

NAVAL POSTGRADUATE SCHOOL
Monterey, California

AD-A236 940



THESIS

DTIC
ELECTE
JUN 12 1991
S B D

EFFECTS OF NON-UNIFORM WINDOWING
IN A RICIAN-FADING CHANNEL AND
SIMULATION OF ADAPTIVE
AUTOMATIC REPEAT REQUEST PROTOCOLS

by

Chris G. Kmiecik

June 1990

Thesis Advisor:

Tri T. Ha

Approved for public release; distribution is unlimited

91-01885



91 6 11 163

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
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			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School		
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School		6b OFFICE SYMBOL (If applicable) EC	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-4000		
6c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-4000		9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION		8b OFFICE SYMBOL (If applicable)	10 SOURCE OF FUNDING NUMBERS		
8c. ADDRESS (City, State, and ZIP Code)		PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
11 TITLE (Include Security Classification) EFFECTS OF NON-UNIFORM WINDOWING IN A RICIAN-FADING CHANNEL AND SIMULATION OF ADAPTIVE AUTOMATIC REPEAT REQUEST PROTOCOLS					
12 PERSONAL AUTHOR(S) KMIECIK, Chris G.					
13a TYPE OF REPORT Master's Thesis		13b TIME COVERED FROM _____ TO _____		14 DATE OF REPORT (Year, Month, Day) 1990 June	
15 PAGE COUNT 67					
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 US Government.					
17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	FFT windowing; ARQ, fading, frequency shifts		
19 ABSTRACT (Continue on reverse if necessary and identify by block number)					
Two aspects of digital communication were investigated. In the first part, a FFT-based, M-ary FSK receiver in a Rician-fading channel was analyzed to determine the benefits of non-uniform windowing of sampled received data. When a frequency offset occurs, non-uniform windowing provided better FFT magnitude separation. The improved dynamic range was balanced against a loss in detectability due to signal attenuation. With large frequency offset, the improved magnitude separation outweighed the loss in detectability. An analysis was carried out to determine what frequency deviation is necessary for non-uniform windowing to out-perform uniform windowing in a slow Rician-fading channel. Having established typical values of probability of bit errors, the					
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS			21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a NAME OF RESPONSIBLE INDIVIDUAL HA, Tri T.			22b TELEPHONE (Include Area Code) 408-646-2788		22c OFFICE SYMBOL EC/HA

DD Form 1473, JUN 86

Previous editions are obsolete

S/N 0102-LF-014-6603

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

19. cont.

second part of this thesis looked at improving throughput in a digital communications network by applying adaptive automatic repeat request (ARQ) protocols. The results of simulations of adaptive ARQ protocols with variable frame lengths is presented. By varying the frame length, improved throughput performance through all bit error rates was achieved.

Approved for public release; distribution is unlimited

Effects of Non-Uniform Windowing
in a Rician Fading Channel
and
Simulation of Adaptive
Automatic Repeat Request Protocols

by

Chris G. Kmiecik
Lieutenant, USCG.
B.S, USCGA, 1984

Submitted in partial fulfillment of the
requirements for the degree of

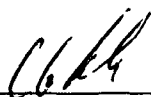
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

June, 1990

Author:

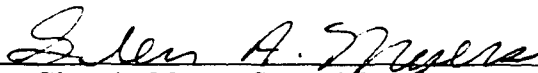


Chris G. Kmiecik

Approved by:



Tri T. Ha, Thesis Advisor



Glen A. Myers, Second Reader



John P. Powers, Chairman
Department of Electrical and Computer Engineering

iii

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

ABSTRACT

Two aspects of digital communication were investigated. In the first part, a FFT-based, M-ary FSK receiver in a Rician-fading channel was analyzed to determine the benefits of non-uniform windowing of sampled received data. When a frequency offset occurs, non-uniform windowing provided better FFT magnitude separation. The improved dynamic range was balanced against a loss in detectability due to signal attenuation. With large frequency offset, the improved magnitude separation outweighed the loss in detectability. An analysis was carried out to determine what frequency deviation is necessary for non-uniform windowing to out-perform uniform windowing in a slow Rician-fading channel. Having established typical values of probability of bit errors, the second part of this thesis looked at improving throughput in a digital communications network by applying adaptive automatic repeat request (ARQ) protocols. The results of simulations of adaptive ARQ protocols with variable frame lengths is presented. By varying the frame length, improved throughput performance through all bit error rates was achieved.

TABLE OF CONTENTS

I.	INTRODUCTION	1
	A. EFFECTS OF NON-UNIFORM WINDOWING	1
	B. ADAPTIVE ARQ	3
II.	RECEIVER ANALYSIS	7
III.	ERROR ANALYSIS	13
IV.	WINDOWING RESULTS	16
V.	ARQ PROTOCOLS AND THEIR PERFORMANCE	21
	A. STOP-AND-WAIT ARQ	21
	1. Performance Analysis Background	22
	a. Assumptions	23
	b. Average Transmission Attempts	23
	c. Throughput Definition	24
	2. Stop-and-Wait Performance	24
	B. GO-BACK- <i>N</i>	25
	C. SELECTIVE-REPEAT ARQ	28
VI.	ADAPTIVE ARQ PROTOCOLS	32
	A. ADAPTIVE STOP-AND-WAIT USING A SEMAPHORE	32
	B. ADAPTIVE SELECTIVE-REPEAT USING THRESHOLDS	33
	C. GO-BACK- <i>N</i> ARQ	35
VII.	CONCLUSIONS	37
	APPENDIX A: ERROR COMPUTATION	38
	1. Bit Error Program Code	39
	APPENDIX B: ARQ SIMULATIONS	47

1. Selective-Repeat	47
2. Go-Back- N	48
3. Stop-and-Wait	48
4. Selective-Repeat Program Code	48
REFERENCES	58
INITIAL DISTRIBUTION LIST	59

I. INTRODUCTION

Two aspects of digital communication were investigated. Chapters II, III and IV present an analysis determining the advantages of non-uniform windowing in reducing the effects of frequency offsets caused by doppler shifts and other sources in a fading channel. Chapters V, VI and VII develop a simulation of adaptive automatic repeat request (ARQ) protocols to improve throughput performance through various bit error rates.

A. EFFECTS OF NON-UNIFORM WINDOWING

Digital communication involves the transmission of information in bit format. M-ary frequency-shift-keying (MFSK) uses $M = 2^k$ different frequencies to represent M different symbols (each containing k bits of information). A proper determination of the frequency sent enables the receiver to determine which symbol was transmitted.

With the advent of real-time Fast Fourier Transform (FFT) processors, an MFSK receiver can be easily implemented. Selected output bins of the FFT will correspond to the M possible frequencies provided the sampling rate and the signal frequencies are closely related. In the case developed here, a bin separation of two will be used (proper determination of sampling rate and corresponding bin locations will be discussed in Chapter II). By comparing the magnitudes of the FFT output bins and choosing the bin with the largest magnitude, the transmitted symbol can be identified.

Noise in the channel causes a variance in the magnitudes of the FFT output and an error can occur when the magnitude of one output bin is greater than the bin corresponding to the frequency of the signal sent. In the presence of frequency offsets

(caused by doppler shifts or receiver oscillator drift, for example), the probability of error is further increased by a shifting of signal power into other frequency bins. Frequency offsets also contribute indirectly to erroneous frequency components by causing leakage. Leakage is encountered in FFT analysis and is a result of a periodic signal not being sampled over an integer number of cycles. A discontinuity will occur at the endpoints of the signal and will contribute erroneous frequency components to the FFT. These contributions can be seen in Figure 1.1(a) where the magnitudes of a FFT are given for a noiseless signal experiencing a frequency offset.

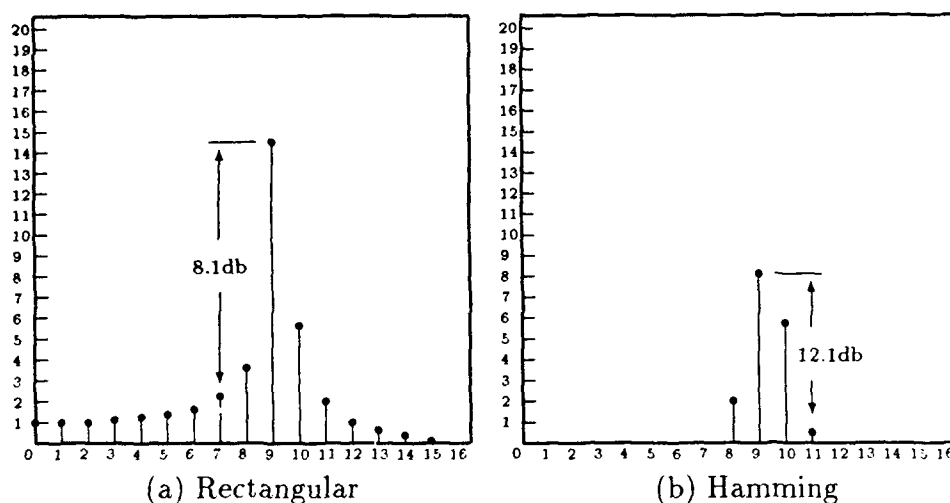


Figure 1.1: Effects of frequency offset with (a) rectangular, and (b) Hamming windowing. Dynamic range is improved using non-uniform windowing with a loss of signal detectability.

Non-uniform windowing can be used on the sampled signal in order to smooth out discontinuities at the endpoints of the sampling interval and help reduce the effects of leakage. The reduced effects of leakage are balanced against signal attenuation caused by non-uniform windowing. With less energy, signal detectability deteriorates and the noise variance has a greater effect on proper signal decoding. This is seen in

Figure 1.1(b) where a Hamming window is applied to the sampled signal. Although the dynamic range between the two components of interest has increased, signal strength has decreased. As the frequency offset becomes more dramatic, the benefits of reducing the amount of leakage by non-uniform windowing outweighs the loss in signal detectability.

This thesis investigates the amount of frequency offset necessary to overcome the performance loss due to signal attenuation in a Rician-fading channel. The results of [1] are applied to a channel experiencing slow Rician fading to determine the effects of fading on the performance trade-off between non-uniform and uniform windowing. A statistical analysis is carried out to determine the probability of bit error as a function of signal-to-noise ratios as well as direct-to-fading ratios associated with Rician fading. From these results, the frequency offset at which non-uniform windowing out-performs uniform windowing is determined. It is shown that as the channel approaches Rayleigh fading (the direct-to-fading ratio decreases), the amount of frequency offset necessary to justify the use of non-uniform windowing decreases despite its signal attenuation characteristic.

B. ADAPTIVE ARQ

Having an understanding of the expected bit error rate experienced by the channel, Chapters V through VII investigate the possibility of improving throughput performance by implementing an adaptive automatic repeat request (ARQ) scheme in computer communication.

Communication between computers can be described using the seven layer open systems interconnection (OSI) model given in Figure 1.2. Information from each node is first processed down to the network layer and into the data link control (DLC). The DLC then prepares the packet of information for transmission. Overhead bits

are appended to the packet which include coding information for error detection and synchronization. In addition, a header and a tail is attached to the packet so that it may be identified by the proper receiving node. The processed packet, now a frame, is released from the DLC to the physical interface for transmission over the physical link. The transmitted frame is then received at the receiving node's physical interface and a reverse process occurs.

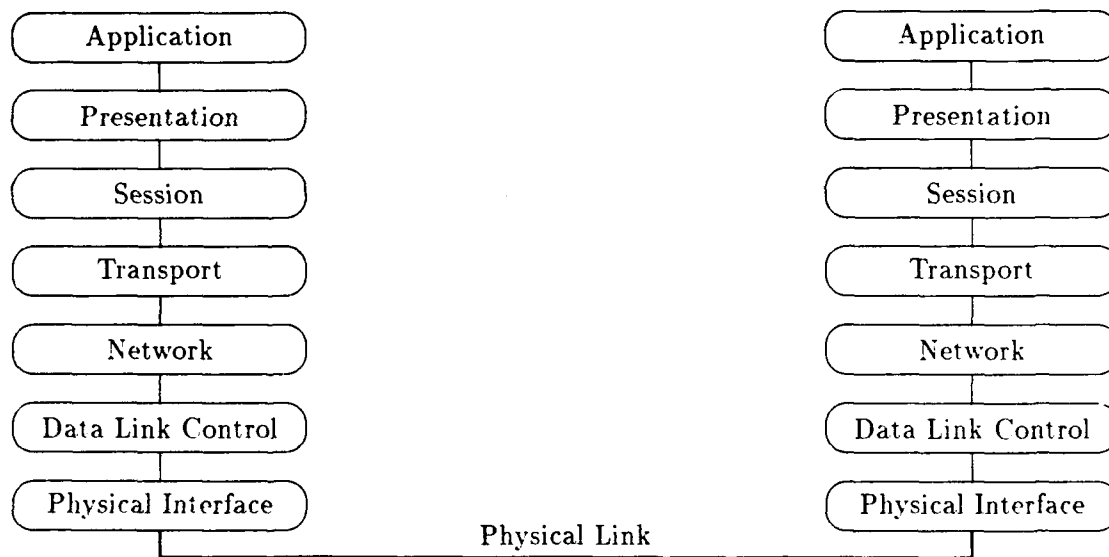


Figure 1.2: In an OSI network communication representation, data packets are passed from the network layer to the data link control (DLC) layer where they are formatted and processed for transmission over the physical link. ARQ protocols are carried out in the DLC layer.

