NTIA Report 94-310

High-Frequency Radio Channel Error Model

E.E. Johnson
D.F. Peach

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

Larry Irving, Assistant Secretary
for Communications and Information

June 1994

PREFACE

This report describes a model for errors occurring over high-frequency (HF) propagation channels that employ the FED-STD-1045 Automatic Link Establishment (ALE) modem. It is part of a series of three NTIA reports on a software package that simulates the various components of high-frequency radio automatic link establishment. The other two reports are: "High-Frequency Channel and Modem Simulator" (NTIA Technical Memorandum 94-163) and "A Software Simulator of High-Frequency Automatic Link Establishment" (NTIA Technical Memorandum 94-162). The simulator package has previously been used to evaluate technologies proposed for standardization in the Federal Standards for HF ALE radio, FED-STD-1045 through FED-STD-1049.

The objective of this report is to describe the design of the channel error model, and to document its implementation in the ALE simulation software.

The need for this software was recognized by the Institute for Telecommunication Sciences (ITS) within the National Telecommunications and Information Administration (NTIA), which established the functional requirements and specifications for the software. The code was then developed by Johnson Research of Las Cruces, New Mexico under contract to ITS. The development work was funded by the Department of Commerce (DOC).

The simulation software described here is currently used for research at NTIA and New Mexico State University (NMSU). Certain commercial names are identified in this report to describe some of the information. Such identification does not imply recommendation or endorsement of the companies or products by NTIA or NMSU.

CONTENTS

FIGURES

Page

TABLES

Page

ACRONYMS AND ABBREVIATIONS

ALE	Automatic Link Establishment.
AWGN	Additive White Gaussian Noise.
BLER	Block Error Rate.
BSC	Binary Symmetric Channel.
CCIR	International Radio Consultative Committee.
dB	Decibels.
DOC	Department of Commerce.
FEC	Forward Error Correction.
FSK	Frequency-Shift Keying.
FSM	Finite State Machine.
HF	High Frequency.
Hz	Hertz. 1 Hz is 1 cycle per second.
ITS	Institute for Telecommunication Sciences.
NTIA	National Telecommunications and Information Administration.
SChEMe	Skywave Channel Error Model. The software simulator that is the subject of this report.
SES	Single Error State.
SNR	Signal-to-Noise Ratio.

HIGH-FREQUENCY RADIO CHANNEL ERROR MODEL

Eric E. Johnson[1]
David F. Peach[2]

Simulation has been extensively employed to evaluate concepts included in the current generation of standards for automated high-frequency (HF) radio systems. As development proceeds from link-layer technology to network- and higher-layer technology, it is no longer necessary to devote great amounts of computer time to detailed simulation of the physical medium. Instead, the error behavior of the medium should be abstracted so that simulation resources can be concentrated on the phenomena of interest at these higher protocol layers.

In this report, a model of channel error behavior is derived that accurately captures the operation of the Automatic Link Establishment (ALE) modem operating over Gaussian noise and skywave channels. When this model is employed in place of a detailed simulation of the modem and channel, an overall simulation speedup of nearly two orders of magnitude results.

Key words: automatic link establishment; HF; high-frequency radio; performance analysis; simulation

1. INTRODUCTION

The current simulator suite for Automatic Link Establishment (ALE) HF radios (described in [1,2]) has been shown to accurately predict the impact on linking performance of a number of variations in the ALE protocols. However, this simulator requires significant computational resources: when executed on an IBM RS/6000 Model 340H workstation (SPECmark 34), each simulated linking attempt requires approximately 2.2 seconds of CPU time. For link-level simulations, this performance may be acceptable; linking performance is the metric of interest, and the simulation of 100 to 1000 linking attempts at perhaps 30 channel conditions requires only a few hours of CPU time on such a workstation.

However, when we are interested in evaluating the performance of networking algorithms and protocols, we want to simulate several days of operations, which corresponds to hundreds of

[1] Johnson Research, Las Cruces, NM 88005

[2] Institute for Telecommunication Sciences, National Telecommunications and Information Administration, U.S. Department of Commerce, Boulder, CO 80303-3328

thousands of link-level transmissions. The prospect of waiting several days for the results of simulating each alternative network protocol design prompted a search for a means to speed up the link-level simulation.

By profiling the execution time of the various functions comprising the simulator, it was found that just over 94 percent of the running time is spent in the HF channel simulator. Another 4 percent is spent in the modem frequency-shift keying (FSK) demodulator, with slightly under 1 percent of the simulation time spent in the data-link-layer functions (see Figure 1). If an abstract model of the HF channel and ALE modem could be developed to replace the HF channel simulator and ALE modem simulator, the time to simulate each link-level transmission could be reduced by up to two orders of magnitude. The goal of the research reported here is the development of such a model.

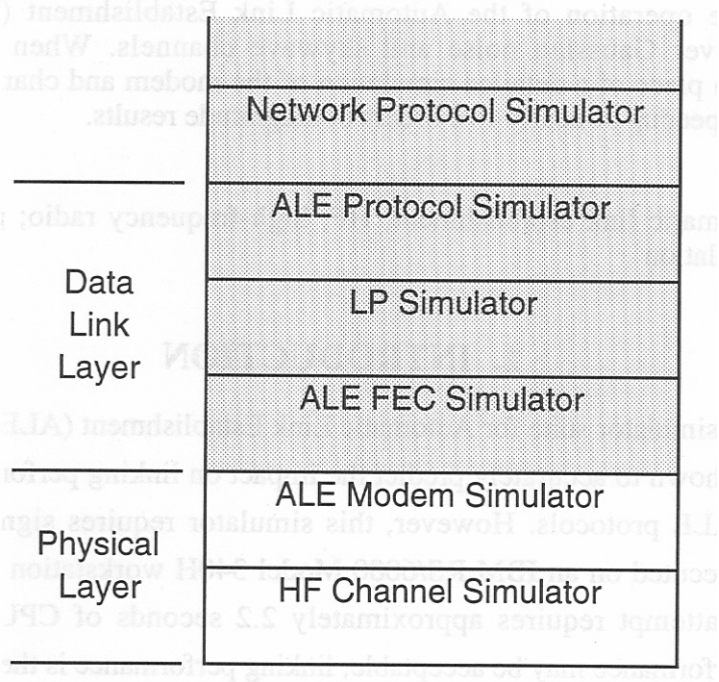

Figure 1. Simulator stack.

