## NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

## AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

# Coding and Multiple Access Protocols for Land-Mobile Satellite Data Networks

by

Salim Fakhouri, B.A.Sc.

A thesis submitted to the
School of Graduate Studies and Research
in partial fulfilment of the requirements
for the degree

Master of Applied Science

Ottawa-Carleton Institute for Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

January 1991

ISBN 0-315-68072-5

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

## Abstract

*In land mobile satellite communications, the mobiles share the same channel for the transmission of their messages towards the satellite. Therefore a multiple access scheme that regulates the access to the channel must be employed. In this thesis, we compare the performance of four different access techniques for the case of land mobile data networks. A hybrid ARQ scheme based on Reed-Solomon codes is used to combat the error bursts resulting from multipath fading and shadowing. The influence on the efficiency of the system of parameters such as code rate, block length, guard time and error rate on the acknowledgement channel is evaluated. The advantages of using symbol interleaving and erasure decoding are also discussed.*

## ACKNOWLEDGEMENTS

I wish to especially thanks Dr. A. Yongacoglu for his valuable support and guidance throughout my research. I also want to acknowledge the help of NSERC, TRIO and OGS for their support through various grants and scholarships.

# Contents

# List of Figures

# Chapter 1

# INTRODUCTION

In this age of information technology, it is becoming essential to communicate information to people who are on the move and may be difficult to locate precisely. In the vicinity of cities, cellular systems can be used for this purpose. However, there is an urgent need to provide global communication networks serving not only urban areas but remote and rural areas as well. Because of their wide coverage and their multiple access capability, satellites are very well suited to fulfill this demand.

In recent years, extensive research has been made in the field of land mobile satellite communication and several systems are expected to become operational in the near future around the world (especially in the United States, Canada and Australia) [5] [6] [7] [8].

An LMSAT network consists of a large fleet of mobiles, a communication satellite and one or many fixed earth stations. The transmission of the information from the mobiles towards the fixed base station takes place on a channel called the return channel. Since this channel is shared by all the users, a multiple access protocol regulating the access to the channel is necessary. The forward channel (from base station to mobiles) operates in a broadcast mode carrying the reverse traffic and control information. Generally, because of the low gain of the antennas mounted on the mobiles, a mobile to mobile communication requires the information to be relayed

by the central base station. The most important factors that influence the choice of the access scheme are:

- The high propagation delay (one hop takes about 0.27 seconds).
- The large number of users.
- The type of traffic.
- The star topology of the network.
- The strong level of noise in the channels.

These factors clearly prevent the use of techniques such as carrier sensing or polling. Possible schemes include random access protocols which require little co-ordination between the users and demand access protocols that rely on a central controller. The robustness of the protocol is also important because of the high level of noise and other impairments in the channels.

The main reasons for the degradation on the links are the limited power available at the mobiles, the low gains of the antennas and the effect of shadowing and multipath fading. Shadowing occurs when the direct view of the satellite is obstructed by trees, bridges, buildings... Multipath fading is a consequence of interferences due to multiple reflections from various objects. As we shall show in this thesis, if the effects of shadowing and multipath fading are not judiciously mitigated, efficient and reliable communication cannot be possible in such channels. The system efficiency can be significantly improved by employing forward error correction (FEC) coding. Reed-Solomon and convolutional codes (or a concatenation of the two) are among the strongest candidates for this purpose.

## 1.1 Motivations

Because of the limited frequency space allocated to land mobile satellite communications, a great deal of research is being performed on the ways of maximizing bandwidth utilization. Error control coding and multiple access are two problems that have direct implications on the overall efficiency of an LMSAT system. When reviewing the literature, one can make the following remarks.

- In papers discussing error control, a mathematical model for the channel is generally adopted and a coding scheme suggested. The resulting bit error rate (or symbol error rate) is then obtained analytically or by computer simulations [18] [19] [20].
- In studies concerning multiple access, the procedure, in most cases, is to describe the suggested protocol and obtain its throughput-delay performance assuming error free channels or a given packet error rate [42] [44] [41] [45].

For the case of land-mobile satellite data networks, such a separation in the study of error control and multiple access is not adequate because of the following reasons.

- The coding rate of the FEC code has a direct influence on the performance of the multiple access scheme since it affects the size of the message to be transmitted and the overall packet error rate resulting from the noise in the channel and the possible collisions due to multiple access.
- Unlike additive white guaussian noise channels (AWGN), LMSAT channels induce error bursts and, therefore, cannot be considered memoryless. Hence, long time average bit error rates are not sufficient for the evaluation of the performance of a packet switching system.
- In the case of data networks, the message length puts a restriction on the maximum size of an eventual interleaver used to spread the errors. Therefore,

the frequent assumption [19] [18] suggesting that the errors in a packet can be randomized using a big enough intereleaver may not be valid.

