develop a mathematical model to analyze the throughput-delay characteristics of the six algorithms proposed in this chapter.

Chapter IV

Approximate Analysis Of The Stack Algorithm

4.1 Assumptions

In developing an analytic model for the modified stack algorithms described in the previous chapter, we make the following additional assumptions about the multiaccess environment:

- The system is synchronous, with a slot time equal to the (fixed) message transmission time. Furthermore, all stations are synchronized in the sense that each station may transmit only at the beginning of each time slot.
- The overall arrival rate of messages to all stations in the network is a time-invariant Poisson process with parameter λ. Furthermore, we assume a large number of stations such that each station has at most one message to transmit.
- The feedback from the channel, giving the "outcome" of a slot (LACK, ACK, NACK or NONE), is available to all active users immediately at the end of the slot.

Given the assumptions above and the error model in Figure 3.1, it is clear that in order to exactly model the dynamics of the stack algorithm, the number of messages at *each* individual stack element of the *composite* stack has to be tracked, since two messages at stack depth k in the composite stack at time T_j may be in different composite stack elements at time T_{j+1} as a result of asymmetric feedback. Hence,

to track each stack level in the composite stack we need a Markov chain with the following state descriptors:

$$(y_1, y_2, \ldots, y_i, \ldots)$$

where y_i denotes the number of messages in the ith element of the composite stack. Although the solution to the system of equations resulting from the Markov chain described above is exact, the chain is infinite-dimensional. Hence the computations needed to arrive at those solutions are quite tedious and of no practical interest to us. Note that truncating the chain for a small value of i cannot guarantee accurate results for sufficiently high arrival rates, and a large value of i clearly makes the solution space more intractable. Another approach is to reduce the size of the chain by reducing the number of state variables to a fixed number, thereby making the model approximate but more tractable for the system.

In the following sections we suggest an approximate model for calculate the throughput-delay characteristics of the stack algorithm and give theoretical and simulation results on the basis of which such an approximation was made.

4.2 Approximate Model for the Stack Algorithm

Figure 4.1 shows simulation results showing the average number of messages in the stack element k, ($k = 0, \ldots, 30$) of the composite stack at the beginning of a slot following a collision slot, (without including any new arrivals during the previous slot, to be added to level 0), given that the composite stack depth is of depth 31 and given that a collision has occurred in the previous slot. Several interesting features are to be noted:

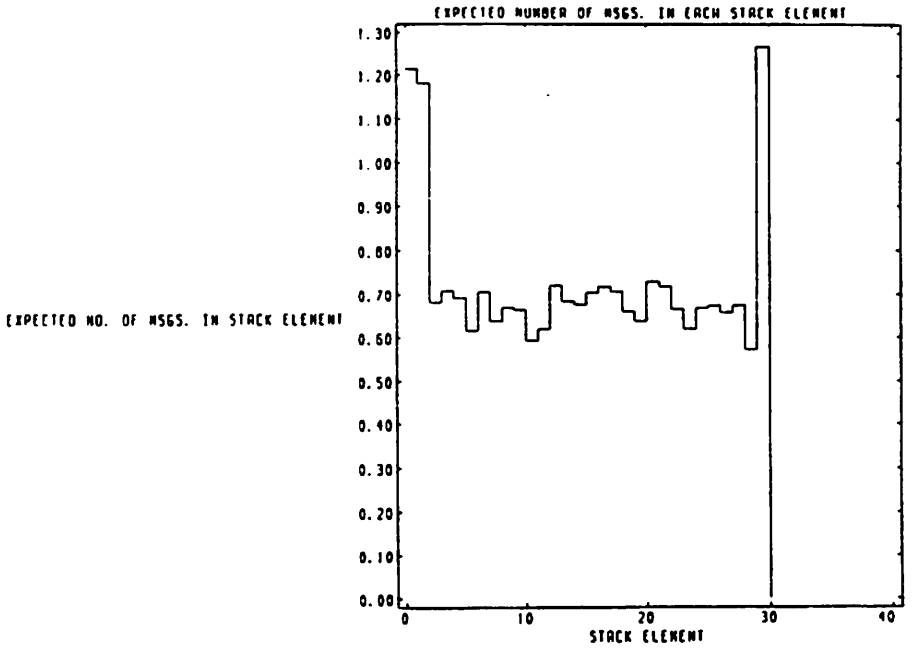

Figure 4.1: Expected occupancy of the composite stack (No Errors)

- The expected number of messages (occupancies) in stack elements 2 through 29 are essentially identical.

- The expected number of messages in the bottommost element of the composite stack is higher; because the bottom of the composite stack is tracked as a stack level with one or more packets (given the composite stack exits), and all levels below the bottommost element are assumed to have zero packets.

- The expected occupancy of the top two stack elements is higher than those in elements 2 through 29. This results from our conditioning on a collision having occurred in the last slot; so there are at least two messages within the top two composite stack elements.

Similar characteristics were noted for arbitrary composite stack depths and various arrival rates. In the case of a success or idle in the previous slot, the top and the bottommost stack elements again showed a higher expected occupancy while the remaining composite stack elements showed essentially identical expected occupan-

cies.

The above considerations suggest an approximate Markov chain model with the state variables:

$$(x_1, x_2, x_3, x_4)$$

where:

- x_1 is the number of messages in composite stack element 0.
- x_2 is the number of messages in composite stack element 1.
- x_3 is the number of messages in all remaining composite stack elements.
- x_4 is the depth of the composite stack.

and that in determining the transition probabilities we fix one message in the bottommost composite stack element and assume that the remaining $x_3 - 1$ messages are uniformly distributed in composite stack elements 2 through $x_4 - 1$.

To strengthen our claim that the stack occupancies for levels 2 through $x_4 - 1$ are similar, in the next chapter we use results from the analysis of the stack algorithm done by Fayolle *et al.* [6], and extend their work to obtain the expected stack occupancies for all stack levels of the composite stack.

In order to test the validity of our approximate model, we first use it to study the case of no channel errors [8]. This follows in the next section.

4.3 Model Verification for the No Error Case

The balance equations from the Markov chain for the case of no feedback errors, are given in Appendix B. For a given λ, these equations may be iteratively solved

to obtain the chain state probabilities, $\{p_{i,j,k,l}\}$, where $p_{i,j,k,l}$ is the probability of being in state with $(x_1 = i, x_2 = j, x_3 = k, x_4 = l)$.

We start by assuming an initial distribution for $p_{i,j,k,l}$'s (we chose to start with $p_{0,0,0,0} = 1$ and all other $p_{i,j,k,l}$'s $= 0$). The new set of $p_{i,j,k,l}$ values is computed using the $p_{i,j,k,l}$ values from the previous iteration. The algorithm terminates if the convergence criterion is satisfied, otherwise the latest values of $p_{i,j,k,l}$ are used for the next iteration.

To terminate the iterative procedure we calculate the relative difference for *each* of the p 's to be $\leq \epsilon$, i.e.,

$$\frac{\left|p^t_{i,j,k,l} - p^{t-1}_{i,j,k,l}\right|}{p^{t-1}_{i,j,k,l}} \leq \epsilon, \ \ \forall i,j,k,l$$

where $p^t_{i,j,k,l}$ denote the latest values after the t iteration of the algorithm. We choose ϵ to be 10^{-3}, so that each of the p 's is accurate to within 0.1 % standard error.

In order to solve the equations derived in Appendix B, the chain was truncated at $(3,3,20,30)$ for arrival rates upto 0.25 packets/slot and at $(5,5,35,50)$ for higher arrival rates. These values of x_i's were chosen from the stack behavior observed through simulation.

Given the probabilities $\{p_{i,j,k,l}\}$, the expected number of backlogged messages $\overline{L}$ (i.e. expected number of messages in the composite stack), can be calculated using the equation:

$$\overline{L} = \sum_{i,j,k=0}^{\infty} (i+j+k)p_{i,j,k,l} \tag{1}$$

The average delay $\overline{T}$ (in number of slots), from the time of the first transmission until a successful transmission occurs for a given arrival rate λ, can be calculated

Table IV.1: Average Delays for the No Error Case

0	Lambda	1 Delay (Model)	2 Delay (Sim)	3 Delay (UConn)
1	0.05	1.681	1.681	1.684
2	0.10	1.966	1.962	1.969
3	0.15	2.463	2.444	2.446
4	0.20	3.355	3.364	3.332
5	0.25	5.932	5.310	5.292
6	0.30	12.434	11.367	11.383
7	0.31	13.672	13.869	
8	0.32	17.930	19.148	
9	0.33	27.492	27.540	
10	0.34	43.153	43.528	

using Little's Result

$$T = L/\lambda \tag{2}$$

The actual average delay $\overline{D}$ (in number of slots), the time between a message arrival until its successful transmission is given by the equation:

$$\overline{D} = \overline{T} + 0.5 \tag{3}$$

The 0.5 term arises because each new message has to wait for half a time slot on the average, between its arrival and its first transmission.

Figure 4.2 plots the average time delay as a function of λ. The results from the model and simulation are shown in the figure. (Also refer to Table IV.1). Both these results agree closely (within 2 % error) with the numerical results in [8]. The close agreement between these results suggest that the approximate model is indeed a good one.

After corroborating our model for the no error case, we look at the delay characteristics of the PN protocol which was described in Chapter III. This follows in

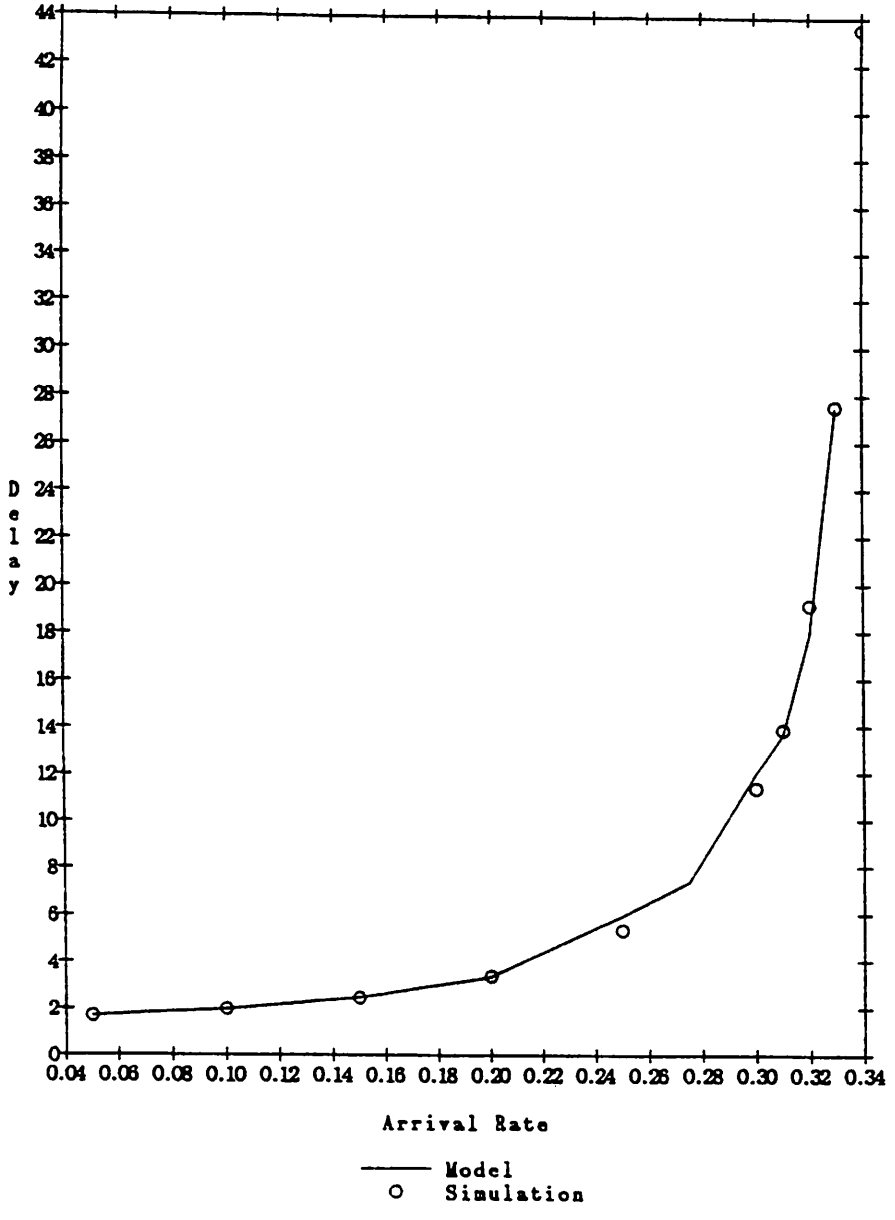

Figure 4.2: Time delay versus throughput: the no error case

the next section.

