NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

AN ADAPTIVE ARQ STRATEGY FOR
PACKET SWITCHING DATA COMMUNICATION
NETWORKS

By

Kurtis J. Guth

June 1989

Thesis Advisor: Tri T. Ha

Approved for public release; distribution is unlimited.

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified		1b Restrictive Markings			
2a Security Classification Authority		3 Distribution Availability of Report Approved for public release; distribution is unlimited.			
2b Declassification Downgrading Schedule					
4 Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)			
6a Name of Performing Organization Naval Postgraduate School	6b Office Symbol (if applicable) 62	7a Name of Monitoring Organization Naval Postgraduate School			
6c Address (city, state, and ZIP code) Monterey, CA 93943-5000		7b Address (city, state, and ZIP code) Monterey, CA 93943-5000			
8a Name of Funding Sponsoring Organization	8b Office Symbol (if applicable)	9 Procurement Instrument Identification Number			
8c Address (city, state, and ZIP code)		10 Source of Funding Numbers			
		Program Element No	Project No	Task No	Work Unit Accession No

11 Title (Include security classification) AN ADAPTIVE ARQ STRATEGY FOR PACKET SWITCHING DATA COMMUNICATION NETWORKS

12 Personal Author(s) Kurtis J. Guth

13a Type of Report	13b Time Covered	14 Date of Report (year, month, day)	15 Page Count
Master's Thesis	From To	June 1989	75

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms (continue on reverse if necessary and identify by block number)
Field	Group	Subgroup	DLC, protocol, Stop and Wait, Adaptive protocol, throughput efficiency

19 Abstract (continue on reverse if necessary and identify by block number) Automatic Repeat reQuest (ARQ) techniques are often used by a packet switching data communication network to provide an error-free communications link between stations. The ARQ technique ensures consistent data quality under varying link conditions. Unfortunately, the information throughput is link dependent and as the noise or interference on the link increases, the throughput decreases. In an effort to improve the throughput on a short range, RF, packet switching data communications network, an adaptive ARQ strategy applied to Stop-and-Wait (SW) protocols was developed. To provide a system designer with flexibility, different adaptive strategies for different system and link conditions were developed. Examples of information transfer between two stations using the adaptive SW protocol are presented. A simulation to compute the throughput efficiency of several adaptive SW protocols was performed. A comparison of the throughput efficiencies of the simulated adaptive SW protocol with the non-adaptive SW protocol showed good gains could be achieved using the adaptive strategy when the networks are subject to high channel bit error rates. Theses (AW)

20 Distribution Availability of Abstract ☒ unclassified unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Tri T. Ha	22b Telephone (include Area code) (408) 646-2788	22c Office Symbol 62Ha

DD FORM 1473,84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete — security classification of this page

Unclassified

An Adaptive ARQ Strategy for
Packet Switching Data Communication Networks

by

Kurtis J. Guth
Lieutenant, United States Coast Guard
B.S., United States Coast Guard Academy, 1981

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1989

Author: Kurtis J. Guth
Kurtis J. Guth

Approved by: Tri T. Ha
Tri T. Ha, Thesis Advisor

Glen A. Myers
Glen A. Myers, Second Reader

Robert D Strum for
John P. Powers, Chairman,
Department of Electrical and Computer Engineering

G. E. Schacher
Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

Automatic Repeat reQuest (ARQ) techniques are often used by a packet switching data communication network to provide an error-free communications link between stations. The ARQ technique ensures consistent data quality under varying link conditions. Unfortunately, the information throughput is link dependent and as the noise or interference on the link increases, the throughput decreases. In an effort to improve the throughput on a short range, RF, packet switching data communications network, an adaptive ARQ strategy applied to Stop-and-Wait (SW) protocols was developed. To provide a system designer with flexibility, different adaptive strategies for different system and link conditions were developed. Examples of information transfer between two stations using the adaptive SW protocol are presented. A simulation to compute the throughput efficiency of several adaptive SW protocols was performed. A comparison of the throughput efficiencies of the simulated adaptive SW protocol with the non-adaptive SW protocol showed good gains could be achieved using the adaptive strategy when the networks are subject to high channel bit error rates.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

In this thesis an adaptive *Automatic Repeat reQuest* (ARQ) strategy for a packet switching network is developed. The adaptive technique is to be applied to a Stop-and-Wait (SW) protocol. The SW protocol was chosen because the adaptive techniques designed are useful for radio frequency (RF) systems transferring data over short distances. In the short range RF systems, the propagation delay is very short, and the inefficient use of bandwidth resulting from a transmitter waiting for a response is not a significant factor. Also, RF systems often operate in a high noise environment and are affected by signal fading or even jamming. The throughput of all ARQ systems is very low under high noise conditions. These high noise conditions will often result in a large number of retransmissions which increases the delays for the data transfer and further justifies the use of adaptive SW type protocols.

The adaptive strategies developed in this thesis all vary the length of the message packets by reducing or increasing the information contained in the packet. These increases or decreases in the packet length are made based on the number of consecutive successful transmissions or consecutive retransmissions of a packet. By keeping track of the consecutive packet transmissions and using this information to base the decision of when to adapt the packet length, no additional hardware should be needed to upgrade an SW protocol to an adaptive SW protocol.

Chapters 2 and 3 discuss the basic requirements to be met by an ARQ protocol and the structure used to meet these requirements. Chapter 4 develops the adaptive strategies and provides examples of their implementation. Chapter 5 describes some simulations of several adaptive SW strategies and computes the throughput efficiencies for each type.

II. THE DATA LINK CONTROL (DLC) LAYER AND PROTOCOLS

A. DLC LAYER

The International Standards Organization (ISO) has developed the Open System Interconnection (OSI) network architecture composed of a hierarchy of seven layers or modules [Ref. 1,2]. Each layer performs certain functions and provides certain services to the layers above. Each function performed by a data communications terminal or a station on a data communications network is typically assigned to a particular layer. The actual operation of transferring data or messages by a station is the responsibility of the first three layers. Figure 1 shows the OSI network architecture and a block diagram representing a communications network between n stations.

The third layer, the network layer, and the second layer, the data link control (DLC) layer, provide the upper layers with a virtual link for messages with the other stations on the communications network. This means the layers above the network and DLC layer only provide a message to be transferred to another station on the network, and expect the network and DLC layers to perform the necessary functions to ensure its proper transfer. The network layer works closely with the DLC layer to break the message into packets for transmission. The DLC layer is responsible for providing the upper layers with an "error-free" communications link. The DLC layer uses an unreliable bit pipe provided by the first layer, the physical layer, to transmit the actual bits through the communications link. The physical layer contains the transmission equipment used to provide the interface between stations on the data communications network. This layer is only concerned with transferring bits from station to station and does not perform any bit error detection to determine the accuracy of the bit transmission. Therefore, the reception of a correct bit is "unreliable" and must be checked by the DLC layer before accepted. The DLC layer uses a specified set of procedures to transmit the information and correct the bit errors that occur on the unreliable bit pipe.

To establish the "error-free" communications link, the network and DLC layers break the message into packets and then add some overhead control bits to each packet. As shown in Figure 2, a packet with the overhead control bits added is called a frame. The DLC layer is responsible for establishing a bit sequence to identify the frame starting and ending points. The bit sequence must be unique and no inadvertent occurrences of the sequences can be transmitted prior to sending the bit sequence marking the end

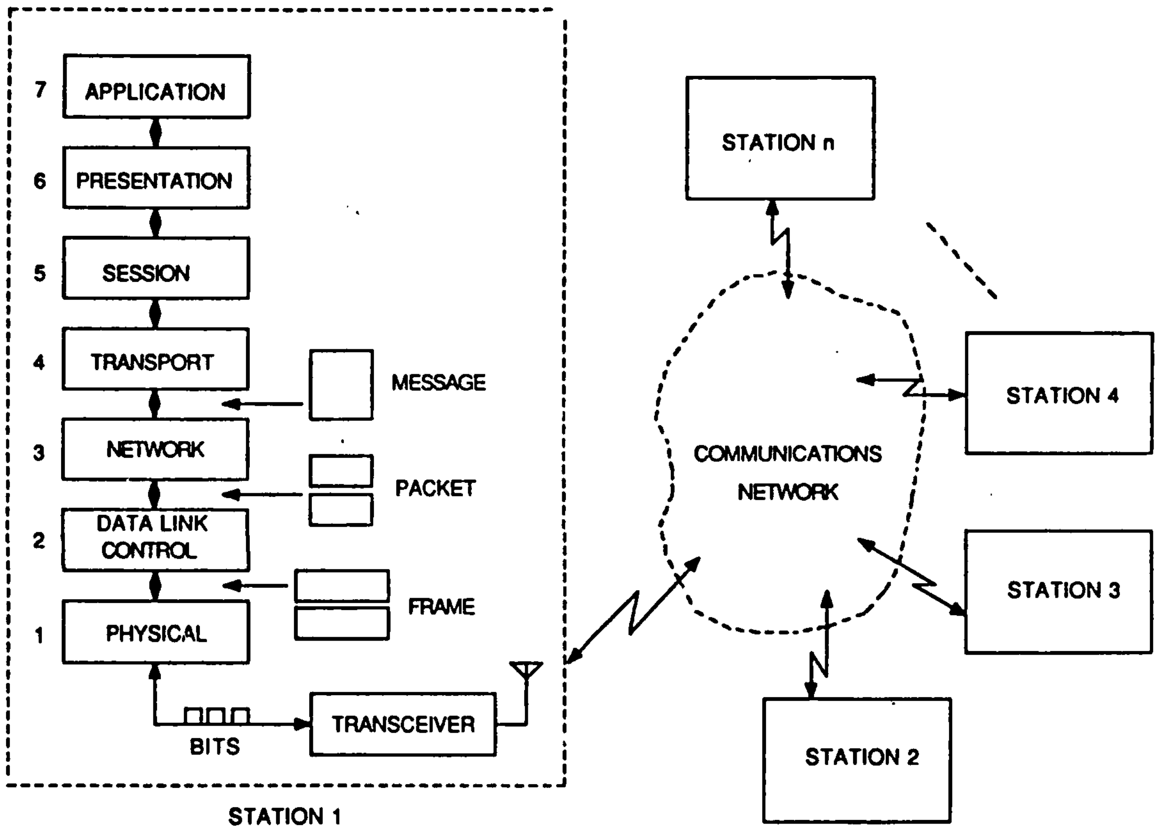


Figure 1. **OSI network architecture and data communications network**

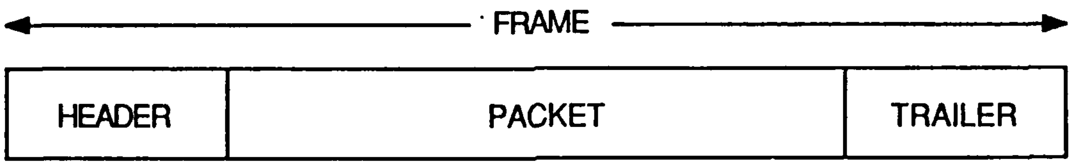


Figure 2. Frame structure [From Ref. 2]

of the frame. The DLC layer must account for the frames that have been transmitted and the acknowledgements identifying receipt from the receiving station. Additionally, the DLC layer is responsible for ensuring the frames are received in the proper order so the message can be reassembled and passed in the correct form to the layer above. Frames which acquire errors during transmission over the unreliable bit pipe must be corrected by the DLC. Finally, the DLC layer must be able to establish and control the communications link between the stations.

B. DLC PROTOCOLS

A protocol is a set of specific procedures for data transfer that the DLC layers use to obtain error free communications when utilizing the unreliable bit pipe. The protocols or sets of procedures must be designed to allow the DLC layers to identify and correct all possible transmission errors. One technique used to correct these transmission errors is called automatic repeat request (ARQ). This technique draws on the ability of the DLC to detect frame errors which occur during transmission and request the sender to retransmit the frame. To implement these ARQ protocols, the DLC establishes a specific format for the frames and transfer procedures to handle the data transfer.

One type of ARQ protocol is called the Stop-and-Wait (SW) protocol. In the SW protocol, a single frame is transmitted and the transmitter stops and waits for a response from the receiving station. If the frame is received without errors, then an acknowledgement (ACK) will be returned from the receiver and the next frame is transmitted. If an error is encountered, then a request for retransmission (NAK) will be returned and the same frame is retransmitted. If no response is received, the sender only waits a certain amount of time referred to as a *time-out* and then the frame is retransmitted. The SW protocol is effective for data communications when the round trip propagation delay time for a packet is short relative to the frame transmission time. This protocol becomes inefficient in using the available bandwidth when the round trip propagation time is long, such as in satellite communications, or when stopping the transmission introduces any additional delays associated with resynchronization of the transmission equipment.

To make better use of the bandwidth on these channels with long propagation delays, continuous or "pipelining" protocols are used. A pipelining type protocol allows the transmitter to send frames continuously without stopping and waiting for a response for each frame. There is a limit on how many frames can be sent without having received a response from the receiver for the first of the unacknowledged frames transmitted. This limit is referred to as a "sliding window" and is determined by the system capabilities and the propagation delay. For the most efficient operation, the window should be greater when the propagation delay is longer. [Ref. 2]

Go-Back-N (GBN) is a pipelining protocol which allows a transmitter to send up to N-1 frames before stopping and waiting for a response from the receiver. The receiver, in contrast to the transmitter, has a receive window equal to one frame. The receiver is looking to receive a specific frame and once that frame is received, it immediately looks for the next one. Typically, the receiver discards a frame received with errors. When a later frame arrives correctly and is out of sequence, or not the frame the receiver was expecting, the transmitter is notified that an error has occurred. When the transmitter learns that one of the frames was received with errors, it backs up and begins to transmit again starting with that frame. The beginning of the window *slides* up to the frame being retransmitted, so this frame is the first of N-1 possible frames in the new window. All the frames that were transmitted following the frame received in error are also retransmitted. If the protocol is operating in an environment which causes several consecutive packets to incur errors during transmission, a large number of frame retransmissions will result. Under these conditions, the transmission delays can be quite large and the protocol efficiency quite low.

Selective Repeat (SR) is another pipelining protocol. Similar to GBN, SR protocol uses a sliding window to allow N-1 frames to be sent without waiting for a response. However, SR protocol uses a different method of error correction than GBN. For the SR protocol, only frames received in error are retransmitted. Since the DLC must still provide the message in the proper form to the layer above, frames received without errors must be stored in a buffer while waiting for the retransmitted frames. This selective repeat feature adds complexity to the system and additional memory requirements which may be undesirable for some systems.