The requirements for the simulator module to be based upon this model are as follows:

1. Plug-compatibility with the remaining modules of the ALE simulator. The new module must accept the symbol stream produced by the transmit forward-error-correction (FEC) simulator and return a symbol stream to the receive FEC simulator.

2. The returned symbol stream must differ from the transmit symbol stream (i.e., contain errors) in the same manner as for the full simulator. That is, the arrival process for errors must be indistinguishable for the two channel/modem (physical layer) simulators, as viewed by the ALE protocol simulator.

3. The new physical layer simulator should read the channel characteristics to be simulated from the same global variables as the current modules, so that no changes are required in the data-link-layer modules when switching between the two physical layer simulators.

2. BACKGROUND

The behavior of a channel is described by the manner in which it introduces errors into the symbols that are passed through it. This behavior is typically described in terms of the distribution of bit errors; however, we are concerned with an 8-ary FSK modem, so it is more natural to work with 3-bit symbols and to discuss the distribution of bit errors within symbols separately from the occurrence of symbol errors.

2.1 Channel Characterization

The error behavior of the ALE modem working in Gaussian noise and fading (skywave) channels was obtained by modifying the full ALE simulator shown in Figure 1 to log every symbol received in error during simulated linking attempts. The resulting log files were processed after the simulation runs that generated them to extract the various channel characteristics used in this research. Each log entry listed the symbol sent, the symbol received, and the number of symbols sent since the immediately previous symbol error. This approach ensured that the sequences of symbols presented to the channel for measurement of channel performance were identical to those that would be encountered in practice.

The first investigation using these log files determined the distribution of bit errors within symbols received with errors. It was found that each of the seven possible error patterns was equally likely, independent of the transmitted symbol.

The arrival process of symbol errors can be characterized in a variety of ways. The simplest is to plot the symbol error rate versus the channel signal-to-noise ratio (SNR), as shown in Figure 2. However, this representation is insufficient to fully characterize the channels, as may be noted from the figure. The good and poor channels[3] produce essentially identical curves, despite having markedly different HF radio ALE linking performance (see Figure 3).

2.2 Binary Symmetric Channel

The simplest model of a communications channel is one in which the probability of an error is independent of the history of errors on the channel. This memoryless model is usually called the Binary Symmetric Channel (BSC). The probability of a symbol error, p, is also termed

[3] Both of these channels exhibit selective fading. The good channel is characterized by a 0.5-ms multipath delay and 0.l-Hz Doppler spread, while the poor channel simulated has a 2-ms multipath delay and 2-Hz Doppler spread. The good and poor channel parameters are similar, but not identical, to the parameters defined by the standard CCIR (now ITU-R) Good and Poor channels.

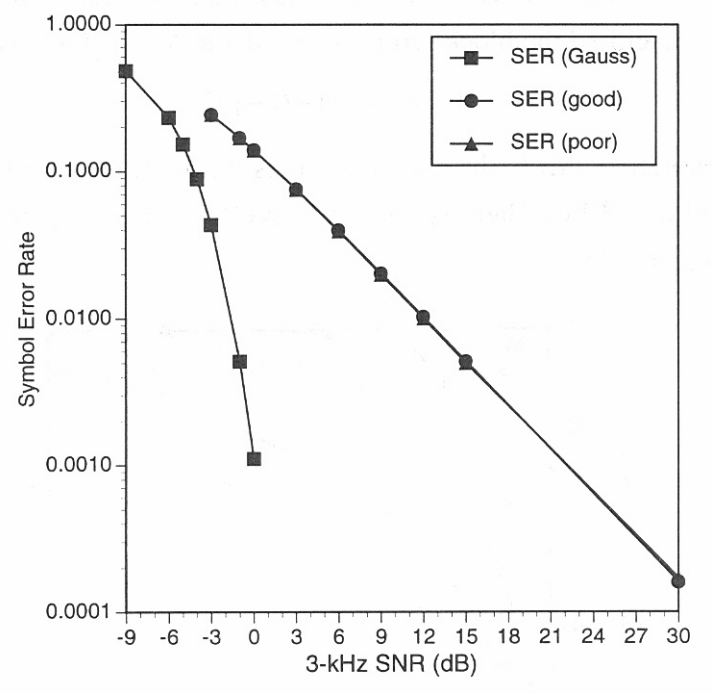

Figure 2. Channel error rate (full simulator).