4.4 Performance Analysis of the PN Protocol

Given the four state-variable approximate model developed in the previous sections, the state transitions may be modified to obtain the balance equations for the six protocols (protocols PN, PL, PP, NN, NL, NP) in the presence of asymmetric feedback errors. The equations for the PN protocol are derived and presented in Appendix C. Note that each balance equation for the error case has more terms compared to the corresponding equation derived for the no error case in Appendix B. This is due to the fact that additional transitions are now possible out of each of the states.

Figure 4.3 and Table IV.2 show both numerical results from the model and simulation results for the PN protocol for error probabilities of π = 0, 0.01, 0.05, 0.10. The chain was again truncated as in the no error case to obtain the numerical results shown. For error rates of 5 and 10 %, the maximum throughputs observed were approximately 0.33 and 0.31 packets/slot respectively. These results compare well with the upper bound on the throughput for the non-error case of $\sim$ 0.36 packets/slot [17,6]. Also note the close correspondence between our numerical and simulation results; confirming that the approximate model we have adopted is indeed appropriate.

We have now developed a mathematical model to analyze delays for the modified stack algorithms and examined the delay characteristics for the PN protocol. In Chapter VI, we look at the delay characteristics of the remaining five protocols and compare their performance. In the following chapter, we look at the evolution of the composite stack during a CRI. This will help us calculate the expected number

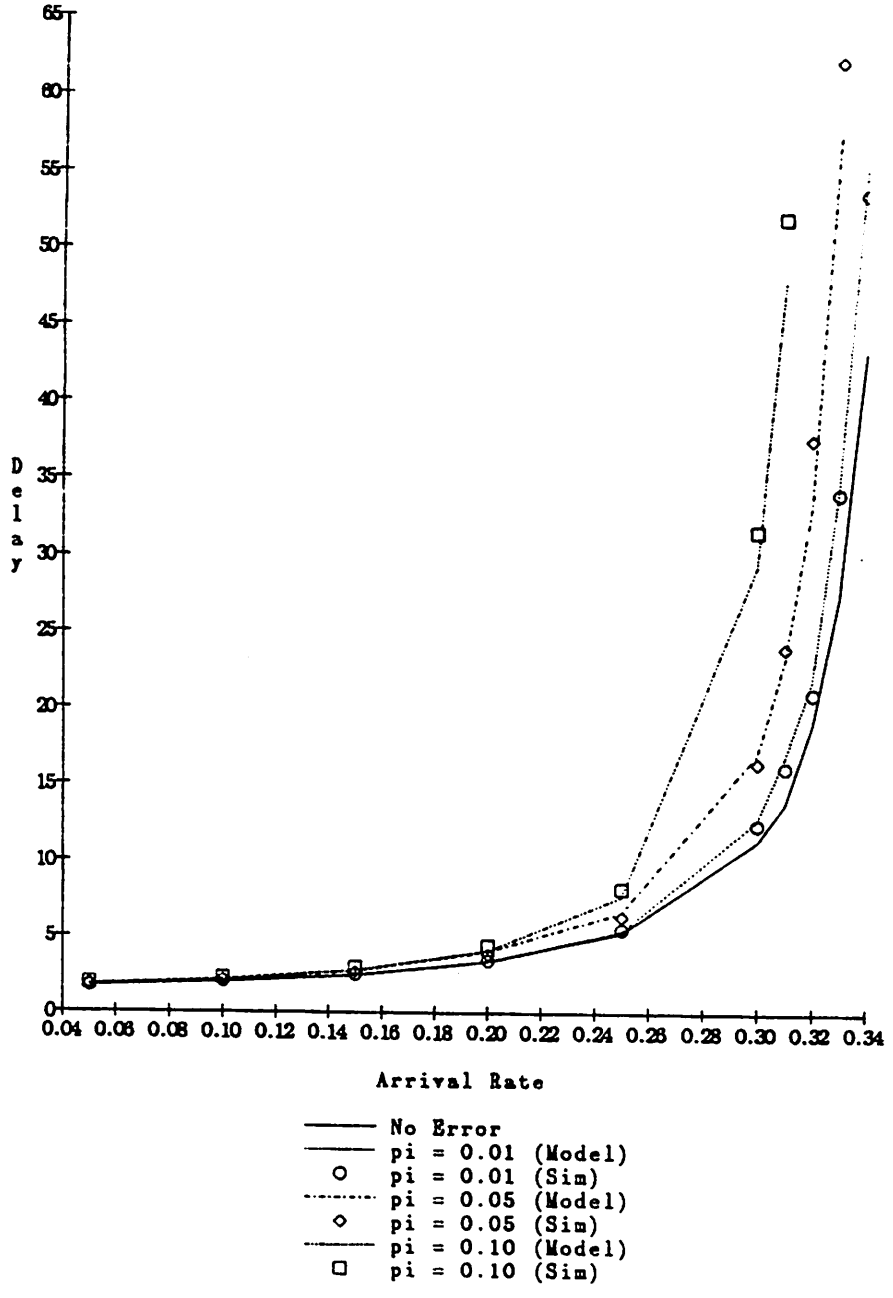

Figure 4.3: Time Delay versus throughput: The PN algorithm with errors

Table IV.2: Time Delay versus throughput: The PN algorithm with errors

0 Lambda		1 Delay 0.01 (Model)	2 Delay 0.01 (Sim)
1	0.05	1.732	1.698
2	0.10	2.055	1.992
3	0.15	2.476	2.484
4	0.20	3.356	3.427
5	0.25	5.457	5.530
6	0.30	12.785	12.453
7	0.31	16.962	16.200
8	0.32	22.043	21.069
9	0.33	34.572	34.256
10	0.34	55.659	53.852

0 Lambda		1 Delay 0.05 (Model)	2 Delay 0.05 (Sim)
1	0.05	1.7980	1.767
2	0.10	2.1500	2.099
3	0.15	2.7080	2.661
4	0.20	4.0320	3.809
5	0.25	6.6820	6.368
6	0.30	17.1040	16.489
7	0.31	23.3280	24.069
8	0.32	33.4456	37.726
9	0.33	57.6349	62.492

0 Lambda		1 Delay 0.10 (Model)	2 Delay 0.10 (Sim)
1	0.05	1.80249	1.86546
2	0.10	2.16938	2.25278
3	0.15	2.76876	2.94700
4	0.20	4.09847	4.41687
5	0.25	7.74438	8.24404
6	0.30	29.47227	31.78337
7	0.31	48.05655	52.21271
8	0.32		

of messages (occupancies) for all levels of the composite stack, needed to justify that the approximate model developed in this chapter is appropriate.

Chapter V

Basis for the approximate Markov Model

To justify the approximation for our Markov model from the last chapter (i.e., the simulation results in Figure 4.1 of Chapter IV), in this chapter we show how the expected occupancy in any intermediate level (l) (those stack levels which lie between levels 2 and stack depth of the composite stack) can be computed *analytically*, given the composite stack depth to be at least l. Our results verify that the expected stack occupancies are, in fact, nearly identical.

We again assume that the channel is perfect, i.e., the channel feedback is ternary and that all stations hear the same feedback transmitted by the common receiver at the end of each slot.

Before examining the evolution of the composite stack during a CRI, we first modify our concept of the composite stack slightly. Until now we defined the bottommost level of the composite stack to be a level with one or more packets (when the composite stack exists) and every level below the bottom of this level to have zero packets identically. In order to simplify our analysis, we now let the depth of the composite stack to unconditionally change at the end of each time slot i.e., increment after a collision slot and decrement after an idle or a success slot. Note that stack depth remains zero after a time slot in which no composite stack exists at the beginning of a slot (i.e. no backlogged messages at the beginning of a slot) and the outcome of the time slot is an idle or a success.

We derive the expected occupancies by looking at the distribution of the length of a CRI and the distribution of number of messages in each stack level in a time slot. In the next section we look at the length of CRI's and the distribution associated with the number of messages at each stack level. In Section 5.2 we derive the equations needed to compute the expected stack occupancies.

5.1 Properties of a CRI

Consider the system to start out with no backlog of packets at time t and let a transmission of multiplicity n occur in the following time slot. For $n = 0$ and 1, a degenerate CRI of length 1 follows. Consider the first slot with $n \geq 2$, and the CRI associated with that initial collision.

The resultant CRI can be visualized as shown in Figure 5.1. The CRI starts with n packets colliding in the first slot. On receiving the feedback signal some $j, (0 \leq j \leq n)$, of the colliding packets stay at level 0, and $n - j$ packets move to level 1 of the composite stack, as a result of the randomization process (step 2 with NACK feedback signal, of the basic stack algorithm in Chapter II). Thus the n -fold CRI is composed of an initial collision of n packets (taking one slot time) and two subsequent sub-CRI's with starting multiplicity $j + x$ and $n - j + y$ (x and y are the new arrivals in the slots just before the start of the two sub-CRI's).

We begin our analysis by examining the statistical properties of various random variables associated with the collision resolution mechanism.

Define:

$a(x)$: The probability of $x, (x \geq 0)$ arrivals in a time slot. Since the arrival process is Poisson with rate λ packets/slot:

$$a(x) = \frac{e^{-\lambda}\lambda^x}{x!}$$

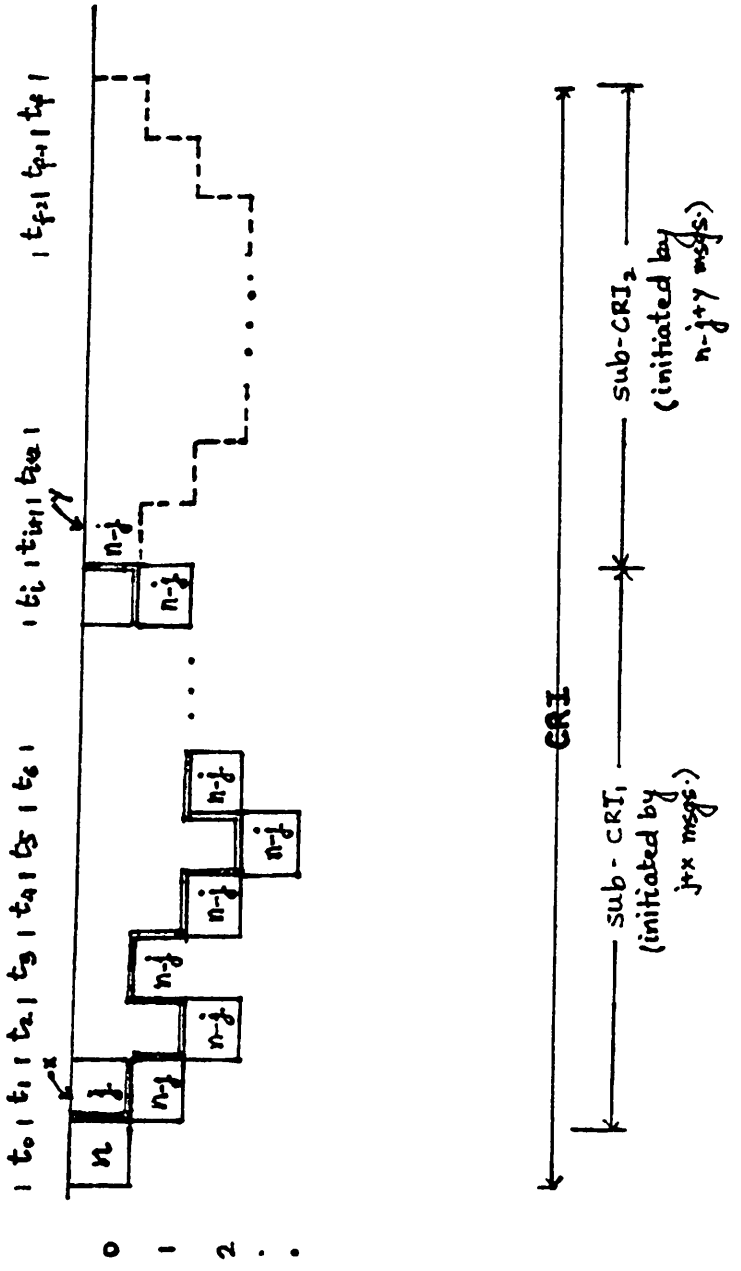

Figure 5.1: The decomposition of a CRI

l_n : Expected length of a CRI (number of slots) needed to resolve an initial n-fold collision. Note that the CRI ends with possibly more than n successful transmissions because of the continuous entry property of the stack algorithm.