As noted in the previous section, noise on the physical link sometimes corrupts the transmitted frame and an error-detection scheme is implemented by the DLC to recognize transmission errors. When an error occurs, the receiving DLC must issue a command to the sender of the frame to request retransmission of the frame.

Several different ARQ protocols currently exist and three will be analyzed here—stop-and-wait, go-back- N and selective-repeat. An analysis of the throughput of each will be developed to compare the performance of each protocol. From this analysis, it is found that different fixed frame lengths provide better performance for various bit error rates. Although one frame length may be efficient at a given bit error rate, that same length may not be as efficient if the bit error rate changes. This is reflected in Figure 1.3 where, although a frame length of 1024 bits is most efficient at low bit error rates, at high bit error rates a substantial improvement in throughput is achieved using a frame length of 128 bits.

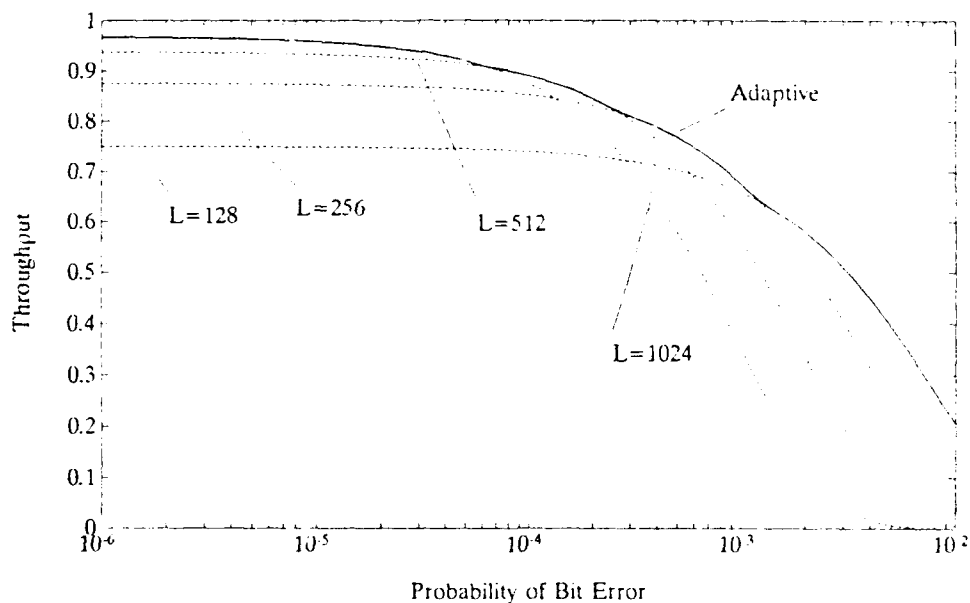


Figure 1.3: Dotted curves represent throughput of a fixed frame length ARQ protocol for various frame lengths L . At low bit error rates, increased percentage overhead in small frames decreases throughput. At higher bit error rates, longer frames have a higher probability of error. The solid curve represents the desired performance from an adaptive ARQ scheme.

To help improve performance through the entire range of bit error rates, an adaptive ARQ protocol can be implemented. With adaptive frame lengths, long

frames are used at lower bit error rates *and* shorter frames at higher bit error rates. By adapting to the appropriate frame length, the throughput achieved can approach the maximum at any BER of the channel (the solid curve in Figure 1.3). The concept of an adaptive ARQ strategy is proposed in [2]. This thesis presents the results from simulations where the length of a frame can adapt to the experienced bit error rate of the channel.

II. RECEIVER ANALYSIS

An analysis similar to that presented in [1] is carried out where, in an M-ary FSK communications scheme, one of M signals is sent during the time interval $0 \leq t < T$. Customarily, $M = 2^k$ so that k -bits of digital information are represented by each signal. The possible signals to be transmitted are of the form

$$s_m(t) = \sqrt{2E_s} \cos 2\pi f_m t \quad (2.1)$$

where E_s is the signal energy ($E_s = \int_0^T s_m^2(t) dt$) and f_m is one of the M frequencies used to distinguish among signals where

$$f_m = \frac{1}{T} + \frac{\Delta_f(m-1)}{T}. \quad (2.2)$$

Here, Δ_f is an integer representing the spacing (relative to $1/T$) between each of the M signals. This signal is modulated with an appropriate carrier frequency f_c which is removed at the receiver.

The receiver used in this analysis is depicted in Figure 2.1. After removing the carrier frequency and low-pass filtering (with cutoff frequency $W > f_{max}$), the signal is then sampled at a rate N/T (N is the size of the FFT) so that the outputs of the FFT correspond to a frequency spacing of $1/T$. With this sampling rate, only the bins with a spacing Δ_f from Equation 2.2 will be used in the decision stage.

The input to the receiver is the transmitted signal corrupted by zero-mean, Gaussian-distributed white noise $\eta(t)$ with power spectral density (p.s.d.) of $N_o/2$. The signal may also experience a frequency offset of f' caused by doppler effects, receiver oscillator drift or other sources. In addition, the signal experiences slow Rician fading causing the amplitude a to be a random variable following a Rician

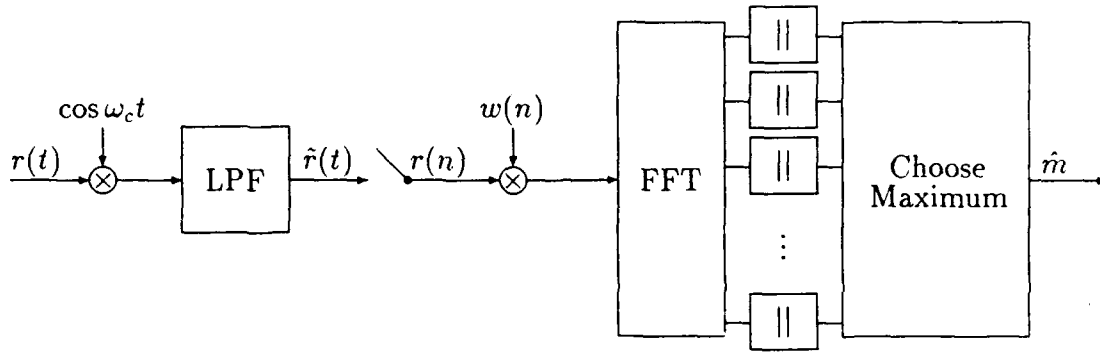


Figure 2.1: With a FFT receiver, the incoming signal is sampled, a window is applied, and the magnitude of the FFT output is used to determine which signal is received.

distribution. Rician-distributed random variables in signal analysis are characterized by the ratio of the power of the direct signal component A^2 to the power of the diffused (or faded) component $2\sigma_f^2$

$$f_a(a) = \frac{a}{\sigma_f^2} \exp \left[-\frac{a^2 + A^2}{2\sigma_f^2} \right] I_0 \left(\frac{aA}{\sigma_f^2} \right) \quad (2.3)$$

where $I_0(x)$ is the modified Bessel function

$$I_0(x) \triangleq \sum_{n=0}^{\infty} \frac{x^{2n}}{2^{2n}(n!)^2}. \quad (2.4)$$

At the receiver's input,

$$r(t) = \sqrt{2E_s} a \cos [2\pi(f_c + f_m + f')t] + \eta(t) \quad (2.5)$$

and after removing the carrier

$$\tilde{r}(t) = \sqrt{2E_s} a \cos [2\pi(f_m + f')t] + \eta(t). \quad (2.6)$$

To simplify the signal representation, the following substitutions are made:

$$q \triangleq 1 + \Delta_f(m-1) \quad (2.7)$$

$$\epsilon \triangleq f'T \quad (2.8)$$

so that Equation 2.6 becomes

$$\tilde{r}(t) = \sqrt{2E_s}a \cos \left[\frac{2\pi(q + \epsilon)t}{T} \right] + \eta(t) \quad (2.9)$$

where ϵ now denotes the fractional (of $1/T$) frequency offset and q represents the multiple of $1/T$ of the sent signal. With a sampling rate of N/T , the discrete form of Equation 2.9 is represented by

$$r(n) = \sqrt{2E_s}a \cos \frac{2\pi(q + \epsilon)n}{N} + \eta(n). \quad (2.10)$$

As seen in the receiver of Figure 2.1, the discrete signal is windowed using a discrete windowing function $w(n)$ and the FFT is taken. The outputs of the FFT can be expressed in real (denoted $X_R(k)$) and imaginary $X_I(k)$ parts

$$\begin{aligned} X_R(k) &\triangleq \sum_{n=0}^{N-1} r(n)w(n) \cos \frac{2\pi kn}{N} \\ &= \sum_{n=0}^{N-1} \left\{ w(n)\sqrt{2E_s}a \cos \frac{2\pi(q + \epsilon)n}{N} \cos \frac{2\pi kn}{N} + \eta(n)w(n) \cos \frac{2\pi kn}{N} \right\} \end{aligned} \quad (2.11)$$

$$\begin{aligned} X_I(k) &\triangleq \sum_{n=0}^{N-1} r(n)w(n) \sin \frac{2\pi kn}{N} \\ &= \sum_{n=0}^{N-1} \left\{ w(n)\sqrt{2E_s}a \cos \frac{2\pi(q + \epsilon)n}{N} \sin \frac{2\pi kn}{N} + \eta(n)w(n) \sin \frac{2\pi kn}{N} \right\}. \end{aligned} \quad (2.12)$$

Keeping in mind that the channel experiences slow Rician fading so that a is assumed to be a constant over the period of interest $0 \leq t < T$, and, because the received signal is being corrupted by random noise following a Gaussian distribution, $X_R(k)$ and $X_I(k)$ are Gaussian distributed as well, since they are derived from linear combinations of Gaussian processes. As a result, the output of the FFT generator is a complex Gaussian process of the form

$$X(k) = X_R(k) + jX_I(k). \quad (2.13)$$

The necessary variable for the decision criteria will be the magnitude of the outputs of the FFT

$$d_k \triangleq |X(k)| = \sqrt{X_R^2(k) + X_I^2(k)}. \quad (2.14)$$

This is the envelope of the complex signal $X(k)$. As stated in [3], for a random complex signal $X(k)$, the distribution of the envelope is

$$f(d_k) = \frac{d_k}{\sigma_X^2} \exp \left[-\frac{1}{2\sigma_X^2} (d_k^2 + \mu^2) \right] I_0 \left(\frac{d_k \mu}{\sigma_X^2} \right) \quad (2.15)$$

where (overbar notation denotes expected value),

$$\mu = |\overline{X(k)}| = \sqrt{\overline{X_R(k)}^2 + \overline{X_I(k)}^2} \quad (2.16)$$

and

$$\sigma_X^2 = \frac{1}{2} E \left[|X(k) - \overline{X(k)}|^2 \right]. \quad (2.17)$$

The mean of $X(k)$ (from Equations 2.11 and 2.12) for a fixed amplitude a are easily found by observing that $\eta(n)$ is the only random quantity and is zero-mean, causing the second term of the sum to go to zero leaving the first term of the sum,

$$\begin{aligned} \overline{X_R} &= a\sqrt{2E_s} \sum_{n=0}^{N-1} w(n) \cos \frac{2\pi n(q+\epsilon)}{N} \cos \frac{2\pi nk}{N} \\ &\triangleq m_R \end{aligned} \quad (2.18)$$

$$\begin{aligned} \overline{X_I} &= a\sqrt{2E_s} \sum_{n=0}^{N-1} w(n) \sin \frac{2\pi n(q+\epsilon)}{N} \sin \frac{2\pi nk}{N} \\ &\triangleq m_I. \end{aligned} \quad (2.19)$$