The above arguments have convinced us that it is necessary to consider the influence of coding on the performance of multiple access techniques. On the other hand, we find that it is difficult to compare the efficiency of the access protocols proposed in the literature since different parameters and assumptions are done in the papers suggesting them. In this thesis, we study the following:

- The combined effect of coding and multiple access on the performance of land mobile satellite systems (Answer of questions like : What is better, to have a code rate of 1/2 and packet sizes of 2L or a code rate of 3/4 and packet sizes of 4L/3 ?).
- The comparison of different multiple access schemes under the same conditions and the same parameters.
- The advantage of using symbol interleaving, erasure decoding, and channel state information.

## 1.2 Plan of the thesis

We start the discussion by describing, in Chapter 2, the topology of the network and the channel model adopted for the computer simulations.

The problem of coding is then discussed in Chapter 3. In this chapter, we first describe the Reed-Solomon based hybrid ARQ scheme suggested for the achievement of a good throughput and the high degree of reliability necessary in data communications. The advantage of using a symbol interleaver in order to allow the use of short codes is then demonstrated. Finally, we evaluate the improvement in the efficiency of the coding scheme that can be achieved by the use of erasure decoding.

In Chapter 4, after a brief overview of the different categories of multiple access protocols, we describe and evaluate separately the performance of four schemes. These schemes are: slotted ALOHA, frequency hopping, adaptive mobile access protocol (AMAP) and a scheme that combines AMAP and frequency hopping. The delay-throughput performance of each protocol is obtained first in the case of an error free channel and then in the land-mobile satellite channel model. The effect of parameters such as the guard time, the coding rate and the error rate on the acknowledgement channel are also studied. We end the chapter by a performance comparison of the four schemes. The conclusions drawn from this research are summarized in Chapter5, and some suggestions for further research are given. The thesis contains two appendices. In Appendix A, an analytical derivation evaluating the efficiency of an uncoded system is given. Appendix B contains the listings of the various simulation programs that were written.

# Chapter 2

# SYSTEM MODEL

## 2.1 Introduction

In this chapter, we define the system model that will be used throughout this thesis. We first begin by describing the topology of the network and its components. Then, we discuss the land-mobile satellite channel and define the channel model that will be adopted for the computer simulations.

## 2.2 Network Topology

The network that is assumed here consists of a large number of small mobile stations, a communication satellite and a large earth station called the hub. The network is illustrated in Figure 2.1. In this thesis, we are concerned with the transmission of data from the mobiles to a common destination which is assumed to be co-located with the hub or connected to it through a fast and reliable terrestrial link. The data packets are transmitted through a channel that is shared by all the mobiles and that is called the return channel. The rules that regulate the access to this return channel is defined by the multiple access protocol which will be discussed in detail in Chapter 4. The acknowledgements sent by the hub to confirm the correct reception of the

packets are transmitted through the forward channel (from the hub to the mobiles). All mobile stations listen to this broadcast channel which also carries the opposite data traffic along with some control information necessary for the correct functioning of the network.