σ_n^l : Expected number of slots with composite stack depth of exactly l, during a CRI starting with n packets.

$\eta_n^l(k)$: Expected number of slots with exactly $k, (k \geq 0)$, packets at level $l, (l \geq 0)$, during a CRI which begins with a conflict of multiplicity n.

From the above definition of l_n, we have the following recurrence relations:

$$l_0 = 1$$

$$l_1 = 1$$

and for $n \geq 2$,

$$l_n = 1 + \sum_{j=0}^{n} \binom{n}{j} (1/2)^n \left(\sum_{x}^{\infty} a(x) l_{j+x} + \sum_{y}^{\infty} a(y) l_{n-j+y}\right) \qquad (1)$$

The above equations for l_n were first derived in [6]; we discuss in detail here because of the structural similarities between the equations for l_n ,σ_n^l and $\eta_n^l(k)$. The n -fold CRI consists of an initial collision slot followed by two sub-CRI's starting with $j+x$ and $n-j+y$ packets respectively. The binomial term represents the probability associated with the splitting of n packets in the first collision slot into exactly j and $n-j$ packets in levels 0 and 1 respectively. $a(x)$ and $a(y)$ represents the Poisson probabilities of x and y arrivals. The equation reflects the fact that an initial n -fold CRI is comprised of an initial collision slot (taking 1 slot), followed by two CRI's with same distribution of lengths as the initial CRI, with multiplicities of $j+x$ and $n-j+y$ packets respectively.

Similarly, we derive relations for σ_n^l ($\delta_{k,n}$ denotes the Kronecker delta function, where $\delta_{k,n} = 1$, when $k = n$ and 0 otherwise)

$$\begin{aligned}
\sigma_0^l &= \delta_{l,0} \\
\sigma_1^l &= \delta_{l,0} \\
\sigma_n^l &= 0, \quad \forall l < 0 \\
\sigma_n^l &= \delta_{l,0} + \sum_{j=0}^{n} \binom{n}{j} (1/2)^n (\sum_{x=0}^{\infty} a(x)\sigma_{j+x}^{l-1} + \sum_{y=0}^{\infty} a(y)\sigma_{n-j+y}^{l}), (n \geq 2) \qquad (2)
\end{aligned}$$

The derivation for σ_n^l is similar, but not identical to, the derivation of l_n:
The expected number of slots with composite stack depth l, during a CRI starting with n-fold collision, is composed of a single slot of depth zero followed by two sub-CRI's. The first sub-CRI starts out with $j + x$ packets on the top of the stack and $n - j$ packets at level 1. Note that all collisions in level 0 of the composite stack will be successfully resolved before collisions in level 1. This is true for any levels l and $l + 1$ of the composite stack.

During the first sub-CRI, the expected number of slots with composite stack depth of $l - 1$ (momentarily ignoring the $n - j$ packets at the bottom of the stack) in fact has exactly $n - j$ packets at level l. Hence the expected number of slots with composite stack of depth l, during the resolution of the sub-CRI starting with $j + x$ packets is by definition, σ_{j+x}^{l-1}. The second sub-CRI starts out with $n - j + y$ packets and the term σ_{n-j+y}^{n} accounts for the slots with stack depth l during that sub-CRI.

We now derive recurrence relations for $\eta_n^l(k)$ for all stack levels l:

$$\begin{aligned}
\eta_0^0(k) &= \delta_{k,0} \\
\eta_1^0(k) &= \delta_{k,1} \\
\eta_n^0(k) &= \delta_{k,n} +
\end{aligned}$$

$$\sum_{j=0}^{n} \binom{n}{j} (1/2)^n (\sum_{x=0}^{\infty} a(x)\, \eta_{j+x}^0(k) + \sum_{y=0}^{\infty} a(y)\, \eta_{n-j+y}^0(k)),\ (n \geq 2) \qquad (3)$$

This relation for $\eta_n^0(k)$ first appeared in [6], can be explained as follows:
The expected number of slots with k packets at the top of the stack during a CRI starting with a n fold collision comprises of a single slot with exactly $k = n$ packets, (hence the $\delta_{k,n}$ term) and the expected number of slots with k packets at the top of the stack in the two sub-CRI's of multiplicity $j + x$ and $n - j + y$ packets.

For $\eta_n^1(k)$ we have:

$$\eta_0^1(k) = \delta_{k,0}$$

$$\eta_1^1(k) = \delta_{k,0}$$

$$\eta_n^1(k) = \delta_{k,0} + (1/2)^n\,(a(0) + a(1))(\delta_{k,n} + \sum_{x=0}^{\infty} a(x)\, \eta_{n+x}^1(k)) +$$

$$n(1/2)^n\, a(0)(\delta_{k,n-1} + \sum_{x=0}^{\infty} a(x)\, \eta_{n-1+y}^1(k)) +$$

$$\sum_{j=2}^{n} \binom{n}{j} (1/2)^n (\sum_{x=0}^{\infty} a(x)(\eta_{j+x}^1(k) + \sigma_{j+x}^0(\delta_{k,n-j} - \delta_{k,0})) +$$

$$\sum_{y=0}^{\infty} a(y)\, \eta_{n-j+y}^1(k)),\ (n \geq 2) \qquad (4)$$

The delta term in the equation for $\eta_n^1(k)$ accounts for the first time slot in the CRI (when level 1 does not exist) and hence the number of messages in level 1 is explicitly 0. The second term represents the event when all the n packets from the initial collision move to level 1 in the following slot and there is an idle or success in that slot. The third term is the event when $n - 1$ of the n packets from the initial collision move to level 1 and there is no arrival in the following slot, thus the event in the next slot is a success. The final term in equation (4) arises from four factors. The initial n colliding messages split into two groups of j and $n - j$ as shown in Figure 5.1. Ignoring the $n - j$ messages in level 1 momentarily, we note

that the expected number of slots with k messages in element 1 is $\eta^1_{j+x}(k)$ for the first sub-CRI. Also note that during the first sub-CRI, whenever the depth of the stack (ignoring the bottommost $n-j$) is zero (as in the slot t_1 in Figure 5.1), the first stack element will in fact have $n-j$ messages. Since the σ^0_{j+x} represents the expected number of such slots during the sub-CRI, $\delta_{k,n-j}\ \sigma^0_{j+x}$ gives the expected number of slots in which there are exactly $n-j$ in level 1 during the first sub-CRI with stack of depth exactly 1. The term $\delta_{k,0}\,\sigma^0_{n+x}$ arises from related considerations; since the $\eta^1_{j+x}(k)$ term counts level 1 stack occupancies ignoring the $n-j$ packets starting out in level 1 (refer Figure 5.1), each time the stack (ignoring the $n-j$ packets) was of depth 0, the occupancy of level 1 was counted as level 0. The term σ^0_{j+x} gives the expected number of slots in which such an event occurred, and thus $\sigma^0_{j+x}\ \delta_{k,0}$ is subtracted to get the correct slot count. The final term in the equation is due to the occupancy of level 1 during the second sub-CRI, which begins with $n-j+y$ packets on top of the stack and all other stack elements empty.

For stack levels l, where $l \geq 2$,

$$\begin{aligned} \eta^l_0(k) &= \delta_{k,0} \\ \eta^l_1(k) &= \delta_{k,0} \end{aligned}$$

$$\eta^l_n(k) = \delta_{k,0} + \sum_{j=0}^{n} \binom{n}{j} (1/2)^n \\ (\sum_{x=0}^{\infty} a(x)(\eta^l_{j+x}(k) + \sigma^{l-1}_{j+x}(\delta_{k,n-j} - \delta_{k,0}) + \sum_{y=0}^{\infty} a(y)\ \eta^l_{n-j+y}(k)), (n \geq 2) \tag{5}$$

Equation 5 can be derived as follows:

The first and the third terms in the equation for $\eta^l_n(k)$ are based on the same

reasoning described previously for equation 4. Other terms appear for the following reasons. The expected number of slots with k packets at level l of the composite stack is a composition of two sub-CRI's with multiplicity of $j+x$ and $n-j+y$ packets. We also account for the additional slots when there are exactly $n-j$ packets at level l. This happens whenever the composite stack in the previous slot is of depth $l-1$, during the resolution of the sub-CRI with $j+x$ packets.

Equations (1) and (3) are solved in [6]. Equations (2), (4) and (5) can be solved using techniques mentioned in [16,17,6]. While solving equations (1) through (5) numerically the solution space is first truncated for some value of n, which in general should depend on the arrival rate λ. We first solve for the $\{\sigma_n^l\}$ and then using those values to solve for $\{\eta_n^l(k)\}$. After solving for $\eta_n^l(k)$, the expected occupancy (number of packets) at any level l, given the composite stack has at least l deep, can be derived as described below.

5.2 Solving for the Expected Stack Occupancies

Recall that our goal is to justify our assumption of uniformly distributed stack occupancies during a CRI, i.e., we want to show that the expected occupancies for all intermediate stack levels given that those composite stack levels exist, are nearly the same.

Consider N CRI's in succession. According to our definition of a CRI, these N CRI's are non-overlapping and occupy the entire time axis from the start of the first CRI till the end of the Nth CRI.

In the derivation of unconditional stack occupancies for all stack levels, the following quantities will help establish the final result:

$Na(n)$	Expected number of CRI's that begin with a n-fold collision. The probability of a CRI starting with a n-fold collision is $a(n)$. Thus in N CRI's, the expected number of CRI's beginning with a n-fold collision is $Na(n)$.
$Na(n)\eta_n^l(k)$	Expected number of slots in those CRI's that begin with a n-fold collision, with k packets in level l. The expected number of slots with k packets in level l during a CRI that starts with a n-fold collision is, by definition, $\eta_n^l(k)$. Thus, in N CRI's the expected number of such slots is $Na(n)\eta_n^l(k)$.
$\sum_{n=0}^{\infty} Na(n)\eta_n^l(k)$	Expected number of slots in N CRI's, with k packets in level l. This is obtained by simply summing the previous result over all values of n.

Also,

$\sum_{j=l+1}^{\infty} \sigma_n^j$	Expected number of slots in a single CRI starting out with a n-fold collision, for which level l exists. The expected number of slots during a CRI beginning with a n-fold collision, with composite stack depth exactly l is σ_n^{l+1}. Thus, the sum equals the expected number of slots for which stack level l exists, during a CRI starting with a n-fold collision.
$Na(n)\sum_{j=l+1}^{\infty} \sigma_n^j$	Expected number of slots in N CRI's that occur in a CRI which began with a n-fold l exists. Multiplying the previous quantity by $Na(n)$ gives the expected number of slots during the N CRI's which begin with a n-fold collision, for which stack level l exists.
$\sum_{n=0}^{\infty} Na(n)\sum_{j=l+1}^{\infty} \sigma_n^j$	Expected number of slots in N CRI's for which level l exists. This quantity is obtained simply summing the previous quantity over all possible values of n.