The variance in Equation 2.17 then becomes

$$\sigma_X^2 = \frac{1}{2} E \left[\left| \sum_{n=0}^{N-1} w(n) \eta(n) \cos \frac{2\pi nk}{N} + \sum_{n=0}^{N-1} w(n) \eta(n) \sin \frac{2\pi nk}{N} \right|^2 \right] \quad (2.20)$$

which, because of white noise, the autocorrelation function of $\eta(n)$ is given by

$$R_\eta(n, p) = 2W \frac{N_o}{2} \delta(n - p) \quad (2.21)$$

where, again, W is the cut-off frequency of the low-pass filter at the receiver and $N_o/2$ is the p.s.d. of the additive white Gaussian noise. Equation 2.20 is reduced to

$$\begin{aligned}\sigma_X^2 &= \frac{WN_o}{2} \left(\sum_{n=0}^{N-1} w^2(n) \cos^2 \frac{2\pi nk}{N} + \sum_{n=0}^{N-1} w^2(n) \sin^2 \frac{2\pi nk}{N} \right) \\ &= \frac{WN_o}{2} \sum_{n=0}^{N-1} w^2(n).\end{aligned}\quad (2.22)$$

Keep in mind that Equations 2.18 and 2.19 are still functions for a given k , m and ϵ . By factoring out a from m_R and m_I given in Equations 2.18 and 2.19, and substituting into Equation 2.16,

$$\begin{aligned}\mu^2 &= m_R^2 + m_I^2 \\ &= a^2(m_R'^2 + m_I'^2) \\ &\triangleq a^2\beta_k.\end{aligned}\quad (2.23)$$

With slow Rician fading, Equation 2.15 is actually a conditional density on a and becomes

$$f_{d_k|a}(d_k|a) = \frac{d_k}{\sigma_X^2} \exp \left[-\frac{1}{2\sigma_X^2} (d_k^2 + a^2\beta_k) \right] I_0 \left(\frac{ad_k\sqrt{\beta_k}}{\sigma_X^2} \right). \quad (2.24)$$

The density function of d_k can then be found by integrating over the Rician-distributed random variable a

$$\begin{aligned}f_{d_k}(d_k) &= \int_0^\infty f_{d_k}(d_k|a) f_a(a) da \\ &= \int_0^\infty \frac{d_k}{\sigma_X^2} e^{-\frac{1}{2\sigma_X^2} (d_k^2 + a^2\beta_k)} I_0 \left(\frac{ad_k\sqrt{\beta_k}}{\sigma_X^2} \right) \frac{a}{\sigma_f^2} e^{-\frac{1}{2\sigma_f^2} (a^2 + A^2)} I_0 \left(\frac{aA}{\sigma_f^2} \right) da \\ &= \frac{d_k}{\sigma_X^2 \sigma_f^2} \exp \left[-\frac{1}{2} \left(\frac{d_k^2}{\sigma_X^2} + \frac{A^2}{\sigma_f^2} \right) \right] \\ &\times \int_0^\infty a e^{-a^2 \left(\frac{\beta_k}{2\sigma_X^2} + \frac{1}{2\sigma_f^2} \right)} I_0 \left(\frac{ad_k\sqrt{\beta_k}}{\sigma_X^2} \right) I_0 \left(\frac{aA}{\sigma_f^2} \right) da.\end{aligned}\quad (2.25)$$

By making the substitutions $I_0(x) = J_0(jx)$,

$$Q^2 \triangleq \frac{\beta_k}{2\sigma_X^2} + \frac{1}{2\sigma_f^2} = \frac{\beta_k\sigma_f^2 + \sigma_X^2}{2\sigma_X^2\sigma_f^2}, \quad (2.26)$$

and using the integral (from [4])

$$\int_0^\infty x e^{-Q^2 x^2} J_\nu(ax) J_\nu(bx) dx = \frac{1}{2Q^2} \exp\left[-\frac{a^2 + b^2}{4Q^2}\right] I_\nu\left(\frac{ab}{2Q^2}\right), \quad (2.27)$$

gives

$$\begin{aligned} f_{d_k}(d_k) &= \left(\frac{d_k}{\sigma_X^2 \sigma_f^2}\right) \left(\frac{1}{2Q^2}\right) \exp\left[-\frac{d_k^2 \sigma_f^2 + A^2 \sigma_X^2}{2\sigma_X^2 \sigma_f^2}\right] \\ &\times \exp\left[\left(\frac{d_k^2 \beta_k}{\sigma_X^4} + \frac{A^2}{\sigma_f^4}\right) \frac{1}{4Q^2}\right] I_0\left(\frac{-d_k A \sqrt{\beta_k}}{2\sigma_X^2 \sigma_f^2 Q^2}\right). \end{aligned} \quad (2.28)$$

From Equation 2.4, it is seen that $I_0(x)$ is an even function. Replacing the Q^2 as defined in Equation 2.26 into Equation 2.28 results in

$$\begin{aligned} f_{d_k}(d_k) &= \left(\frac{d_k}{\beta_k \sigma_f^2 + \sigma_X^2}\right) \exp\left[\frac{-d_k^2 \sigma_f^2 - A^2 \sigma_X^2}{2\sigma_X^2 \sigma_f^2} + \frac{d_k^2 \beta_k \sigma_f^4 + A^2 \sigma_X^4}{2\sigma_X^2 \sigma_f^2 (\beta_k \sigma_f^2 + \sigma_X^2)}\right] \\ &\times I_0\left(\frac{d_k A \sqrt{\beta_k}}{\beta_k \sigma_f^2 + \sigma_X^2}\right) \end{aligned} \quad (2.29)$$

which simplifies to

$$f_{d_k}(d_k) = \left(\frac{d_k}{\beta_k \sigma_f^2 + \sigma_X^2}\right) \exp\left[-\frac{1}{2} \left(\frac{d_k^2 + \beta_k A^2}{\beta_k \sigma_f^2 + \sigma_X^2}\right)\right] I_0\left(\frac{d_k A \sqrt{\beta_k}}{\beta_k \sigma_f^2 + \sigma_X^2}\right). \quad (2.30)$$

Note that by making the substitutions

$$\begin{aligned} \sigma_k^2 &\triangleq \beta_k \sigma_f^2 + \sigma_X^2 \\ \alpha_k &\triangleq A \sqrt{\beta_k}, \end{aligned} \quad (2.31)$$

Equation 2.30 is seen to be Rician-distributed since

$$f_{d_k}(d_k) = \frac{d_k}{\sigma_k^2} \exp\left(-\frac{1}{2} \frac{d_k^2 + \alpha_k^2}{\sigma_k^2}\right) I_0\left(\frac{d_k \alpha_k}{\sigma_k^2}\right). \quad (2.32)$$

A similar result was found in [1] where the parameters σ_k^2 and α_k are now altered as a result of Rician fading. Note that if no fading is present ($2\sigma_f^2 = 0$ and $A = 1$), then Equation 2.32 is the solution found in [1].

III. ERROR ANALYSIS

Now having an expression for the distribution of the magnitudes of the FFT output, an error occurs when the desired frequency component (located at bin q of the FFT) is less than any of the other FFT bins of interest. With a sampling rate of N/T , the bins of interest are those corresponding to a frequency integer multiple of Δ_f . In addition, the FFT displays symmetry about the $N/2$ frequency bin so only the first $N/2$ outputs are of consequence. Finally, the first bin corresponding to a d.c. component is disregarded, as well as the bin located at $N/2$, since a frequency component in this bin will not satisfy the Nyquist criteria.

As further discussed in [1] (referencing [5]), the union bound solution is carried out for the probability P_s of symbol error. From Equations 2.18 and 2.19, the distribution of d_k will differ with a positive or negative frequency offset ϵ . It is assumed that a positive or negative frequency offset is equally likely. In addition, each of the M signals are assumed equally likely to occur with probability $1/M$. An error occurs when the magnitude d_k of a FFT bin is greater than the magnitude d_q (referring to Equation 2.7) of the bin of interest. The probability of symbol error is then given by

$$P_s \leq \frac{1}{2M} \sum_{\left\{ \begin{array}{l} k, q \in 1 + \Delta_f(m-1) \\ m=1, 2, \dots, M \\ k \neq q \end{array} \right\}} \{Pr[d(q) < d(k) + \epsilon, q] + Pr[d(q) < d(k) - \epsilon, q]\} \quad (3.1)$$

where, from [5], for Rician-distributed random variables

$$Pr[d(q) < d(k)] = Q(\sqrt{a}, \sqrt{b}) - \frac{c^2}{c^2 + 1} \exp\left(-\frac{a+b}{2}\right) I_0(\sqrt{ab}) \quad (3.2)$$

with the following definitions made

$$\begin{aligned} a &\triangleq \frac{\alpha_k^2}{\sigma_q^2 + \sigma_k^2} \\ b &\triangleq \frac{\alpha_q^2}{\sigma_q^2 + \sigma_k^2} \\ c &\triangleq \frac{\sigma_q^2}{\sigma_k^2} \end{aligned}$$

Evaluation of the Marcum-Q function is described in Appendix A. Also, the dependency of ϵ , q and k is found in the β_k term in the definitions made in Equation 2.23 (with reference to Equations 2.18 and 2.19).

Parameters for analysis include the direct-to-fading ratio

$$DTF \triangleq \frac{A^2}{2\sigma^2} \quad (3.3)$$

and the signal-to-noise ratio at the receiver

$$SNR \triangleq \frac{E_s}{N_o} = \frac{A^2 + 2\sigma^2}{N_o} \quad (3.4)$$

By solving each of the above equations for A^2 , it is found that

$$\frac{\sigma^2}{N_o} = \frac{SNR}{2(DTF + 1)} \triangleq \gamma \quad (3.5)$$

which is the diffused signal-to-noise density ratio. Using this, making the substitutions from Equation 2.31, and normalizing the transmitted energy ($E_s = 1$ in Equation 2.18 and 2.19), then a , b and c can be expressed in terms of the given parameters

$$a = \frac{2DTF\beta_k}{\beta_k + \beta_q + \gamma^{-1}W \sum_{n=0}^{N-1} w^2(n)} \quad (3.6)$$

$$b = \frac{2DTF\beta_q}{\beta_k + \beta_q + \gamma^{-1}W \sum_{n=0}^{N-1} w^2(n)} \quad (3.7)$$

$$c = \frac{2\beta_q\gamma + W \sum_{n=0}^{N-1} w^2(n)}{2\beta_k\gamma + W \sum_{n=0}^{N-1} w^2(n)} \quad (3.8)$$

A bound on bit error is then found by

$$P_b \leq \frac{M/2}{M-1} P_s. \quad (3.9)$$

IV. WINDOWING RESULTS

Graphs of the probability of bit error as a function of the received signal-to-noise ratio (per bit) were constructed for various direct-to-fading ratios $A/2\sigma_f^2$ for the rectangular and Hamming windowing cases. These results were generated from Equations 3.1 and 3.2 with evaluation of the Marcum-Q function and the modified Bessel function described in Appendix A. A plot with low fading channel ($A^2/2\sigma^2 = 100$) is given in Figure 4.1 below. The case studied in [1] ($\Delta_f = 2$, $M = 8$) is used so that a direct comparison of results can be made. In the case of low fading, the results are comparable.

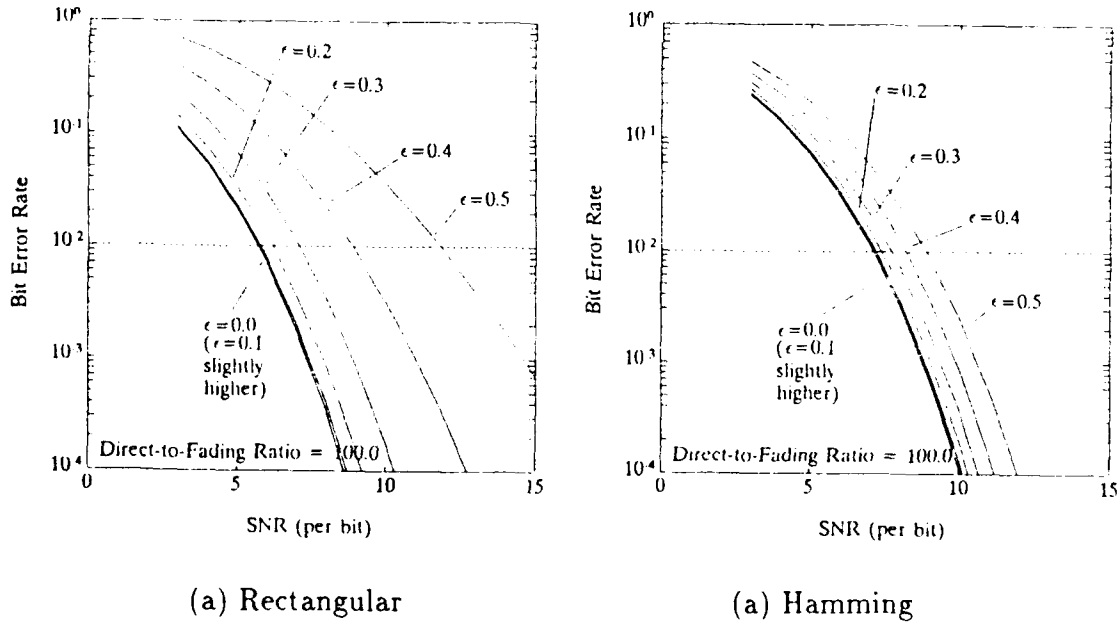


Figure 4.1: Probability of bit error for low fading channel (a) rectangular, and (b) Hamming windowing. 32-point FFT, $M = 8$, $\Delta_f = 2$.