A short range, radio frequency (RF) network will often operate in a noisy environment with probability of bit error P_b rates in the range of 10^{-2} to 10^{-5}. The short distance for this RF network means the propagation delay is very small. When the frame processing time can also be kept small relative to the frame transmission time, the average delay in transferring a packet using a SW protocol is almost the same as using a GBN protocol. Considering the similar delay time and the fact that SW is a simple, easy to implement protocol with low storage requirements, then the SW protocol is often preferable for this type of RF network. [Ref. 3]

III. NETWORK CONFIGURATIONS AND PROTOCOL STRUCTURES

A. CONFIGURATIONS AND DATA TRANSFER MODES

Networks are configured in many different ways and require flexible data transfer modes and procedures that can handle a varied range of applications. Networks are generally configured as multi-point or point-to-point systems and operate in a full-duplex or half-duplex mode. Depending on the configuration and type of stations operating on the network, different types of data transfer modes may be used. For the multi-point configuration where only one station may be considered the primary station communicating with outlying secondary stations, the normal response mode (NRM) is often the best suited data transfer mode. In the NRM, a primary station controls the data transfer link and informs each secondary station on the network when it is allowed to transfer data. This mode is particularly useful in a polled operation where each secondary station has only the limited capability of responding to a poll and then transmitting or receiving data when directed. The primary station directs all the interaction between itself and the outlying stations, relays all the transmissions between stations, and performs all error recovery procedures on the link. This mode does have the disadvantage of high overhead associated with the polling operation. [Ref. 2]

To take advantage of secondary stations with slightly more capability, the asynchronous response mode (ARM) can be used to limit the polling overhead required. In the ARM, the secondary stations are capable of initiating responses without the strict direction from the primary station as in the NRM. In the ARM, there is still a primary station controlling the data link between itself and the remote secondary stations. The primary station is typically still superior in capability to the secondary stations and when communication with a secondary is desired, the secondary is activated. Once a secondary is activated, the data transfer between the primary and the secondary flows freely back and forth without any polling or direction required. The advantage of the ARM is the ability to communicate in an asynchronous manner between the primary and the activated secondary stations in the multi-point configuration, without the polling operation. [Ref. 4]

For point-to-point communications when stations have comparable capabilities to initiate data transfer and control the communication link, the asynchronous balanced mode (ABM) is a very effective mode of data transfer. In the ABM, the stations act as both the primary and the secondary stations during the data transfer. Therefore, stations operating in the ABM are called combined stations [Ref. 4].

In a radio frequency (RF) network, it is often desirable for stations to communicate over a full-duplex, point-to-point communications link. Both stations will often be of comparable capabilities and will operate as combined stations, so the data transfer is conducted in the ABM. For short range RF networks, the ABM is most efficient. This is the network of interest in this research.

B. STRUCTURE

A protocol establishes a specific frame structure and transfer procedures for the proper error-free transmission of information and control of the communications link. The protocol uses three different types of frames to perform the data transfer and the communications link control:

- Information frame (I frame).
- Supervisory frame (S frame).
- Unnumbered frame (U frame).

Information or I frames are used to transfer the data and can be of variable length depending on the length of the message packet. An I frame can also be used to return the appropriate acknowledgment for a frame received from the other station. Supervisory or S frames are used to control the data flow on the communications link and to provide message acknowledgments or rejections. They also can be used to recover from conditions causing errors. Unnumbered or U frames provide the necessary commands and responses to establish and terminate the communications session and to specify the various parameters and restrictions to be used during the session. Depending on the type of frame, two different formats can be used. The first format type, used for the S or U frames when no information is to be transmitted, is shown in Figure 3. The second format type, used for I frames when information is to be transmitted, is shown in Figure 4.

The starting and ending bits of both frame formats are delineated by the flag field. The flag field shown in Figure 5 is a unique 8-bit sequence from which the receiver can acquire frame synchronization. The bit sequence is a "0" bit followed by six "1" bits and another "0" bit. To ensure that the flag sequence is unique and not reproduced anywhere else in the frame, the transmitter uses a "bit stuffing" technique. In bit stuffing, the transmitter inserts a "0" bit after five consecutive "1" bits in all fields after the be-

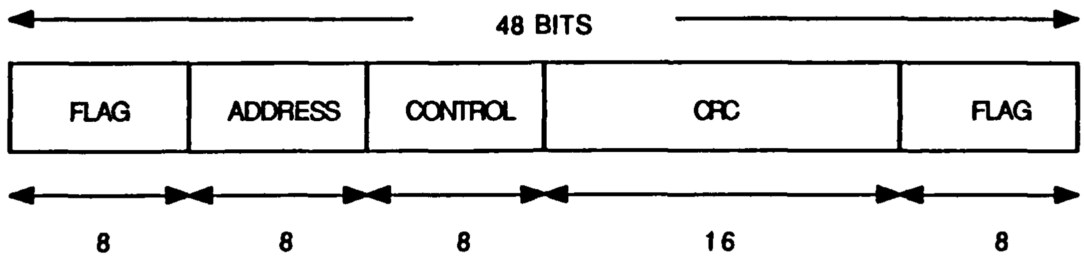


Figure 3. Supervisory and unnumbered frame format

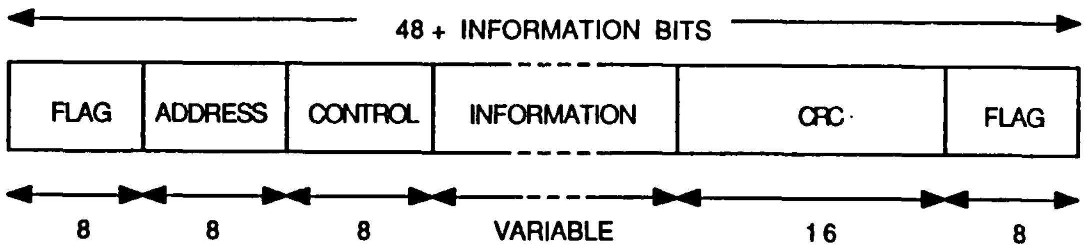


Figure 4. Information frame format

ginning flag and before the ending flag. The receiver, to "destuff" the frame, simply removes the "0" bit after five consecutive "1" bits for all bits received between the flag sequences. [Ref. 1]

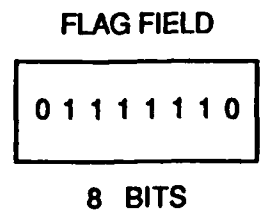


Figure 5. Flag field

The address field is typically eight bits or a byte in length and used to identify the transmitter or the receiver of the frame. For conditions where more than 256 addresses are required, the address field can be extended by any number of additional bytes. By convention, when using the extended addressing, all bytes preceding the final byte of the address field have the first bit set to "0" and the first bit of the final byte is set to "1" [Ref. 4].

In systems with multi-point configurations, the address field is used to identify the station receiving the frame. In point-to-point configurations, where the stations are combined stations, the address field is used to determine if the frame being transmitted contains a command or a response. All frames contain either a command or a response and are one of the three types of frames: I, S, or U frames. When a local station transmits a frame with the remote station specified in the address, that frame contains a command. When a station uses its own address, then the frame contains a response.

The control field is another eight bit field and this field actually identifies the type and function of the frame. The first bit of the control field is used to distinguish between an I frame and an S or U frame. The control field format is specified in Figure 6. If a frame is determined not to be an I frame, the second bit of the control field is used to distinguish between an S and a U frame.

An I frame is identified by a "0" in the first bit position of the control field. The next three bit positions of the I frame control field compose the send sequence number N(s) which identifies the frame. Only an I frame contains a send sequence number. The Poll/Final (P/F) is the fifth bit position of the control field in all the frame types and used to generate a checkpointing system. In the checkpointing system, the P/F bit is P

CONTROL FIELD BITS

CONTROL FIELD FOR	1	2	3	4	5	6	7	8
INFORMATION TRANSFER FRAME	0	N(s)			P/F	N(r)		
SUPERVISORY FRAME	1	0	TYPE		P/F	N(r)		
UNNUMBERED FRAME	1	1	TYPE		P/F	TYPE		

Figure 6. Control field format

for frames containing commands and F for the related response frame. When a station sends a frame containing a command and has the P bit set to "1", the remote station can only send a frame containing a response and must set the F bit to "1". The final three bit positions of an I frame and an S frame compose a receive sequence number N(r). By convention, the N(r) is the sequence number of the next expected frame and acknowledges receipt of all the previous frames with sequence numbers less than or equal to N(r)-1 [Ref. 2].

An S frame is identified by a "1" and a "0" in the first and second bit positions of the control field. The third and fourth bits of the control field encode the four *types* of S frames used for data flow control:

- Receive Ready (RR) indicates that the station is ready to receive I frames and is used to acknowledge I frames received when the station does not have information to transfer.
- Receive Not Ready (RNR) indicates that the station is unable to receive I frames at this time and acknowledges the I frames that have been received.
- REJect (REJ) is used to indicate that a transmission error has occurred and that retransmission is to begin with the frame indicated in the N(r) field.

- Selective REJect (SREJ) indicates a frame has been received in error and retransmission of this frame is requested.

The use of RR and RNR are required for a system to provide proper data flow control. REJ and SREJ are typically optional and best suited for pipelining protocols to enhance error recovery. The final three bit positions contain the receive sequence number N(r) which enable a station to use an S frame to acknowledge receipt of N(r)-1 frames when it has no data to transfer.

A U frame has the first bit of the control field set to "1" and is distinguished from the S frame by setting the second bit to "1". The next two and the last three bit positions of the control field are used to encode the commands and responses required for data link control. To initiate a communications session, a station will use a U frame SET Mode (SETM) command. The SETM command is used to represent the set of mode setting commands which specifies the mode of communications and the modulus used for the sequence numbering [Ref. 2]. The U frame response Disconnected Mode (DM) to the SETM command is used when a station is temporarily unable to participate in a communications session. The response to the SETM command when a station is ready for a communications session is the Unnumbered Acknowledgment (UA) response. The UA response is used also for acknowledging other U frame commands. To terminate a communications session, the U frame DISConnect (DISC) command is used. To recover from a frame error condition which can not be solved by retransmitting the frame, the FRaMe Reject (FRMR) response is used. A more detailed description of the types of U frames available can be found in Ref. 4.

The information field only exists in an I frame. The information field contains the information or the message packet to be transferred across the link and is typically of variable length. Since the information is composed of bits occurring in any order with no particular pattern, the bit stuffing technique must be used to ensure a bit sequence resembling a flag sequence is not inadvertently transmitted in the information field.

The Cyclic Redundancy Check (CRC) field of the frame is a 16-bit checksum used to detect bit errors between the flags of the frame. To determine the bit sequence in the CRC field, the transmitting station performs a long division on the frame's data bits (excluding the flag field) by a generator polynomial. The remainder resulting from this long division then becomes the bit sequence for the CRC field. The generator polynomials used to create this checksum are typically one of two types of polynomials:

- CRC-CCITT polynomial $x^{16} + x^{12} + x^5 + 1$.

- CRC-16 polynomial $x^{16} + x^{15} + x^2 + 1$.

The receiver, to determine if a frame has suffered any transmission errors, performs a similar long division using the same generator polynomial as the transmitter on all the fields of the frame between the flags (including the CRC field). If a frame is received without any transmission errors, the remainder of the long division will equal zero and the frame is accepted. If a non-zero remainder results from the long division, then a transmission error has occurred and the frame is not accepted. If a zero remainder is undesirable to satisfy the CRC, other methods of calculating the checksum can be used which results in the receiver obtaining a remainder when performing the division on the received frame. The receiver then compares this remainder to the bit sequence in the CRC field to determine if a transmission error has occurred. If the remainder and the bit sequence in the CRC field match, then the frame is accepted. If they do not match, then the frame has errors and is rejected. [Ref. 2]

IV. ADAPTIVE ARQ STRATEGY DEVELOPMENT

A. GENERAL CONSIDERATIONS

1. Throughput Efficiency

The effectiveness of information transfer of the typical ARQ protocol largely depends on the bit error rate of the channel. For a low and generally stable channel bit error rate, the optimum packet size for the maximum throughput efficiency can be determined. Throughput efficiency is defined as the amount of actual information transferred relative to the total number of bits transmitted. Since every packet transmitted has the same amount of overhead required by the ARQ protocol, the larger the information field, the better the packet efficiency. Unfortunately, as the packet size increases the probability of an error occurring during transmission increases correspondingly. Generally, the maximum throughput efficiency is realized by utilizing the optimum packet length for the link conditions and minimizing the number of retransmissions required. For long term changes in the channel bit error rate, the optimum packet lengths can be modified manually to maximize the throughput efficiency. If the channel bit error rate increases quickly, the typical ARQ strategy will continue to retransmit packets which were received in error, at the packet length determined to be optimum for the average channel bit error rate. For these typical ARQ strategies the throughput efficiency suffers greatly during the periods of operation with high bit error rate. [Ref. 5]

2. Signal Fading and Noise

The channel bit error rate does not often remain constant for a radio frequency (RF) system, particularly for a system affected by signal fading or subject to jamming. Destructive interference of the RF signal as a result of multi-path interference or jamming may result in an abrupt, large increase in the channel bit error rate. This condition may only last for several packet transmissions followed by an equally abrupt decrease in the bit error rate. A less destructive degradation of the channel, such as an increase in noise level, may result in a less abrupt and severe increase in the bit error rate. This noisy degradation of the channel may last longer than the quick destructive interference causing an increase of the bit error rate to persist for a large number of packet transmissions. Since the bit error rate does not remain constant over the channel and actually may vary widely, an ARQ strategy which adapts the packet length for a changing bit error rate would increase the throughput efficiency over one that does not.

Different types of signal fading conditions and channel noise dictate that different adaptive strategies be employed. Since many different channel conditions exist, developing a different strategy for each condition is nearly impossible and certainly impractical. Therefore, two strategies with some optional design variations were developed in an attempt to counteract the effects of signal fading and increased channel noise conditions.

B. ADAPTIVE STRUCTURE

1. Strategy Structure

An adaptive ARQ strategy decreases the length of the packets by decreasing the amount of data transferred in the information field. This reduction of data occurs for packets which have already failed a certain number of retransmission attempts indicating an increase in the bit error rate on the forward channel. Also, the lengths of packets not yet transmitted for the first time may be increased (up to a maximum size allowed), if the previous packets are transmitted without error, indicating that the forward channel bit error rate has decreased.

The structure for the adaptive strategies can be viewed as consisting of various levels where each level equates to a packet length as shown in Figure 7. The top level of the structure is associated with the largest information field and is typically the starting point for a communications session. Each level below the top level has an information field length associated with it which is less than the largest information field.

Each successive level is connected by a series of steps. The number of steps between different levels do not have to be the same. Once the structure is established, the number of steps are fixed for that structure. These steps connecting the levels determines how many retransmission attempts at a certain length are made prior to the modification of the packet length to the new length associated with the new level. A step up to the next step or level of the structure is made for each successful packet transmission until the top level is reached. For each packet which suffers an error in transmission, a step down is made until the bottom level is reached.

In Figure 7, the set of steps between two levels have a packet length equal to the length associated with the upper level. In this figure, to go from level L(0) to level L(1) requires four steps. This means that four consecutive attempts are made to transfer a packet with length equal to the maximum length associated with L(0). If four consecutive attempts fail to transfer the packet error-free, then level L(1) is reached and the packet length is modified prior to the next retransmission attempt.