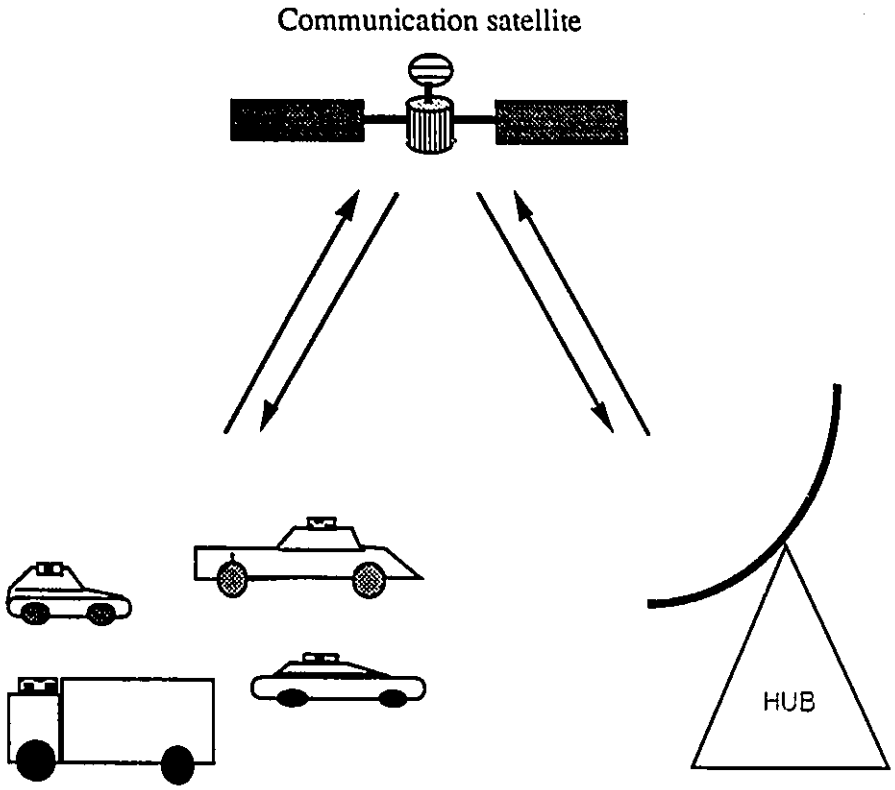


Figure 2.1: The topology of the land-mobile satellite network.

## 2.3 The Land-Mobile Satellite Channel

Propagation characteristics of land-mobile satellite links play an important role in the design of such systems. The main problem arises due to shadowing and multipath effects caused by the nature of the environment in the vicinity of the mobile. The shadowing is caused by obstacles such as bridges, foliage and buildings which obstruct the clear view of the satellite and thus cause a deep attenuation of the direct path signal (also called the line-of-sight signal). Figure 2.2 illustrates this phenomenon. The multipath effect is caused by the interference at the receiver of the reflections of the satellite signal at a large number of points such as hills, buildings, water surfaces etc... This effect is illustrated in Figure 2.3.

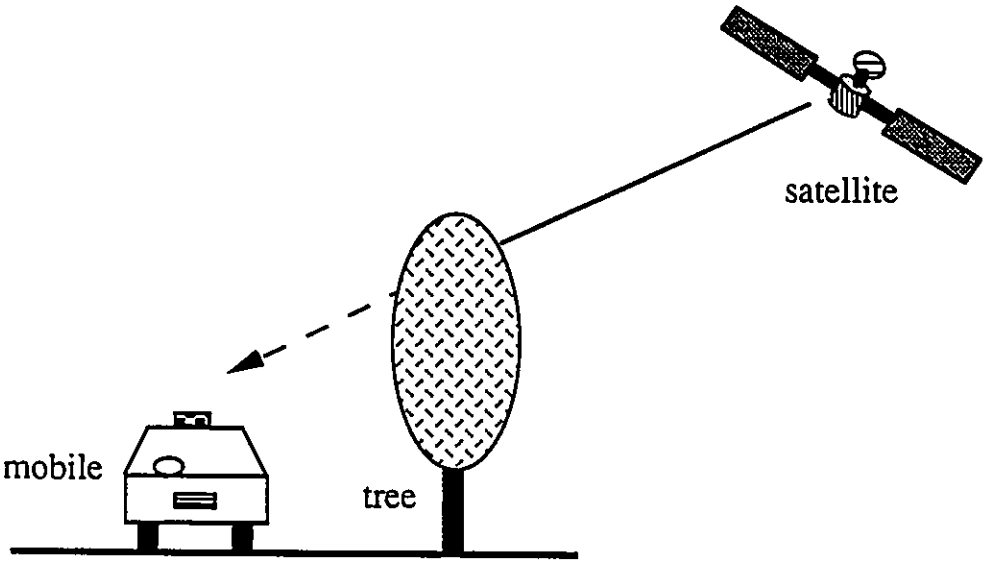


Figure 2.2: Shadowing in land-mobile satellite communication.