From the first principles, the fraction of slots in N CRI's with k packets at level l, given stack level l exists, can be expressed as

$$\frac{\textit{Number of slots with } k \textit{ packets at level } l}{\textit{Number of slots when level } l \textit{ exists}}$$

For example, in Figure 5.2, there are 4 slots during the CRI in which level 3 ($l = 3$) exists, and 2 of those 4 slots contain exactly 3 ($k = 3$) packets in the composite stack

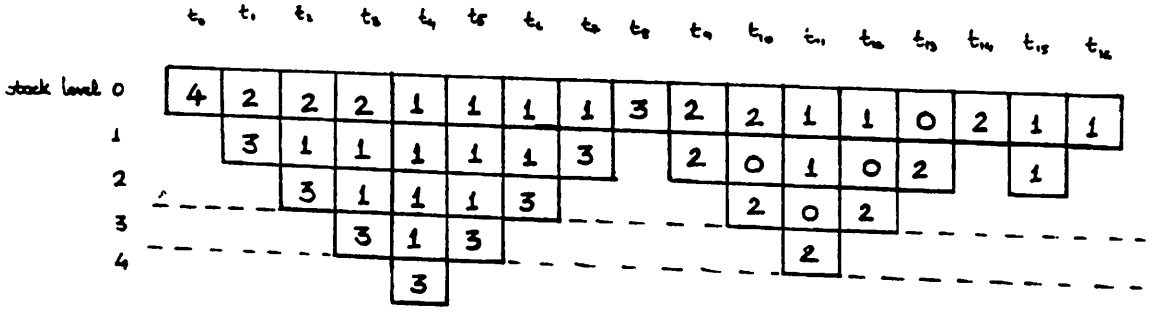

Figure 5.2: Calculation of Stack Occupancy

during the CRI.

Using the terms derived above, the fraction of slots in N CRI's with k packets at level l, given stack level l exists, can be mathematically expressed as

$$\frac{\sum_{n=0}^{\infty} Na(n)\eta_n^l(k)}{\sum_{n=0}^{\infty} Na(n)\sum_{j=l+1}^{\infty}\sigma_n^j}$$

Define $P^l(k)$ to be probability of k packets at level l, given stack level l exsits. Also define E^l to be the expected number of packets at level l, given level l exists.

Now, letting $N \rightarrow \infty$,

$$\begin{aligned} P^l(k) &= \lim_{N\rightarrow\infty} \frac{\sum_{n=0}^{\infty} Na(n)\eta_n^l(k)}{\sum_{n=0}^{\infty} Na(n)\sum_{j=l+1}^{\infty}\sigma_n^j} \\ &= \frac{\sum_{n=0}^{\infty} a(n)\eta_n^l(k)}{\sum_{n=0}^{\infty} a(n)\sum_{j=l+1}^{\infty}\sigma_n^j} \end{aligned} \tag{6}$$

The quantity E^l can then be computed as follows:

$$E^l = \sum_{k=0}^{\infty} kP^l(k) \tag{7}$$

Equations (6) and (7) can be solved to obtain E^l for any value l. The difference in values between E^l and E^{l+1} for all intermediate levels l and $l+1$ tells how good the approximate Markov model is. Again,

$$\left|E^l - E^{l+1}\right| \leq \epsilon$$

for small values of ϵ indicates that the approximation is a good one.

5.3 Numerical Results

In this section, we solve for l_n, σ_n^l and $\eta_n^l(k)$, using the previously derived equations, and then use these values to calculate the expected stack occupanices for a composite

stack of given depth. As we wil see, these numerical results provide compelling support for our modeling assumption that the expected stack occupancies for the intermediate levels of a given stack depth are identical.

A solution to the system of linear equations for l_n is proposed in [16], in which conditions are derived on the arrival rate λ, to establish maximal throughput results for the stack algorithm. The conditions for which a nonnegative solution to the system of equations exists and is unique are also derived.

Tsybakov *et. al.* also present an iterative method for solving the system of equations after truncating the chain for an appropriate value of n (the initial multiplicity of conflict for a CRI). They note that due to the nature of the equations (namely the non-negative coefficients of each l_n in the system of equations), the value for each l_n in the abridged system is bounded from above by the corresponding l_n in the solution for the complete set of equations. Due to the similar form of equations (1), (2) and (4) we use an iterative procedure similar to that in [16], to solve these equations.

In order to provide actual numerical results for E_l, we will look at the expected occupancy for each level of the composite stack for arrival rate of 0.1 and 0.3. Equation (1) was first solved iteratively after truncating the system for $n = 30$, 40 and 50. The maximum error between corresponding values of l_i's when truncating the equations for $n = 30$ (as opposed to when the chain was truncated at 40) was less than 5/1000 th of one percent. We denote this maximum value of n as $n_{max}(= 30)$. Results for l_n using Equation (1) for $n_{max} = 30$ are given in Table V.1.

We next solved equation (2) iteratively after again truncating the system of equations using $n = n_{max}$. In order to determine an appropriate truncating value

Table V.1: Length of CRI starting with n packets

n	l_n	
	$\lambda = 0.1$	$\lambda = 0.3$
0	1.000	1.000
1	1.000	1.000
2	6.478	24.205
3	10.198	40.492
4	14.152	57.584
5	18.155	74.858
6	22.159	92.140
7	26.155	109.402
8	30.147	126.651
9	34.138	143.895
10	38.129	161.139
11	42.121	178.386
12	46.114	195.635
13	50.108	212.885
14	54.101	230.135
15	58.094	247.385
16	62.087	264.635
17	66.080	281.884
18	70.073	299.133
19	74.066	316.381
20	78.058	333.630
21	82.051	350.878
22	86.044	368.127
23	90.036	385.375
24	94.029	402.624
25	98.022	419.873
26	102.015	437.122
27	106.008	454.371
28	110.001	471.620
29	113.994	488.869
30	117.987	506.119

of l, we make use of the following relationship between σ_n^l and l_n:

$$\sum_{j=0}^{\infty} \sigma_n^j = l_n, \quad n = 0, 1, \ldots \tag{8}$$

Equation (2) (now truncated at $n = n_{max}$) was solved for increasing values of l (starting from with $l = 0$), until the value of the sum on the left hand side of Equation (8) was within 0.01 percent of the corresponding value of l_n (previously obtained by solving Equation (1)). The above condition was satisfied for $l = 50$. This value of l is denoted by l_{max}.

Finally Equation (4) was solved iteratively for all values of $\eta_n^l(k)$ after truncating the system of equations at n_{max} and l_{max}.

The numerical results for all three quantities (namely l_n, σ_n^l and $\eta_n^l(k)$) were verified through simulation independently. All corresponding results from analysis and simulation were within 1 % of each other.

We finally compute the expected number of packets in each stack level (l), given level l exists, using Equations (8) and (9). In Equation (8), when calculating the $\sum_{j=l+1}^{\infty} \sigma_n^j$ term, we make use of the fact that:

$$\sum_{j=l+1}^{\infty} \sigma_n^j = l_n - \sum_{j=0}^{l} \sigma_n^j$$

which follows from Equation 8. The expected stack occupancies are presented in Table V.2.

5.4 Discussion of Results

In Table V.2, note that the stack occupancies for top (level 0) and level 1 of the composite stack are not required (since the Markov model developed in last chapter, explicitly tracks packets at level 0 and 1). We also note that our simulation results

Table V.2: E^l for level l of composite stack

l	E^l	
	$\lambda = 0.1$	$\lambda = 0.3$
2	0.5522084561	0.7626038173
3	0.5368101768	0.7481873350
4	0.5323016033	0.7449770624
5	0.5309292976	0.7448231234
6	0.5305016996	0.7446669096
7	0.5303619216	0.7446048951
8	0.5303044251	0.7445683073
9	0.5302591186	0.7443490709
10	0.5301908182	0.7442242573
11	0.5300621159	0.7441368753
12		0.7441301091
13		0.7441220208
14		0.7441132089
15		0.7441027520
16		0.7440615653
17		0.7440527555
18		0.7440501100
19		0.7440368953
20		0.7440233765
21		0.7440002586
22		0.7439734353
23		0.7439427828
24		0.7439077550
25		0.7438677357
26		0.7438219605
27		0.7437692383
28		0.7437089408
29		0.7436400251
30		0.7435116124

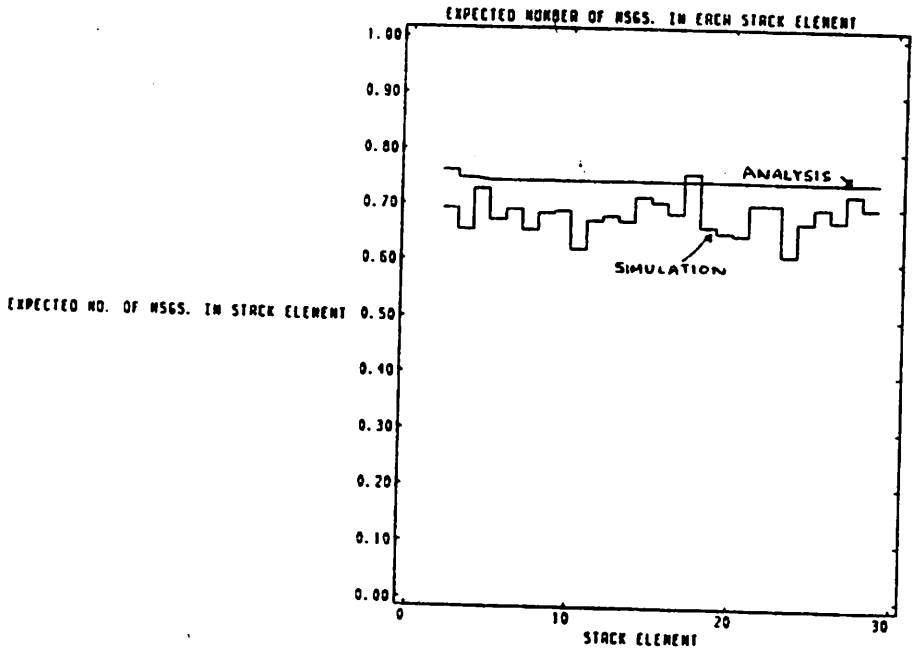

Figure 5.3: Expected stack occupancy for $\lambda = 0.3$

have shown that for arrival rate of 0.1, the composite stack never grows deeper than level 11 (with 300,000 arrivals). Hence from a practical standpoint, the expected occupancies in the second column of Table V.2 for level 11 and below need not be computed for $\lambda = 0.1$.

As shown in Table V.2, the maximum difference between the expected occupancy between any two intermediate levels for $\lambda = 0.1$ and 0.3 is 4 and 2.5 % respectively. This shows that our approximate model is good even for very low arrival rates and becomes even better at higher arrival rates.

A comparision of the analytical results of Table V.2 with the results shown in Figure 4.1 in Chapter IV, is shown in Figure 5.4. The expected occupancies for intermediate levels 2 through 30 of a composite stack of depth 31 from analysis and simulation are compared. The expected occupancies agree within 5 % of each other; we conjecture that the slight differences in the numerical values arise due to the differing definitions of the stack depth used in the simulation and analytic models. Note that comparing the results for the bottom of the stack (level 30)

are meaningless because during simulation we always track the bottom of the stack marker as the deepest stack level which has one or more packets, whereas for analysis we do not impose the restriction of at least one packet in the bottom of the stack.

Thus the numerical results from our analysis help support our claim that the approximate Markov model developed in Chapter IV is indeed a good one.

Chapter VI

Comparison of the Six Protocols

In this chapter we study the delays for the other five schemes, namely the PL, PP, NN, NL and NP protocols. We choose to study the performance of these protocols only through simulation results, for different error rates. Solving the balance equations arising from the Markov chain associated with each of the individual protocols is straightforward but tedious. We also believe that the results from simulation and the Markov model would not differ significantly, noting the similarity of the two results for the no error and the PN algorithm case. We shall compare the delays of the protocols with one another.

In Figure 6.1 and Table VI.1, we show the simulation results comparing PN, PL, PP, NN, NL and NP algorithms for 1, 5 and 10 % error rates. We note that for an error rate of 0.01, there is little difference among the performances of the protocols. However, for higher error rates (π = 0.05 or 0.10), the performance differences become more noticeable. For the range of error rates considered, and arrival rates greater than 0.2, the PN and NN protocols clearly perform better than the other four protocols.

The results have great intuitive appeal. For a fixed high error probability (for e.g. $\pi = 0.05$), the delays associated with each protocol are controlled by:

1. The different action taken by stations at the top of the composite stack ($stack\_depth_i = 0$) on receiving a NONE feedback signal.