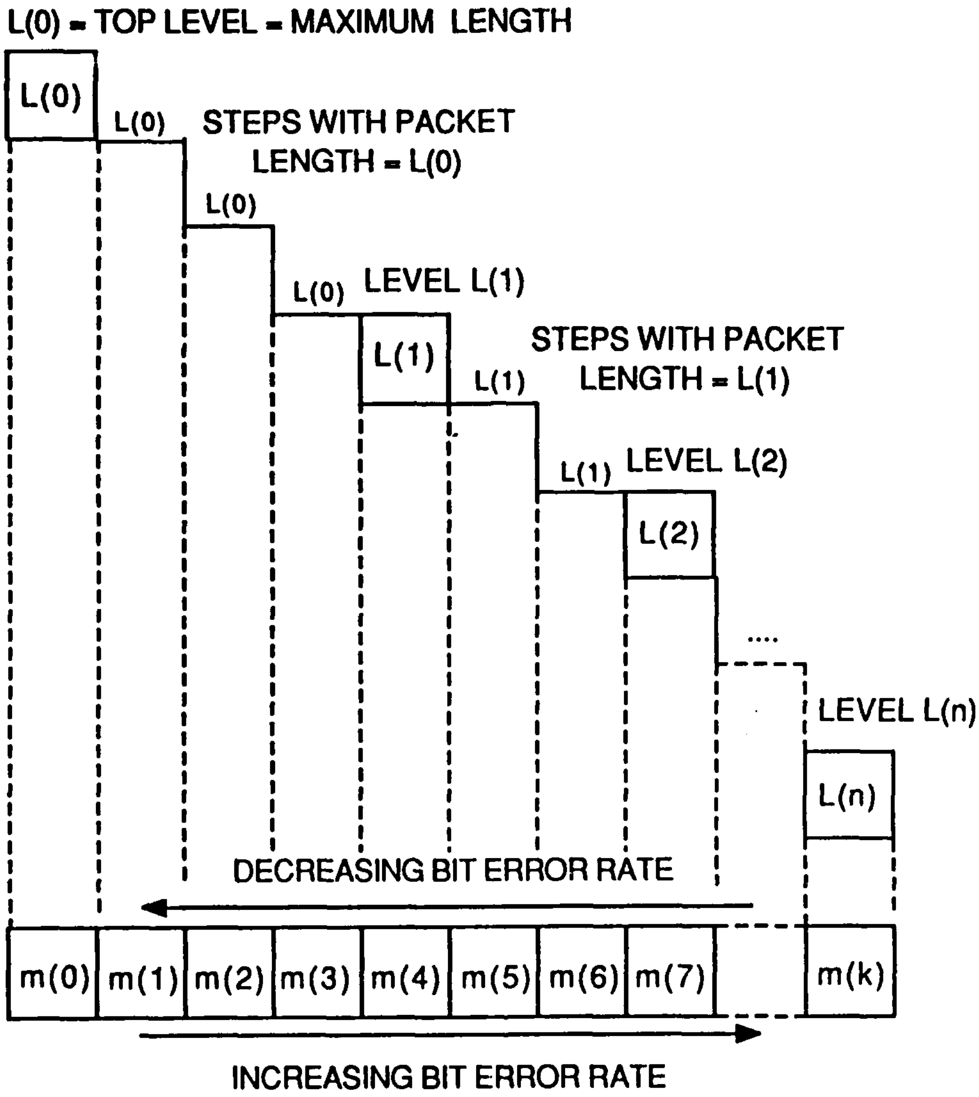


Figure 7. Adaptive strategy structure and state vector

The adaptive structure can be represented by a state vector **m**. Each of the elements or the states of the state vector m(i), i = 0,1,...,k, correspond to either one of the levels or the steps of the adaptive structure. The number of states (k + 1) in the state vector is equal to the number of levels and steps in the adaptive structure. Moving left or right to a new state in the vector corresponds to stepping up or down the adaptive structure respectively. The value of each state m(i) is the packet length associated with the corresponding level or step of the structure. Since some levels and steps have the same associated packet length, the values of some states will be the same. For example, as shown in Figure 7, when the current state is m(3), then the packet is transmitted at the length associated with level L(0). If the packet is received in error, then the state is updated to m(4) and the length is modified to that associated with L(1). [Ref. 6]

This state vector will be used by the station software to implement the adaptive strategy. The packet length to be transmitted by the station is determined by the value of the current state of the vector. The ends of the state vector correspond to the levels associated with the maximum and minimum packet lengths of the structure. When the state at the left-hand end of the vector is reached, this corresponds to reaching the top of the adaptive structure and the system state will remain at this vector element until a frame suffers an error during transmission. Likewise, when the right-hand end of the vector is reached, the state corresponds to the bottom of the structure and the frame is retransmitted at the minimum length associated with this state until it is successfully transferred.

An example of one alternative method of designing the steps between levels is shown in Figure 8. In this structure, a different set of steps or path is provided when ascending from one level to another. This strategy may be useful in the situations where additional packet transmissions at the length associated with the lower level are desired prior to modifying the packet lengths to the longer length of the level above. The steps ascending to the next level can be variable in number and may intersect any of the descending steps or go all the way up to the next level. The state vector **m** for this structure is made up of different segments. The states m(i) in each segment of the state vector correspond to one level and the steps which ascend and descend from that level, which all have the same associated packet length. As in the previous vector, moving left or right corresponds to stepping up and down the structure. For this vector, the arrows indicate which is the next state when moving from one segment to another. This method of transition between levels provides the system designer with added flexibility to optimize the system throughput efficiency.

2. Two-Level Adaptive Strategy Structure

When the signal is subjected to fading or jamming on the forward channel, there is a correspondingly large increase in the bit error rate which occurs quickly relative to the time required for packet transmission. Under these conditions, the packet length must be modified substantially to gain an increase in the information throughput. The type of adaptive strategy to counteract the conditions imposed by this large changing bit error rate is a *two-level* adaptive strategy, shown in Figure 9.

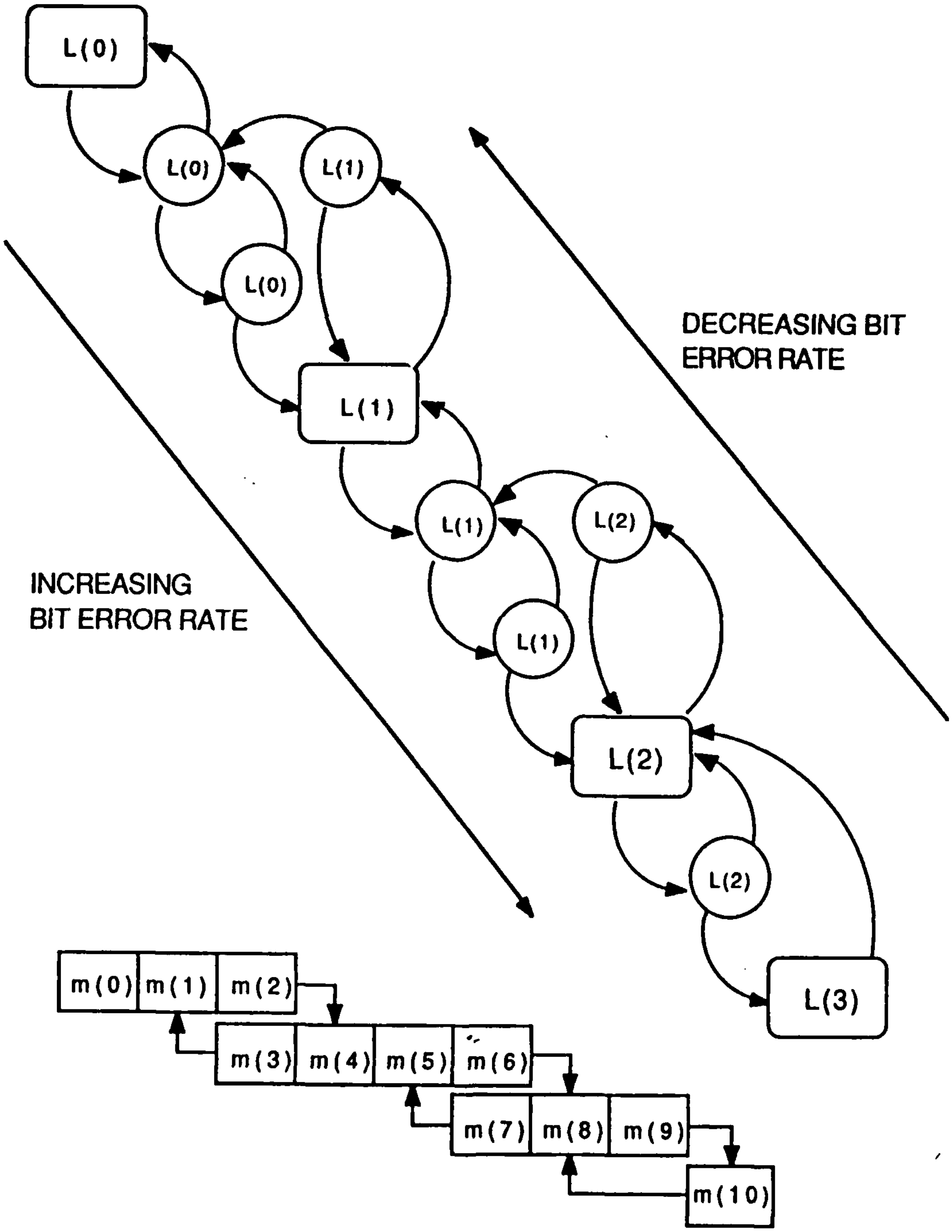
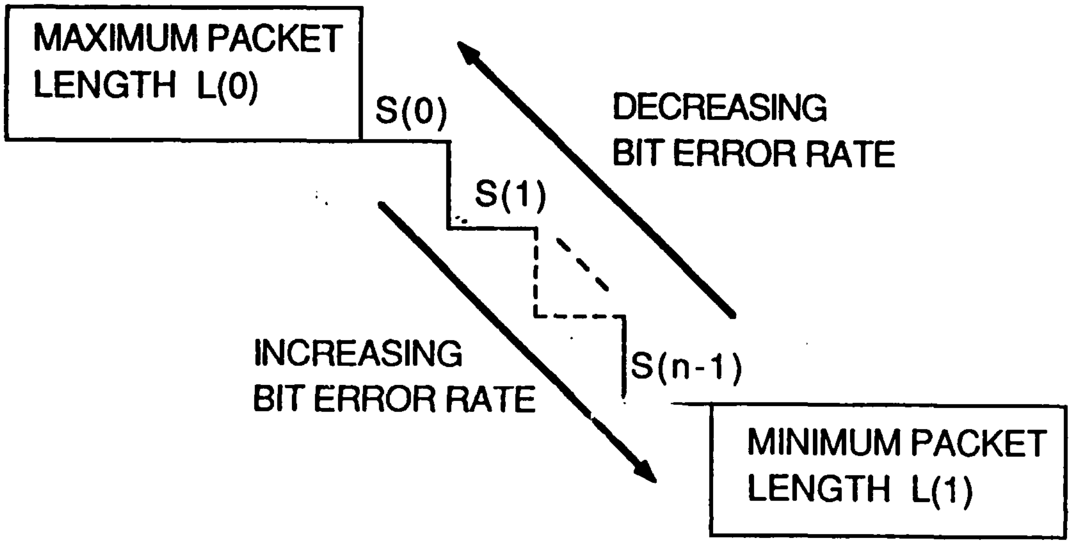


Figure 8. Alternative adaptive strategy structure design



S(0),S(1),...,S(n-1) = STEPS (NUMBER OF TRANSMISSION
ATTEMPTS TO CHANGE LEVELS)

Figure 9. Two level adaptive strategy

The structure of the two-level strategy, as the name implies, transmits only packets with two lengths. The top level is the maximum packet length and would typically be the length determined to be optimum for the average channel bit error rate of the system employed. The packet length associated with the lower level is the length determined to be optimum for the system during periods of operation under the high bit error rate conditions which occur most frequently. These channel bit error rates are usually not known for a system and must typically be determined from empirical testing of the system and results accumulated for long periods of time.

The steps connecting the levels which determine the number of retransmission attempts or the number of consecutive, successful packet transmissions prior to a length modification are completely variable. Using the alternative structure design, the number of steps to descend to the lower level can be specified to be different than the number of steps to ascend to the top level. This variability allows a system designer to specify how quickly the system will react to a perceived increase or decrease in the forward channel bit error rate. Determining the number of steps between the levels which optimizes the system throughput efficiency, similar to the level determination, is very system dependent. System testing under as many conditions as possible would be required to find the number of steps which optimizes the system throughput.

3. Multi-level Adaptive Strategy Structure

A different strategy is needed in the cases where varying noise, which may be modeled as Gaussian noise, causes an improvement or degradation of the channel quality resulting in a forward channel bit error rate which changes accordingly. The changing bit error rate may vary slowly relative to the time required for packet transmission and in smaller amounts than in the signal fading case. For this type of changing bit error rate condition the adaptive strategy must vary the packet length in smaller increments than the previous strategy in an attempt to optimize the packet length for the channel conditions. The type of adaptive strategy shown in Figure 10 is a *multi-level* adaptive strategy and will step or modify the packet length to various lengths associated with each level ranging from the maximum to the minimum packet length allowed.

Similar to the two-level strategy, the top level is associated with the maximum packet length allowed. This packet length is typically chosen to optimize the system operating with the most frequently occurring bit error rate. Because this strategy targets the cases where the changes in the bit error rate are less severe than in the two-level strategy, the information field lengths of each lower level are typically no less than one-half the length of the level above. The number of levels and steps in the multi-level

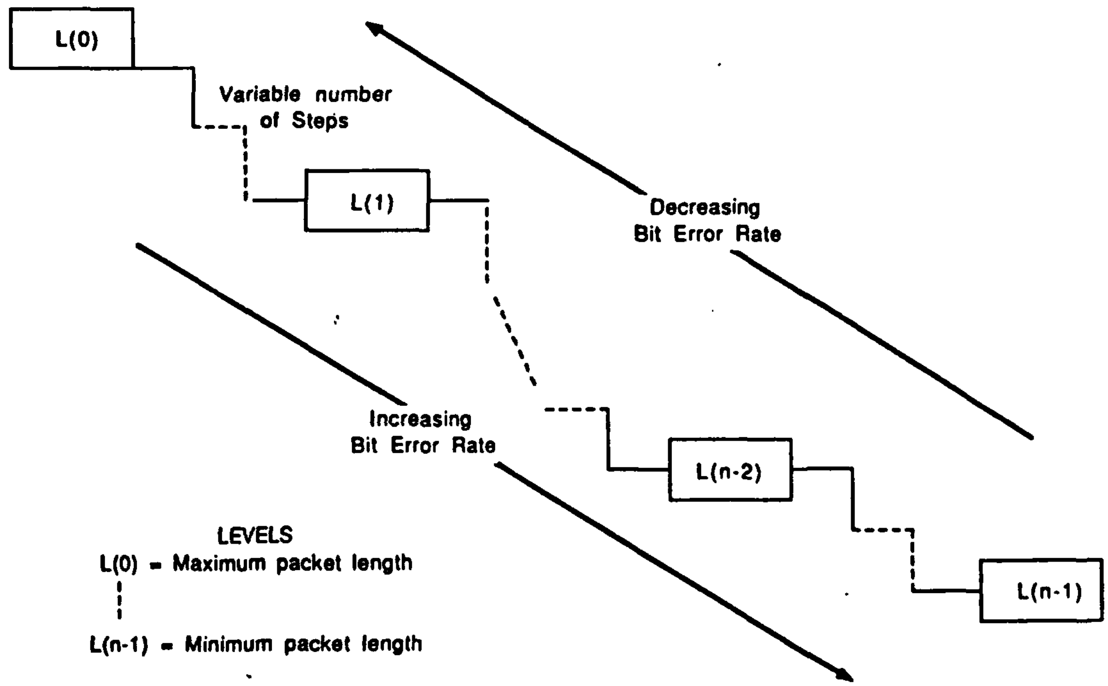


Figure 10. Multi-level adaptive strategy

strategy are completely variable. As the channel bit error rate increases, after a specified number of failed retransmission attempts of a packet with length greater than the minimum length allowed, the packet length is changed to the next lower level and retransmitted. This procedure is continued until the packet is successfully transmitted or the lowest level is reached. Once a specified number of packets at a certain level have been successfully transmitted, the packet length of the next packet to be transmitted is increased to the next level. As long as packets are transmitted successfully, stepping up the packet length to the next level continues until the highest level is reached and the packet length is the maximum for the system.