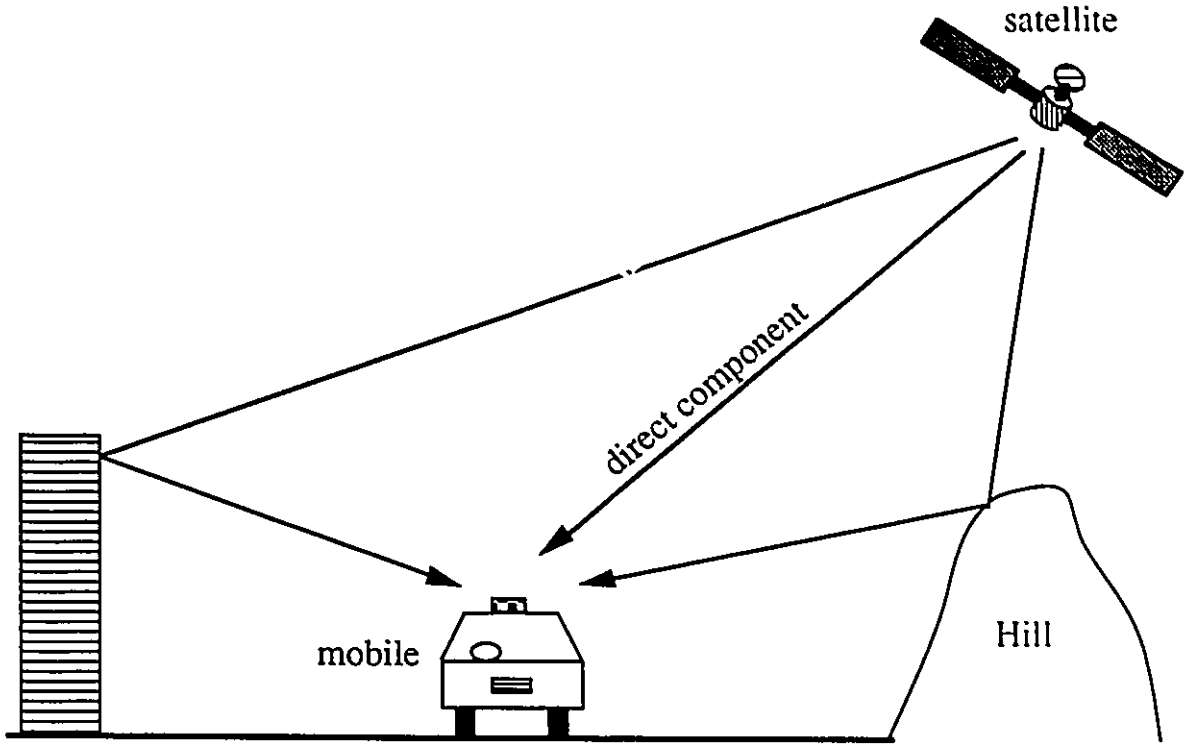


Figure 2.3: Multipath fading in land-mobile satellite communication.

### 2.3.1 Land-Mobile Satellite Channel Models

Besides the method of using stored data from real experimental measurements [20], many mathematical models have been suggested for the characterization and the analysis of the land-mobile satellite channel. The most common models used are:

- The Rician model: In this model, the received signal is the sum of a direct component and a fading component due to reflections. The direct component is only attenuated due to free space loss and the doppler shift due to the movement

of the mobile. The multipath component is Rayleigh distributed in amplitude and uniformly distributed in phase. The sum of the two signals result in a received envelope which has a Rician distribution [10] [11].

- Shadowed Rician fading model: The previous model did not consider the shadowing effect which can be predominant in many cases. In this model, the line-of-sight component is subjected to a log-normal transformation to simulate the shadowing phenomenon. The multipath signal is Rayleigh distributed as in the Rician model. The mathematical derivations required to describe the model can be found in [12] [13] [14].

- Other models: Some other models emphasize on the importance of shadowing and distinguish between intervals in which the direct satellite signal is shadowed and those without any shadowing [18] [15]. In [18] for example, the received power is considered to have a Rician density in non-shadowing intervals and Rayleigh-lognormal density if shadowing is present.

### 2.3.2 Adopted Channel Model

The values of the dispersion factor and the Rician factor in the composite log-normal and Rician channel are very terrain dependent and are very difficult to generalize. Moreover, the satellite link is influenced by other parameters such as:

- the elevation angle,
- the type of antenna,
- the type of modulation (analog, digital, BPSK, QPSK ...),
- the bit rate,
- the frequency band,

- the speed of the mobile.

A slight modification of one of these parameters can change the statistics of the channel considerably . Therefore, computer simulations using a model such as the Rician model should take these parameters into consideration. Furthermore, the synchronization at the receiver should also be taken into account since the frequent deep fades due to shadowing will result in the loss of synchronization and thus in longer error bursts. Because of these reasons, we choose here to use a channel model that is derived from various experimental measurements. This simple empirical bursty channel model is used for land mobile Standard-M testing and is based on an ON-OFF model in which fading is in a binary state [16]. The OFF state corresponds to the state where the signal is severely attenuated or where the receiver is out of synchronization and the bit error rate (BER) corresponding to this state is assumed to be 50%. No transmission errors occur during the ON state. The density distribution of the fade lengths is derived from averaging real measurements done by the European Space Agency during the PROSAT program and by the German Aerospace Research Establishment (DFVLR). The results are based on an elevation angle of 25 to 28 degrees with a vehicle speed of 50 km/hr in open/rural areas. The simplified model assumes 5 possible fade lengths with the probabilities shown in the table below.