From Figure 4.1 the advantages of the non-uniform windowing can be seen as the frequency offset becomes larger for a fixed f_m and T . In both cases, as ϵ grows linearly, the energy necessary to maintain a given bit error rate grows exponentially. However, the advantage of the Hamming window is reflected in a smaller exponential increase than the rectangular window. With lower values of frequency offset the signal energy necessary is larger than with uniform windowing, but, as ϵ increases, the advantages of non-uniform windowing become apparent.

A visual representation of this was developed in [1] by graphing the amount of SNR degradation necessary to maintain a fixed probability of bit error as a function of increasing ϵ and is repeated here. Figure 4.2 gives the result for the low fading case with a fixed BER of 10^{-2} (from Figure 4.1).

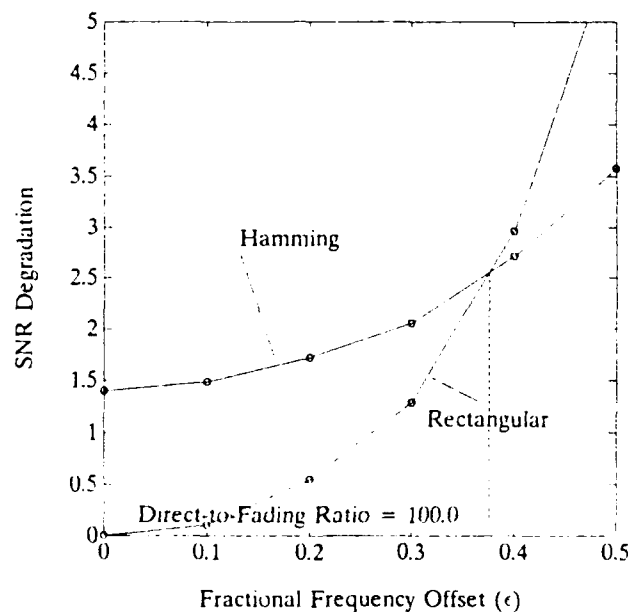
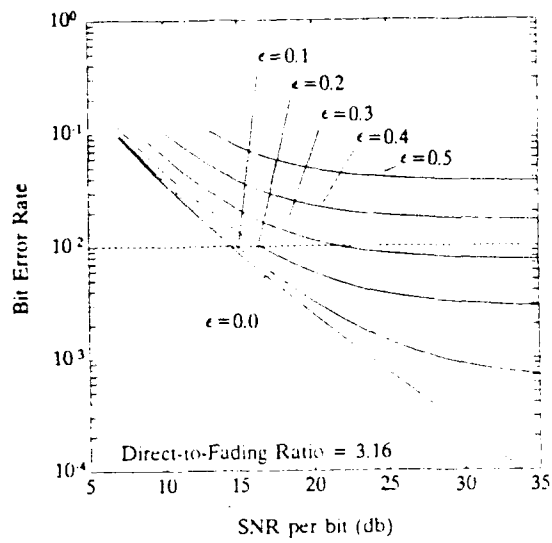
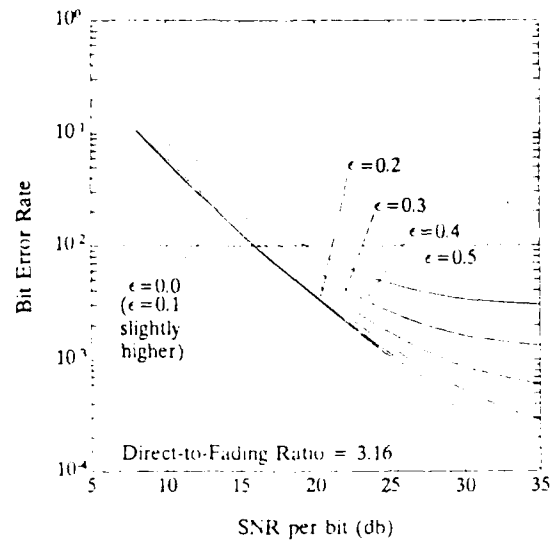


Figure 4.2: Trade-off of uniform and non-uniform windowing for bit error rate of 10^{-2} . Non-uniform windowing becomes more advantageous as ϵ increases.

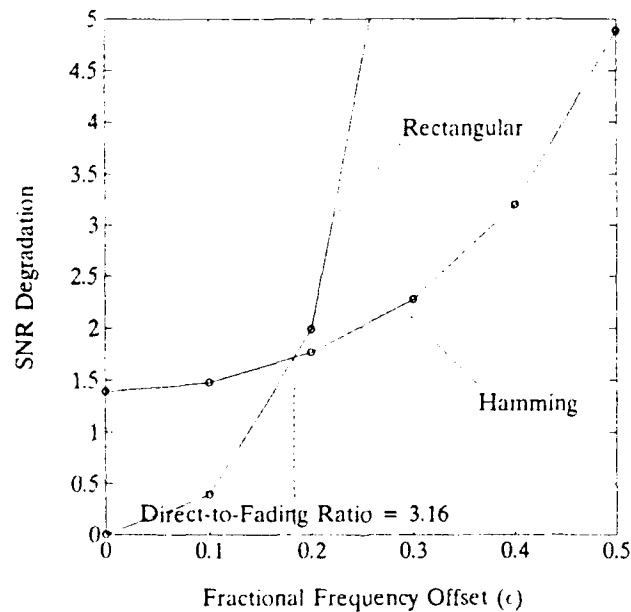
Similar graphs of bit error rate as a function of SNR are given for the case of a slightly diffused channel ($A^2/2\sigma^2 = 3.16$) in Figure 4.3. The results displaying SNR degradation are given as well. From Figure 4.3 we see that with a more diffused signal, the cross-over fractional frequency offset decreases. This phenomena was studied closer by generating additional results in the same manner for various direct-to-diffused ratios and plotting the frequency offset cross-over point of each in Figure 4.4. From this graph we see that the justification to use non-uniform windowing is very sensitive in the area of equal direct and diffused components. However, as the direct component approaches zero (Rayleigh fading), non-uniform windowing becomes more advantageous when minimal frequency offset is expected.



(a)



(b)



(c)

Figure 4.3: Probability of bit error for fading channel. $A/2\sigma_f^2 = 3.16$ with (a) rectangular and (b) Hamming windowing. SNR degradation for a fixed BER of 10^{-2} given in (c).

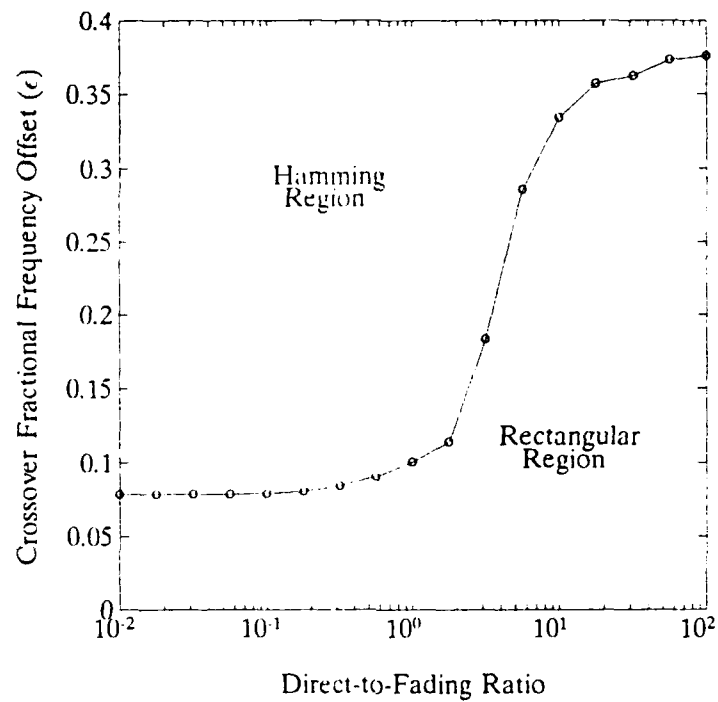


Figure 4.4: Trade-off for various direct-to-fading ratios $A/2\sigma_f^2$. As the channel approaches Rayleigh fading, the frequency offset necessary to justify non-uniform windowing decreases (maintaining a BER of 10^{-2}).

V. ARQ PROTOCOLS AND THEIR PERFORMANCE

As seen in the previous section, noise on a communication channel can cause errors in the reception of signals. When an error occurs in computer communications, the receiving DLC must issue a command to the sender for retransmission of the frame. This negative acknowledgement (NAK) of the received frame, or acknowledgement (ACK) if no errors occurred, can be a relatively short frame with a minimal chance of error or imbedded in a longer information frame with a higher probability of error occurring in the frame. As a safeguard, the original sending node assigns a maximum allowable waiting time, or *timeout*, where, upon its expiration, the frame is assumed to have been received in error and is then retransmitted.

The efficiency of ARQ schemes is dependent upon the complexity of the DLC to implement it; the more complex the implementation, the higher the throughput rate. Current ARQ protocols—stop-and-wait, go-back-N and selective-repeat—are based on a fixed frame length. As will be seen from the throughput curves generated for each protocol, a higher performance may be achieved in areas of different bit error rates with different frame lengths.

A. STOP-AND-WAIT ARQ

The stop-and-wait ARQ protocol is the simplest protocol to implement and serves as a good example to illustrate ARQ schemes. In the stop-and-wait protocol, each frame is transmitted and the sending DLC then stands idle waiting for a response. If a negative response is made (either by a NAK or by a timeout), the frame is retransmitted. This process continues until all frames have been transmitted. As an

example, a communication is illustrated in Figure 5.1 between a sending DLC (node A) and a receiving DLC (node B). Here, frame '0' is sent and received correctly by node B who ACKs the packet by requesting the next frame (frame '1'). An error was encountered with frame '1' and node B NAKs (negatively acknowledges) the frame by re-requesting frame '1' which is retransmitted by node A. The ACK from node B encounters an error and at the conclusion of the timeout, frame A retransmits the frame.

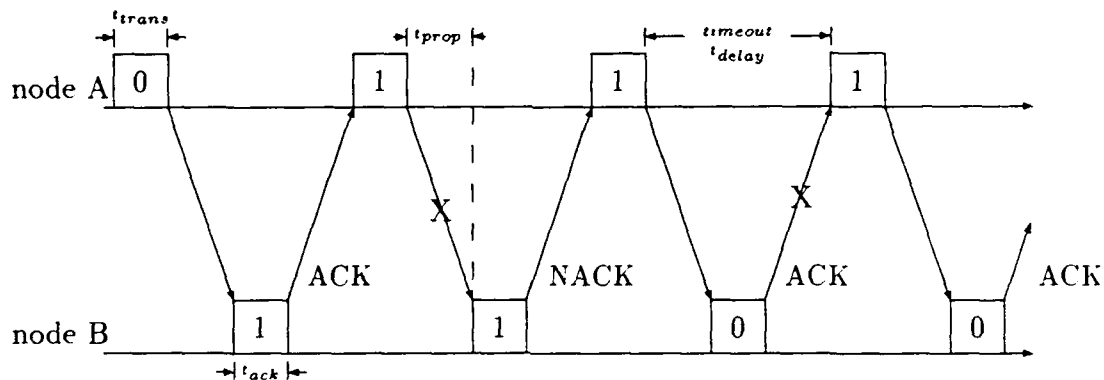


Figure 5.1: Stop-and-wait protocol. Frame '0' is sent and ACKed by node B. An error occurs with frame '1' and is NAKed by B. The ACK from node B encounters an error, and at the conclusion of the timeout, frame A retransmits the frame.