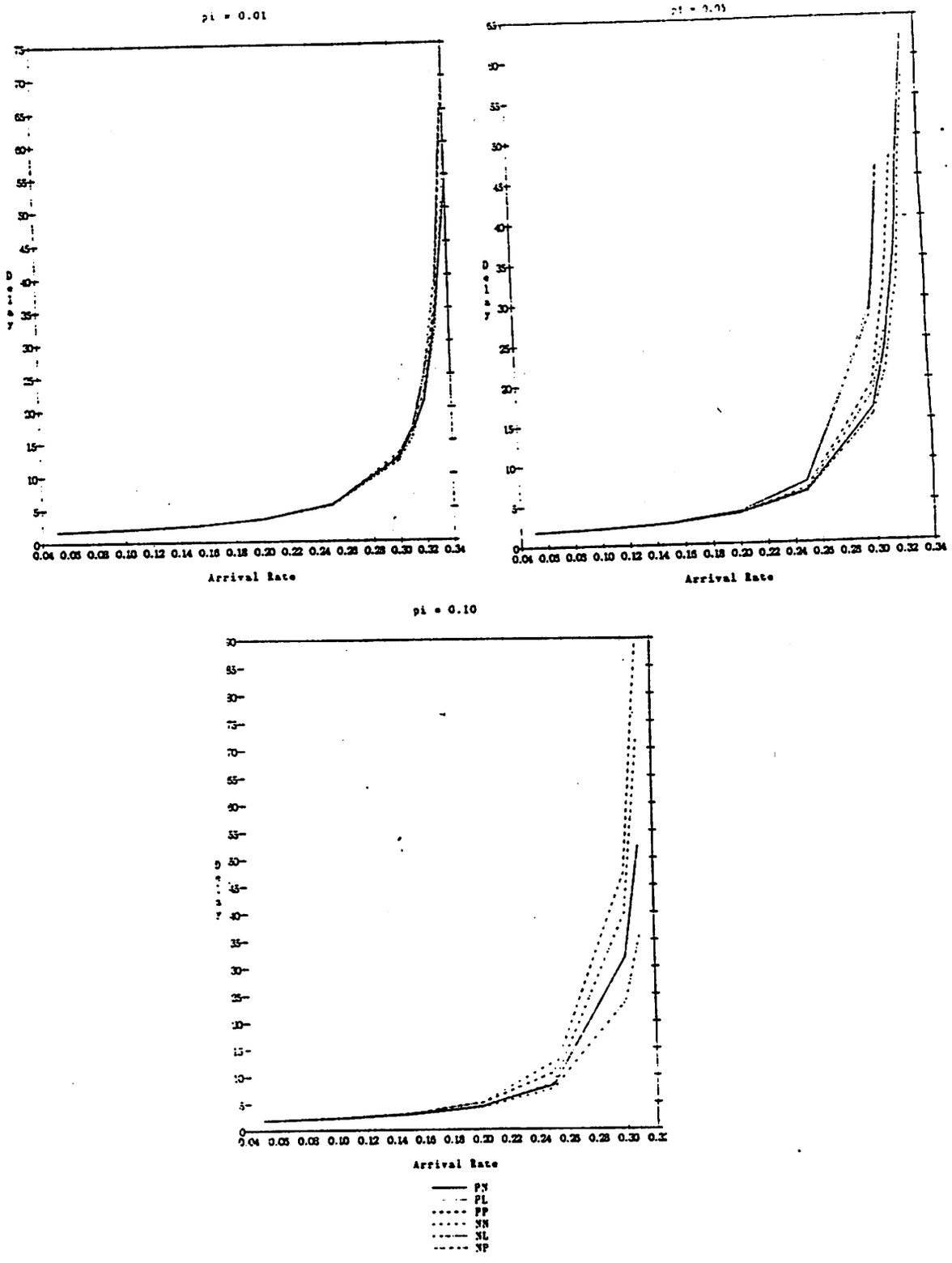

Figure 6.1: Time delay versus throughput: Comparison of six protocols

Table VI.1: Time delay versus throughput: Comparison of six protocols

0 Lambda	1 PN	2 PL	3 PP	4 NN	5 NL
1 0.05	1.698	1.699	1.697	1.703	1.704
2 0.10	1.992	1.998	1.990	1.989	1.994
3 0.15	2.484	2.494	2.487	2.493	2.508
4 0.20	3.427	3.473	3.442	3.430	3.482
5 0.25	5.530	5.628	5.592	5.437	5.582
6 0.30	12.453	12.849	12.696	12.093	13.248
7 0.31	16.200	16.697	16.863	15.297	16.645
8 0.32	21.069	24.575	22.401	21.365	24.791
9 0.33	34.256	36.091	35.803	32.204	39.105
10 0.34	53.852	74.333	68.629	58.391	70.056

pi = 0.01

0 Lambda	1 PN	2 PL	3 PP	4 NN	5 NL	6 NP
1 0.05	1.767	1.772	1.763	1.788	1.787	1.80474
2 0.10	2.099	2.115	2.093	2.110	2.127	2.12928
3 0.15	2.661	2.741	2.670	2.661	2.744	2.67530
4 0.20	3.809	4.074	3.854	3.731	4.007	3.79800
5 0.25	6.368	7.637	6.808	6.290	7.521	6.54758
6 0.30	16.489	28.805	19.554	15.671	27.856	18.27448
7 0.31	24.069	48.508	30.811	21.308	46.882	26.00409
8 0.32	37.726		47.820	32.229		
9 0.33	62.492			57.548		

pi = 0.05

0 Lambda	1 PN	2 PL	3 PP	4 NN	5 NL	6 NP
1 0.05	1.86550	1.87652	1.89945	1.91325	1.90779	1.89199
2 0.10	2.25280	2.29441	2.33813	2.28973	2.32092	2.27409
3 0.15	2.94700	3.16372	3.17502	2.94303	3.16263	2.95409
4 0.20	4.41687	5.27913	5.02882	4.22949	5.03516	4.36109
5 0.25	8.24404	13.36544	10.51064	7.59501	12.45333	8.62150
6 0.30	31.78337		47.16825	23.67694		39.82302
7 0.31	52.21271		88.81518	35.78525		71.79263

pi = 0.10

2. The action taken by stations at other levels of the composite stack ($stack\_depth_i \geq 1$) on receiving a NONE feedback signal.

For low arrival rates ($\lambda \leq 0.2$, a Utilization [1] of $\leq \sim 55$ %), an idle or success is the most probable event to occur in any time slot. Thus, protocols which persist on the top level of the composite stack (namely PN, PL, PP), have better delay characteristics (although marginally) over the other three protocols (NN, NL, NP).

For higher arrival rates ($\lambda \geq 0.2$), collisions take over as the most probable event. Thus those protocols in which packets with $stack\_depth_i \geq 1$, which on receiving a NONE feedback signal decrement their stack depths i.e., treat the NONE signal as a success or an idle slot (namely the PL, NL protocols), have higher delays (perform poorly) over other protocols. Protocols which treat this event by moving down a stack level i.e. treat the NONE signal as a collision (namely the PN and NN protocols) have lower delays (better performance) than protocols in which such packets persist at the same level on receiving a NONE feedback (PP and NP protocols).

Thus, the results indicate that in the absence of of a definite feedback (i.e. upon receiving a NONE feedback signal), the best option is to treat the outcome of the previous slot as if were the most probable one.

[1] Utilization $\equiv \lambda/\lambda_{max}$, where λ = arrival rate, and λ_{max} = max. throughput (0.36)

Chapter VII

Conclusions And Extensions

In this thesis, we have examined stack random access protocols in multiaccess networks in which individual stations may receive *asymmetric* feedback from the channel. We proposed six possible modifications to the basic stack algorithm for such environments and quantitatively examined the performance of each of the six protocols both through simulation and by developing and solving an approximate Markov chain model. It was found that for arrival rates above 0.2 packets/slot and high error rates (5 % or so) protocols which treat receiving a NONE message as a collision showed lower delays and thus had superior performance to the other schemes, while for low arrival rates the difference in delays of the six protocols was marginal. It was also found from the simulation studies that most of these algorithms remained stable for arrival rates of $\sim$ 0.3 packets/slot, at 10 % error rates.

The Markov model developed to analyze the modified stack algorithms was approximate, but the approximations used were good due to the similar stack occupancies for intermediate levels of the composite stack. This allows us to collapse all the state variables associated with those intermediate levels into two state variables, namely the total number of messages in those intermediate levels and the depth of the composite stack. This behavior was observed in our simulation studies and proved theoretically in Chapter V.

Several open problems, which arose during the course of this thesis are yet to be addressed.

- Development of an exact model to analyze the problem. This exact model could possibly be used to determine the conditions under which the modified stack algorithms are stable with asymmetric feedback errors occurring on the channel.

- Study the behavior of the protocol in the case that LACK, ACK and NACK messages may be mistaken for one another. Past work [13] shows that the error conditions occurring in such cases are exceedingly complex. For example some stations may interpret a NACK as an ACK feedback and leave the system prematurely i.e., even before their messages were successfully transmitted. In such a case, the protocol cannot guarantee successful transmission of every message.

- Since the error model proposed in this thesis is realistic (the symmetry feedback requirements for a system with geographically separated stations is too 'idealistic' for real situations), it would be interesting to study the behavior of the Capetanakis' tree protocol [3] and Gallager's window algorithm [7] in the presence of asymmetric feedback errors. Since these algorithms are characterized by continuous sensing and blocked entry, different stations have different views of the length and starting times of each CRI's. This in particular makes the problem of asymmetric feedback errors more difficult for the tree and time window algorithms.

Appendix A

Different Schemes For Asymmetric Feedback

The other five schemes to handle the NONE feedback signal are described here. An algorithmic description of each scheme similar to that of the PN algorithm in Chapter III is presented. An brief outline of each of these algorithms can also be found in Chapter III.

SCHEME II: MODIFIED STACK ALGORITHM PL

Each active station, i, executes the following steps:

1. /* At the *beginning* of each slot */
   if ($stack\_depth_i = 0$) then transmit the message
2. /* At the *end* of each slot */
   if ($stack\_depth_i = 0$)
      if ACK feedback then stop executing the algorithm since station i's message has just been successfully transmitted.
      if NACK feedback then with probability p
        set $stack\_depth_i = stack\_depth_i + 1$.
        otherwise leave $stack\_depth_i$ unchanged.
      if NONE feedback, keep $stack\_depth_i$ unchanged.
   else /*($stack\_depth_i \geq 1$)*/
      if LACK or ACK or NONE feedback then
        set $stack\_depth_i = stack\_depth_i - 1$

if NACK set $stack\_depth_i = stack\_depth_i + 1$

SCHEME III: MODIFIED STACK ALGORITHM PP

Each active station, i, executes the following steps:

1. /* At the *beginning* of each slot */

   if $(stack\_depth_i = 0)$ then transmit the message

2. /* At the *end* of each slot */

   if $(stack\_depth_i = 0)$

      if ACK feedback then stop executing the algorithm since station i's message has just been successfully transmitted.

      if NACK feedback then with probability p

        set $stack\_depth_i = stack\_depth_i + 1$.

        otherwise leave $stack\_depth_i$ unchanged.

      if NONE feedback, keep $stack\_depth_i$ unchanged.

   else /*$(stack\_depth_i \geq 1)$*/

      if LACK or ACK feedback then set $stack\_depth_i = stack\_depth_i - 1$

      if NACK set $stack\_depth_i = stack\_depth_i + 1$

      if NONE leave $stack\_depth_i$ unchanged.