| Burst-length | Probability |
|---|---|
| 10 ms | 0.8 |
| 20 ms | 0.1 |
| 40 ms | 0.05 |
| 100 ms | 0.04 |
| 200 ms | 0.01 |

The generation of the channel states is made as follows: Every 10 ms, a test is made to decide whether a fade is occuring or not. The probability of beginning a fade interval is $p = 5.944\ 10^{-2}$. This process is a binomial process so that the distribution of the interfade lengths approaches an exponential distribution. The length of the

fade is decided according to the probabilities shown on the table above. After the end of the burst, the channel is back in an interfade state and the process repeats itself. Using this burst generation process results in an overall fading probability of 10%. The channel model is illustrated in Figure 2.4. Figure 2.5 and 2.6 show simulation runs for 2 and 10 seconds respectively. These two figures show clearly how noisy the channel is.

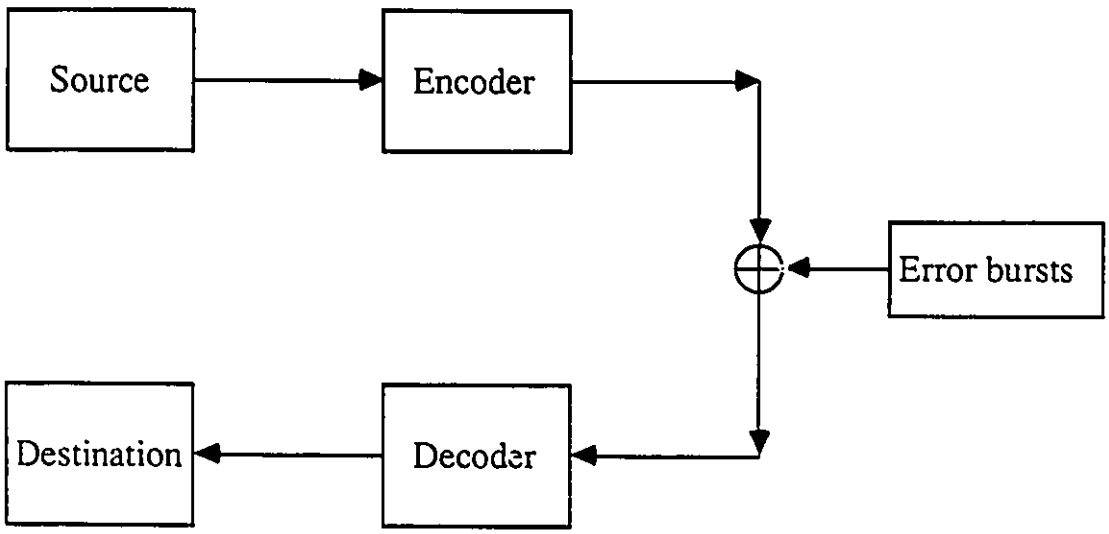


Figure 2.4: The simplified binary model for the land mobile satellite channel.