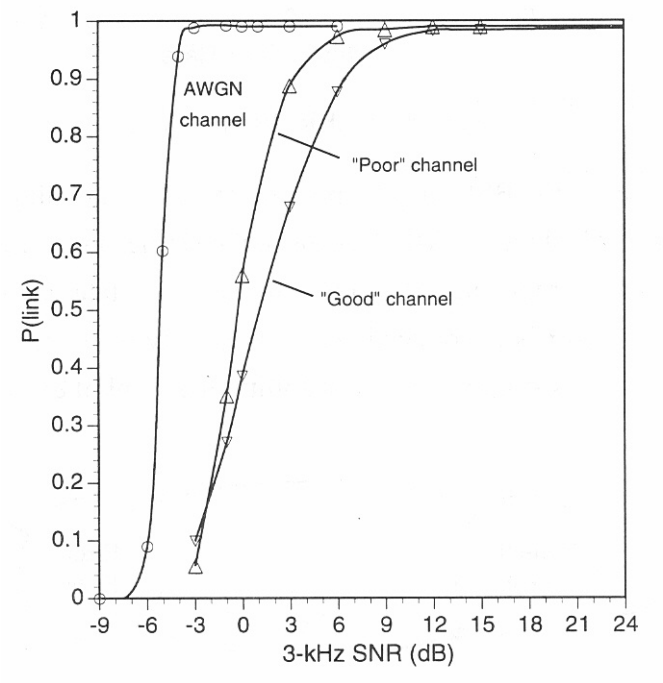

Figure 3. Linking performance (full simulator).

the symbol error rate, or SER. Because of its memoryless behavior, the probability of one or more errors in a block of N consecutive symbols ($N = 49$ is used in this report to simulate the use of the ALE waveform word), termed the block error rate or BLER(N), is given by

$$\text{BLER}(N) = 1-(1-p)^N.$$

The BLERs computed from the simulation logs for our three example channels are shown in Figure 4. The Additive White Gaussian Noise (AWGN) channel fits the BSC model well, but the fading channels clearly do not.

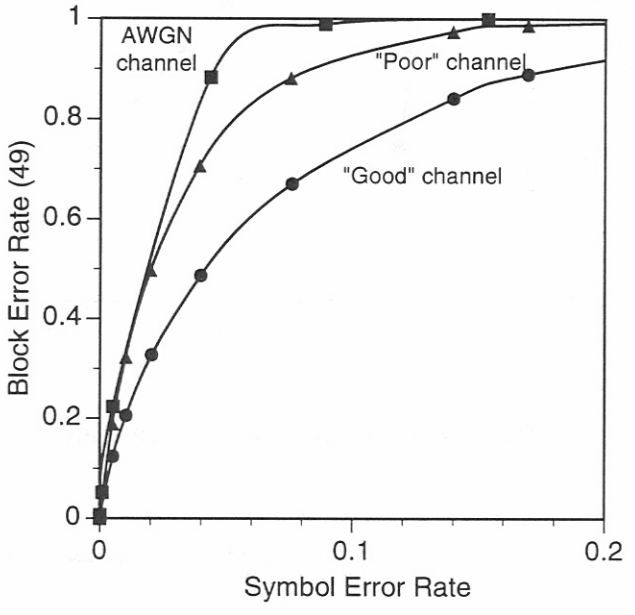

Figure 4. BLER(49) vs. SER.

The error behavior of the BSC may be represented as a state diagram, as seen in Figure 5. In this Moore Finite State Machine (FSM) [3, p. 281], the single state variable determines whether or not to introduce an error for the current symbol: in the error state, an error is introduced, while in the non-error state, no error is introduced. The probability of making a state transition to the error state (from either state) is simply p, the probability of a symbol error.

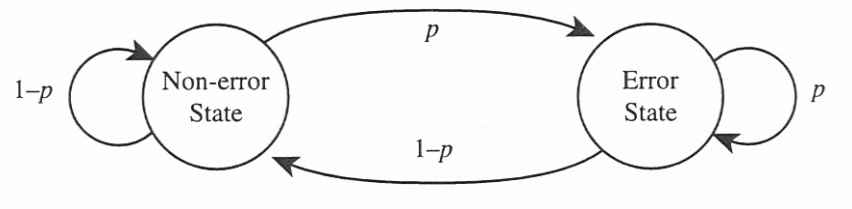

Figure 5. BSC state diagram.

2.3 Single-Error-State Models

The BSC model can be generalized to a multi-state Markov model with a single error state as described in [4]. The key feature of such a single-error-state (SES) model is that transitions are permitted only between the error state and one of the non-error states (Figure 6); no transitions are permitted among the non-error states. This constraint has the effect of introducing N_s–l different geometric distributions for error-free runs between each pair of error bursts, where N_s is the number of states in the model.

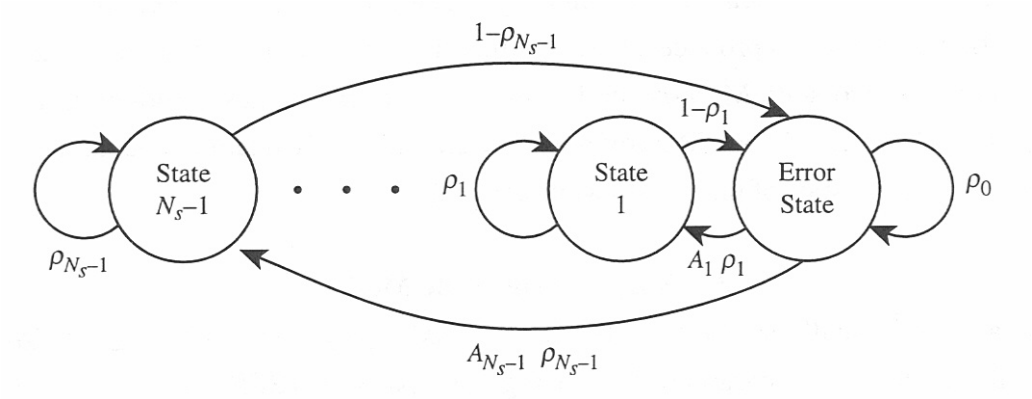

Figure 6. SES model state diagram.

A procedure is described in [4] by which pairs of SER and BLER measurements can be converted into the A_i and ρ_i parameters of an SES model. The number of non-error states resulting from this procedure will equal the number of pairs of measurements used.

For use in a simulator, this initial model must be modified to generate precisely the SER desired. This is achieved by removing any states that lead to an SER lower than desired, and then adding one state with its A_i and ρ_i parameters calculated to bring the SER exactly to the desired value.