C. ADAPTIVE STRATEGY IMPLEMENTATION

The main feature of the adaptive strategies is the modification of the length of the message packet after a specified number of consecutive retransmission attempts or successful transmissions. All ARQ protocols require that the transmitting station store all transmitted packets until their successful transfer is acknowledged by the receiving station. Since the transmitted information is stored until acknowledged, it can be broken up into smaller packets to be retransmitted at lengths associated with the new level of the adaptive structure. For the case where the bit error rate decreases and the packet length is to be increased, the number of information bits in the information field is simply increased accordingly.

The addition of the adaptive feature is designed to be strictly a software modification which adds only some additional software overhead requirements to the transmitting station. The transmitting station must know what information has been successfully transferred, the information that is attempting to be transferred, and the beginning of the next block to be transferred. In addition, the number of attempted transfers at the current packet length, as well as the number of consecutive packets transmitted successfully, must be accounted for. The addition of a state vector for the two-level or multi-level strategy can take care of most of the additional packet accounting required. The state vector is used by the transmitting station software to implement the adaptive strategy and identifies the proper length of the packets to be transmitted at any particular time. The adaptive strategy can be designed to be as complex or as simple as the system designer determines necessary to optimize the system throughput.

The state vector is always updated when an acknowledgment for receipt or rejection of a transmitted packet is received. For an accepted packet, the state is updated by moving one position to the left in the vector and the next packet of information is assembled into a frame at the length specified by this new state and transmitted. If a frame is rejected, the state is updated by moving to the right and the new state specifies if the packet length is to be modified. When modification of the packet length is indicated prior to retransmitting the frame, the local station must verify that the receive sequence number N(r) of the remote station is the same as the send sequence number N(s) of the frame that the local station is attempting to transfer. If the received frame from the remote station which contained the negative acknowledgment (NAK) was received without error, then the verification is complete. But, if the frame which contained the NAK is received with errors or not received at all, the verification procedure can be accomplished through the use of a P/F exchange. A flow diagram of the verification procedure is shown in Figure 11. The local station transmits a supervisory command frame with the P set to "1", which forces the remote station to send a supervisory response frame with the F bit set to "1". The remote station also provides its current N(r) value in this response frame. Then if a modification of the packet length is required, the frame is reassembled with the new packet length, a new CRC is performed and the checksum is put in the CRC field. The frame is then retransmitted and the data transfer process continues.

D. ADAPTIVE STOP-AND-WAIT (SW) PROTOCOL

1. Adaptive SW Operations and Example Notation

Point-to-point communications using adaptive SW protocols operate in a very similar manner to typical communications with SW protocols [Ref. 1]. The sessions are initiated in the same manner and when operating in the asynchronous balanced mode (ABM), the information can flow in both directions [Ref. 2]. Since a SW protocol can be thought of as a GBN protocol with a send window equal to one, the modulus used for the sequence numbering sets N(s) and N(r) would be modulo 2. Acknowledgments (ACK) for accepted frames and negative acknowledgments (NAK) for rejected frames are returned by the N(r) value in the control field of the frame transmitted from the remote station. Error recovery from the frames that are lost and consequently never acknowledged, or discarded due to errors received in transmission, is handled by the time-out function. The minimum length of time set for the time-out function is typically equal to two times the propagation delay plus the transmission time required for the largest frame plus a small amount of time for processing delay and a small safety margin to help eliminate premature time-outs [Ref. 1].

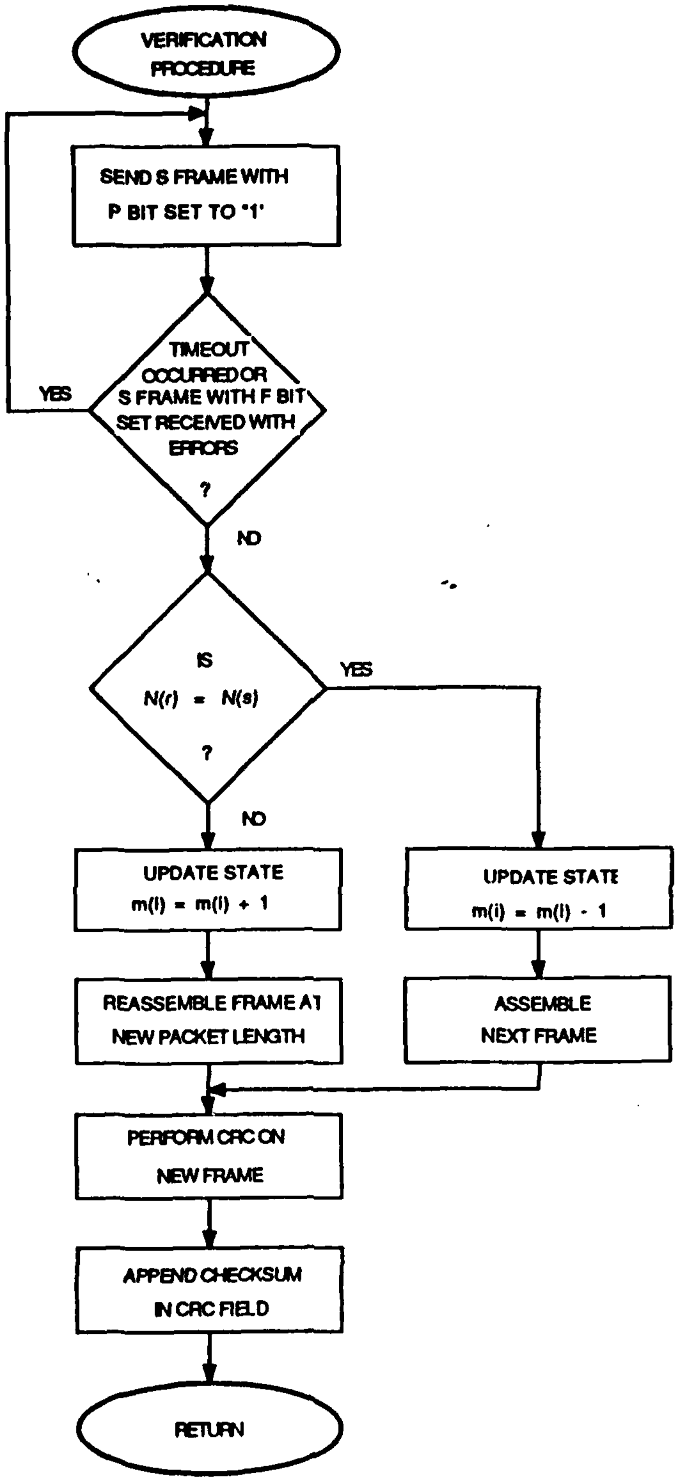


Figure 11. Verification procedure and packet length modification

Figure 12 and Figure 13 illustrate some typical operations for point-to-point communications sessions between two stations A and B. These examples are for short range, RF packet switching networks which may be subject to noise and signal fading so the probability P_b of bit error may be different for the channel from A to B than from B to A. Both stations are combined stations so the mode used for the data transfer is ABM. In these examples, I frames are represented by a single "I" symbol of various lengths for each frame. The maximum length for each frame is specified by the current state m(i) of the state vector **m**. The S and U frames are specified by the short, double "II" symbols and are all transmitted at a length of 48 bits. The length of the frame symbols represents the time required to transmit the frame, including the small amount of time used to process the frame. The slanted lines indicate the amount of propagation delay in the transmission of the frames. These lines are only slightly angled which reflects how small the propagation delay is relative to the frame transmission time for the short range system. The disruptions in some of the propagation lines, illustrated by a rotated "Z" symbol, indicate that the packet has received at least one error in transmission.

Often times a quick exchange of S or U frames is desired for a station to verify that the frame it is attempting to send is the same as the frame that the remote station is expecting to receive. This exchange can best be handled by using the P/F bit of the control field which initiates the checkpointing mechanism. To force the remote station to respond with either an S or U frame, the I frames are restricted to only containing commands. This restriction never allows an I frame to be used in responding to a received frame with the P bit set to "1".

The notation used in the examples of protocol operations to describe the frames is as follows: [Ref. 4]

L T N(s) N(r) **P/F**.

- **L** represents the address of the frame. For frames which contain commands, the address is specified as the station receiving the frame. For frames which contain responses, the address is the station sending the frame.
- **T** represents the abbreviation of the frame function. For I frames T is "I" and for S and U frames it is the specific function the frame performs (e.g., RR, RNR, SETM, UA, DISC, etc.).
- N(s) is the send sequence number and will only be used for I frames.
- N(r) is the receive sequence number and is present for all I and S frames.

- **P/F** represents either the P or F bit and is only present when the frame sets this bit to "1".

2. An Example of One-Way Information Transfer

Figure 12 illustrates an example of a point-to-point communications session between stations A and B with information being transferred only from A to B. The protocol established for these stations is multi-level adaptive SW, with each station using a four-level structure specified in Table 1. The adaptive structure is implemented by use of the state vector specified in row three of Table 1.

Table 1. ADAPTIVE SW STRUCTURE REPRESENTATION AND STATE VECTOR

Adaptive Structure Levels	L(0)	L(0)	L(0)	L(1)	L(2)	L(3)
Max Information Field Length (bits)	1000	1000	1000	500	250	125
State Vector	m(0)	m(1)	m(2)	m(3)	m(4)	m(5)

Station A initiates the communication session by sending a set mode command frame with the P bit set to "1". B is ready to receive data and responds with an Unnumbered Acknowledgement (UA) response frame. Using its own address in the frame, B is indicating that this is a response frame and the F bit is set to "1" in response to the P bit in the command frame from A. Upon transfer of the UA frame, B sets its N(s) and N(r) values to zero, initializes its current state to m(0), and enters the information transfer state. When A receives the UA response from B, its performs the same initializations and commences transferring I frames.

The stations commence transferring frames according to the adaptive SW protocol which requires they stop after each transmitted frame and wait for a response. Since B does not have any data to transfer to A during this session, B provides a response for each frame received utilizing a supervisory frame and the appropriate N(r) value. Essentially, B provides an acknowledgement (ACK) for each frame received without errors and a negative acknowledgement (NAK) for each frame received with errors. For the system errors caused by transmitted frames that are lost and not received, recovery is performed by activating the time-out function.

A transmits the first I frame BI00, at the length specified by the current state m(0). Since this frame is received at B with errors being detected, B responds with a

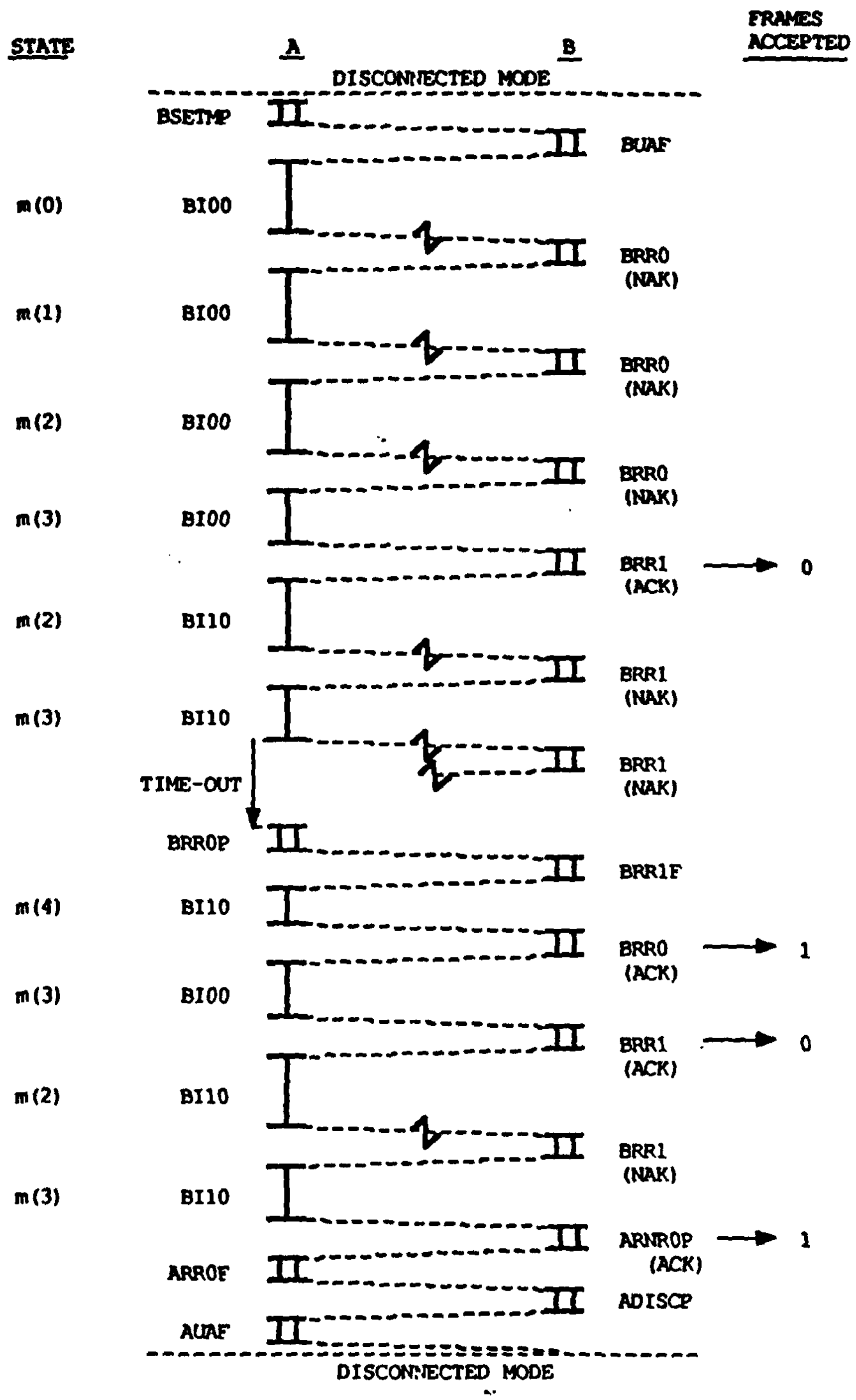


Figure 12. Adaptive SW - an example of one-way information transfer

NAK, by setting the N(r) value equal to N(s) value of the frame just received. Upon receipt of frame BRR0, A updates its state by moving one state to the right. The current state is now m(1) which specifies that the frame is to be retransmitted using the same maximum packet length constraint. The attempted data transfer continues and after the third NAK is received by A, an updating of the current state to m(3) specifies a modification to the frame length prior to retransmitting the frame. To modify the length, A divides the information field into two blocks, places the first block into the information field, performs a CRC on the new frame, appends the new checksum, and transmits BI00. The second block of information is now the next block of information waiting to be transferred.