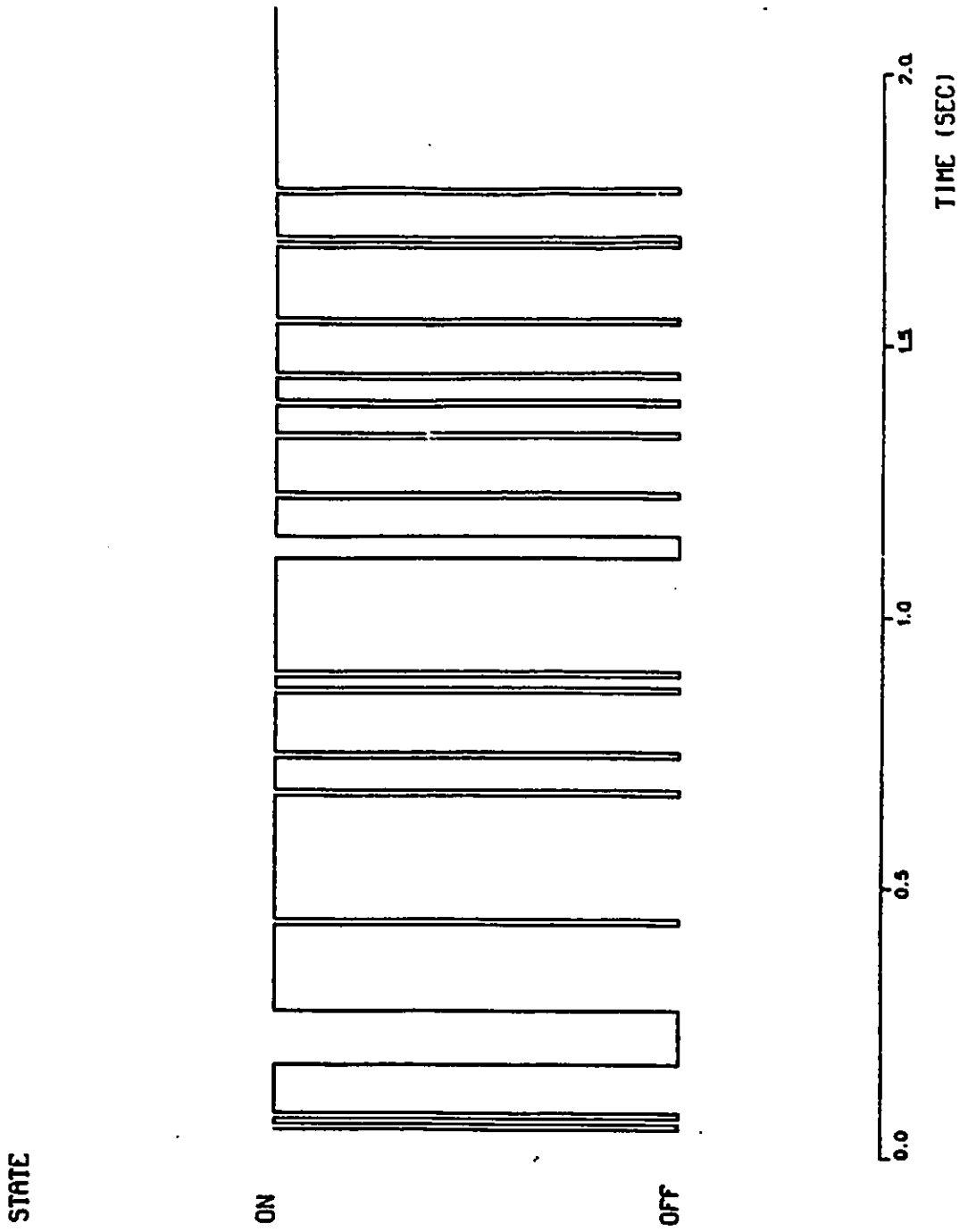


Figure 2.5: A two seconds run for the binary LMSAT channel.