3. SKYWAVE MODEL

Although the BSC was found to accurately model the AWGN channel, it does not accurately model the skywave channel. In this section, the design of a model for the fading (skywave) channels is presented.

3.1 Fixed-State Markov Models

Because of the complications of handling varying numbers of states as the SER value changes, we first attempted to produce a model with a fixed number of states: either one non-error state (resulting in a two-state Markov model) or two non-error states (producing a three-state Markov model). These fixed-state Markov models were ultimately unable to accurately represent the error burst characteristics of the HF skywave channel.

3.2 Single-Error-State Model

The approach finally selected for the HF SChEMe (Skywave Channel Error Model) is a version of the SES model, parameterized using four pairs of (SER, BLER) measurements. Corresponding measurements were collected from channel error logs of four fading channels, and are listed in Table 1. The SER values for the four channels did not differ significantly, so the mean values of the four SER measurements are used for all fading channels.

By interpolating among the BLER values listed in Table 1, estimates of BLER(49) can be obtained for any channel conditions within the bounds of the measured channels. (However, extrapolation beyond these bounds may be unreliable.)

Table 1. Channel Error Measurements

Multipath delay (ms)		0.5000	2.0000	0.5000	2.0000
Doppler Spread (Hz)		0.1000	0.1000	2.0000	2.0000
SNR (dB)	SER	BLER (49)			
–3	0.24323	0.9572	0.9010	0.9995	0.9978
–1	0.16943	0.8895	0.7784	0.9947	0.9875
0	0.13968	0.8415	0.7015	0.9880	0.9746
+3	0.07565	0.6707	0.4694	0.9251	0.8816

4. PERFORMANCE MEASUREMENT

The performance of the simulator developed from this model can be gauged in two ways: its speedup relative to the full simulation and the fidelity of its results to those obtained using the full simulator.

Addressing speedup first, the time per simulated linking attempt for the full simulator is 2.2 seconds on an IBM RS/6000 Model 340H, using SChEMe in place of the channel and modem simulators. Each linking attempt is simulated in 0.024 second. This represents a speedup factor of 92. Both measurements are averaged over 1000 linking attempts using a poor channel with 0-dB SNR. Similar results are obtained for other channels.

The fidelity of the SChEMe simulator is shown in Figures 7 through 9, which show simulated linking probabilities for the full simulator, the SChEMe simulator, and the two fixed-state Markov simulators. Clearly, the fidelity of the HF SChEMe is excellent for the Gaussian and poor channels, and is also quite good for the good channel.

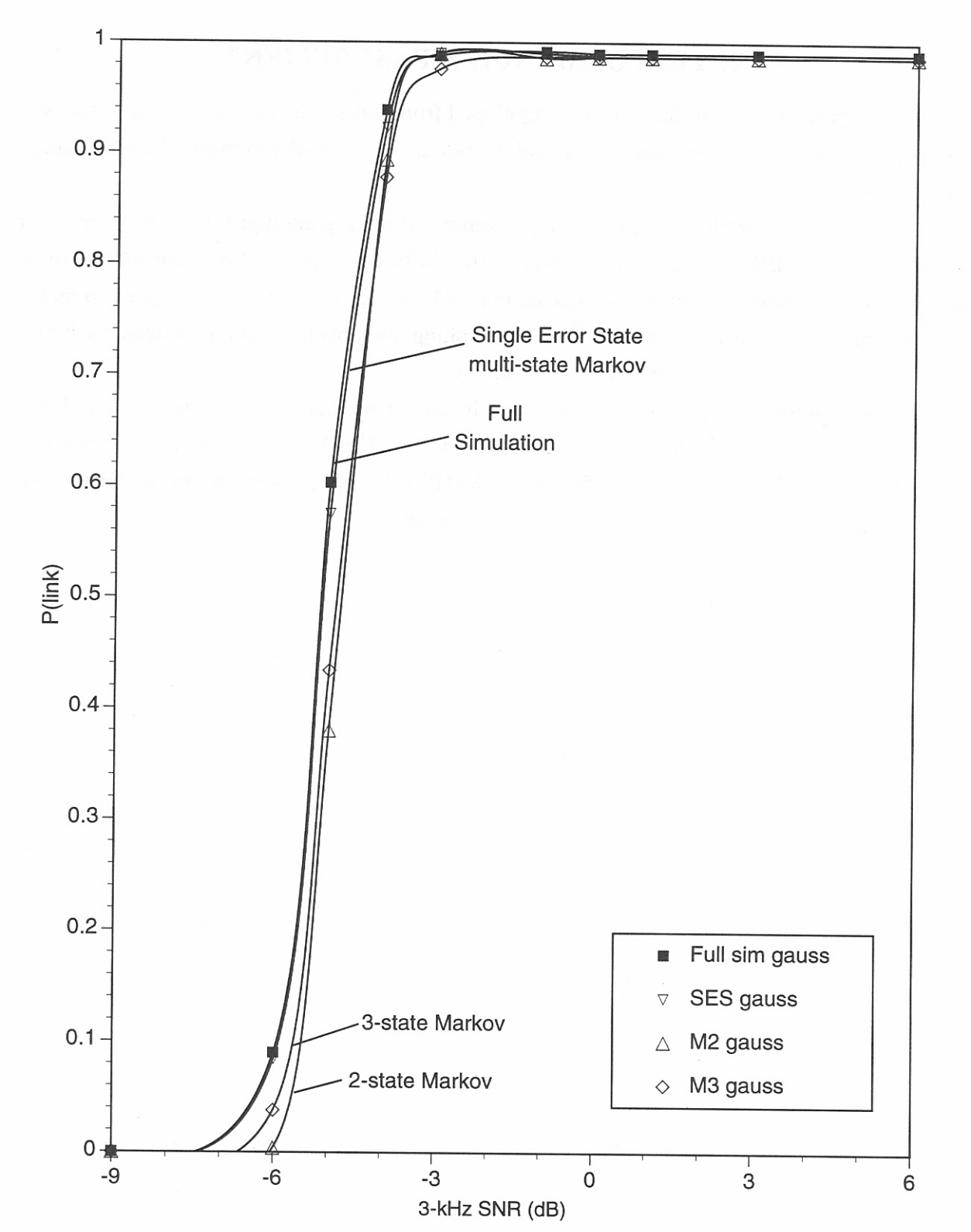

Figure 7. A comparison of simulated linking possibilities for the Gaussian channel.