1. Performance Analysis Background

Performance for each of the three protocols will be developed in their respective sections. Before proceeding with the analysis for the stop-and-wait protocol, the following background used in all three performance analyses is developed.

a. Assumptions

The following assumptions are made throughout the performance analysis:

- Node (A) continuously transmits to node (B). — With node A continuously transmitting, there are no queuing delays. Because node B has no information to send to node A, acknowledgement frames will only contain the ACK or NAK.
- An acknowledgement is always received and is never received in error. — This assumption can be made because of the relatively short length of the acknowledgement frame.
- The propagation delay is known. — This stipulation is only necessary to establish an appropriate window size with the go-back- N and selective repeat protocols.
- Processing time at each node is negligible and can be considered included in the propagation delay.
- Bit error is independently and identically distributed for all bits within a frame.

b. Average Transmission Attempts

If the probability P_b of bit error is independent and identically distributed for each bit within a frame (L bits long), the probability of a frame being received in error is given by

$$P_e = 1 - P_{\text{success}} = 1 - (1 - P_b)^L. \quad (5.1)$$

The probability that it will take i attempts to successfully transmit a packet can be expressed as

$$Pr[\text{number of attempts} = i] = P_e^{i-1}(1 - P_e). \quad (5.2)$$

The expected number of attempts can be computed by ¹

$$E [P_e^{i-1}(1 - P_e)] = (1 - P_e)E [P_e^{i-1}] \quad (5.3)$$

$$= (1 - P_e) \sum_{i=1}^{\infty} i P_e^{i-1} \quad (5.4)$$

$$= \frac{1}{1 - P_e}. \quad (5.5)$$

c. Throughput Definition

Throughput is defined as

$$\rho \triangleq \frac{\text{total bits successfully transmitted in a given time interval}}{\text{bit capacity of the channel in the same interval}}. \quad (5.6)$$

This can also be expressed as the ratio of the time to transmit a frame t_{trans} to the average time to transmit a frame without error $\bar{t}$.

Because the information packet is appended with overhead bits ℓ , some of which contains coding information, the throughput is scaled by this 'coding' rate. This results in a throughput efficiency given by

$$\rho \triangleq \frac{t_{trans}}{\bar{t}} \frac{(L - \ell)}{L}. \quad (5.7)$$

2. Stop-and-Wait Performance

The total time the transmitter expends for each frame is a sum of the transmission time of the frame, transmission time of the acknowledgement t_{ack} and two propagation delays t_{prop} (which are assumed to contain the processing delay) for the round-trip cycle. This gives a total time for each transmission attempt of

$$T_L = t_{trans} + 2t_{prop} + t_{ack}. \quad (5.8)$$

¹The infinite sum is shown to be convergent by taking the derivative of the geometric infinite series solution where $x < 1$

$$\frac{\partial}{\partial x} \sum_{n=0}^{\infty} x^n = \frac{\partial}{\partial x} \frac{1}{1-x} \longrightarrow \sum_{n=1}^{\infty} n x^{n-1} = \frac{1}{(1-x)^2}$$

Using the results of Equation 5.5 for the average number of attempts, the average time to transmit a frame successfully is given by

$$\bar{t}_{sw} = \frac{t_{trans} + 2t_{prop} + t_{ack}}{1 - P_e}. \quad (5.9)$$

By applying the definition of throughput given in Equation 5.7 and defining the number of frames that can be sent in one round-trip propagation delay as

$$a \triangleq \frac{2t_{prop} + t_{ack}}{t_{trans}}, \quad (5.10)$$

the throughput of the stop-and-wait strategy is

$$\rho_{sw} = \frac{(1 - P_e)(L - \ell)}{1 + a} \frac{1}{L} = \frac{(1 - P_e)^L (L - \ell)}{1 + a} \frac{1}{L}. \quad (5.11)$$

Theoretical curves for Equation 5.11 are given in Figure 5.2 for various frame lengths. The simulations described in Appendix B were performed and the results are given here as well.

Although the stop-and-wait protocol is easy to implement, the efficiency of the system is compromised by waiting for a response after each packet is transmitted. The following two protocols do not stand idle waiting for a response and, subsequently, have better expected throughput than the stop-and-wait.

B. GO-BACK- N

In the go-back- N protocol, node A continuously sends frames to node B. Node B keeps node A aware of its status by responding with an acknowledgement which contains an RN (received number) indicating that all frames preceding RN have been received correctly. RN also indicates that all frames from RN onward need to be sent by A. As frames are received without error, RN is incremented.

In the event of an error, RN will remain constant as illustrated in Figure 5.3. There is a limit to the number of frames node A can send beyond RN so that a frame

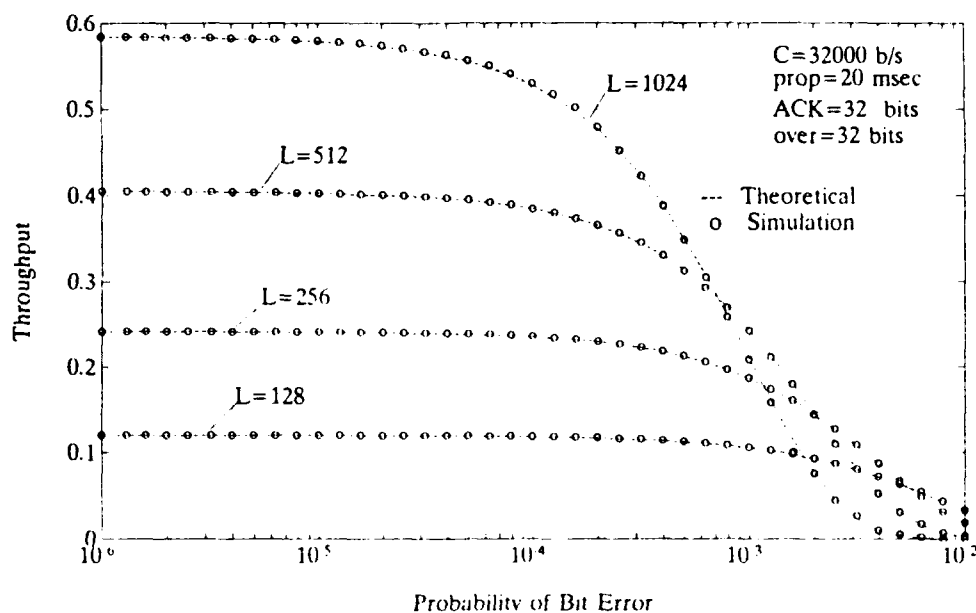


Figure 5.2: Performance of stop-and-wait for frame lengths of 1024, 512, 256 and 128. Channel capacity is 32000 b/s and a round-trip propagation delay of 0.02 seconds.

in error can be detected and retransmitted by node A. The number of frames that can be transmitted beyond RN is the N in go-back- N and is referred to as the window length. Because RN remains constant, when an error occurs, node A will stop as soon as the end of the window is reached ($SN + 1 = RN + N$) and go back N frames and begin retransmission of frame RN and all subsequent frames. To minimize the number of frames needed to be retransmitted in the event of an error, the value of N is set to the number of frames that can be transmitted in the time interval given in Equation 5.8. Using Equation 5.10,

$$N = a + 1. \quad (5.12)$$

With the go-back- N protocol, the transmitter can send frames continuously. However, upon each frame received in error, the protocol requires that the sending DLC go back N frames and resend them. Thus, when successful on the i th attempt,

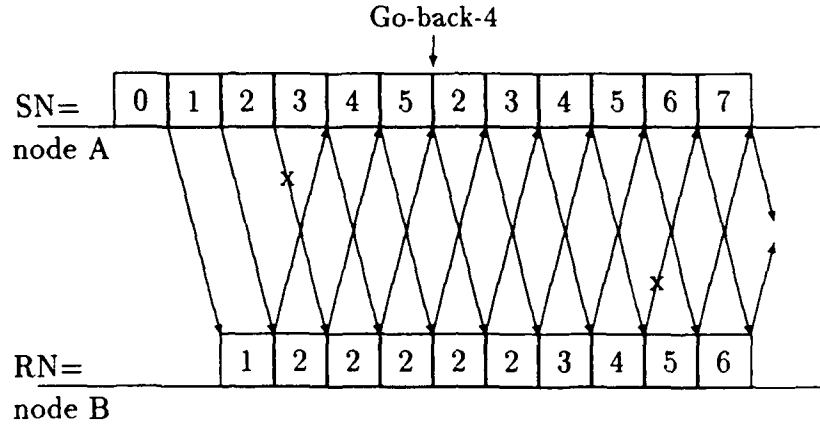


Figure 5.3: Go-back- N protocol for $N = 4$. With frame 2 received in error. RN stays constant at 2 until node A has reached the end of the window. Node A detects the error and must "go back 4" frames.

the DLC has sent a total of $1 + (i - 1)N$ frames. The average number of attempts was given in Equation 5.5 which results in an average number of frames ($\bar{f}$) that need to be sent of

$$\begin{aligned}\bar{f} &= 1 + \left(\frac{1}{1 - P_e} - 1\right)N \\ &= \frac{1 + P_e(N - 1)}{1 - P_e}.\end{aligned}\tag{5.13}$$

Using Equation 5.12, the average time to transmit a frame successfully is

$$\begin{aligned}\bar{t} &= t_{trans}\bar{f} \\ &= \frac{t_{trans}(1 + aP_e)}{1 - P_e}.\end{aligned}\tag{5.14}$$

Applying the definition given by Equation 5.7, the theoretical throughput for the go-back- N protocol becomes

$$\rho_{gbn} = \frac{(1 - P_e)(L - \ell)}{1 + aP_e} \frac{1}{L}.\tag{5.15}$$

where the dependency of P_e on L is shown in Equation 5.1. A graph of theoretical throughputs for various frame lengths is given in Figure 5.4 with the simulation results superimposed. It can be seen from the graph that an adaptive strategy will not improve the performance of the go-back- N protocol. A frame length of 1024 could be used at all bit error rates and only a fractional percent of throughput would be lost at higher bit error rates.

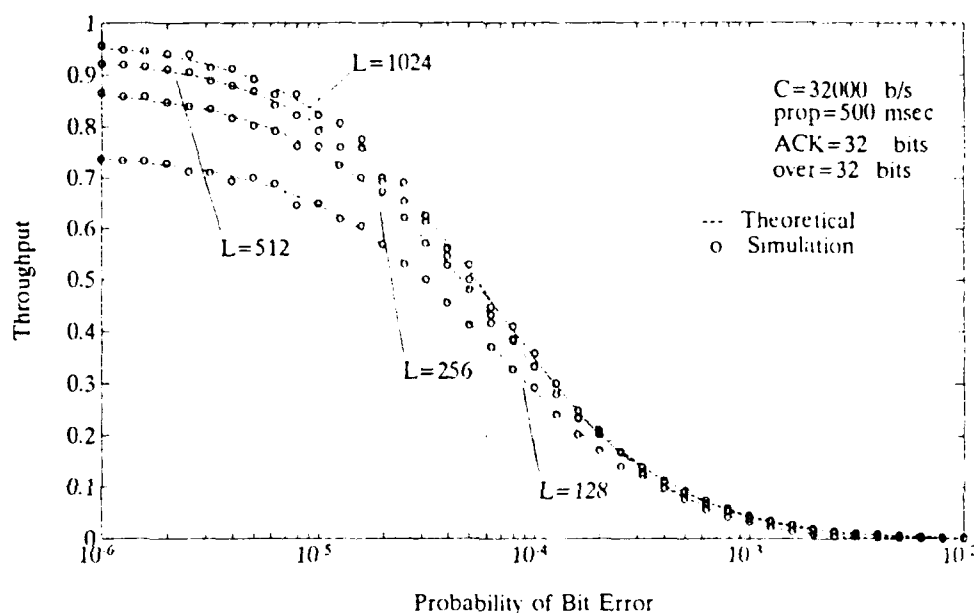


Figure 5.4: Performance of go-back- N for various frame lengths. A frame length of 1024 could be used at all bit error rates with only minimal loss of performance at higher BER.

C. SELECTIVE-REPEAT ARQ

The selective-repeat protocol is similar to the go-back- N except that when an error occurs, only the frame that is in error needs to be retransmitted. Here it is important to acknowledge each frame (both positive and negative) so that the transmitter accounts for all frames sent out. If a response is not received, that frame reaches its timeout and is retransmitted.

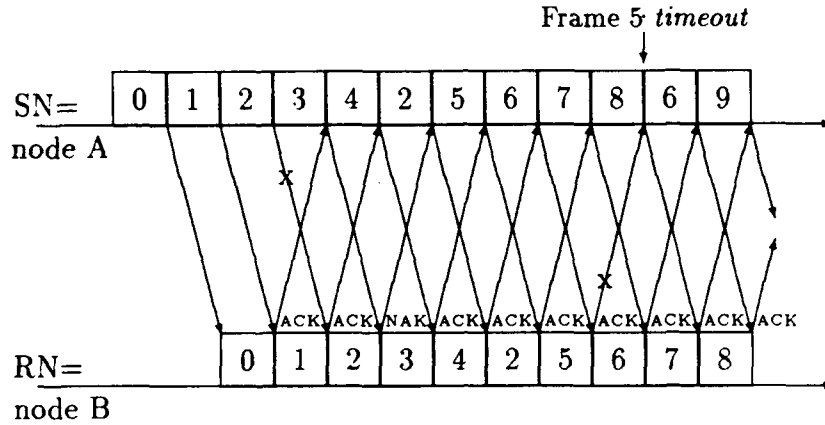


Figure 5.5: Selective repeat protocol.