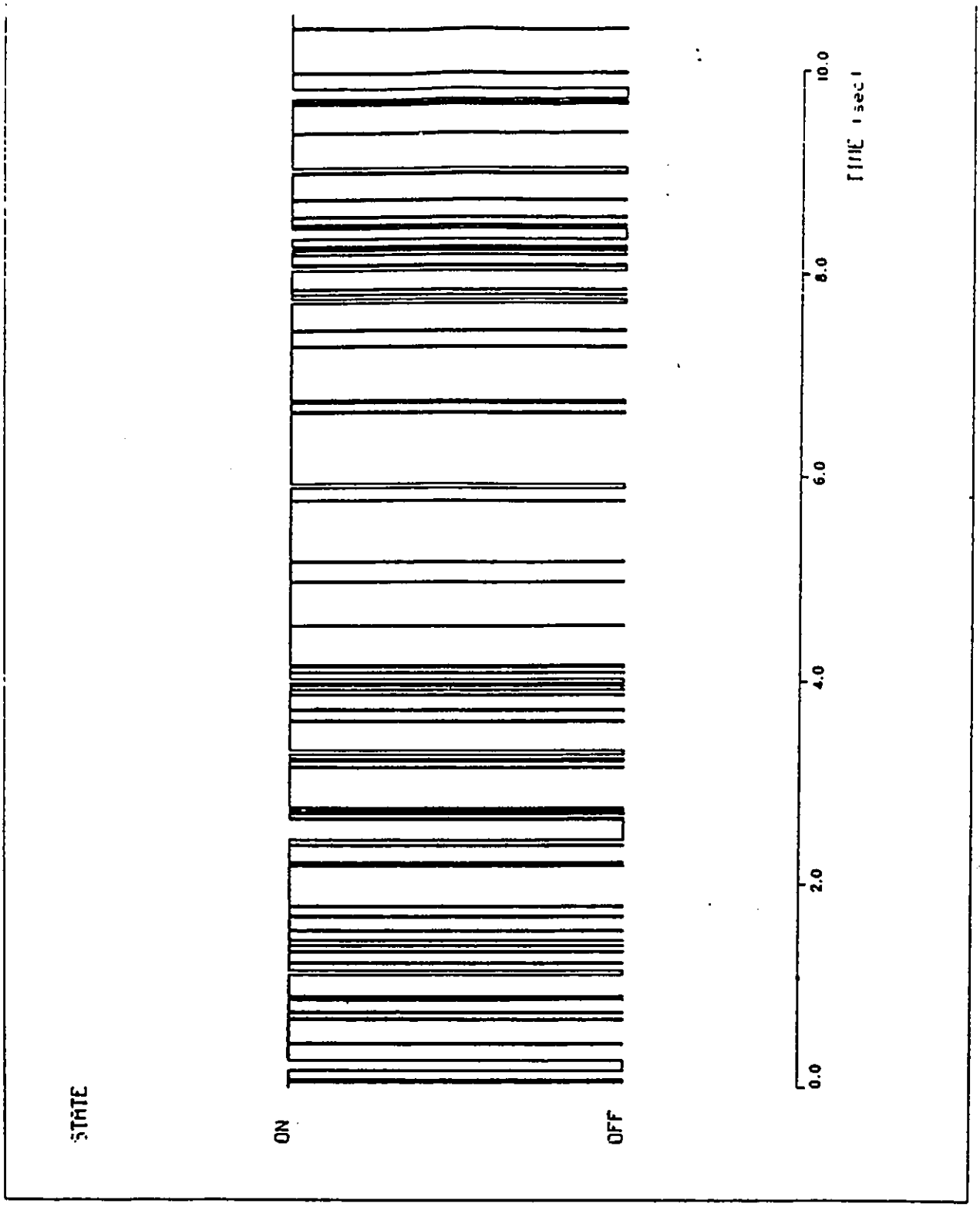


Figure 2.6: A 10 seconds run for the binary LMSAT channel.

# Chapter 3

# CODING

## 3.1 Introduction

Unlike voice or TV signals, data communication requires a very high degree of reliability, therefore, appropriate measures should be taken in order to deliver virtually error free data to the user. This chapter discusses ways of controlling the transmission errors caused by our LMSAT channel. First, a brief description of the different error control schemes is given, then, the choice of such schemes for our channel is discussed, finally, results of simulations are shown and interpreted.

## 3.2 Error Control Schemes

There are two basic categories of error control schemes for data communications: automatic-repeat-request (ARQ) schemes and forward-error-correction (FEC) schemes. Below, a brief description of both schemes is given.

### 3.2.1 ARQ Schemes

In an automatic-repeat-request scheme, an (n,k) error detecting code is used in order to check the validity of the received data. When a message of k information bits is ready for transmission, n-k parity-check bits are appended to it to form an n-bit codeword (packet). These n-k parity-check bits are formed based on the code used by the system. After transmission through the channel, the receiver checks if the received data constitutes a valid codeword of the (n,k) code or not. If not, the message is discarded and the sender is asked to retransmit the data one more time, otherwise, the received word is assumed to be error free and is delivered to the user after the parity-check bits are removed. With this error control scheme, erroneous data is delivered to the user only if the original codeword that has been sent is transformed into another codeword. The probability of such undetected errors can be made very small by choosing proper error detecting codes.
Three basic types of ARQ schemes can be distinguished [26]:

- Stop-and-wait ARQ
- Go-back-N ARQ
- Selective-repeat ARQ.

Below we describe briefly each of these protocols.

#### Stop-and-Wait ARQ

In a stop-and-wait system, the transmitter sends a codeword and waits for an acknowledgement (ACK) as shown in Figure 3.1. If a positive acknowledgement is received, the transmission is considered successful and the transmitter can send the next packet in the queue (if any). If a negative acknowledgement (NAK) is received or if no ACK is received within a specified time, the sender retransmits the packet

and again waits for an acknowledgement. Retransmissions continue until the transmitter receives an ACK. Because of the idle time wasted in waiting for ACKs, this stop-and-wait scheme is not efficient when the transmitter has more than one message to transmit and if the propagation delay is not negligible.