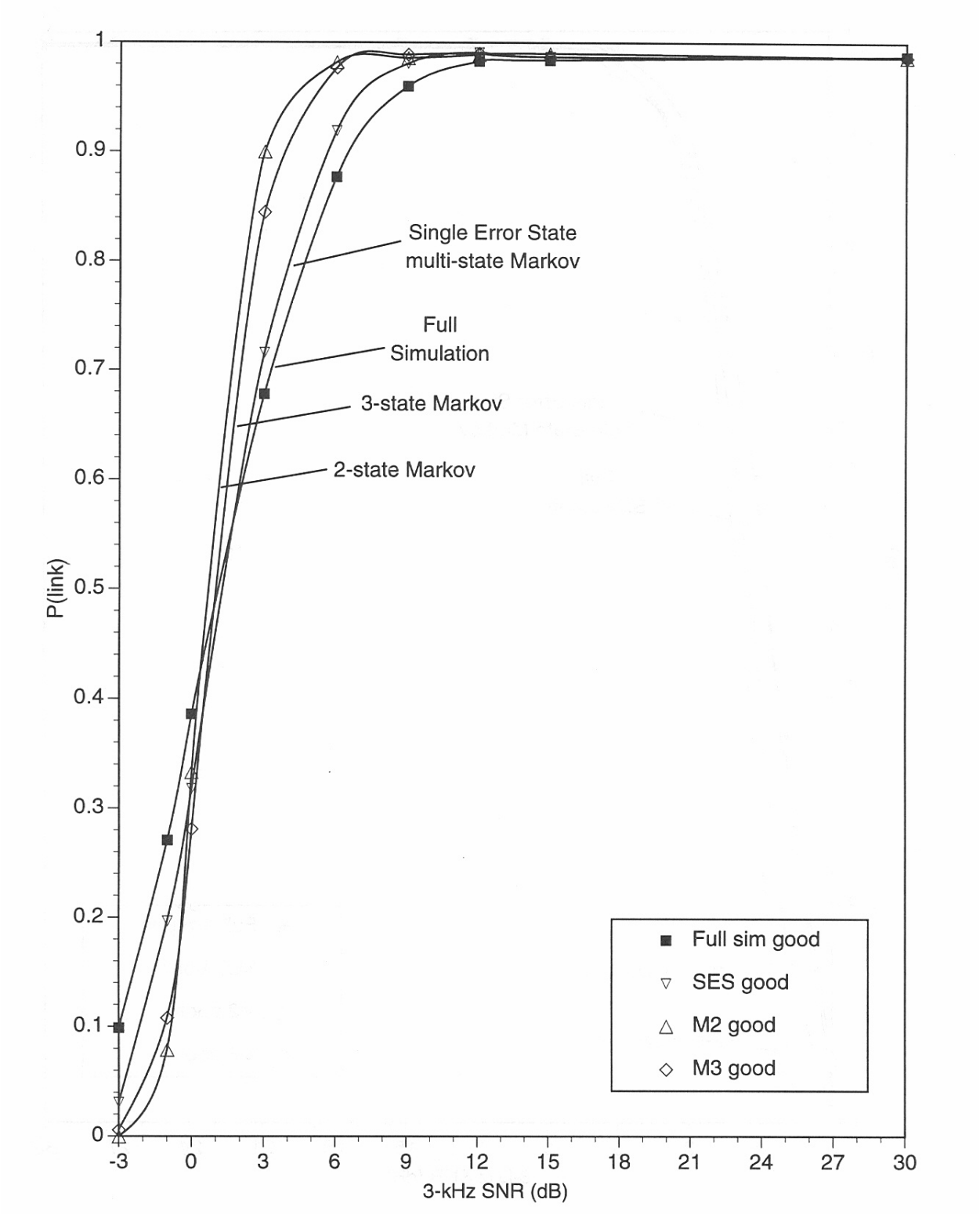

Figure 8. A comparison of simulated linking possibilities for the good channel.