After the successful transfer of BI00 and the ACK response is received from B, A updates its state to m(2). This state specifies that the maximum frame length is increased to the next level. Therefore A assembles and transmits frame BI10 at the length associated with level L(0). Frame BI10 is received at B with errors and B responds with a NAK. After updating its state to m(3), A must modify the frame length to the length associated with level L(1) in the same manner as before. A then retransmits the modified frame BI10. The frame BI10 again suffers transmission errors, so B responds with a NAK. Since the NAK sent by B is never received at A, the time-out function at A expires and a recovery action is initiated. In this case, A sends a supervisory type command frame BRR0P, with the P bit set to "1". This ensures the remote station B is still active and updates the N(r) value. B responds with BRR1F, with the F bit set to "1" in response to the set P bit of the previous frame.

Data transfer from A to B continues with the length of the I frames increasing and decreasing as specified by the state vector. After A has successfully transferred frame BI10, station B is interested in ending the communications session. B sends the command ARNR0P with the P bit set, indicating that B is not ready to receive any additional data from A. A responds with ARR0F indicating receipt of the command frame. B then initiates the disconnect procedures with station A. When A responds with the UA response, both stations enter the disconnect mode.

3. An Example of Two-Way Information Transfer

Figure 13 illustrates another example of a point-to-point communications session between stations A and B. In this example, the data link has already been established and both stations have information to transfer. Whenever both stations have information to transfer, the ACK(s) and NAK(s) for the received frames are returned with or "piggybacked" on the transmitted I frames. This case requires both local stations to receive, without errors, an I frame transmitted by the remote station to determine if the previous frame transmitted by the local station was accepted. A local station will discard frames received with errors and uses the time-out function to initiate a retransmission. For this situation, setting the optimal length of the time-out function is very important for maximizing the information throughput.

In this adaptive SW example, both stations A and B are using multi-level strategies with the adaptive structure represented by Table 2. As in the previous example, the state vector shown in the third row of Table 2 is used to implement the adaptive strategy.

Table 2. ADAPTIVE SW STRUCTURE AND STATE VECTOR

Adaptive Structure Levels	L(0)	L(0)	L(1)	L(2)	L(3)
Max Information Field Length (bits)	1000	1000	500	250	125
State Vector	m(0)	m(1)	m(2)	m(3)	m(4)

A sends frame BI00 at the frame length specified by the state m(0). Since the frame arrives at B suffering from errors which occurred during transmission, B waits for the time-out to expire before sending AI00. The frame AI00 sent by B provides a piggybacked negative acknowledgment displayed by the N(r) value remaining equal to "0". As B transmits the last bit of frame AI00, the time-out function is reset and started. Upon receipt of AI00, A performs the CRC on the frame and accepts the frame since no errors are detected. A then updates its state to m(1) and retransmits the previous frame at the same length. B detects errors in the received frame BI01 and again discards the frame. Since B is unaware that its previous frame AI00 was accepted, after the time-out expires, B updates its state and retransmits the frame.

A detects errors in the received frame AI00, and discards the frame. After the time-out expires, A updates its state to m(2) and determines that a packet length change is required for the next retransmission. Since the N(r) value at B has not been verified to be the same as the N(s) value at A following the last transmission of this packet, A transmits supervisory command frame BRR1P. This supervisory frame contains a command and with the P bit set to "1", forces B to respond. Since an I frame is restricted to containing a command only, B must use a S or U frame with the F bit set to "1" for the response. Limiting this checkpointing exchange to the 48 bit S or U frames provides

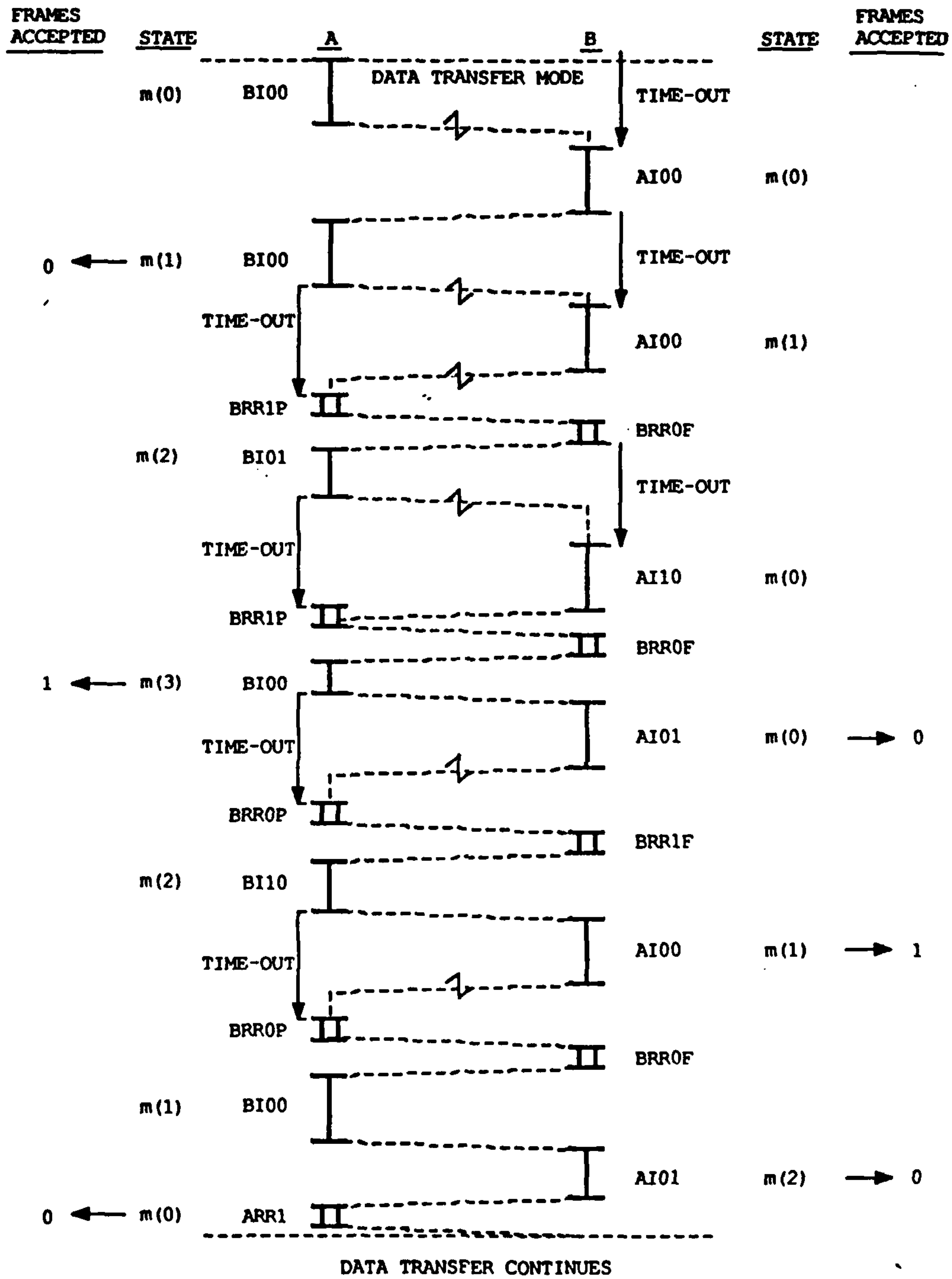


Figure 13. Adaptive SW - an example of two-way information transfer

high probability that the frames will be received without errors and the amount of time for the exchange is very short. When A receives the response frame BRR0F from B, the frame length is modified to the length specified by state m(2) using the same length modification procedures as described in the previous example. A then retransmits this modified frame at the new length and resets the time-out function.

Since the modified frame from A is received with errors, B discards the frame and waits for the time-out to expire before sending frame AI10, A's time-out expires prior to receiving frame AI10 from B and this causes A to initiate the necessary verification procedure prior to modifying the frame length for retransmission. When A receives the error-free frame AI10 from B, A updates the current state to m(3) and retransmits frame BI00 at the new frame length. B responds to the command frame BRR1P from A and prepares to retransmit the previous frame after receipt of the next frame. B then receives frame BI00 from A which indicates B's previous frame was accepted. B then updates the current state to m(0) and transmits frame AI01. The I frame AI01 is discarded by A once errors are detected and upon expiration of the time-out, A sends a supervisory command to determine B's N(r) value. When B responds with BRR1F, A updates its state to m(2) and transmits frame BI10.

The transfer of data continues and when a station no longer has any I frames to transfer, it responds to received frames from the remote station with supervisory frames and the appropriate N(r). If any new information should arrive and require transmission, it is put into I frames with lengths specified by the current state of the state vector and given the next N(s). If both stations run out of I frames to transfer, then one station will initiate the disconnect procedure as in the previous example. Once both stations acknowledge the disconnect commands, then they enter the disconnect mode.

V. ADAPTIVE SW SIMULATION

A. INTRODUCTION

The methods available to determine the throughput efficiency of any ARQ protocols, short of using or building an actual packet switching data communications network and collecting empirical data, are through analysis and computing numerical results or through simulation. The difficulty in performing the analysis of the system operating with the adaptive protocol is in being able to mathematically or statistically model the system properly. Often, to model all states and processes of the system successfully, certain assumptions and preconditions must be made. Many times the assumptions pose no restrictions on the results and the analysis results closely resemble actual system performance.

Simulation of the adaptive protocol provides another method of determining the throughput efficiency of the system. To simulate the adaptive protocol operating on a system, the system operation and parameters must be reproduced. In addition, the modeling of the statistical nature of the channel allows the simulation to reflect the actual efficiency of the system. As in analysis, assumptions and preconditions are made to perform the simulation and to the extent that these conditions are accurate, the simulation results may be indicative of actual system performance.

B. SIMULATION DEVELOPMENT

1. Model Design

The simulation model was designed to produce results from which the throughput efficiency of the system operating with one of the adaptive SW strategies could be computed. The model was not developed to show that the adaptive protocols worked correctly. The simulation modeled a point-to-point, RF, data communications system with only two stations. The model is for a system operating over a short distance where the propagation delay is very short relative to the time required for packet transmission. The bit rate for the data transfer is 4800 bits per second. The computer systems modeled in the simulation are capable of processing the packets with almost negligible delay and they are assumed to be dedicated to the data communications network. The RF channel connecting the stations is modeled to be a noisy channel, and the probability of bit error for the channel can be specified. To model the adaptive SW protocol, the packet sizes had to be controllable and the overhead associated with each packet held constant. The conditions used for the system in developing the simulation model are:

- Only one station transfers data.
- The station transmitting data is always saturated with information.
- The ACK(s) and NAK(s) are only 48 bits and always received without errors.
- Delays associated with the processing time are considered negligible and not used for the throughput efficiency calculations.
- The bit errors which occur during transmission are considered independent.

These conditions were applied to the model to provide results which could be used to calculate the throughput efficiency.

2. Simulation Program

The simulation for the data communications system was performed using the PC-NETWORK II.5 simulation software package from the CACI Products Company [Ref. 7]. This simulation package is intended to model computer system configurations and local area networks. Because the package has the capability to model the instructions required for the local area network, modeling some packet switching features was possible.

To simulate the adaptive SW protocol, two stations A and B, called processing elements (PE)s, were created. Within these PE(s) the instructions for sending the various sized frames, receiving data frames or acknowledgments, and sending ACK(s) or NAK(s) were developed. To simulate the channel connecting stations A and B, a transfer device (TD) was created. The parameters specified in the creation of the TD established the bit rate, the overhead associated with the parity checks for each word, and the control overhead for each frame. These TD parameters established the transmission time required for the data and control frames.

The software which simulates the ARQ adaptive SW protocol operating on the two PE(s) is in the form of software modules. A module contains a list of executable instructions to perform the various functions on a specified PE. Modules were created for each size packet required by the adaptive structure. Other modules were used to decide what size packet to send based on the current value of the state vector. Each module has a set of preconditions which must be satisfied prior to its execution. Through the setting of the preconditions for the modules, the software logic for the adaptive protocol could be implemented.

The module preconditions used to implement the adaptive protocol are described as message and semaphore based preconditions. The message preconditions require that a specified message be received prior to that condition being satisfied. The semaphore preconditions require that a set of semaphore values be checked and satisfied before execution of the module begins.

Semaphores are programming elements which can be used to "flag" certain events or conditions, or used as counters. Semaphores have a count associated with them which is incremented when the semaphore is set and decremented when reset. Semaphores are set or reset through the execution of the appropriate semaphore instruction. Through the use of semaphores, updating the state, which implements stepping up or down the adaptive structure, can be simulated.

The TD provided in the simulation package did not have any capability of simulating a noisy channel which would cause some frames to be received with errors. To simulate this noisy channel condition, a transmitter would send randomly a number of I frames which contain errors. This number of transmitted I frames which have errors would be a percentage of the total number of frames sent. The percentage represents the probability of a frame error P_e for a specified P_b and a frame length. Using the assumption that all bit errors are independent, the probability of a frame error P_e for a certain P_b of the channel is determined from

$$P_e = 1 - (1 - P_b)^{l+h} \tag{1}$$

where l specifies the information field size of the frame and h specifies the frame overhead. As can be seen from Equation (1), the P_e of an I frame decreases as the frame length decreases which means a lower percentage of I frames containing errors are transmitted. Since the percentages used to select the I frames which contain errors are set for a simulation run, the channel P_b had to also be set and not varied while performing the simulation.

C. SIMULATION RESULTS

The adaptive SW protocol for several different adaptive structures were simulated using the simulation model and program described in the previous section. In all simulations, the configuration was point-to-point and operated under the aforementioned assumptions.

These simulations provide results that are used to calculate the gain or loss in a station's information throughput efficiency ρ when using an adaptive SW strategy versus a non-adaptive SW protocol. The throughput efficiency for the simulation is defined as the number of information bits a station transfers divided by the total number it transmits. The simulation program used the output reports provided by PC-Network II.5 to record the total number of frames transmitted, the number of frames received without errors, and the number of frames received with errors.

To calculate the station throughput efficiency for a particular adaptive strategy, the number of information bits transferred is first computed. This result is obtained by summing the product of the number of frames successfully transferred for each different length times its respective frame efficiency times the number of bits in each frame. Frame efficiency is the ratio of the number of information bits to the total number of bits in a frame. Then the total number of bits transmitted by a station is computed. This number is obtained by summing the product of the number of frames transmitted for each different length times the number of bits in each frame. The throughput efficiency is obtained from the quotient of the number of bits transferred divided by the number of bits transmitted. The throughput efficiency results computed as described above for three different adaptive SW strategies and a non-adaptive SW protocol are shown in Figure 14.

Three adaptive strategies were used in the simulation and all three showed a higher throughput efficiency than the non-adaptive SW when the P_b was greater than 1×10^{-3}. The three adaptive strategies were different and the relative merit of each can be seen from the results. The results indicate that all the adaptive strategies are very similar in throughput efficiency when the channel P_b is less than about 1×10^{-3}. For data transfer operations when the P_b gets larger than 1×10^{-3} , the multi-level strategy with the state vector that was a single segment representing a structure with the same ascending and descending steps between levels, provided the best throughput efficiency. This strategy is referred to as "Multi-level Strategy 1" in Figure 14. The multi-level strategy with the state vector composed of multiple segments representing a structure with different ascending steps than descending steps between levels, indicated a ρ value consistently less than or equal to the multi-level strategy with a single segment state vector. This strategy is referred to as "Multi-level Strategy 2" in Figure 14. The two-level strategy indicated the worst ρ performance of the adaptive strategies, but as the P_b got very high, approaching 1×10^{-2} , the efficiency curve appeared to level off. This leveling off might indicate that the two-level adaptive strategy might be quite effective for large swings of the channel P_b.