SCHEME IV: MODIFIED STACK ALGORITHM NN

Each active station, i, executes the following steps:

1. /* At the *beginning* of each slot */

   if $(stack\_depth_i = 0)$ then transmit the message

2. /* At the *end* of each slot */

if ($stack\_depth_i = 0$)

if ACK feedback then stop executing the algorithm since station i's message has just been successfully transmitted.

if NACK or NONE feedback then with probability p,

set $stack\_depth_i = stack\_depth_i + 1$.

otherwise leave $stack\_depth_i$ unchanged.

else /*($stack\_depth_i \geq 1$)*/

if LACK or ACK feedback then set $stack\_depth_i = stack\_depth_i - 1$

if NACK or NONE set $stack\_depth_i = stack\_depth_i + 1$

SCHEME V: MODIFIED STACK ALGORITHM NL

Each active station, i, executes the following steps:

1. /* At the *beginning* of each slot */

if ($stack\_depth_i = 0$) then transmit the message

2. /* At the *end* of each slot */

if ($stack\_depth_i = 0$)

if ACK feedback then stop executing the algorithm since station i's message has just been successfully transmitted.

if NACK or NONE feedback then with probability p,

set $stack\_depth_i = stack\_depth_i + 1$.

otherwise leave $stack\_depth_i$ unchanged.

if NONE feedback, keep $stack\_depth_i$ unchanged.

else /*($stack\_depth_i \geq 1$)*/

if LACK or ACK or NONE feedback then

set $stack\_depth_i = stack\_depth_i - 1$

SCHEME VI: MODIFIED STACK ALGORITHM NP

Each active station, i, executes the following steps:

1. /* At the *beginning* of each slot */

   if $(stack\_depth_i = 0)$ then transmit the message

2. /* At the *end* of each slot */

   if $(stack\_depth_i = 0)$

   if ACK feedback then stop executing the algorithm since station i's message has just been successfully transmitted.

   if NACK feedback then with probability p

   set $stack\_depth_i = stack\_depth_i + 1$.

   otherwise leave $stack\_depth_i$ unchanged.

   if NONE feedback, keep $stack\_depth_i$ unchanged.

   else /*$(stack\_depth_i \geq 1)$*/

   if LACK or ACK feedback then set $stack\_depth_i = stack\_depth_i - 1$

   if NACK or NONE set $stack\_depth_i = stack\_depth_i + 1$

Appendix B

Balance Equations for the No Error Case

Let us define:

$p_{i,j,k,l}$ as the steady state probability for state $(x_1 = i, x_2 = j, x_3 = k, x_4 = l)$.

q_i as the probability of i arrivals (network-wide) during a time slot. Given Poisson assumptions:

$$q_i = \frac{e^{-\lambda}\lambda^i}{i!}$$

$t_{l,m}$ as the probability that m of l stations toss heads using a fair binary coin. This probability will be used in distributing colliding messages in stack level 0 of the composite stack among stack levels 0 and 1. (See algorithm statement for the case that $stack\_depth_i = 0$ with NACK feedback).

$$t_{l,m} = \left(\begin{array}{c} l \\ m \end{array} \right) (1/2)^m (1/2)^{m-l} \ , (l \geq m)$$

$f_{l,m,n}$ as the probability of finding l messages in composite stack level 3, assuming there are m messages $(m \geq l)$ uniformly distributed among n composite stack elements, numbered 3 through $n + 2$.

$$f_{l,m,n} = \left(\begin{array}{c} l+m \\ l \end{array} \right) (1/n)^l (1 - 1/n)^m \ , (l \geq 0, m, n \geq 2)$$

Given the above definitions, the balance equations for the Markov chain for the no error case are:

$$p_{0,0,0,0} = q_0\, p_{1,0,0,1} + (q_0 + q_1)\, p_{0,0,0,0}$$

$$p_{1,0,0,1} = q_0\, p_{1,1,0,2} + (q_0 + q_1)\, p_{0,1,0,2}$$

$$p_{k,0,0,1} = q_0\, p_{1,k,0,2} + (q_0 + q_1)\, p_{0,k,0,2} + q_k\, t_{k,k}\, p_{0,0,0,0} + \sum_{i=0}^{k-1} q_i\, t_{k,k}\, p_{k-i,0,0,1}, (k \geq 2)$$

$$p_{0,1,0,2} = q_0\, p_{1,0,1,3} + (q_0 + q_1)\, p_{0,0,1,3}$$

$$p_{0,l,0,2} = q_0\, p_{1,0,l,3} + (q_0 + q_1)\, p_{0,0,l,3} + q_l\, t_{l,0}\, p_{0,0,0,0} + \sum_{i=0}^{l-1} q_i\, t_{l,0}\, p_{l-i,0,0,1}\ , (l \geq 2)$$

$$p_{k,l,0,2} = q_0\, p_{1,k,l,3} + (q_0 + q_1)\, p_{0,k,l,3} + q_{k+l}\, t_{k+l,k}\, p_{0,0,0,0} + \sum_{i=0}^{k+l-1} q_i\, t_{k+l,k}\, p_{k+l-i,0,0,1}\ , (k,l \geq 1)$$

$$p_{0,0,m,3} = q_0\, f_{0,m-1,2}\, p_{1,0,m,4} + (q_0 + q_1)\, f_{0,m-1,2}\, p_{0,0,m,4}\ , (m \geq 1)$$

$$p_{0,1,m,3} = q_0\, f_{1,m-1,2}\, p_{1,0,m+1,4} + (q_0 + q_1)\, f_{1,m-1,2}\, p_{0,0,m+1,4}\ , (m \geq 1)$$

$$p_{0,l,m,3} = q_0\, f_{l,m-1,2}\, p_{1,0,m+l,4} + (q_0 + q_1)\, f_{l,m-1,2}\, p_{0,0,l+m,4} + \sum_{i=0}^{l} q_i\, t_{l,0}\, p_{l-i,m,0,2}\ , (l \geq 2, m \geq 1)$$

$$p_{1,0,m,3} = q_0\, f_{0,m-1,2}\, p_{1,1,m,4} + (q_0 + q_1)\, f_{0,m-1,2}\, p_{0,1,m,4}\ , (m \geq 1)$$

$$p_{k,0,m,3} = q_0\, f_{0,m-1,2}\, p_{1,k,m,4} + (q_0 + q_1)\, f_{0,m-1,2}\, p_{0,k,m,4} + \sum_{i=0}^{k} q_i\, t_{k,k}\, p_{k-i,m,0,2}\ , (k \geq 2, m \geq 1)$$

$$p_{k,l,m,3} = q_0\, f_{l,m-1,2}\, p_{1,k,l+m,4} + (q_0 + q_1)\, f_{l,m-1,2}\, p_{0,k,l+m,4} + \sum_{i=0}^{k+l} q_i\, t_{k+l,k}\, p_{k+l-i,m,0,2}\ , (k,l,m \geq 1)$$

$$p_{0,0,m,n} = q_0\, f_{0,m-1,n-1}\, p_{1,0,m,n+1} + (q_0 + q_1)\, f_{0,m-1,n-1}\, p_{0,0,m,n+1}\ , (m \geq 1, n \geq 4)$$

$$p_{0,1,m,n} = q_0\, f_{1,m-1,n-1}\, p_{1,0,m+1,n+1} + (q_0 + q_1)\, f_{1,m-1,n-1}\, p_{0,0,m+1,n+1}\ , (m \geq 1, n \geq 4)$$

$$p_{0,l,m,n} = q_0\, f_{l,m-1,n-1}\, p_{1,0,m+l,n+1} + (q_0 + q_1)\, f_{l,m-1,n-1}\, p_{0,0,l+m,n+1} + \sum_{i=0}^{l} \sum_{j=0}^{m-1} q_l\, t_{l,0}\, p_{l-i,j,m-j,n-1}\ , (l \geq 2, m \geq 1, n \geq 4)$$

$$p_{1,0,m,n} = q_0\, f_{0,m-1,n-1}\, p_{1,1,m,n+1} + (q_0 + q_1)\, f_{0,m-1,n-1}\, p_{0,1,m,n+1}\ , (m \geq 1, n \geq 4)$$

$$p_{k,0,m,n} = q_0 f_{0,m-1,n-1} p_{1,k,m,n+1} + (q_0 + q_1) f_{0,m-1,n-1} p_{0,k,m,n+1} + \sum_{i=0}^{k} \sum_{j=0}^{m-1} q_i t_{k,k} p_{k-i,j,m-j,n-1} \quad , (k \geq 2, m \geq 1, n \geq 4)$$

$$p_{k,l,m,n} = q_0 f_{l,m-1,n-1} p_{1,k,l+m,n+1} + (q_0 + q_1) f_{l,m-1,n-1} p_{0,k,l+m,n+1} + \sum_{i=0}^{k+l} \sum_{j=0}^{m-1} q_i t_{k+l,k} p_{k+l-i,j,m-j,n-1} \quad , (k, l, m \geq 1, n \geq 4)$$

We briefly describe how several representative equations above may be derived. Consider first the equation for $p_{0,0,0,0}$. The composite stack may enter state $x(0,0,0,0)$ if it was either in state $x(1,0,0,1)$ during the previous time slot and there were 0 arrivals (this event occurs with probability $p_{1,0,0,1}q_0$) or if the stack was in state $x_(0,0,0,0)$ and there were either zero or one arrivals.

Consider next the equation for $p_{0,0,m,3}$. One way in which the composite stack may enter $x(0,0,m,3)$ is if it was in state $x(1,0,m,4)$ during the previous time slot and there were zero arrivals (in which case the single message at the top of the composite stack was transmitted successfully) and there were also 0 messages in stack elements 2 and 3, and m messages in stack level 4. This event occurs with probability $p_{1,0,m,4}q_0 f_{0,m-1,2}$. Similarly the composite stack could have been in state state $x(0,0,m,4)$ at time T_j, and there were zero or 1 arrivals, with all other conditions as above.

Finally, consider the equation for $p_{k,l,m,n}$. The stack may reach state $x(k,l,m,n)$ if there was exactly one message at the top of the stack, no arrivals, k messages in stack element 2, l messages in stack element 3, and $l + m$ messages in stack elements 3 through $n+1$ during the previous time slot. This occurs with probability $q_0 f_{l,m-1,n-1} p_{1,k,l+m,n+1}$. The second set of terms in the equation for $p_{k,l,m,n}$ may be similarly obtained. Finally, if a collision occurred during the previous time slot, then

there must have been $k + l$ colliding messages in the top stack element (including new arrivals) of which k tossed heads and l tossed tails. There could have been any number, j, messages previously in stack element 2, with $m - j$ messages in the remaining stack elements.

Similar arguments are used to derive the remaining equations for the no error case.

Appendix C

Balance Equations for the PN Protocol

In this appendix we present the balance equations for the PN protocol. The full set of balance equations may be obtained in a straightforward manner from the balance equations in Appendix B. In addition to the terms defined in Appendix B, we define:

$e_{l,m}$ as the probability that m of l messages receive a (NONE) feedback signal.

$$e_{l,m} = \begin{pmatrix} l \\ m \end{pmatrix} \pi^m (1-\pi)^{l-m}, (l \geq m)$$

$\alpha_{l,m,n}$ as the probability that l of the m messages in the n stack elements, 2 through $n+1$, move from stack element 3 to stack element 2 *and* at least one of the messages in the bottom of the stack receives a NONE feedback signal.

$$\alpha_{l,m,n} = \sum_{i=0}^{m-1} \sum_{j=l}^{m-i-1} f_{i,m-i-1,n-2}\, f_{j,m-i-1-j,n-3}\, e_{j,j-l}\, (1 - e_{i+1,0}), (n \geq 4)$$

$\beta_{l,m,n}$ as the probability that l of the m messages in the n stack elements, 2 through $n+1$ move from stack element 3 to stack element 2 *and* no messages in the two bottommost stack elements receive a NONE feedback signal. As a result, if the correct feedback signal is LACK or ACK, the depth of the composite stack decreases by one.

$$\beta_{l,m,n} = \sum_{i=0}^{m-1} \sum_{j=l}^{m-i-1} \sum_{k=0}^{m-i-j-1} f_{i,m-i-1,n-2}\, f_{j,m-i-1-j,n-3}\, f_{k,m-i-j-i-k,n-4}\, e_{k,0}\, e_{j,j-l}\, e_{i+1,0}, (n \geq 4)$$

$\nu_{l,m,n}$ as the probability that l of the m messages in the n stack elements, 2 through $n+1$ move from stack element 3 to stack element 2 *and* no messages in the bottommost stack elements receive a NONE feedback signal and at least one message in the next to bottom stack element receives a NONE feedback signal. As a result, if the correct feedback signal is LACK or ACK, the depth of the composite stack remains the same.