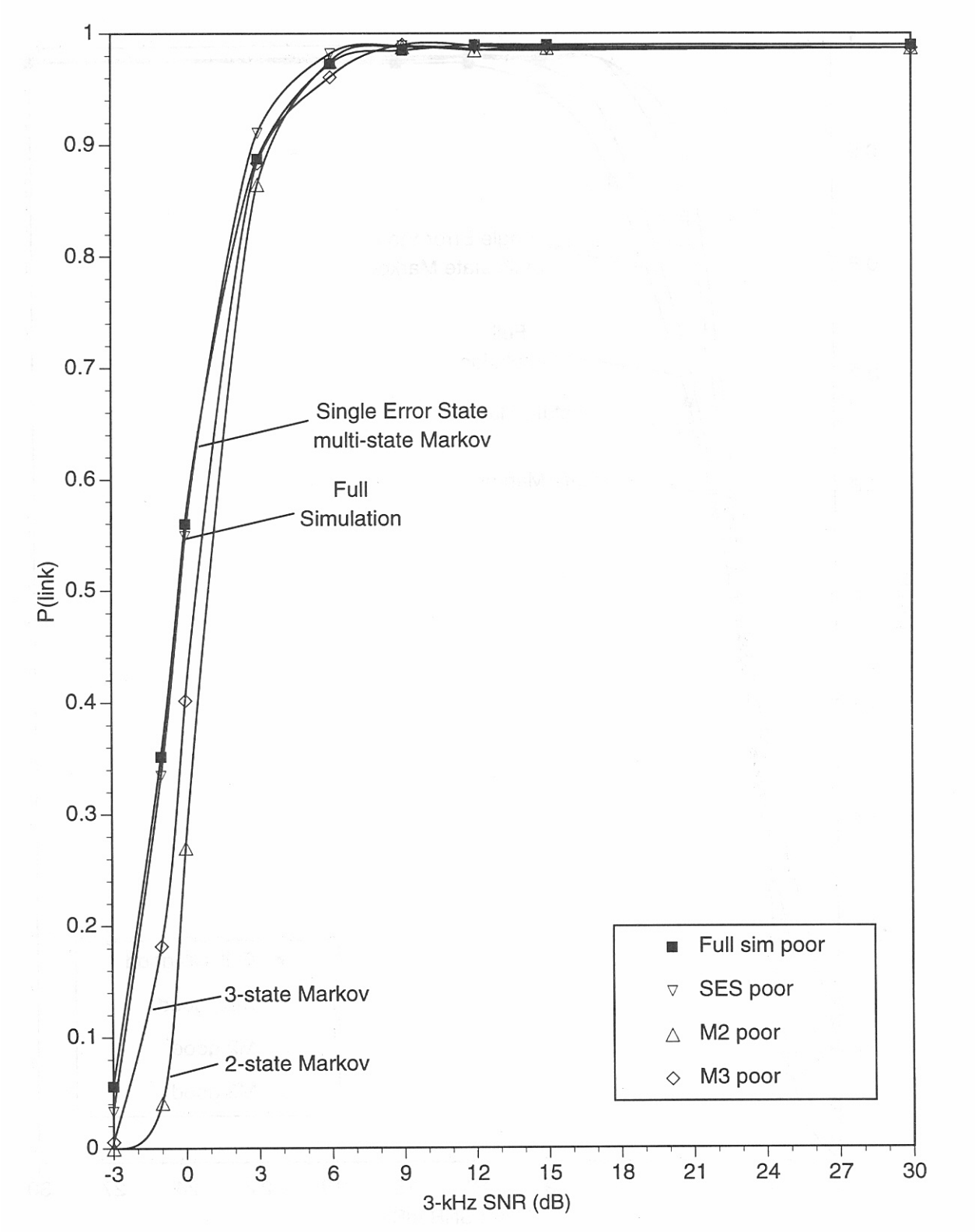

Figure 9. A comparison of simulated linking possibilities for the poor channel.

5. CONCLUSION

The use of a single-error-state model of HF skywave channel performance in simulations of data-link-layer (and higher) protocols should yield approximately two orders of magnitude improvement in simulation speed, while introducing little error into performance statistics collected at levels of abstraction above the physical layer.

6. FUTURE WORK

A research project is planned at New Mexico State University that will use SChEMe in HF network simulations run on parallel computers to determine the magnitude of the speedup on such a system. We expect that this system will be able to provide timely simulations of much more complicated systems than can be practically simulated using current technology.

7. REFERENCES

1. E.E. Johnson and D.F. Peach, "High-Frequency Radio Channel and Modem Simulator," NTIA Report 94-308, 1994.

2. E.E. Johnson and D.F. Peach, "A Software Simulator for High-Frequency Automatic Link Establishment," NTIA Technical Memorandum 94-162, 1994.

3. Z. Kohavi, *Switching and Finite Automata Theory,* New York: McGraw Hill Book Co., 1970

4. L.E. Vogler, "An Extended Single-Error-State Model for Bit Error Statistics," NTIA Report 86-195, July 1986.

APPENDIX: PROGRAM LISTINGS

(Only new or revised code is included.)