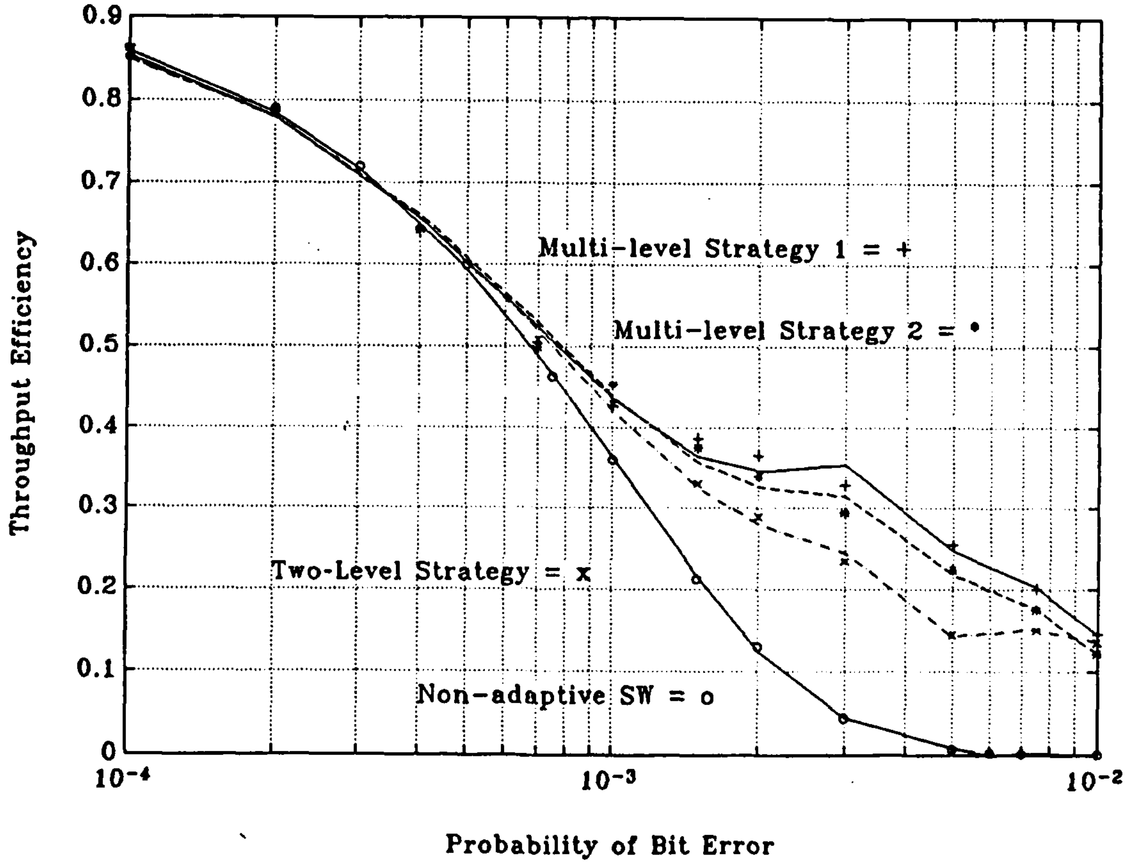


Figure 14. Simulation throughput efficiency vs probability of bit error P_b

VI. CONCLUSIONS

Adaptive SW protocols used for short range, RF networks performing packet switching data communications, where the propagation delays are short relative to the packet transmission time, can increase the information throughput efficiency over a typical SW protocol. As shown by the simulation results this improvement is particularly evident for a communication system operating in an environment with high channel P_b. The addition of the adaptive features to an existing SW protocol should be a fairly inexpensive modification as it is designed to be only a software modification with no additional hardware requirements.

Several different adaptive strategies were presented; all adapt the packet length as the channel P_b increases or decreases. To determine how many levels or steps to use in an adaptive structure for a system would require an analysis of the noise or interference of the specific system or empirical data which reflects system conditions. The different strategies were presented to provide a system designer with the ability to optimize a specific system for the particular noise or fading conditions experienced.

The simulation program used to model the adaptive SW protocol can possibly be modified to simulate other ARQ protocols. This would allow comparisons with other protocols in an attempt to further optimize the system.