The obvious advantage of selective-repeat over go-back- N is that only the frame in error is retransmitted as opposed to sending all following $N - 1$ frames in the go-back- N protocol. However, this requires that the receiving DLC have adequate memory to store frames that it receives and to be able to sort them as they become available. If memory in the receiver is full, then all incoming frames are NAKed, regardless of any errors.

The throughput analysis for selective-repeat (with an infinite receiver buffer size) is straightforward from Equation 5.5. Because the only delay experienced is that of the retransmission of a single frame when an error occurs,

$$\bar{t} = \frac{t_{trans}}{1 - P_e} \quad (5.16)$$

which results in a theoretical throughput for selective-repeat with an infinite receiver buffer as

$$\rho_{sr} = (1 - P_b)^L \frac{L - \ell}{L}. \quad (5.17)$$

Plots of theoretical throughputs with infinite receiver buffer length are given in Figure 5.6 with the results obtained in the simulation.

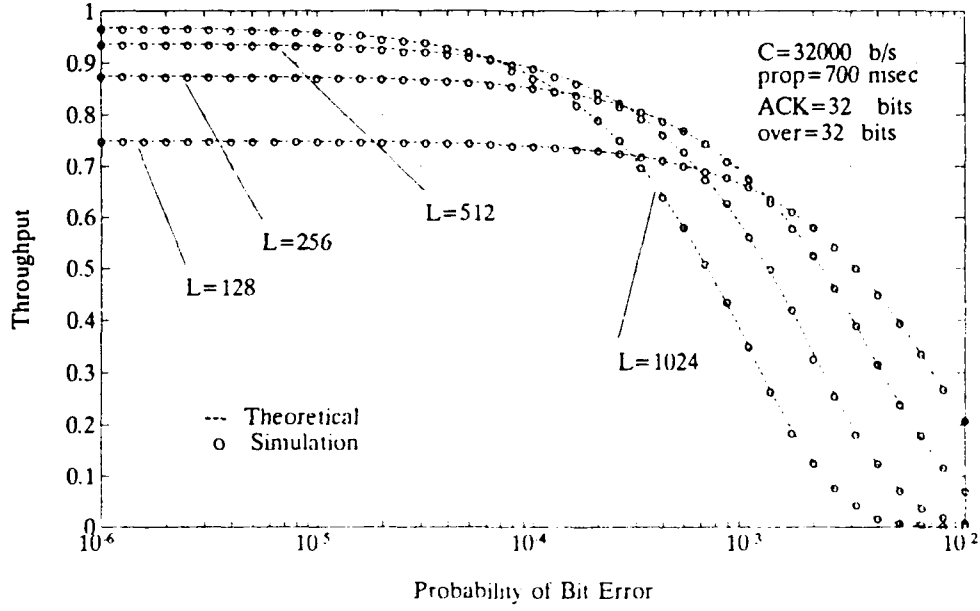


Figure 5.6: Performance of selective repeat with an infinite buffer.

By limiting the size of the buffer to the number of frames that can be transmitted in a round-trip delay N , a lower bound on the throughput is given in [6] by

$$\rho \geq \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2 N} \quad (5.18)$$

where

$$\begin{aligned} \lambda_0 &= \frac{\beta}{1-\gamma} (1 - \beta\gamma^{N-1}) \\ \lambda_1 &= P_{succ}^2 (\alpha^{N-2} + P_e \beta^{N-2} + P_e^2 \gamma^{N-2}) \\ \lambda_2 &= 3 - P_{succ}^2 \alpha^{N-2} - \alpha^2 \beta^{N-2} - \beta^2 \gamma^{N-2} \end{aligned}$$

and

$$\begin{aligned} \alpha &= 1 - P_e^2 \\ \beta &= 1 - P_e^3 \\ \gamma &= 1 - P_e^4 \end{aligned}$$

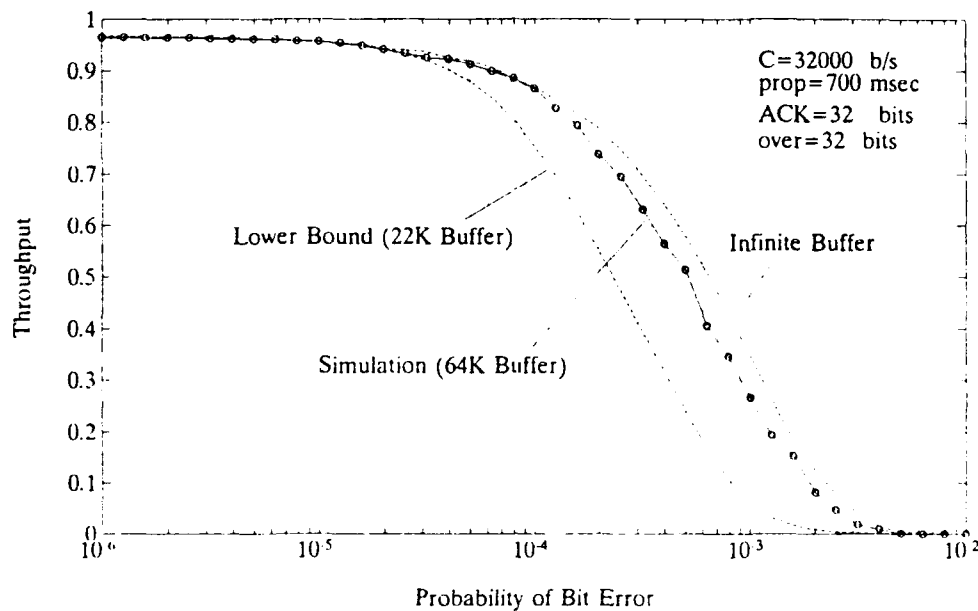


Figure 5.7: Performance of SR with finite receiver buffer (fixed length $L = 1024$). $N = 22$ is the number of frames transmitted in a round-trip delay. From the analysis of [6], a lower bound on throughput is given.

A comparison of the upper bound (infinite receiver buffer) and the lower bound given in Equation 5.18 is given in Figure 5.7 for a fixed length of $L = 1024$. For the simulation, a receiver buffer size of 64K is used instead of restricting it to N ($N = 22$ for the parameters of the simulation). A buffer size of 64K was believed not to be too excessive, yet imposes some restriction on the system.

VI. ADAPTIVE ARQ PROTOCOLS

Implementation of the previously described ARQ schemes remains the same in an adaptive environment. The adaptation takes the form of varying the length L of the frames. With an increased bit error rate, the increase in frame error rate in Equation 5.1 can be offset by decreasing the length of the frame. Alternatively, if the bit error rate of the channel decreases, a longer frame length will decrease the percentage of a frame dedicated to overhead (assuming the overhead remains constant for varying lengths).

A. ADAPTIVE STOP-AND-WAIT USING A SEMAPHORE

The decision to change the length of the frame was made in three different schemes. The first scheme is taken from [7] and invokes a counting semaphore m that is incremented each time a frame is acknowledged and decreased each time a frame needs to be retransmitted. The length of the frame is then dependent upon the current value of the semaphore by

$$L = \begin{cases} l & \gamma_0 \leq m < \gamma_1 \\ 2l & \gamma_1 \leq m < \gamma_2 \\ \vdots & \vdots \\ nl & \gamma_{n-1} \leq m \leq \gamma_n. \end{cases} \quad (6.1)$$

For this thesis, $l = 128$ and $n = 4$.

Figure 6.1 shows the gain associated with an adaptive stop-and-wait protocol. This scheme adds little complexity to the DLC implementation and provides the desired results.

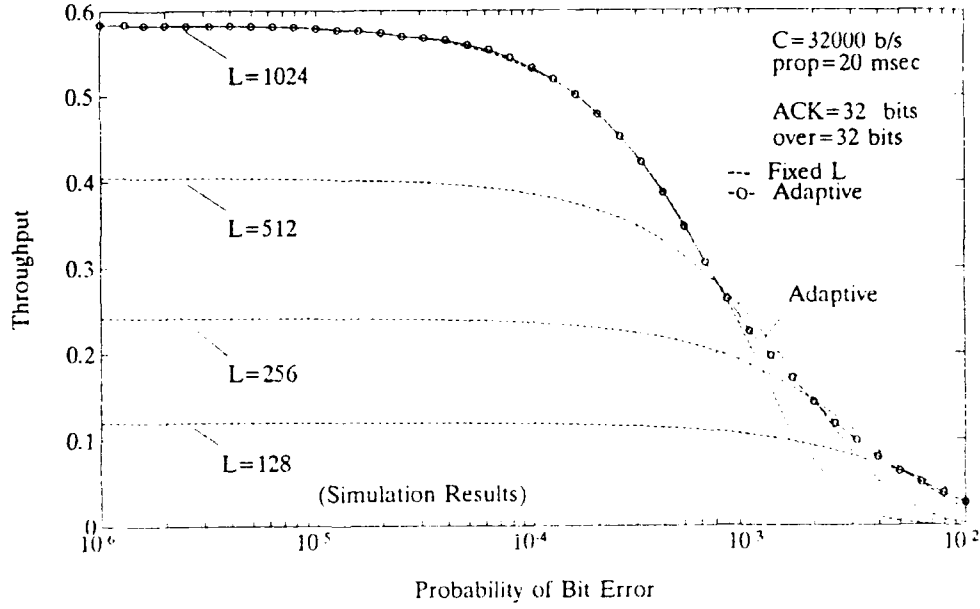


Figure 6.1: Adaptive stop-and-wait performance. Improved performance is obtained in the region of high BER.

B. ADAPTIVE SELECTIVE-REPEAT USING THRESHOLDS

Another criteria compared the number r of frames transmitted after a fixed value b of frames were received in error to a threshold to determine if the frame length should be increased (doubled) or decreased (halved). Two schemes were implemented where either a single threshold

$$L = \begin{cases} L/2 & b/r > \gamma \\ 2L & b/r \leq \gamma \end{cases} \quad (6.2)$$

or a dual threshold

$$L = \begin{cases} L/2 & \rho_{L/2} > \rho_L \\ 2L & \rho_{2L} > \rho_L \\ L & \text{otherwise} \end{cases} \quad (6.3)$$

was used. The computation of $\rho_{L/2}$, ρ_L and ρ_{2L} are based on the experienced bit error rate. The experienced P_b can be computed from the ratio of transmitted frames to frames received in error by applying the inverse of Equation 5.1

$$P_{success} = \frac{r-b}{r} = 1 - P_e = (1 - P_b)^L$$

$$P_b = 1 - \left(\frac{r-b}{r} \right)^{1/L}. \quad (6.4)$$

Throughputs for lengths $2L$ and $L/2$ can then be computed for the selective repeat protocol with Equation 5.17. The decision criteria of Equation 6.3 then simplifies to

$$L = \begin{cases} L/2 & 1 - \ell/(L - \ell) > [(r-b)/r]^{1/2} \\ 2L & 1 - \ell/(2L - \ell) < (r-b)/r \\ L & \text{otherwise} \end{cases} \quad (6.5)$$

Table 6.1 can be constructed for the dual threshold selective-repeat case. The determination of γ in Equation 6.2 was taken as an average ($\gamma = 0.88$) of the thresholds necessary to cause transition from table 6.1.

	$L \rightarrow 2L$	$L \rightarrow L/2$
$L = 128$	0.857	N/A
$L = 256$	0.933	0.735
$L = 512$	0.968	0.871
$L = 1024$	N/A	0.936

TABLE 6.1: Adaptive SR transition levels. The necessary value of $(r-b)/r$ to cause transition either up or down with the dual threshold implementation ($\ell = 32$ bits).

The performance of the adaptive selective-repeat for a fixed buffer length is given in Figure 6.2. Implementation of this scheme further increases the complexity of the selective-repeat protocol. Now the transmitting node must keep track of the length of each frame it is sending out. In addition, re-arrangement of the frames at the receiving node becomes quite complex. For example, if a frame is NAKed and the decision to change frame lengths is made, the packet sent in error will now be divided between two frames each of which may or may not be received correctly in order upon retransmission.