A.1 SChEMe Simulator

```
/*  SChEMe.c              eej          7/17/92              */
/*                              revised 9/11/93              */
/*                                                           */
/*  generates symbol errors using single-error-state model  */

#include <stdio.h>
#include <math.h>
#include "hfdef.h"
#include "uniformN.c"

/* RN streams:  0  RandInt                  */
/*              1  select error-free state  */
/*              2  generate error-free run  */
/*              3  generate error burst     */

#define SELECT_STATE 1
#define ERROR_FREE   2
#define ERROR_BURST  3

#define min(x, y) (x < y ? x : y)


/*************   PARAMETERS FOR SINGLE-ERROR-STATE MODEL   **************/

#define NSMAX 5

static double rho[NSMAX+1], /* probability of staying in state   */

              A[NSMAX+1],   /* scale factor for transitions from */
                            /*  error state to non-error state   */

              p[NSMAX+1]    /* cumul. error prob. given i states */
                  = {1.0,0,0,0,0,0},

              t[NSMAX+1];   /* cumulative transition prob. distn */


/************************   STATIC VARIABLES   ************************/

static double SNR;          /* 3 KHz signal-to-noise ratio */

static int Ns;              /* number of error-free states */
                            /* actually used in model      */


/********************   INITIALIZATION FUNCTIONS   ********************/

double GaussianSNR2SER(SigLev)  /* computes Symbol Error Rate from SigLev */
double SigLev;                   /*  (from curve fit to simulator data) */
{
  double lnSER;

  lnSER = (((((0.2196943 * SigLev) - 9.349242) * SigLev) +
            2.927393) * SigLev) - 0.6029639;
  return exp(lnSER);
}

double FadingSNR2SER(SNR)     /* computes Symbol Error Rate from SNR */
double SNR;                   /*  (from curve fit to simulator data) */
{
  double logSER;
```

```
  logSER = (((((1.821874e-5 * SNR) - 9.635476e-4) * SNR) - 8.542218e-2) * SNR)
           - 8.577524e-1;
  return pow(10.0, logSER);
}

#define NOMDELAY 0.5
#define NOMSPREAD 0.1

static double NomSNR[NSMAX]       = {0,     -3.0,     -1.0,      0.0,      3.0},
              NomBLER[NSMAX]      = {0,   0.9572,   0.8895,   0.8415,   0.6707},
              DeltaDelay[NSMAX]   = {0, -0.0375, -0.0741, -0.0933, -0.1342},
              DeltaSpread[NSMAX]  = {0,   0.0223,   0.0553,   0.0771,   0.1339},
              DeltaBoth[NSMAX]    = {0,   0.0191,   0.0365,   0.0444,   0.0554};

#define BLOCKSIZE 49

void GenErrorTable()
{
  double BLER, SER, Sb = 0.0, Sp = 0.0, Rx, Bf, Pf;
  int i;
  for (i=1; i<NSMAX; i++) {
    SER = FadingSNR2SER( NomSNR[i] );
    BLER = NomBLER[i] + DeltaDelay[i] * (DlyTime - NOMDELAY) +
              DeltaSpread[i] * (FreqSpread - NOMSPREAD) +
           DeltaBoth[i] * (FreqSpread - NOMSPREAD) * (DlyTime - NOMDELAY);
    Pf = (1.0 - SER) / SER;
    Bf = (1.0 - BLER) / SER;
    Rx = (Bf - Sb) / (Pf - Sp);
    rho[i] = exp( log(Rx) / (BLOCKSIZE-1) );
    A[i] = (Pf - Sp) * (1.0 - rho[i]) / rho[i];
    Sb =+ A[i] * exp( BLOCKSIZE*log(rho[i]) ) / (1.0 - rho[i]);
    Sp += A[i] * rho[i] / (1.0 - rho[i]);
  }
  rho[NSMAX] = A[NSMAX] = 1.0;
}
```

```
void InitModem() {
  int i;
  double Sp, SER, C, C1, temp, rhohat, Ahat, Gdns;

  SNR = 20.0*log10(SigLev);  /* SigLev is global (ale.c) */
  if ((DlyTime == 0.0) && (FreqSpread == 0.0)) {  /* AWGN channel */
    SER = GaussianSNR2SER(SigLev);
    rho[0] = SER;     /* error state */
    rho[1] = 1.0 - rho[0];
    A[1] = 1.0;
    t[1] = A[1] * rho[1];
    Ns = 1;
  }
  else {  /* fading channel */
    SER = FadingSNR2SER(SNR);
    GenErrorTable();
    Sp = 0.0;
    t[0] = A[0] * rho[0];
    Gdns = 0.0;
    Ns = NSMAX;
    for (i=1; i<Ns; i++) {
      Sp += (temp = A[i] * rho[i] / (1.0 - rho[i]));
      Gdns += A[i] * rho[i];
      p[i] = 1.0 / (1.0 + Sp);
      t[i] = t[i-1] + A[i] * rho[i];
      if (SER >= p[i]) Ns = i;
    }
    if (Ns < NSMAX) {    /* SER falls within array */
      Sp -= temp;
      C = log(SER/p[Ns-1]) / log(p[Ns]/p[Ns-1]);
    }
    else C = 1.0;
    C1 = ((1.0 - SER)/SER - Sp) * (rho[Ns] - rho[Ns-1]) / (C * A[Ns]);
    rhohat = (rho[Ns-1] + C1) / (1.0 + C1);
    Ahat = C * A[Ns] * (rhohat - rho[Ns-1]) / (rhohat*(rho[Ns] - rho[Ns-1]));

    rho[Ns] = rhohat;
    if (Ns < NSMAX) {
      A[Ns] = Ahat;
    }
    else {
      A[Ns] = ((1.0 - rho[Ns])/rho[Ns]) * ((1.0/SER) - (1.0 + Sp));
      while ((t[Ns] = t[Ns-1] + A[Ns] * rho[Ns]) > 1.0) {
      rho[Ns] = sqrt(rho[Ns]);
      A[Ns] = ((1.0 - rho[Ns])/rho[Ns]) * ((1.0/SER) - (1.0 + Sp));
      }
    }
    Sp += A[Ns] * rho[Ns] / (1.0 - rho[Ns]);
    p[Ns] = 1.0 / (1.0 + Sp);
    printf("p = %lf\n", p[Ns]);

    t[Ns] = t[Ns-1] + A[Ns] * rho[Ns];
    for (i=1; i<=Ns; i++) {
      printf("t[%d] = %lf\n", i, t[i]);
    }
    rho[0] = 1.0 - t[Ns];
  }
}
```

```
/*****************   ERROR GENERATING FUNCTIONS   *****************/

long geometricIAT(i, Pfail)
int i;
double Pfail;
{
    return (long)((log(1.0 - uniform(i))/log(Pfail)) + 1.0);
}


#define ErrorPattern ((int)((7.0 * uniform(3)) + 1.0))

static long ErrorFreeRun()       /* # good symbols between errors */
{
  double temp;
  int i;

  while ((temp = uniform(SELECT_STATE)) > t[Ns])
    ;
  for (i=1; i<Ns; i++)     /* find the non-error state selected by temp */
    if (temp < t[i])
      return geometricIAT(ERROR_FREE, rho[i]);

  return geometricIAT(ERROR_FREE, rho[Ns]);
}


static long ErrorBurst()         /* # error symbols in burst */
{
    return geometricIAT(ERROR_BURST, rho[0]);
}


/*******************   SIMULATED MODEM FUNCTIONS   *******************/

int ALEmodem(InSym)
int InSym;
{
    static long count = 2;          /* # symbols in current run */
    static int errors = 0,          /* 0 => error free run; else error burst */
               OutSym;              /* return value */

    if (errors) OutSym = InSym ^ ErrorPattern;
    else        OutSym = InSym;

    if (!(--count)) {
      if (errors ^= 1) count = ErrorBurst();
      else             count = ErrorFreeRun();
    }

    return OutSym;
}
```