APPENDIX A. SIMULATION INPUT FILES

CACI NETWORK II.5 RELEASE 4.01

4 Level Adaptive SW strategy with multi-segment state vector - Probability of Bit Error is 1×10^{-4}.

This is the input file generated on the CACI NETWORK II.5 editor NETIN, used to run the simulation on the NETWORK program.

```
* 4 Level Adaptive SW  strategy
* NETIN RELEASE 4.01

***** GLOBAL.FLAGS
GLOBAL FLAGS =
  RANDOMIZER =  3
  BATCH = NO

***** PROCESSING ELEMENTS - SYS.PE.SET
HARDWARE TYPE = PROCESSING
  NAME = STATION A
    BASIC CYCLE TIME =      1.000000 MICROSEC
    INPUT CONTROLLER = YES
    MESSAGE LIST SIZE = 640000.0
    LOSE OVERFLOW MESSAGES = NO
    INSTRUCTION REPERTOIRE =
      INSTRUCTION TYPE = MESSAGE
        NAME ; SEND GOOD DATA FRAME L(0)
          MESSAGE ; GOOD DATA
          LENGTH ; 1000 BITS
          DESTINATION PROCESSOR ; STATION B
          QUEUE FLAG ; YES
```

```
      RESUME FLAG ; YES
      ALLOWABLE BUSSES ;
        CHANNEL 1
    NAME ; SEND BAD DATA FRAME L(0)
      MESSAGE ; BAD DATA
      LENGTH ; 1000 BITS
      DESTINATION PROCESSOR ; STATION B
      QUEUE FLAG ; YES
      RESUME FLAG ; YES
      ALLOWABLE BUSSES ;
        CHANNEL 1
    NAME ; SEND GOOD DATA FRAME L(1)
      MESSAGE ; GOOD DATA
      LENGTH ; 500 BITS
      DESTINATION PROCESSOR ; STATION B
      QUEUE FLAG ; YES
      RESUME FLAG ; YES
      ALLOWABLE BUSSES ;
        CHANNEL 1
    NAME ; SEND BAD DATA FRAME L(1)
      MESSAGE ; BAD DATA
      LENGTH ; 500 BITS
      DESTINATION PROCESSOR ; STATION B
      QUEUE FLAG ; YES
      RESUME FLAG ; YES
      ALLOWABLE BUSSES ;
        CHANNEL 1
    NAME ; SEND GOOD DATA FRAME L(2)
      MESSAGE ; GOOD DATA
      LENGTH ; 250 BITS
      DESTINATION PROCESSOR ; STATION B
      QUEUE FLAG ; YES
      RESUME FLAG ; YES
      ALLOWABLE BUSSES ;
```

```
      CHANNEL 1
  NAME ; SEND BAD DATA FRAME L(2)
    MESSAGE ; BAD DATA
    LENGTH ; 250 BITS
    DESTINATION PROCESSOR ; STATION B
    QUEUE FLAG ; YES
    RESUME FLAG ; YES
    ALLOWABLE BUSSES ;
      CHANNEL 1
  NAME ; SEND GOOD DATA FRAME L(3)
    MESSAGE ; GOOD DATA
    LENGTH ; 125 BITS
    DESTINATION PROCESSOR ; STATION B
    QUEUE FLAG ; YES
    RESUME FLAG ; YES
    ALLOWABLE BUSSES ;
      CHANNEL 1
  NAME ; SEND BAD DATA FRAME L(3)
    MESSAGE ; BAD DATA
    LENGTH ; 125 BITS
    DESTINATION PROCESSOR ; STATION B
    QUEUE FLAG ; YES
    RESUME FLAG ; YES
    ALLOWABLE BUSSES ;
      CHANNEL 1
  NAME ; START DATA TRANSFER
    MESSAGE ; DATA
    LENGTH ; 1 BITS
    DESTINATION PROCESSOR ; STATION A
    QUEUE FLAG ; YES
    RESUME FLAG ; YES
INSTRUCTION TYPE = SEMAPHORE
  NAME ; A SET MESSAGE TRANSFERRED
    SEMAPHORE ; A MESSAGE TRANSFERRED
```

```
    SET/RESET FLAG ; SET
NAME ; A RESET MESSAGE TRANSFERRED
    SEMAPHORE ; A MESSAGE TRANSFERRED
    SET/RESET FLAG ; RESET
NAME ; A SET LOCNTR
    SEMAPHORE ; A LOCNTR
    SET/RESET FLAG ; SET
NAME ; A RESET LOCNTR
    SEMAPHORE ; A LOCNTR
    SET/RESET FLAG ; RESET
NAME ; A SET L1GOOD
    SEMAPHORE ; A L1GOOD
    SET/RESET FLAG ; SET
NAME ; A RESET L1GOOD
    SEMAPHORE ; A L1GOOD
    SET/RESET FLAG ; RESET
NAME ; A SET L1BAD
    SEMAPHORE ; A L1BAD
    SET/RESET FLAG ; SET
NAME ; A RESET L1BAD
    SEMAPHORE ; A L1BAD
    SET/RESET FLAG ; RESET
NAME ; A SET L2GOOD
    SEMAPHORE ; A L2GOOD
    SET/RESET FLAG ; SET
NAME ; A RESET L2GOOD
    SEMAPHORE ; A L2GOOD
    SET/RESET FLAG ; RESET
NAME ; A SET L2BAD
    SEMAPHORE ; A L2BAD
    SET/RESET FLAG ; SET
NAME ; A RESET L2BAD
    SEMAPHORE ; A L2BAD
    SET/RESET FLAG ; RESET
```

```
      NAME ; A SET L3GOOD
        SEMAPHORE ; A L3GOOD
        SET/RESET FLAG ; SET
      NAME ; A RESET L3GOOD
        SEMAPHORE ; A L3GOOD
        SET/RESET FLAG ; RESET
NAME = STATION B
  BASIC CYCLE TIME =      1.000000 MICROSEC
  INPUT CONTROLLER = YES
  MESSAGE LIST SIZE = 640000.0
  LOSE OVERFLOW MESSAGES = NO
  INSTRUCTION REPERTOIRE =
    INSTRUCTION TYPE = MESSAGE
      NAME ; SEND GOOD DATA FRAME L(0)
        MESSAGE ; GOOD DATA
        LENGTH ; 1000 BITS
        DESTINATION PROCESSOR ; STATION A
        QUEUE FLAG ; YES
        RESUME FLAG ; YES
        ALLOWABLE BUSSES ;
          CHANNEL 1
      NAME ; SEND BAD DATA FRAME L(0)
        MESSAGE ; BAD DATA
        LENGTH ; 1000 BITS
        DESTINATION PROCESSOR ; STATION A
        QUEUE FLAG ; YES
        RESUME FLAG ; YES
        ALLOWABLE BUSSES ;
          CHANNEL 1
      NAME ; SEND GOOD DATA FRAME L(1)
        MESSAGE ; GOOD DATA
        LENGTH ; 500 BITS
        DESTINATION PROCESSOR ; STATION A
        QUEUE FLAG ; YES
```

```
    RESUME FLAG ; YES
    ALLOWABLE BUSSES ;
      CHANNEL 1
  NAME ; SEND BAD DATA FRAME L(1)
    MESSAGE ; BAD DATA
    LENGTH ; 500 BITS
    DESTINATION PROCESSOR ; STATION A
    QUEUE FLAG ; YES
    RESUME FLAG ; YES
    ALLOWABLE BUSSES ;
      CHANNEL 1
INSTRUCTION TYPE = PROCESSING
  NAME ; DATA FRAME PROCESSING
    TIME ; 1100 CYCLES
  NAME ; START DATA TRANSFER
    TIME ; 1 CYCLES
INSTRUCTION TYPE = SEMAPHORE
  NAME ; SET MESSAGE TRANSFERRED
    SEMAPHORE ; MESSAGE TRANSFERRED
    SET/RESET FLAG ; SET
  NAME ; RESET MESSAGE TRANSFERRED
    SEMAPHORE ; MESSAGE TRANSFERRED
    SET/RESET FLAG ; RESET
  NAME ; SET LOCNTR
    SEMAPHORE ; LOCNTR
    SET/RESET FLAG ; SET
  NAME ; RESET LOCNTR
    SEMAPHORE ; LOCNTR
    SET/RESET FLAG ; RESET
  NAME ; SET L1GOOD
    SEMAPHORE ; L1GOOD
    SET/RESET FLAG ; SET
  NAME ; RESET L1GOOD
    SEMAPHORE ; L1GOOD
```

```
          SET/RESET FLAG ; RESET
        NAME ; SET L1BAD
          SEMAPHORE ; L1BAD
          SET/RESET FLAG ; SET
        NAME ; RESET L1BAD
          SEMAPHORE ; L1BAD
          SET/RESET FLAG ; RESET

***** BUSSES - SYS.BUS.SET
HARDWARE TYPE = DATA TRANSFER
  NAME = CHANNEL 1
    CYCLE TIME =    208.333000 MICROSEC
    BITS PER CYCLE =       1
    CYCLES PER WORD =       7
    WORDS PER BLOCK =     125
    WORD OVERHEAD TIME =    208.333000 MICROSEC
    BLOCK OVERHEAD TIME =   9999.990000 MICROSEC
    BUS CONNECTIONS =
      STATION A
      STATION B

***** MODULES - SYS.MODULE.SET
SOFTWARE TYPE = MODULE
  NAME = A TRANSFER GOOD DATA L(0)
    PRIORITY =     0
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    START TIME = 0.0
    ORED PREDECESSOR LIST =
      PROCESSOR 0
    REQUIRED SEMAPHORE STATUS =
      WAIT FOR ; A MESSAGE TRANSFERRED
      TO BE ; RESET
    INSTRUCTION LIST =
```

```
      EXECUTE A TOTAL OF ; 1 SEND GOOD DATA FRAME L(0)
      EXECUTE A TOTAL OF ; 1 A SET MESSAGE TRANSFERRED
      EXECUTE A TOTAL OF ; 1 A RESET LOCNTR
    ANDED SUCCESSORS =
      CHAIN TO ; PROCESSOR 0
      WITH ITERATIONS THEN CHAIN COUNT OF ;       0
  NAME = PROCESSOR 0
    PRIORITY =     1
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    START TIME = 0.0
    STATISTICAL SUCCESSOR STREAM =  21
    ORED PREDECESSOR LIST =
      A TRANSFER GOOD DATA L(0)
      A TRANSFER BAD DATA L(0)
      A START
      A COUNTER RESETTER 1
    REQUIRED MESSAGES =
      DATA*
    REQUIRED SEMAPHORE STATUS =
      CHAIN IF ; A LOCNTR
      IS ; < 3
    STATISTICAL SUCCESSORS =
      CHOOSE AS SUCCESSOR ;  82.00 % A TRANSFER GOOD DATA L(0)
      CHOOSE AS SUCCESSOR ;  18.00 % A TRANSFER BAD DATA L(0)
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 A RESET MESSAGE TRANSFERRED
  NAME = A TRANSFER BAD DATA L(0)
    PRIORITY =     0
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    ANDED PREDECESSOR LIST =
      PROCESSOR 0
    REQUIRED SEMAPHORE STATUS =
```

```
      WAIT FOR ; A MESSAGE TRANSFERRED
      TO BE ; RESET
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 SEND BAD DATA FRAME L(0)
      EXECUTE A TOTAL OF ; 1 A SET MESSAGE TRANSFERRED
      EXECUTE A TOTAL OF ; 1 A SET LOCNTR
    ANDED SUCCESSORS =
      CHAIN TO ; PROCESSOR 0
      WITH ITERATIONS THEN CHAIN COUNT OF ;       0
      CHAIN TO ; PROCESSOR 1
      WITH ITERATIONS THEN CHAIN COUNT OF ;       0
  NAME = A START
    PRIORITY =     9
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    START TIME = 0.0
    ALLOWED PROCESSORS =
      STATION A
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 START DATA TRANSFER
    ANDED SUCCESSORS =
      CHAIN TO ; PROCESSOR 0
      WITH ITERATIONS THEN CHAIN COUNT OF ;       0
  NAME = A TRANSFER GOOD DATA L(1)
    PRIORITY =     0
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    STATISTICAL SUCCESSOR STREAM =   1
    ANDED PREDECESSOR LIST =
      PROCESSOR 1
    REQUIRED SEMAPHORE STATUS =
      WAIT FOR ; A MESSAGE TRANSFERRED
      TO BE ; RESET
    INSTRUCTION LIST =
```

```
    EXECUTE A TOTAL OF ; 1 A SET L1GOOD
    EXECUTE A TOTAL OF ; 1 SEND GOOD DATA FRAME L(1)
    EXECUTE A TOTAL OF ; 1 A SET MESSAGE TRANSFERRED
    EXECUTE A TOTAL OF ; 1 A RESET L1BAD
  ANDED SUCCESSORS =
    CHAIN TO ; A COUNTER RESETTER 1
    WITH ITERATIONS THEN CHAIN COUNT OF ;       0
    CHAIN TO ; PROCESSOR 1
    WITH ITERATIONS THEN CHAIN COUNT OF ;       0
NAME = PROCESSOR 1
  PRIORITY =     4
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = NO
  STATISTICAL SUCCESSOR STREAM =  35
  ORED PREDECESSOR LIST =
    A TRANSFER BAD DATA L(0)
    A TRANSFER GOOD DATA L(1)
    A TRANSFER BAD DATA L(1)
    A COUNTER RESETTER 2
  REQUIRED MESSAGES =
    DATA*
  REQUIRED SEMAPHORE STATUS =
    CHAIN IF ; A LOCNTR
    IS ; > 2
    CHAIN IF ; A L1BAD
    IS ; < 3
    CHAIN IF ; A L1GOOD
    IS ; < 3
  STATISTICAL SUCCESSORS =
    CHOOSE AS SUCCESSOR ;  90.00 % A TRANSFER GOOD DATA L(1)
    CHOOSE AS SUCCESSOR ;  10.00 % A TRANSFER BAD DATA L(1)
  INSTRUCTION LIST =
    EXECUTE A TOTAL OF ; 1 A RESET MESSAGE TRANSFERRED
NAME = A TRANSFER BAD DATA L(1)
```

```
    PRIORITY =      0
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    STATISTICAL SUCCESSOR STREAM =    1
    ANDED PREDECESSOR LIST =
      PROCESSOR 1
    REQUIRED SEMAPHORE STATUS =
      WAIT FOR ; A MESSAGE TRANSFERRED
      TO BE ; RESET
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 SEND BAD DATA FRAME L(1)
      EXECUTE A TOTAL OF ; 1 A SET MESSAGE TRANSFERRED
      EXECUTE A TOTAL OF ; 1 A RESET L1GOOD
      EXECUTE A TOTAL OF ; 1 A SET L1BAD
    ANDED SUCCESSORS =
      CHAIN TO ; PROCESSOR 1
      WITH ITERATIONS THEN CHAIN COUNT OF ;       0
      CHAIN TO ; PROCESSOR 2
      WITH ITERATIONS THEN CHAIN COUNT OF ;       0
  NAME = A COUNTER RESETTER 1
    PRIORITY =      5
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    ORED PREDECESSOR LIST =
      A TRANSFER GOOD DATA L(1)
    REQUIRED SEMAPHORE STATUS =
      CHAIN IF ; A LOCNTR
      IS ; > 2
      CHAIN IF ; A L1GOOD
      IS ; > 2
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 3 A RESET LOCNTR
      EXECUTE A TOTAL OF ; 3 A RESET L1GOOD
    ANDED SUCCESSORS =
```

```
    CHAIN TO ; PROCESSOR 0
    WITH ITERATIONS THEN CHAIN COUNT OF ;      0
NAME = A COUNTER RESETTER 2
  PRIORITY =     5
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = NO
  ORED PREDECESSOR LIST =
    A TRANSFER GOOD DATA L(2)
  REQUIRED SEMAPHORE STATUS =
    CHAIN IF ; A L1BAD
    IS ; > 2
    CHAIN IF ; A L2GOOD
    IS ; > 1
  INSTRUCTION LIST =
    EXECUTE A TOTAL OF ; 3 A RESET L1BAD
    EXECUTE A TOTAL OF ; 2 A RESET L2GOOD
  ANDED SUCCESSORS =
    CHAIN TO ; PROCESSOR 1
    WITH ITERATIONS THEN CHAIN COUNT OF ;      0
NAME = B TRANSFER GOOD DATA L(0)
  PRIORITY =     0
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = NO
  STATISTICAL SUCCESSOR STREAM =   7
  ANDED PREDECESSOR LIST =
    B PROCESSOR 0
  REQUIRED SEMAPHORE STATUS =
    WAIT FOR ; MESSAGE TRANSFERRED
    TO BE ; RESET
  INSTRUCTION LIST =
    EXECUTE A TOTAL OF ; 1 SEND GOOD DATA FRAME L(0)
    EXECUTE A TOTAL OF ; 1 SET MESSAGE TRANSFERRED
    EXECUTE A TOTAL OF ; 1 RESET LOCNTR
  ANDED SUCCESSORS =
```

```
      CHAIN TO ; B PROCESSOR 0
      WITH ITERATIONS THEN CHAIN COUNT OF ;      0
  NAME = B PROCESSOR 0
    PRIORITY =     1
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    ORED PREDECESSOR LIST =
      B TRANSFER GOOD DATA L(0)
      B TRANSFER BAD DATA L(0)
      B START
      B COUNTER RESETTER
    REQUIRED MESSAGES =
      DATA*
    REQUIRED SEMAPHORE STATUS =
      CHAIN IF ; LOCNTR
      IS ; < 3
    STATISTICAL SUCCESSORS =
      CHOOSE AS SUCCESSOR ;  82.00 % B TRANSFER GOOD DATA L(0)
      CHOOSE AS SUCCESSOR ;  18.00 % B TRANSFER BAD DATA L(0)
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 RESET MESSAGE TRANSFERRED
  NAME = B TRANSFER BAD DATA L(0)
    PRIORITY =     0
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    STATISTICAL SUCCESSOR STREAM =   5
    ANDED PREDECESSOR LIST =
      B PROCESSOR 0
    REQUIRED SEMAPHORE STATUS =
      WAIT FOR ; MESSAGE TRANSFERRED
      TO BE ; RESET
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 SEND BAD DATA FRAME L(0)
      EXECUTE A TOTAL OF ; 1 SET MESSAGE TRANSFERRED
```

```
      EXECUTE A TOTAL OF ; 1 SET LOCNTR
    ANDED SUCCESSORS =
      CHAIN TO ; B PROCESSOR 0
      WITH ITERATIONS THEN CHAIN COUNT OF ;       0
      CHAIN TO ; B PROCESSOR 1
      WITH ITERATIONS THEN CHAIN COUNT OF ;       0
  NAME = B START
    PRIORITY =     9
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    START TIME = 0.0
    ALLOWED PROCESSORS =
      STATION B
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 START DATA TRANSFER
    ANDED SUCCESSORS =
      CHAIN TO ; B PROCESSOR 0
      WITH ITERATIONS THEN CHAIN COUNT OF ;       0
  NAME = B TRANSFER GOOD DATA L(1)
    PRIORITY =     0
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    STATISTICAL SUCCESSOR STREAM =   3
    ANDED PREDECESSOR LIST =
      B PROCESSOR 1
    REQUIRED SEMAPHORE STATUS =
      WAIT FOR ; MESSAGE TRANSFERRED
      TO BE ; RESET
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 SEND GOOD DATA FRAME L(1)
      EXECUTE A TOTAL OF ; 1 SET MESSAGE TRANSFERRED
      EXECUTE A TOTAL OF ; 1 SET L1GOOD
    ANDED SUCCESSORS =
      CHAIN TO ; B PROCESSOR 1
```

```
    WITH ITERATIONS THEN CHAIN COUNT OF ;       0
    CHAIN TO ; B COUNTER RESETTER
    WITH ITERATIONS THEN CHAIN COUNT OF ;       0
NAME = B PROCESSOR 1
  PRIORITY =      1
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = NO
  ORED PREDECESSOR LIST =
    B TRANSFER BAD DATA L(0)
    B TRANSFER GOOD DATA L(1)
    B TRANSFER BAD DATA L(1)
  REQUIRED MESSAGES =
    DATA*
  REQUIRED SEMAPHORE STATUS =
    CHAIN IF ; LOCNTR
    IS ; > 2
    CHAIN IF ; L1GOOD
    IS ; < 3
  STATISTICAL SUCCESSORS =
    CHOOSE AS SUCCESSOR ;  90.00 % B TRANSFER GOOD DATA L(1)
    CHOOSE AS SUCCESSOR ;  10.00 % B TRANSFER BAD DATA L(1)
  INSTRUCTION LIST =
    EXECUTE A TOTAL OF ; 1 RESET MESSAGE TRANSFERRED
NAME = B TRANSFER BAD DATA L(1)
  PRIORITY =      0
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = NO
  STATISTICAL SUCCESSOR STREAM =   9
  ANDED PREDECESSOR LIST =
    B PROCESSOR 1
  REQUIRED SEMAPHORE STATUS =
    WAIT FOR ; MESSAGE TRANSFERRED
    TO BE ; RESET
  INSTRUCTION LIST =
```

```
      EXECUTE A TOTAL OF ; 1 SEND BAD DATA FRAME L(1)
      EXECUTE A TOTAL OF ; 1 SET MESSAGE TRANSFERRED
      EXECUTE A TOTAL OF ; 1 RESET L1GOOD
    ANDED SUCCESSORS =
      CHAIN TO ; B PROCESSOR 1
      WITH ITERATIONS THEN CHAIN COUNT OF ;       0
  NAME = B COUNTER RESETTER
    PRIORITY =     5
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    ORED PREDECESSOR LIST =
      B TRANSFER GOOD DATA L(1)
    REQUIRED SEMAPHORE STATUS =
      CHAIN IF ; LOCNTR
      IS ; > 2
      CHAIN IF ; L1GOOD
      IS ; > 2
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 3 RESET LOCNTR
      EXECUTE A TOTAL OF ; 3 RESET L1GOOD
    ANDED SUCCESSORS =
      CHAIN TO ; B PROCESSOR 0
      WITH ITERATIONS THEN CHAIN COUNT OF ;       0
  NAME = A TRANSFER GOOD DATA L(2)
    PRIORITY =     0
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    STATISTICAL SUCCESSOR STREAM =   9
    ANDED PREDECESSOR LIST =
      PROCESSOR 2
    REQUIRED SEMAPHORE STATUS =
      WAIT FOR ; A MESSAGE TRANSFERRED
      TO BE ; RESET
    INSTRUCTION LIST =
```

```
      EXECUTE A TOTAL OF ; 1 SEND GOOD DATA FRAME L(2)
      EXECUTE A TOTAL OF ; 1 A SET MESSAGE TRANSFERRED
      EXECUTE A TOTAL OF ; 1 A SET L2GOOD
      EXECUTE A TOTAL OF ; 1 A RESET L2BAD
    ANDED SUCCESSORS =
      CHAIN TO ; A COUNTER RESETTER 2
      WITH ITERATIONS THEN CHAIN COUNT OF ;      0
      CHAIN TO ; PROCESSOR 2
      WITH ITERATIONS THEN CHAIN COUNT OF ;      0
  NAME = A TRANSFER BAD DATA L(2)
    PRIORITY =     0
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    STATISTICAL SUCCESSOR STREAM =   8
    ANDED PREDECESSOR LIST =
      PROCESSOR 2
    REQUIRED SEMAPHORE STATUS =
      WAIT FOR ; A MESSAGE TRANSFERRED
      TO BE ; RESET
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 SEND BAD DATA FRAME L(2)
      EXECUTE A TOTAL OF ; 1 A SET MESSAGE TRANSFERRED
      EXECUTE A TOTAL OF ; 1 A RESET L2GOOD
      EXECUTE A TOTAL OF ; 1 A SET L2BAD
    ANDED SUCCESSORS =
      CHAIN TO ; PROCESSOR 2
      WITH ITERATIONS THEN CHAIN COUNT OF ;      0
      CHAIN TO ; PROCESSOR 3
      WITH ITERATIONS THEN CHAIN COUNT OF ;      0
  NAME = A COUNTER RESETTER 3
    PRIORITY =     9
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    ORED PREDECESSOR LIST =
```

```
    A TRANSFER GOOD DATA L(3)
  REQUIRED SEMAPHORE STATUS =
    CHAIN IF ; A L2BAD
    IS ; > 1
    CHAIN IF ; A L3GOOD
    IS ; > 0
  INSTRUCTION LIST =
    EXECUTE A TOTAL OF ; 2 A RESET L2BAD
    EXECUTE A TOTAL OF ; 1 A RESET L3GOOD
  ANDED SUCCESSORS =
    CHAIN TO ; PROCESSOR 2
    WITH ITERATIONS THEN CHAIN COUNT OF ;      0
NAME = PROCESSOR 2
  PRIORITY =     1
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = NO
  STATISTICAL SUCCESSOR STREAM =  12
  ORED PREDECESSOR LIST =
    A TRANSFER BAD DATA L(1)
    A TRANSFER GOOD DATA L(2)
    A TRANSFER BAD DATA L(2)
    A COUNTER RESETTER 3
  REQUIRED MESSAGES =
    DATA*
  REQUIRED SEMAPHORE STATUS =
    CHAIN IF ; A L1BAD
    IS ; > 2
    CHAIN IF ; A L2GOOD
    IS ; < 2
    CHAIN IF ; A L2BAD
    IS ; < 2
  STATISTICAL SUCCESSORS =
    CHOOSE AS SUCCESSOR ;  95.00 % A TRANSFER GOOD DATA L(2)
    CHOOSE AS SUCCESSOR ;   5.00 % A TRANSFER BAD DATA L(2)
```

```
  INSTRUCTION LIST =
    EXECUTE A TOTAL OF ; 1 A RESET MESSAGE TRANSFERRED
NAME = PROCESSOR 3
  PRIORITY =     1
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = NO
  STATISTICAL SUCCESSOR STREAM =  45
  ORED PREDECESSOR LIST =
    A TRANSFER BAD DATA L(2)
    A TRANSFER GOOD DATA L(3)
    A TRANSFER BAD DATA L(3)
  REQUIRED MESSAGES =
    DATA*
  REQUIRED SEMAPHORE STATUS =
    CHAIN IF ; A L2BAD
    IS ; > 1
    CHAIN IF ; A L3GOOD
    IS ; < 1
  STATISTICAL SUCCESSORS =
    CHOOSE AS SUCCESSOR ;  97.50 % A TRANSFER GOOD DATA L(3)
    CHOOSE AS SUCCESSOR ;   2.50 % A TRANSFER BAD DATA L(3)
  INSTRUCTION LIST =
    EXECUTE A TOTAL OF ; 1 A RESET MESSAGE TRANSFERRED
NAME = A TRANSFER GOOD DATA L(3)
  PRIORITY =     0
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = NO
  ANDED PREDECESSOR LIST =
    PROCESSOR 3
  REQUIRED SEMAPHORE STATUS =
    WAIT FOR ; A MESSAGE TRANSFERRED
    TO BE ; RESET
  INSTRUCTION LIST =
    EXECUTE A TOTAL OF ; 1 SEND GOOD DATA FRAME L(3)
```

```
      EXECUTE A TOTAL OF ; 1 A SET MESSAGE TRANSFERRED
      EXECUTE A TOTAL OF ; 1 A SET L3GOOD
    ANDED SUCCESSORS =
      CHAIN TO ; A COUNTER RESETTER 3
      WITH ITERATIONS THEN CHAIN COUNT OF ;      0
      CHAIN TO ; PROCESSOR 3
      WITH ITERATIONS THEN CHAIN COUNT OF ;      0
  NAME = A TRANSFER BAD DATA L(3)
    PRIORITY =     0
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    ANDED PREDECESSOR LIST =
      PROCESSOR 3
    REQUIRED SEMAPHORE STATUS =
      WAIT FOR ; A MESSAGE TRANSFERRED
      TO BE ; RESET
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 SEND BAD DATA FRAME L(3)
      EXECUTE A TOTAL OF ; 1 A SET MESSAGE TRANSFERRED
      EXECUTE A TOTAL OF ; 1 A RESET L3GOOD
    ANDED SUCCESSORS =
      CHAIN TO ; PROCESSOR 3
      WITH ITERATIONS THEN CHAIN COUNT OF ;      0
```