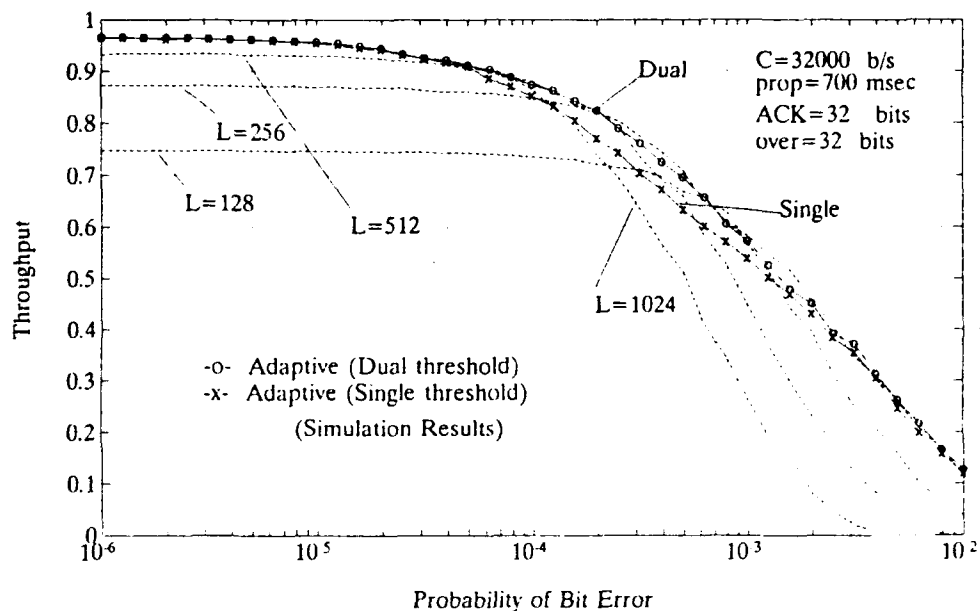


Figure 6.2: Adaptive selective-repeat. Both dual and single threshold results are given. Performance of the single threshold implementation is slightly lower in the areas where the decision to change lengths can "toggle" between two lengths.

C. GO-BACK- N ARQ

The go-back- N protocol does not lend itself to an adaptive environment as was seen in Figure 5.3. The difference in throughput rates between the largest frame length used (1024) and the maximum achievable throughput is negligible. This is because the inefficiency of go-back- N arises from the necessity to go back N frames when an error in a frame occurs. This is depicted in Figure 6.3 where, if the first frame is in error, the $N - 1$ following frames composed of L bits each result in a possibly unnecessary retransmission of $(N - 1)L$ bits. If the frame length is halved (and N doubled to satisfy Equation 5.12), the probability of having to re-transmit the same number of bits is the same because the probability of transmitting two successive frames of length $L/2$ correctly is equal to the probability of transmitting a single frame of length L without error. Although some advantage may be gained

in successfully transmitting the first frame of length $L/2$, there is also an increase in percentage overhead for each frame. For example, with a BER of 10^{-4} , an initial length of $L = 1024$ and overhead $\ell = 32$, if the frame length is halved, P_{succ} increases by only 4.7% while the percentage of frame overhead increases 3.1%. This results in negligible improvement in the areas of higher bit error rates.

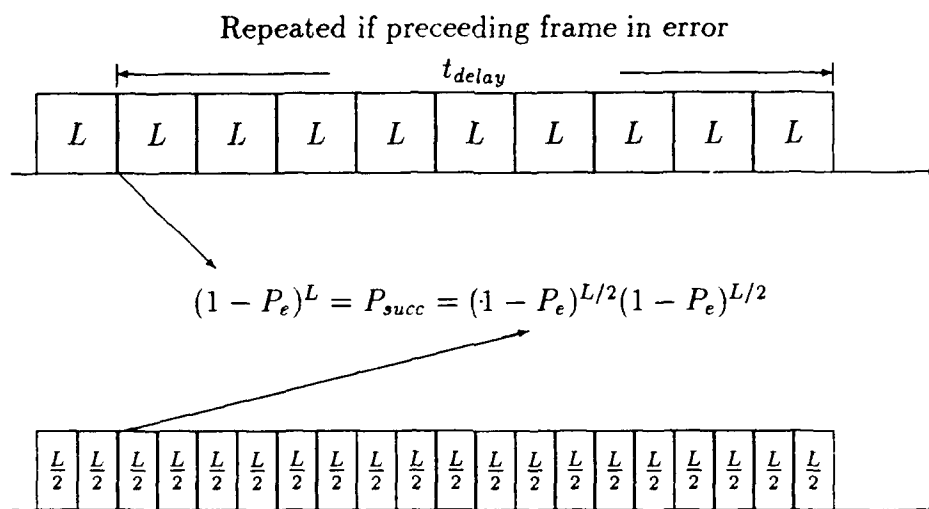


Figure 6.3: Inefficiency of the go-back- N protocol occurs because frames following an error must be repeated. If the frame length is halved, the probability of retransmitting the same number of bits remains the same.

VII. CONCLUSIONS

In an M-ary FSK scheme implemented using sampled data at the receiver and a FFT to determine which signal was sent, it was found that non-uniform windowing can be used to increase the performance of the system in the presence of large frequency offsets. Implementing non-uniform windowing causes greater separation between frequency components of the FFT when frequency shifts occur between transmission and reception. Non-uniform windowing also causes attenuation of the signal making detectability more difficult. A balance between improved frequency separation and attenuation is achieved as the frequency offset becomes more dramatic. In the case of a fading channel, the frequency offset necessary to justify non-uniform windowing becomes smaller as the channel approaches Rayleigh fading.

The application of an adaptive ARQ protocol was shown to improve the throughput of the stop-and-wait and selective-repeat ARQ protocols. With an adaptive frame length, throughput can be increased by decreasing the probability of frame error in high bit error rate situations through a reduction in the frame length. With low bit error rates, a larger frame length reduces the percentage overhead in the frame thereby increasing the throughput. The go-back- N protocol does not lend itself to adaption. From theoretical analysis and simulation, it was found that a fixed frame length can be used in the go-back- N protocol through all bit error rates with minimal loss of performance.

APPENDIX A: ERROR COMPUTATION

The probability given in Equation 3.2 was computed using the enclosed C program code. Values for the Marcum-Q function and the Modified Bessel Function were calculated to within a designated tolerance (10^{-8}) using power series expansions of the functions from [4] and [5]

$$Q(\alpha, \beta) = \begin{cases} 1 & \text{if } \beta = 0 \\ e^{-\beta^2/2} & \text{if } \alpha = 0 \\ \frac{1}{2}(1 + I_0(\alpha^2))e^{-\alpha^2} & \text{if } \alpha = \beta \\ \exp\left[-\frac{\beta^2 + \alpha^2}{2}\right] \sum_{n=0}^{\infty} \left(\frac{\alpha}{\beta}\right)^n I_n(\alpha\beta) & \text{if } \alpha < \beta \\ \exp\left[-\frac{\beta^2 + \alpha^2}{2}\right] \sum_{n=1}^{\infty} \left(\frac{\beta}{\alpha}\right)^n I_n(\alpha\beta) & \text{if } \alpha > \beta \end{cases} \quad (\text{A.1})$$

where

$$I_n(x) = \begin{cases} 1 & \text{if } x = 0 \\ (x/2)^n \sum_{i=0}^{\infty} \frac{(x/2)^{2i}}{i!(v+i)!} & \text{otherwise} \end{cases} \quad (\text{A.2})$$

1. Bit Error Program Code

```

/*****
 * Pefad.c
 * Calculates the probability of bit error for a Rician
 * fading channel and a DFT based receiver for MFSK
 *
 * by: Chris G. Kmieciak
 * date: 22 April, 1990
 *****/
#include <stdio.h>
#include <math.h>

#define p1 3.141592654
#define sqrt2 1.414213562
#define epsilon 0.00000001

#define N 32
#define M 8
#define l 3
#define W 16
#define df 2
double window[N];

main()
{
double mip(), mrp(), I(), Q(), power(), rectangle(), hamming(), hanning();
double Perror, lastPerror, Enoise;
double a, b, c, numb, numc, denom;
double meanrq, meaniq, betaq;
double meanrk, meanik, betak;
long fact();
FILE *output;
int n, k, q, einc;
float SNRdb, SNR, DTF, DTFinc, ratio, e, freqe;

system("rm PefadoutHAMM.mat");

Enoise=0.0;
for (n=0 ; n <= M-1 ; n++) {
    window[n+1]=hamming(n);
    Enoise=Enoise+W*window[n+1]*window[n+1]/2.0;
}

```

```

for (DTFinc=2.0 ; DTFinc >= -2.0; DTFinc = DTFinc-0.25){/* Specify Direct to */
    DTF=pow(10.0,DTFinc);                                /* fading ratio */

for (e=0.0 ; e <= 0.5 ; e=e+0.1) {                    /* Cycle through various */
                                                    /* frequency shifts */

    Perror=3.0; lastPerror=4.0;
    SNRdb=2;

    while((Perror >= 0.00001)                          /* Continue computing Pb */
        &&(SNRdb <= 99)                                /* until SNR becomes large */
        &&(lastPerror-Perror >= .00001)){                /* or error becomes small */
            lastPerror=Perror;                          /* or error does not change */
            SNRdb++;
            SNR=1*pow(10.0,SNRdb/10.0);                /* Convert Eb to Es */

ratio=SNR/(DTF+1);
Perror=0.0;

for (q=1 ; q <= M*df-1 ; q=q+df) {                    /* Each value of q donates */
    meanrq=mrp(q,e,q);                                /* to Pe. First */
    meaniq=mip(q,e,q);                                /* find the mean of the real*/
    betaq=meanrq*meanrq+meaniq*meaniq;                 /* and imaginary parts then */
    numc=0.5*betaq*ratio+Enoise;                       /* specify all parameters */
    numb=(SNR-ratio)*betaq;                             /* changing with q only. */

    for (einc = 1; einc <= 2; einc++) {                /* Include both plus and */
        freqe=e*power(-1.0,einc);                     /* minus Doppler shift */

        for (k=1 ; k <= (N/2)-1 ; k=k+df) {
            if (k != q) {
                meanrk=mrp(q,freqe,k);                /* Calculate values in */
                meanik=mip(q,freqe,k);                /* equations 2.17, 2.18, */
                betak=meanrk*meanrk+meanik*meanik;     /* 2.22, 3.1. */

                c=numc/(0.5*betak*ratio+Enoise);
                denom=0.5*(betaq+betak)*ratio+2.0*Enoise;
                a=(SNR-ratio)*betak/denom;
                b=numb/denom;

                Perror=Perror+Q(sqrt(a),sqrt(b))        /* Equation 3.2 */
                    -(c*exp(-0.5*(a+b))*I(0,sqrt(a*b)))/(c+1);

            } /* if (k != q) */
        } /* for (k=1 ; k <= (N/2)-1 ; k=k+df) */
    } /* for (einc = 1; einc <= 2; einc++) */
} /* for (q=1 ; q <= M*df-1 ; q=q+df) */

Perror=Perror/(2.0*M);                                /* Equation 3.1 */
if (Perror <= 0.2) {
    output=fopen("PefadoutHAMM.mat","a");

```

```

        fprintf(output,"%10.8f  %4.1f  %4.2f  %5.2f\n",Perror,SNRdb,e,DTF);
        fclose(output);
    }

    } /* while (Perror >= 0.00001) */
    } /* for (e=0.0; e <= 0.5 ; e=e+0.1) */
    } /* for (DTFinc=2 ; DTFinc >= -2; DTFinc = DTFinc + 0.02) */
} /* close main */

```



```

/*****
*
*           Function mrp
*
*****/
double mrp(q,e,k)
int    k,q;
float  e;
{
int    n;
double ans;

ans=0.0;
for (n=0; n <= N-1; n++) {
    ans=ans+cos(2*pi*n*(q+e+k)/N)*window[n+1]+cos(2*pi*n*(q+e-k)/N)*window[n+1];
}

if ((e == 0.0) && (k != q)) ans = 0.0;

ans=ans/sqrt2;
return(ans);
}

/*****
*
*           Function mip
*
*****/
double mip(q,e,k)
int    k,q;
float  e;
{
int    n;
double ans;

ans=0.0;
for (n=0; n <= N-1; n++) {
    ans=ans+sin(2*pi*n*(q+e+k)/N)*window[n+1]-sin(2*pi*n*(q+e-k)/N)*window[n+1];
}

if ((e == 0.0) && (k != q)) ans = 0.0;

ans=ans/sqrt2;
return(ans);
}

```

```

/*****
*
*           Marcum-Q Function
*
*****/
double      Q(alpha,beta)
double      alpha, beta;
{
double      ans, e, inc;
int         n;

if (beta == 0.0)
    ans = 1.0;
else if (alpha == 0.0)
    ans = exp(-0.5*beta*beta);
else if (alpha == beta)
    ans = 0.5*(1.0+I(0,alpha*alpha)*exp(-1*alpha*alpha));

else if (alpha <= beta) {
    ans = 0.0;
    e = exp(-0.5*((alpha*alpha)+(beta*beta)));
    n=0; inc = 1.0;
    while (inc >= epsilon) {
        inc=e*power(alpha/beta,n)*I(n,alpha*beta);
        ans = ans + inc;
        n++;
    }
}
else {
    ans = 0.0;
    e = exp(-0.5*((alpha*alpha)+(beta*beta)));
    n=1; inc = 1.0;
    while (inc >= epsilon) {
        inc=e*power(alpha/beta,n)*I(n,alpha*beta);
        ans = ans + inc;
        n++;
    }
    ans=1.0-ans;
}

return(ans);
}

```

```

/*****
*
*          Modified Bessel Function
*
*****/
double    I(v,x)
int        v;
double     x;

{
double     ans, inc;
int        n;

if (x == 0.0)
    ans = 1;

else {
    x=x/2.0;
    ans=power(x,v)/fact((long) v);

    inc=ans; n=1;
    while (inc >= epsilon) {
        inc=inc*x*x/(n*(v+n));
        ans=ans+inc;
        n++;
    }
}

return(ans);
}

```

```

/*****
*
*           Function rectangular
*
*****/
double rectangle(n)
int n;

{
return(1.0);
}

/*****
*
*           Function Hamming
*
*****/
double hamming(n)
int n;

{
return(0.54-0.46*cos(2*pi*n/(N-1)));
}

```

```

/*****
 *
 *          Factorial
 *
 *****/
long fact(x)
long x;

{
long i, ans;

ans=1;
if (x == 0)
    ans = 1;
else
    for (i = 1; i <= x; i++) ans = ans*i;

return(ans);
}

/*****
 *
 *          POWER
 *
 *****/
double power(x,n)
int    n;
double x;
{
double p;
int    recip;

if (n < 0) {
    recip=1;
    n=abs(n);
}

for (p=1.0; n > 0; --n) p=p*x;

if (recip == 1) p=1.0/p;

return(p);
}

```

APPENDIX B: ARQ SIMULATIONS

The ARQ simulations were carried out using a C-program. Although one node would be transmitting to another in the application of an ARQ protocol, the simulation only involves one node. A doubly-linked list maintains a record of jobs to be performed in chronological order by the node. The node can either transmit ('SEND') or receive an acknowledgement ('RECV') for a given frame.