A.2 uniformN

```
/*  uniformN.h    eej    7/6/92  */
/*                               */
/*  starting points for multiple */
/*  uniform RN generators        */

#define LIMIT 16

long s1[LIMIT] = { 123456787, 1230880130,  322352050, 1591950360,
                   1761280518,  216186585, 1527679572, 1718551313,
                   1179131321,  763160215, 1183010142,  423519796,
                     22966675,  633104697,  699250793, 1652958743},

     s2[LIMIT] = { 987654323,  217139478,  680034137,  902024935,
                   1925217876,  640144839, 1463505957, 1971215030,
                    356297119, 1426751771, 1710812483,  943733665,
                   1564591690,   85346316,  735968472,  669421119};
```

```
/*  uniformN.c       eej         7/6/92                  */
/*  set of N independent random number generators with  */
/*  uniform distribution; N determined by header file.  */
/*  derived from single generator found                 */
/*  in CACM June 1988, Vol 31, No 6, pp. 742-749+       */

#include "uniformN.h"    /* contains LIMIT and seeds */

double uniform(i)
int i;
{
    register long r1, r2, z, k;

    if (i>=LIMIT) {
      printf("\n\nIllegal random stream %d\n", i);
      exit(-1);
    }

    k = (r1 = s1[i]) / 53668;
    r1 = 40014 * (r1 - k * 53668) - (k * 12211);
    if (r1 < 0) r1 += 2147483563;
    s1[i] = r1;

    k = (r2 = s2[i]) / 52774;
    r2 = 40692 * (r2 - k * 52774) - (k * 3791);
    if (r2 < 0) r2 += 2147483399;
    s2[i] = r2;

    z = r1 - r2;
    if (z < 1) z += 2147483562;

    return (z * 4.656613E-10);
}
```

A.3 Header Files

```
/*  localdef.h for SunOS     eej     8/12/91  */
/*       AIX, HP-UX, IRIX, unicos    9/18/93  */

#define shareCPU() ;

#include <time.h>
#ifdef CLK_TCK
#define HZ CLK_TCK
#else
#define HZ 60
#endif

#include <sys/types.h>
#include <sys/times.h>

#define TIME(x) {struct tms t; times(&t); x = t.tms_utime;}
```

```
/*  hfdef.h      definitions for HF channel simulator routines    */
/*  eej          8/3/91                                          */

#include "localdef.h"

#define Pi  3.141592654
#define TPi 6.283185307

#define Ts  0.008
#define NPTS 32

typedef double  SymVect[2][NPTS];

extern double DlyTime, FreqSpread, SigLev, TxLevel;  /* declared in hflib.c
*/
```

```
/*  hflib.h      header file for HF propagation simulator routines  */
/*                                    eej      7/30/91             */
/*               version for SChEMe   eej      9/12/93             */


/* hfdef.h must be #included before this file */

extern void Initialize();
extern double uniform();

#define RandInt(i)    ((int) (i * uniform(0)))
                /* Generates random integers in the range 0 to i-1.     */
                /* In run of 100,000 with 32 bins, had variation of     */
                /* 6% of mean from max bin to min bin.                  */
```

```
/*  ALEmodem.h          eej       6/16/91          */
/*          8-ary FSK modem for HF ALE simulator */
/*                                                */
/*          version for SChEMe  eej  9/12/93  */

extern void     InitModem();
extern int      ALEmodem();
```

FORM **NTIA-29**
(4-80)

U.S. DEPARTMENT OF COMMERCE
NAT'L. TELECOMMUNICATIONS AND INFORMATION ADMINISTRATION

BIBLIOGRAPHIC DATA SHEET

	1. PUBLICATION NO.	2. Gov't Accession No.	3. Recipient's Accession No.
	NTIA Report 94-310		

4. TITLE AND SUBTITLE

High-Frequency Radio Channel Error Model

5. Publication Date

6. Performing Organization Code

7. AUTHOR(S)

Eric E. Johnson and David F. Peach

9. Project/Task/Work Unit No.

8. PERFORMING ORGANIZATION NAME AND ADDRESS

National Telecommunications and Information Admin.
Institute for Telecommunication Sciences
325 Broadway
Boulder, CO 80303

10. Contract/Grant No.

11. Sponsoring Organization Name and Address

National Communications System
Attn: NT
701 South Court House Road
Arlington, VA 22204

12. Type of Report and Period Covered

13.

14. SUPPLEMENTARY NOTES

15. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

Simulation has been extensively employed to evaluate concepts included in the current generation of standards for automated high-frequency (HF) radio systems. As development proceeds from link-layer technology to network- and higher-layer technology, it is no longer necessary to devote great amounts of computer time to detailed simulation of the physical medium. Instead, the error behavior of the medium should be abstracted so that simulation resources can be concentrated on the phenomena of interest at these higher protocol layers.

In this report, a model of channel error behavior is derived that accurately captures the operation of the Automatic Link Establishment (ALE) modem operating over Gaussian noise and skywave channels. When this model is employed in place of a detailed simulation of the modem and channel, an overall simulation speedup of nearly two orders of magnitude results.

16. Key Words *(Alphabetical order, separated by semicolons)*

automatic link establishment; HF; high-frequency radio; performance analysis; simulation

17. AVAILABILITY STATEMENT

☒ UNLIMITED.

☐ FOR OFFICIAL DISTRIBUTION.

18. Security Class. *(This report)*

Unclassified

20. Number of pages

24

19. Security Class. *(This page)*

Unclassified

21. Price:

*U.S. GOVERNMENT PRINTING OFFICE:1994-573-013/00036