APPENDIX B. SIMULATION FLOW DIAGRAMS

```
                    CACI NETIN            RELEASE 4.01

               STATION A - MULTI LEVEL ADAPTIVE ARQ STRATEGY

                     ***************
                    *              *  STATION A
                   *     A START    *
                   *                * START DATA TRANS
                   *                *
                   *                *
                    *              *
                     ***************   DATA
                               |
                 CI:A LOCNTR:I|     ST: 0.  MIC
                               |       DATA*
                    ORED.PRED v       /
                     ***************
                    *              *
                   *   PROCESSOR 0  *
                   *                * A RESET MESSAGE
                   *                *
                   *                *
                    *              *
                     ***************   A MESSAGE TR:R
                               |
                               |
          ---------------------------------------
          |BER %                                 |BER %
          |                                      |
          |     ST: 0.  MIC                      |
          |         A MESSAGE TRAN:R             |          A MESSAGE TRAN:R
          |        /                             |         /
 ANDED.PRED v     /                     ANDED.PRED v      /
       ***************                       ***************
      *              *                      *              *
     *   A TRANSFER   *                    *   A TRANSFER   *
     *   GOOD DATA    * SEND GOOD DATA FR  *   BAD DATA     * SEND BAD DATA FR
     *     L(0)       * A SET MESSAGE TR   *     L(0)       * A SET MESSAGE TR
     *                * A RESET LOCNTR     *                * A SET LOCNTR
      *              *                      *              *
       ***************                       ***************
PROCESSOR|           \                PROCESSOR|     |     \
    0    |            GOOD DATA           0    |     |      BAD DATA
    |    |            A MESSAGE TR:S      |    |     |      A MESSAGE TR:S
    |____|            A LOCNTR:R          |____|     |      A LOCNTR:S
                      etc.                           |      etc.
                                                     |
                                                     |
                                                     |
                                                     v
                                              TO PROCESSOR 1
```

```
                                  |
                   CI:A LOCNTR:I|
                    CI:A L1BAD:I|         DATA*
                   CI:A L1GOOD:I|            /
                                |           /
                      ORED.PRED v          /
                        ***************
                       *               *
                      *   PROCESSOR 1   *
                      *                 * A RESET MESSAGE
                      *                 *
                      *                 *
                       *               *
                        ***************
                                |      \
                                |       A MESSAGE TR:R
                                |
              ---------------------------------------
              |BER %                                  |BER %
              |                                       |
              |        A MESSAGE TRAN:R               |        A MESSAGE TRAN:R
              |       /                               |       /
   ANDED.PRED v      /                     ANDED.PRED v      /
      ***************                         ***************
     *               *                       *               *
    *   A TRANSFER    *                     *   A TRANSFER    *
    *   GOOD DATA     * A SET L1GOOD        *   BAD DATA      * SEND BAD DATA FR
    *     L(1)        * SEND GOOD DATA FR   *     L(1)        * A SET MESSAGE TR
    *                 * A SET MESSAGE TR    *                 * A RESET L1GOOD
     *               *  A RESET L1BAD        *               *  A SET L1BAD
      ***************                         ***************
PROCESSOR|    |      \                PROCESSOR|    |      \
    1    |    |       A L1GOOD:S          1    |    |       BAD DATA
    |    |    |       GOOD DATA           |    |    |       A MESSAGE TR:S
    |____|    |       A MESSAGE TR:S      |____|    |       A L1GOOD:R
              |       etc.                          |       etc.
 CI:A LOCNTR:I|                                     |
 CI:A L1GOOD:I|                                     |
              |                                     |
   ANDED.PRED v                                     |
      ***************                               |
     *               *                              |
    *   A COUNTER     *                             |
    *   RESETTER 1    * A RESET LOCNTR              |
    *                 * A RESET L1GOOD              |
    *                 *                             |
     *               *                              |
      ***************                               |
PROCESSOR|           \                              |
    0    |            A LOCNTR:R                    |
    |    |            A L1GOOD:R                    v
    |____|                                   TO PROCESSOR 2
```

```
                                  |
                     CI:A L2GOOD:I|
                      CI:A L2BAD:I|        DATA*
                      CI:A L1BAD:I|        /
                                  |       /
                        ORED.PRED v      /
                          ***************
                         *               *
                        *   PROCESSOR 2   *
                        *                 * A RESET MESSAGE
                        *                 *
                        *                 *
                         *               *
                          ***************
                                  |      \
                                  |       A MESSAGE TR:R
                                  |
               ----------------------------------------
              |BER %                                   |BER %
              |                                        |
              |        A MESSAGE TRAN:R                |        A MESSAGE TRAN:R
              |       /                                |       /
   ANDED.PRED v      /                      ANDED.PRED v      /
      ***************                          ***************
     *               *                        *               *
    *    A TRANSFER   *                      *    A TRANSFER   *
    *    GOOD DATA    * SEND GOOD DATA FR    *    BAD DATA     * SEND BAD DATA FR
    *      L(2)       * A SET MESSAGE TR     *      L(2)       * A SET MESSAGE TR
    *                 * A SET L2GOOD         *                 * A RESET L2GOOD
     *               *  A RESET L2BAD         *               *  A SET L2BAD
      ***************                          ***************
PROCESSOR|    |      \                PROCESSOR|    |      \
    2    |    |       GOOD DATA           2    |    |       BAD DATA
    |    |    |       A MESSAGE TR:S      |    |    |       A MESSAGE TR:S
    |____|    |       A L2GOOD:S          |____|    |       A L2GOOD:R
              |       etc.                          |       etc.
  CI:A L1BAD:I|                                     |
 CI:A L2GOOD:I|                                     |
              |                                     |
   ANDED.PRED v                                     |
      ***************                               |
     *               *                              |
    *    A COUNTER    *                             |
    *    RESETTER 2   * A RESET L1BAD               |
    *                 * A RESET L2GOOD              |
    *                 *                             |
     *               *                              |
      ***************                               |
PROCESSOR|           \                              |
    1    |            A L1BAD:R                     |
    |    |            A L2GOOD:R                    v
    |____|                                 TO PROCESSOR 3
```

```
                                   |
                       CI:A L2BAD:I|       DATA*
                      CI:A L3GOOD:I|       /
                                   |      /
                        ORED.PRED v      /
                          ***************
                         *               *
                        *   PROCESSOR 3   *
                        *                 * A RESET MESSAGE
                        *                 *
                        *                 *
                         *               *
                          ***************
                                   |     \
                                   |      A MESSAGE TR:R
                                   |
               ----------------------------------------
              |BER %                                  |BER %
              |                                       |
              |        A MESSAGE TRAN:R               |        A MESSAGE TRAN:R
              |       /                               |       /
   ANDED.PRED v      /                     ANDED.PRED v      /
      ***************                         ***************
     *               *                       *               *
    *    A TRANSFER   *                     *    A TRANSFER   *
    *    GOOD DATA    * SEND GOOD DATA FR   *    BAD DATA     * SEND BAD DATA FR
    *      L(3)       * A SET MESSAGE TR    *      L(3)       * A SET MESSAGE TR
    *                 * A SET L3GOOD        *                 * A RESET L3GOOD
     *               *                       *               *
      ***************                         ***************
PROCESSOR|    |      \                  PROCESSOR|           \
    3    |    |       GOOD DATA             3    |            BAD DATA
    |    |    |       A MESSAGE TR:S        |    |            A MESSAGE TR:S
    |____|    |       A L3GOOD:S            |____|            A L3GOOD:R
              |       etc.                                    etc.
  CI:A L2BAD:I|
 CI:A L3GOOD:I|
              |
   ANDED.PRED v
      ***************
     *               *
    *    A COUNTER    *
    *    RESETTER 3   * A RESET L2BAD
    *                 * A RESET L3GOOD
    *                 *
     *               *
      ***************
PROCESSOR|           \
    2    |            A L2BAD:R
    |    |            A L3GOOD:R
    |____|
```

CACI NETIN   RELEASE 4.01

STATION B - TWO LEVEL ADAPTIVE ARQ STRATEGY

```
                         ***************
                        *               *  STATION B
                       *     B START     *
                       *                 * START DATA TRANS
                       *                 *
                       *                 *
                        *               *
                         ***************
                                |
                      CI:LOCNTR:I|
                                |         DATA*
                       ORED.PRED v       /
                         ***************
                        *               *
                       *   B PROCESSOR   *
                       *        0        * RESET MESSAGE TR
                       *                 *
                       *                 *
                        *               *
                         ***************   MESSAGE TRAN:R
                                |
                                |
             ---------------------------------------
             |BER %                                 |BER %
             |        MESSAGE TRANSF:R              |        MESSAGE TRANSF:R
             |       /                              |       /
   ANDED.PRED v     /                     ANDED.PRED v     /
     ***************                        ***************
    *               *                      *               *
   *    B TRANSFER   *                    *    B TRANSFER   *
   *    GOOD DATA    * SEND GOOD DATA FR  *    BAD DATA     * SEND BAD DATA FR
   *      L(0)       * SET MESSAGE TRAN   *      L(0)       * SET MESSAGE TRAN
   *                 * RESET LOCNTR       *                 * SET LOCNTR
    *               *                      *               *
     ***************                        ***************
   B     |          \                     B     |    |      \
PROCESSOR|           GOOD DATA         PROCESSOR|    |       BAD DATA
   0     |           MESSAGE TRAN:S       0     |    |       MESSAGE TRAN:S
   |     |           LOCNTR:R             |     |    |       LOCNTR:S
   |_____|           etc.                 |_____|    |       etc.
                                                     |
                                                     |
                                                     v
                                             TO B PROCESSOR 1
```

```
                              |
                  CI:LOCNTR:I|
                  CI:L1GOOD:I|
                              |         DATA*
                              |        /
                   ORED.PRED  v       /
                    ***************
                   *               *
                  *   B PROCESSOR   *
                  *        1        * RESET MESSAGE TR
                  *                 *
                  *                 *
                   *               *
                    ***************
                              |       \
                              |        MESSAGE TRAN:R
                              |
           --------------------------------------
          |BER %                                 |BER %
          |                                      |
          |          MESSAGE TRANSF:R            |          MESSAGE TRANSF:R
          |         /                            |         /
ANDED.PRED v        /              ANDED.PRED    v        /
     ***************                    ***************
    *               *                  *               *
   *   B TRANSFER    *                *   B TRANSFER    *
   *   GOOD DATA     * SEND GOOD DATA FR  *   BAD DATA   * SEND BAD DATA FR
   *     L(1)        * SET MESSAGE TRAN   *     L(1)     * SET MESSAGE TRAN
   *                 * SET L1GOOD         *              * RESET L1GOOD
    *               *                  *               *
     ***************                    ***************
   B     |    |       \               B     |            \
PROCESSOR|    |        GOOD DATA   PROCESSOR|             BAD DATA
   1     |    |        MESSAGE TRAN:S  1    |             MESSAGE TRAN:S
   |     |    |        L1GOOD:S        |    |             L1GOOD:R
   |_____|    |        etc.            |____|             etc.
              |
   CI:LOCNTR:I|
   CI:L1GOOD:I|
              |
    ORED.PRED v
     ***************
    *               *
   *   B COUNTER     *
   *   RESETTER      * RESET LOCNTR
   *                 * RESET L1GOOD
   *                 *
    *               *
     ***************
   B     |            \
PROCESSOR|             LOCNTR:R
   0     |             L1GOOD:R
   |     |
   |_____|
```

LIST OF REFERENCES

1. A.S. Tanenbaum, *Computer Networks*, Prentice Hall, New Jersey, 1981

2. D.P. Bertsekas and R.G. Gallager, *Data Networks*, Prentice Hall, New Jersey, 1987

3. R.L. Borchardt and T.T. Ha, "Packet communications in a multipath fading mobile radio network," *Phoenix Conference on Computers and Communications*, pp.160-164, IEEE Computer Society Press, Washington, D.C., March 1989

4. D.E. Carlson, "Bit oriented data link control procedures," *IEEE Trans. on Communications*, Vol. COM-28, No. 4, pp. 455-467, April 1980

5. E.R. Berlekamp, R.E Peile, and S.P. Pope, "The application of error control to communications," *IEEE Communications Magazine*, Vol. 25, No. 4, pp. 44-57, April 1987

6. Y. Ishibashi and A. Iwabuchi, "Throughput analysis of adaptive ARQ schemes," *Institute of Electronics and Communications Engineers of Japan*, Vol. 71B, No. 6, pp. 698-707, June 1988

7. NETWORK II.5 Version 4.0, CACI Products Company, La Jolla, CA, October 1988

BIBLIOGRAPHY

Bux, W., Kümmerle, K., and Troung, H.L., "Balanced HDLC Procedures: A performance Analysis," *IEEE Trans. on Communications,* Vol. COM-28, No. 11, November 1980.

Kulkarni, U.M., *Performance Analysis of HDLC Protocol Operating in Asynchronous Balanced Mode,* M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, May 1988.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Professor Tri T. Ha, Code 62Ha Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	5
5.	Professor Glen A. Myers, Code 62Mv Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Commandant (G-PTE-1) U.S. Coast Guard 2100 2nd St. SW. Washington, D.C. 20593-0001	2
7.	LT. Kurtis J. Guth Coordinator of Chain Operations US Coast Guard LORMONSTA Kodiak Kodiak, AK 99619	2