$$\nu_{l,m,n} = \sum_{i=0}^{m-1} \sum_{j=1}^{m-i-1} \sum_{k=l}^{m-i-j-1} f_{i,m-i-1,n-2}\, f_{j,m-i-1-j,n-3}\, f_{k,m-i-j-k-1,n-4}\, e_{k,k-l}$$
$$(1-e_{j,0})\, e_{i+1,0}, (n \geq 4)$$

Given the above definitions, we have for the following (representative) balance equations for state occupancy probabilities:

$$p_{0,0,0,0} = (q_0 + q_1\, e_{1,0})\, p_{0,0,0,0} + q_0\, e_{1,0}\, p_{1,0,0,1}$$

$$p_{1,0,0,1} = q_1\, e_{1,1}\, p_{0,0,0,0} + q_0\, e_{1,1}\, p_{1,0,0,1} + e_{1,0}\, (q_0 + q_1\, e_{1,0})\, p_{0,1,0,2} + q_0\, e_{2,0}\, p_{1,1,0,2}$$

$$p_{k,0,0,1} = (q_0 + q_1\, e_{1,0})\, e_{k,0}\, p_{0,k,0,2} + q_0\, e_{k+1,0}\, p_{1,k,0,2} + \sum_{i=o}^{k} q_k\, e_{k,i}\, t_{k-i,k-i}\, p_{0,0,0,0} +$$
$$e_{k-1,0}\, e_{1,1}\, (q_0\, p_{1,k-1,0,2} + q_1\, p_{0,k-1,0,2}) + \sum_{i=0}^{k-1} \sum_{j=0}^{k} q_i\, e_{k,j}\, t_{k-j,k-j}\, p_{k-i,0,0,1}, (k \geq 2)$$

$$p_{0,1,0,2} = (q_0 + q_1\, e_{1,0})\, e_{1,0}\, p_{0,0,1,3} + q_0\, e_{2,0}\, p_{1,0,1,3}$$

$$p_{0,l,0,2} = (q_0 + q_1\, e_{1,0})\, e_{l,0}\, p_{0,0,l,3} + q_0\, e_{l+1,0}\, p_{1,0,l,3} + q_l\, e_{l,0}\, t_{l,0}\, p_{0,0,0,0} +$$
$$\sum_{i=0}^{l-1} q_i\, e_{l,0}\, t_{l,0}\, p_{l-i,0,0,1}, (l \geq 2)$$

$$p_{1,l,0,2} = (q_0 + q_1\, e_{1,0})\, e_{l+1,0}\, p_{0,1,l,3} + q_0\, e_{l+2,0} p_{1,1,l,3} + e_{l,0}\, e_{1,1}\, (q_0\, p_{1,0,l,3} + q_1\, p_{0,0.l,3}) +$$
$$q_{l+1}(e_{l+1,0}\, t_{l+1,1} + e_{l+1,1}\, t_{l,0}) p_{0,0,0,0} +$$
$$\sum_{i=0}^{l} q_i (e_{l+1,0}\, t_{l+1,1} + e_{l+1,1}\, t_{l,0}) p_{l+1-i,0,0,1}, (l \geq 1)$$

$$p_{k,l,0,2} = (q_0 + q_1 e_{1,0})\, e_{k+l,0}\, p_{0,k,l,3} + q_0\, e_{k+l+1,0}\, p_{1,k,l,3} +$$
$$e_{k+l-1,0}\, e_{1,1}(q_0\, p_{1,k-1,l,3} + q_1\, p_{0,k-1,l,3}) + \sum_{i=0}^{k} q_{k+l}\, e_{k+l,i}\, t_{k+l-i,k-i}\, p_{0,0,0,0} +$$
$$\sum_{i=0}^{k+l-1} \sum_{j=0}^{k} q_i\, e_{k+l,j}\, t_{k+l-j,k-j}\, p_{k+l-i,0,0,1}, (k \geq 2, l \geq 1)$$

$$p_{0,0,m,3} = e_{m,m}((q_0 + q_1\, e_{1,0})\, p_{0,m,0,2} + q_0\, e_{1,0}\, p_{1,m,0,2}) + \sum_{i=0}^{m-1} f_{0,m-i-1,2}\, e_{i,i}\, e_{m-i,0}$$
$$((q_0 + q_1\, e_{1,0}) p_{0,i,m-i,4} + q_0\, e_{1,0}\, p_{1,i,m-i,4}), (m \geq 1)$$

$$p_{0,1,m,3} = e_{m,m}\, e_{1,0}\, ((q_0 + q_1\, e_{1,0}) p_{0,m,1,3} + q_0\, e_{1,0}\, p_{1,m,1,3}) + \sum_{i=0}^{m-1} e_{i,i}\, e_{m-i+1,0}\, f_{1,m-i-1,2}$$
$$((q_0 + q_1\, e_{1,0}) p_{0,i,m-i+1,4} + q_0\, e_{1,0} p_{1,i,m-i+1,4}), (m \geq 1)$$

$$p_{0,l,m,3} = e_{m,m}\, e_{l,0}\, ((q_0 + q_1\, e_{1,0}) p_{0,m,l,3} + q_0\, e_{1,0}\, p_{1,m,l,3}) + \sum_{i=0}^{l} q_i\, e_{l,0}\, t_{l,0}\, p_{l-i,m,0,2} +$$
$$\sum_{i=0}^{m-1} e_{i,i}\, e_{m-i+l,0}\, f_{l,m-i-1,2}((q_0 + q_1\, e_{1,0}) p_{0,i,m-i+l,4} + q_0\, e_{1,0} p_{1,i,m-i+l,4}),$$
$$(l \geq 2, m \geq 1)$$

$$p_{1,0,m,3} = e_{m+1,0}\, f_{0,m-1,2}((q_0 + q_1\, e_{1,0}) p_{0,1,m,4} + q_0\, e_{1,0}\, p_{1,1,m,4}) +$$
$$e_{m+1,m}((q_0 + q_1\, e_{1,0}) p_{0,m+1,0,2} + q_0\, e_{1,0}\, p_{1,m+1,0,2}) +$$
$$e_{m+1,m+1}(q_0\, p_{1,m,0,2} + q_1\, p_{0,m,0,2}) +$$
$$\sum_{i=0}^{m-1} e_{i,i}\, e_{m-i,0}\, e_{1,1}\, f_{0,m-i-1,2}(q_0\, p_{1,i,m-i,4} + q_1\, p_{0,i,m-i,4}) +$$
$$\sum_{i=0}^{m-1} e_{i+1,i}\, e_{m-i,0}\, f_{0,m-i-1,2}((q_0 + q_1\, e_{1,0}) p_{0,i+1,m-i,4} + q_0\, e_{1,0}\, p_{1,i+1,m-i,4}), (m \geq 1)$$

$$p_{k,0,m,3} = e_{m+k,0}\, f_{0,m-1,2}((q_0 + q_1\, e_{1,0}) p_{0,k,m,4} + q_0\, e_{1,0}\, p_{1,k,m,4}) +$$
$$e_{m+k,m}((q_0 + q_1\, e_{1,0}) p_{0,m+k,0,2} + q_0\, e_{1,0}\, p_{1,m+k,0,2}) +$$
$$e_{m+k-1,m}\, e_{1,1}(q_0\, p_{1,m,0,2} + q_1\, p_{0,m,0,2}) +$$
$$\sum_{i=0}^{m-1} e_{k+i-1,i}\, e_{m-i,0}\, e_{1,1}\, f_{0,m-i-1,2}(q_0\, p_{1,k+i-1,m-i,4} + q_1\, p_{0,k+i-1,m-i,4}) +$$
$$\sum_{i=0}^{m-1} e_{k+i,i}\, e_{m-i,0}\, f_{0,m-i-1,2}((q_0 + q_1\, e_{1,0})\, p_{0,k+i,m-i,4} + q_0\, e_{1,0}\, p_{1,k+i,m-i,4}) +$$

$$\sum_{i=0}^{k}\sum_{j=0}^{k} q_i\, e_{k,j}\, t_{k-j,k-j}\, p_{k-i,m,0,2}, (k \geq 2, m \geq 1)$$

$$p_{k,l,m,3} = e_{m+k,m}\, e_{l,0}\, ((q_0 + q_1\, e_{1,0}) p_{0,k+m,l,3} + q_0\, e_{1,0}\, p_{1,k+m,l,3}) +$$

$$e_{m+k-1,m}\, e_{l,0}\, e_{1,1} (q_0\, p_{1,k+m-1,l,3} + q_1\, p_{0,k+m-1,l,3}) +$$

$$\sum_{i=0}^{m-1} e_{k+i-1,i}\, e_{l+m-i,0}\, e_{1,1}\, f_{l,m-i-1,2} (q_0\, p_{1,k+i-1,l+m-i,4} + q_1\, p_{0,k+i-1,l+m-i,4}) +$$

$$\sum_{i=0}^{m-1} e_{k+i,i}\, e_{l+m-i,0}\, f_{l,m-i-1,2} ((q_0 + q_1\, e_{1,0}) p_{0,k+i,l+m-i,4} + q_0\, e_{1,0}\, p_{1,k+i,l+m-i,4}) +$$

$$\sum_{i=0}^{k+l}\sum_{j=0}^{k} q_i\, e_{k+l,j}\, t_{k+l-j,k-j}\, p_{k+l-i,m,0,2}, (k, l, m \geq 1)$$

$$p_{0,0,m,n} = e_{m,0}\, f_{0,m-1,n-1} ((q_0 + q_1\, e_{1,0}) p_{0,0,m,n+1} + q_0\, e_{1,0}\, p_{1,0,m,n+1}) +$$

$$\sum_{i=0}^{m-1} e_{i,i}\, \alpha_{0,m-i,n-1} ((q_0 + q_1\, e_{1,0}) p_{0,i,m-i,n-1} + q_0\, e_{1,0}\, p_{1,i,m-i,n-1}) +$$

$$\sum_{i=0}^{m-1} e_{i,i}\, \beta_{0,m-i,n-1} ((q_0 + q_1\, e_{1,0}) p_{0,i,m-i,n+1} + q_0\, e_{1,0}\, p_{1,i,m-i,n+1}) +$$

$$\sum_{i=0}^{m-2} e_{i,i}\, \nu_{0,m-i,n} ((q_0 + q_1\, e_{1,0}) p_{0,i,m-i,n} + q_0\, e_{1,0}\, p_{1,i,m-i,n}), (m \geq 1, n \geq 4)$$

$$p_{0,1,m,n} = e_{m+1,0}\, f_{1,m-1,n-1} ((q_0 + q_1\, e_{1,0}) p_{0,0,m+1,n+1} + q_0\, e_{1,0}\, p_{1,0,m+1,n+1}) +$$

$$\sum_{i=0}^{m-1} e_{i,i}\, \alpha_{1,m-i+1,n-1} ((q_0 + q_1\, e_{1,0}) p_{0,i,m-i+1,n-1} + q_0\, e_{1,0}\, p_{1,i,m-i+1,n-1}) +$$

$$\sum_{i=0}^{m-1} e_{i,i}\, \beta_{1,m-i+1,n-1} ((q_0 + q_1\, e_{1,0}) p_{0,i,m-i+1,n+1} + q_0\, e_{1,0}\, p_{1,i,m-i+1,n+1}) +$$

$$\sum_{i=0}^{m-2} e_{i,i}\, \nu_{1,m-i+1,n} ((q_0 + q_1\, e_{1,0}) p_{0,i,m-i+1,n} + q_0\, e_{1,0}\, p_{1,i,m-i+1,n}), (m \geq 1, n \geq 4)$$

$$p_{0,l,m,n} = e_{m+l,0}\, f_{l,m-1,n-1} ((q_0 + q_1\, e_{1,0}) p_{0,0,m+l,n+1} + q_0\, e_{1,0}\, p_{1,0,m+l,n+1}) +$$

$$\sum_{i=0}^{m-1} e_{i,i}\, \alpha_{l,m-i+l,n-1} ((q_0 + q_1\, e_{1,0}) p_{0,i,m-i+l,n-1} + q_0\, e_{1,0}\, p_{1,i,m-i+l,n-1}) +$$

$$\sum_{i=0}^{m-1} e_{i,i}\, \beta_{l,m+l-i,n-1} ((q_0 + q_1\, e_{1,0})\, p_{0,i,m-i+l,n+1} + q_0\, e_{1,0}\, p_{1,i,m-i+l,n+1}) +$$