For each transmission, the time of completion of transmission and the computed time of reception (based on Equation 5.8) are added to the list of jobs. At the completion of transmission, the node then transmits the next frame if the protocol allows it.

At the execution of 'RECV', a uniformly distributed, pseudorandom number between zero and one is compared to the theoretical frame error rate (Equation 5.1). If the comparison dictates, a NAK is recorded by the node and the appropriate retransmission protocol is applied. Particulars for implementing retransmission in each protocol are given below.

1. Selective-Repeat

This simulation served as the template for the others (see Section B.4) in that go-back- N and stop-and-wait are "subsets" of this program. The adaptive nature of the program is contained in a section added after receiving each frame's acknowledgement. (see Figure B.1).

2. Go-Back-N

Before sending a frame, the window is checked to see if it is full. If the window is full SN is reset to the beginning of the window (Figure B.2).

3. Stop-and-Wait

Here, only the time to receive each packet is loaded onto the linked list. When a frame is received, it is either ACKed or NAKed and a frame is sent out (see Figure B.3). If the frame is ACKed, the number of successful bits is incremented.

4. Selective-Repeat Program Code

```

/*****
 * sr.c
 *
 * Simulation of selective repeat protocol with data generated
 * for BER from 1E-6 to 1E-2.
 *
 * Receiver buffer size is kept track of by the sender
 *
 * by: Chris G. Kmiecik, LT, USCG
 * date: 05 February, 1990
 *****/
#include <stdio.h>
#include <malloc.h>
#include <math.h>

#define MALLOC(x) ((x *)malloc(sizeof(x)))
#define max_FN 4096
#define max_buffer 65536.0

typedef struct job {
    float      time, length;
    char       type[5];
    int        FN;
    struct job *previous, *following;
} job_type;

float      logBER;
job_type   *top_job;
FILE       *of, *tf, *rf, *ttf;

main()
{
    int      framestatus[max_FN], framelength[max_FN];

```

```

int          queue;
int          buffer, begin_window, SN, i;
float        completed, m, Ps;
float        BER, simulation_length;
float        capacity, propagation, overhead, ack_length;
float        rand, length;
float        successful_bits, throughput, tr, tt;
job_type     *temp, *previous_job, *job, *newjob();
unsigned long int seed;
char         *strcat();

seed=1234;
system("rm throughsr");

simulation_length=250.0;          /*          */
capacity = 32000;                 /* Simulation Parameters */
propagation = 0.7;                /*          */
overhead = ack_length = 32.0;     /*          */

for (length =1024.0; length >= 128.0; length=length/2.0) {
    for (logBER = -6.0; logBER <= -2.0; logBER=logBER+.1) {

        BER = pow(10.0,logBER);          /*          */
        begin_window = queue = SN = i =1; /* Initialize variables */
        successful_bits = buffer = 0.0;   /*          */
        while (i <= max_FN) framestatus[i++]=0;
        m=completed=0;

        top_job = newjob(0.0,"send",0,0.0);
        top_job->previous = top_job;
        previous_job = top_job;

        job = newjob(simulation_length+1,"send",0,0.0);
        job->previous = previous_job;
        previous_job->following = job;
    }
}

```



```

/***** Begin Simulating *****/
while (top_job->time <= simulation_length){

/*****
*
*   Time to Transmit a Packet
*
*****/
    if (!(strcmp(top_job->type,"send"))){

i = begin_window;
queue = SN;
while (i != SN) {
    if (framestatus[i] == 3) {
        queue = i;
        break;
    }
    if (++i > max_FN) i=1;
}

/*
*   Re-transmit Packet in Queue
*
*****/
    if (queue != SN) {
        tt = length/capacity;
        job = newjob(top_job->time+tt,"send",0,0.0);
        place_job(job);
        tr = propogation+((length+ack_length)/capacity);
        job = newjob(top_job->time+tr,"recv",queue,length);
        place_job(job);
        framestatus[queue] = 4;
        framelength[queue] = length;
    }

/*
*   Transmit if Window not Full
*
*****/
    else if (buffer+length < max_buffer) {
        tt = length/capacity;
        job = newjob(top_job->time+tt,"send",0,0.0);
        place_job(job);
        tr = propogation+((length+ack_length)/capacity);
        job = newjob(top_job->time+tr,"recv",SN,length);
        place_job(job);
        buffer=buffer+length;
        framestatus[SN] = 1;
        framelength[SN] = length;
        if (++SN > max_FN) SN = 1;
    }
}

```

```

/*
 *      IF Window is full
 *      Try to Send After Receiving
 *      Next Packet
 *
 *****/
else {
    job = newjob(top_job->following->time,"send",0,0.0);
    place_job(job);
}
} /* Packet transmission complete */

/*****
 *
 *      Time to Recieve a Packet
 *
 *****/
else if (!(strcmp(top_job->type,"recv"))) {
    seed=seed*1103515245+12345; /* Psuedo-random */
    rand=(seed/65536) % 32768; /* generator */

/*
 *      NAK the received packet
 *
 *****/
if (rand/32767 > pow((1.0-BER),top_job->length)) {
    framestatus[top_job->FN]=3;
}

/*
 *      ACK the received packet
 *
 *****/
else {
    framestatus[top_job->FN]=2;
}

} /* Packet Received */

/*****
 *
 *      Update Window Beginning
 *
 *****/
while (framestatus[begin_window] == 2) {
    successful_bits=successful_bits+framelength[begin_window]-overhead;
    framestatus[begin_window]=0;
    buffer=buffer-framelength[begin_window];
    if (++begin_window > max_FN) begin_window=1;
}

```

```

/*****
*
*      Get next job
*      Free current job memory
*
*****/
temp=top_job->following;
free(top_job);
top_job=temp;

} /* repeat for next job -- "while(top_job->time<=simulation_length)" */

/***** Simulation Completed *****/

/*****
*
*      Record Throughput Results
*
*****/
ttf = fopen("throughsr","a");
throughput=successful_bits/(capacity*simulation_length);
fprintf(ttf,"%6.5f  %11.8f\n",throughput,BER);
fclose(ttf);

} /* next BER -- "for(logBER=-4.0;logBER<-2.0;logBER=logBER+.1)" */
} /* for (length =1024.0; length >= 128.0; length=length/2.0) */

} /* close main */

```

```

/*****
*
*           Function newjob
*
*****/
job_type  *newjob(time,type,FN,length)
int        FN;
float      time, length;
char       type[5];

{
job_type  *new;

if (!(new=MALLOC(job_type))) {
    fprintf(ttf,"out of the storage \n");
    fclose(ttf);
    fclose(rf);
    fclose(of);
    fclose(tf);
    exit(1);
}

new->time=time;
strcpy(new->type,type);
new->FN=FN;
new->length=length;
new->previous=new->following=NULL;
return(new);
}

```

```

/*****
*
*           Function place_job
*
*****/
place_job(job)
job_type  *job;
{
job_type  *current;

current=top_job;
while (job->time >= current->time) {
    current=current->following;
    if ((current->following == NULL) && (job->time >= current->time)) {
        current->following=job;
        job->previous=current;
        job->following= NULL;
        return;
    }
}

current->previous->following = job;
job->previous = current->previous;
current->previous=job;
job->following=current;
}

```

```

/*
 * Decision Criteria for Level
 * Change
 *
 *****/
if (m >= 1) {
    Ps=(completed-m)/completed;
    if (Ps < 0.87) {
        length=length/2.0;
        if (length < 128.0) length=128.0;
    }
    else {
        length = length*2.0;
        if (length > 1024.0) length=1024.0;
    }
    m=completed=0;
}

```

Figure B.1: Decision segment for single threshold.

```

/*****
 *
 *   Time to Transmit a Packet
 *
 *****/
if (!(strcmp(top_job->type,"send"))){

    /*
     *   Transmit if Window not Full
     *
     *****/
    if (SN-RN <= window) {

        :

    }

    /*
     *   IF Window is full
     *   Go-back-N
     *
     *****/
    else {
        SN=RN=1;
        job = newjob(top_job->time,"send",0,0.0);
        place_job(job);
    }
} /* Packet transmission complete */

```

Figure B.2: Go-back-N implementation.

```

/*****
 *
 *   Time to Recieve a Packet
 *
 *****/
if (!(strcmp(top_job->type,"recv"))) {
    seed=seed*1103515245+12345;
    rand=(seed/65536) % 32768;

/*
 *   ACK the received Packet
 *****/
    if (rand/32767 < pow((1.0-BER),top_job->length)) {
        successful_bits=successful_bits+top_job->length-overhead;
        if (SN == 1) SN=0;
        else SN=1;
    }

    tr=((length+ack_length)/capacity)+propogation;
    job = newjob(top_job->time+tr,"recv",SN,length);
    place_job(job);

temp=top_job->following;
free(top_job);
top_job=temp;
} /* repeat for next job */

```

Figure B.3: Stop-and-wait simulation block does not require a "Time to Transmit a Packet" section.

REFERENCES

1. Richard A. Yost, "On Nonuniform Windowing M-ary FSK Data in a DFT Based Detector," *IEEE Trans. on Communications*, Vol. COM-28, No. 12, pp. 2014-2019, December 1980
2. 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
3. S. Stein, "Unified Analysis of Certain Coherent and Noncoherent Binary Communication Systems," *IEEE Trans. on Information Theory*, Vol. IT-10, pp. 43-51, January 1964
4. I.S. Gradshteyn and I.M. Ryzhik, *Table of Integrals, Series and Products*, pp. 718 and 961, Academic Press, New York, 1980
5. Harry L. Van Trees, *Detection, Estimation, and Modulation Theory*, pp. 394-396, John Wiley and Sons Inc., New York, 1968
6. Philip S. Yu and Shu Lin, "An Efficient Selective-Repeat ARQ Scheme for Satellite Channels and Its Throughput Analysis," *IEEE Trans. on Communications*, Vol. COM-29, No. 3, pp.353-363, March, 1981
7. Kurtis J. Guth. "An Adaptive ARQ Strategy for Packet Switching Data Communications Networks," Master's thesis, Naval Postgraduate School, Monterey, California, 1989
8. David T. Tsuda, "Adaptive Go-Back-N: an ARQ Protocol for a Tactical VSAT Network," Master's thesis, Naval Postgraduate School, Monterey, California, 1989.
9. Dimitri Bertsekas and Robert Gallager, *Data Networks*, pp. 50-102, Prentice-Hall, New Jersey, 1987

INITIAL DISTRIBUTION LIST

	No. of Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3. Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
4. Superintendent, Naval Postgraduate School Atten: Professor Tri T. Ha, Code EC/Ha Naval Postgraduate School Monterey, California 93943-5000	5
5. Superintendent, Naval Postgraduate School Atten: Professor Glen A. Myers, Code EC/Mv Naval Postgraduate School Monterey, California 93943-5000	1
6. Commandant, G-PTE-1 U.S. Coast Guard Washington, D.C. 20593-0001	2
7. Commandant, G-TPP-2 U.S. Coast Guard Washington, D.C. 20593-0001	1

8. U.S. Department of Transportation Library
Code M-493.3, Room 2200
400 7th Street South-West
Washington, D.C. 20590

1