$$\sum_{i=0}^{m-2} e_{i,i}\, \nu_{l,m+l-i,n} ((q_0 + q_1\, e_{1,0}) p_{0,i,m-i+l,n} + q_0\, e_{1,0}\, p_{1,i,m-i+l,n}) +$$

$$\sum_{i=0}^{l}\sum_{j=0}^{m-1} q_i\, e_{l,0}\, t_{l,0}\, p_{l-i,j,m-j,n-1}, (l \geq 2, m \geq 1, n \geq 4)$$

$$
\begin{aligned}
p_{1,0,m,n} \;=\;& e_{m,0}\, e_{1,0}\, f_{0,m-1,n-1}\big((q_0 + q_1\, e_{1,0})p_{0,1,m,n+1} + q_0\, e_{1,0}\, p_{1,1,m,n+1}\big) + \\
& \sum_{i=0}^{m-1} e_{i+1,i}\, \alpha_{0,m-i,n-1}\big((q_0 + q_1\, e_{1,0})p_{0,i+1,m-i,n-1} + q_0\, e_{1,0}\, p_{1,i+1,m-i,n-1}\big) + \\
& \sum_{i=0}^{m-1} e_{i,i}\, \alpha_{0,m-i,n-1}\, e_{1,1}\big(q_0\, p_{1,i,m-i,n-1} + q_1\, p_{0,i,m-i,n-1}\big) + \\
& \sum_{i=0}^{m-1} e_{i+1,i}\, \beta_{0,m-i,n+1}\big((q_0 + q_1\, e_{1,0})p_{0,i+1,m-i,n+1} + q_0\, e_{1,0}\, p_{1,i+1,m-i,n+1}\big) + \\
& \sum_{i=0}^{m-1} e_{i,i}\, \beta_{0,m-i,n+1}\, e_{1,1}\big(q_0\, p_{1,i,m-i,n+1} + q_1\, p_{0,i,m-i,n+1}\big) + \\
& \sum_{i=0}^{m-2} e_{i+1,i}\, \nu_{0,m-i,n}\big((q_0 + q_1\, e_{1,0})p_{0,i+1,m-i,n} + q_0\, e_{1,0}\, p_{1,i+1,m-i,n}\big) + \\
& \sum_{i=0}^{m-2} e_{i,i}\, \nu_{0,m-i,n}\, e_{1,1}\big(q_0\, p_{1,i,m-i,n} + q_1\, p_{0,i,m-i,n}\big), (m \geq 1, n \geq 4)
\end{aligned}
$$

$$
\begin{aligned}
p_{k,0,m,n} \;=\;& e_{m,0}\, e_{k,0}\, f_{0,m-1,n-1}\big((q_0 + q_1\, e_{1,0})p_{0,k,m,n+1} + q_0\, e_{1,0}\, p_{1,k,m,n+1}\big) + \\
& \sum_{i=0}^{m-1} e_{i+k,i}\, \alpha_{0,m-i,n-1}\big((q_0 + q_1\, e_{1,0})p_{0,i+k,m-i,n-1} + q_0\, e_{1,0}\, p_{1,i+k,m-i,n-1}\big) + \\
& \sum_{i=0}^{m-1} e_{k+i-1,i}\, \alpha_{0,m-i,n-1}\, e_{1,1}\big(q_0\, p_{1,k+i-1,m-i,n-1} + q_1\, p_{0,k+i-1,m-i,n-1}\big) + \\
& \sum_{i=0}^{m-1} e_{i+k,i}\, \beta_{0,m-i,n+1}\big((q_0 + q_1\, e_{1,0})p_{0,i+k,m-i,n+1} + q_0\, e_{1,0}\, p_{1,i+k,m-i,n+1}\big) + \\
& \sum_{i=0}^{m-1} e_{k+i-1,i}\, \beta_{0,m-i,n+1}\, e_{1,1}\big(q_0\, p_{1,k+i-1,m-i,n+1} + q_1\, p_{0,k+i-1,m-i,n+1}\big) + \\
& \sum_{i=0}^{m-2} e_{i+k,i}\, \nu_{0,m-i,n}\big((q_0 + q_1\, e_{1,0})p_{0,i+k,m-i,n} + q_0\, e_{1,0}\, p_{1,i+k,m-i,n}\big) + \\
& \sum_{i=0}^{m-2} e_{k+i-1,i}\, \nu_{0,m-i,n}\, e_{1,1}\big(q_0\, p_{1,k+i-1,m-i,n} + q_1\, p_{0,k+i-1,m-i,n}\big) + \\
& \sum_{i=0}^{k} \sum_{j=0}^{m-1} q_i\, e_{k,j}\, t_{k-j,k-j}\, p_{k-i,j,m-j,n-1}, (k \geq 2, m \geq 1, n \geq 4)
\end{aligned}
$$

$$
\begin{aligned}
p_{k,l,m,n} \;=\;& e_{l+m,0}\, e_{k,0}\, f_{l,m-1,n-1}\big((q_0 + q_1\, e_{1,0})p_{0,k,l+m,n+1} + q_0\, e_{1,0}\, p_{1,k,l+m,n+1}\big) + \\
& \sum_{i=0}^{m-1} e_{i+k,i}\, \alpha_{l,m+l-i,n-1}\big((q_0 + q_1\, e_{1,0})p_{0,k+i,m+l-i,n-1} + q_0\, e_{1,0}\, p_{1,k+i,m+l-i,n-1}\big) + \\
& \sum_{i=0}^{m-1} e_{k+i-1,i}\, \alpha_{l,m+l-i,n-1}\, e_{1,1}\big(q_0\, p_{1,k+i-1,m+l-i,n-1} + q_1\, p_{0,k+i-1,m+l-i,n-1}\big) +
\end{aligned}
$$

$$\sum_{i=0}^{m-1} e_{k+i,i}\,\beta_{l,m+l-i,n+1}((q_0+q_1\,e_{1,0})p_{0,k+i,m+l-i,n+1}+q_0\,e_{1,0}\,p_{1,i+k,m+l-i,n+1})+$$
$$\sum_{i=0}^{m-1} e_{k+i-1,i}\,\beta_{l,m+l-i,n+1}\,e_{1,1}(q_0\,p_{1,k+i-1,m+l-i,n+1}+q_1\,p_{0,k+i-1,m+l-i,n+1})+$$
$$\sum_{i=0}^{m-2} e_{i+k,i}\,\nu_{l,m+l-i,n}((q_0+q_1\,e_{1,0})p_{0,k+i,m+l-i,n}+q_0\,e_{1,0}\,p_{1,k+i,m+l-i,n})+$$
$$\sum_{i=0}^{m-2} e_{k+i-1,i}\,\nu_{l,m+l-i,n}\,e_{1,1}(q_0\,p_{1,k+i-1,m+l-i,n}+q_1\,p_{0,k+i-1,m+l-i,n})+$$
$$\sum_{i=0}^{k+l}\sum_{j=0}^{k}\sum_{w=0}^{m-1} q_i\,e_{k+l,j}\,t_{k+l-j,k-j}\,p_{k+l-i,w,m-w,n-1},\,(k,l,m\geq 1,n\geq 4)$$

The equation for $p_{0,0,0,0}$ is the same as for the no error case, except that the (at most) one message being transmitted during a slot must receive correct feedback in order to leave the system at the end of the slot; correct feedback is obtained with probability $e_{1,0}$.

The sum in the equation for $p_{0,0,m,3}$ sums over the number of possible messages which receive the NONE feedback. Note that if there are i messages in stack element 1 in the preceding time slot and all i messages receive the NONE feedback signal, there will then be 0 elements in this stack element as a result of the PN algorithm. The term outside the sum is for the case in which all m of the messages in stack element 1 and each receives the NONE feedback signal.

Finally, we examine the equation for $p_{k,l,m,n}$. The terms on the first line on the right hand side are for the case of no errors and are thus analogous to those discussed in Appendix B. The first six sums are for the case that a single message is successfully transmitted. The first sum is for the case in which i of the $k+i$ elements in stack level 1 receive NONE feedback and at least one message in stack element $n-1$ receives NONE feedback (and hence the stack depth increases from

$n-1$ to n.) The second sum is identical except that the single message in the topmost stack element receives NONE feedback. The third and fourth sums are similar to the first two sums except that the stack depth decreases from $n+1$ to n. The fifth and sixth sums are also similar except that the stack depth remains the same. The final (triple) sum is for the case of a collision among the $k+l$ messages in the top stack element.

Bibliography

[1] N. Abramson,"The ALOHA System-Another alternative for computer communications", *Proceedings of the AFIPS Fall Joint Computer Conference,*, Vol 37, AFIPS Press, pp. 281-285, 1970.

[2] T. Berger, N. Mehraravi, J. Wolf and D. Towsley, "Random Multiple Access Communications and Group Testing", *IEEE Trans. Commun.*, Vol, COM-32, No 7, pp. 769-779, July 1984.

[3] J.I. Capetanakis, "Generalized TDMA: The Multi-Accessing Tree Protocol", *IEEE Trans. Commun.*, Vol. COM-27, No. 10, pp. 1476-1484, October 1979.

[4] I. Cidon and M. Sidi, "Erasures and Noise in Splitting Multiple Access Algorithms", to appear in *IEEE Transactions on Information Theory.*

[5] R. Cruz, "Protocols for Multiaccess Channels with Continuous Entry and Noisy Feedback", MIT Report LIDS-TH-1213, June 1982.

[6] G. Fayolle, P. Flajolet, M. Hofri, P. Jacquet, "Analysis of a Stack Algorithm for Random Multiple Access Communication" *IEEE Trans. on Information Theory,*, Vol. IT-31, No. 2, pp. 244-254, March 1985.

[7] R. Gallager, "A Perspective on Multiaccess Channels", *IEEE Trans. on Information Theory,*, Vol. IT-31, No. 2, pp. 124-142, March 1985.

[8] L. Georgiadis, L. Merakos, P. Papantoni-Kazakos, "Unified Method for Delay Analysis of Random Multiple Access Algorithms", Technical Report, Univ. of Conn., 1985.

[9] K. Ho, R. Rao, and J.K. Wolf, "An Upper Bound on Capacity and a Robust Collision Resolution Algorithm for the Random Access System with a Noisy Channel", *Proc. INFOCOM '87*, pp. 372-380, April 1987.

[10] L. Klienrock and F. A. Tobagi, "Packet Switching in Radio Channels: Part I – Carrier sense multiple access models and their throughput-delay characteristics", *IEEE Trans. Commun.*,, COM-23, 12(Dec.), pp. 1400-1416.

[11] J. F. Kurose, M. Schwartz and Y. Yemini, "Multiple-Access Protocols and Time-Constrained Communication", *Computing Surveys* ,,Vol 16, No. 1, pp. 43-70, March 1984.

[12] T. Liu and D. Towsley, "Window and Tree Protocols for Satellite Channels", *Proc. INFOCOM '83*, pp. 215-221, April 1983.

[13] J. Massey, "Collision Resolution Algorithms and Random Access Communications", in *Multi-User Communications*, New York: Springer-Velag, pp. 73-137, 1981.

[14] L. G. Roberts, "Aloha Packet system with and without slots and capture". ASS Note 8, Advanced Research Projects Agency, Network Info. Ctr.,Stanford Research Inst., Stanford CA, June 1972.

[15] D. Ryter, "A Conflict Resolution Algorithm for Noisy Multiaccess Channels", MIT Report LIDS-TH-1007, June 1980.

[16] B. Tsybakov and N. Vvedenskaya, "Stack Algorithm for Random Multiple Access", *Prob. Peredachi Inf.*,, Vol. 16, No. 3, pp. 80-94, 1980.

[17] B. Tsybakov, "Survey of USSR Contributions to Random Multiple Access Communications", *IEEE Trans. on Information Theory*,, Vol. IT-31, No. 2, pp. 143-165, March 1985.

[18] N. Vvedenskaya and B. Tsybakov, "Random Multiple Access of Packets to a Channel with Errors", *Prob. Peredachi Inf.*,, Vol. 19, No. 2, April-June 1983.