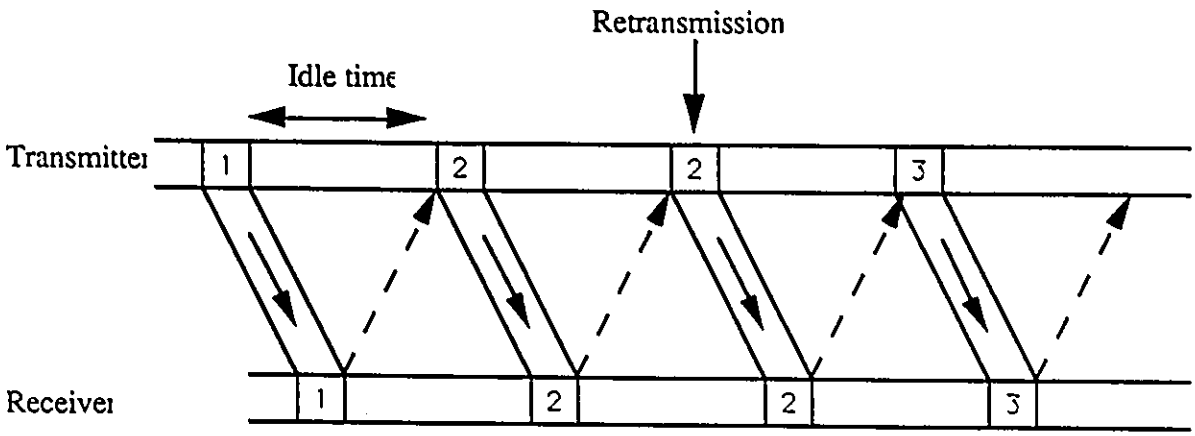


Figure 3.1: The Stop and wait ARQ protocol.

### Go-Back-N ARQ

In a go-back-N ARQ scheme, the transmitter is allowed to transmit codewords continuously in a given order and keep them pending receipt of an ACK/NAK for each packet (see Figure 3.2). When an ACK for a codeword arrives after a round-trip delay, the codeword is removed from the transmitter's buffer. Whenever the transmitter receives a NAK indicating that a particular codeword, say codeword i, was received in error, it stops transmitting new codewords, then it goes back to codeword i and proceeds to retransmit that codeword and the succeeding N-1 codewords which were transmitted during one round-trip delay. The main drawback of the go-back-N ARQ is that, whenever a received word is detected in error, the receiver also rejects the

next N-1 received words irrespective of whether they were correctly transmitted or not. This drawback can be overcome by using the selective-repeat ARQ protocol.

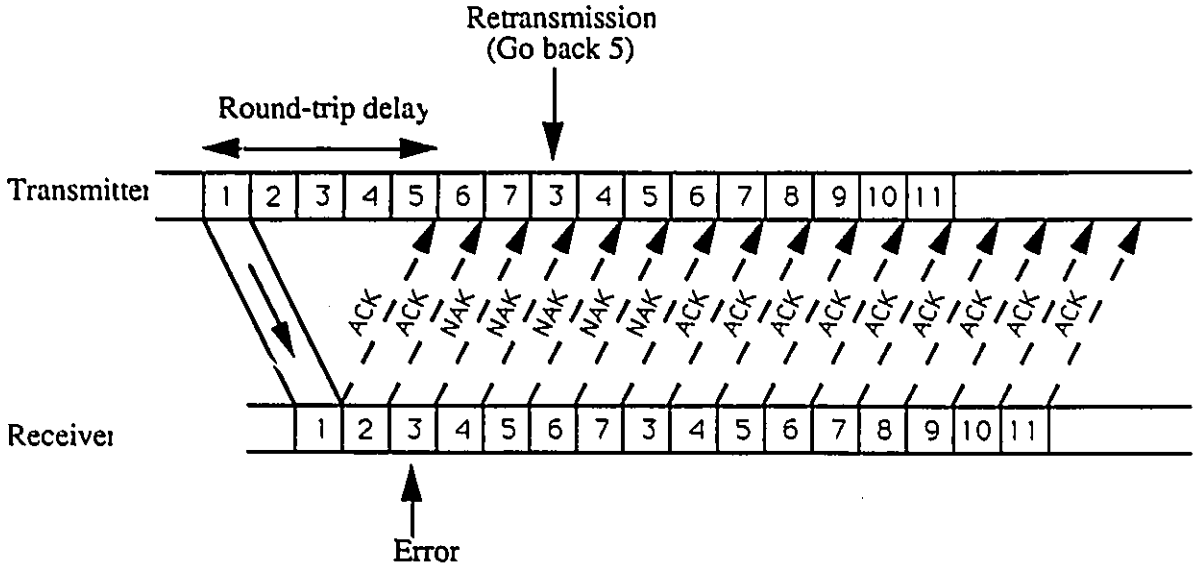


Figure 3.2: The Go-back N ARQ protocol (N = 5).

### Selective Repeat ARQ

In a selective-repeat ARQ error control system, codewords are also transmitted continuously. However, the transmitter only resends those codewords that are negatively acknowledged (NAK'ed) as shown in Figure 3.3. With this system, a buffer must be provided at the receiver to store the error free codewords following a received word detected in error, because, ordinarily, the codewords must be delivered to the end user in a correct order.

In all three ARQ schemes discussed above, there is one serious drawback. The throughput falls rapidly with increasing channel error rate. This is caused by the time wasted in retransmitting the codewords detected in error.

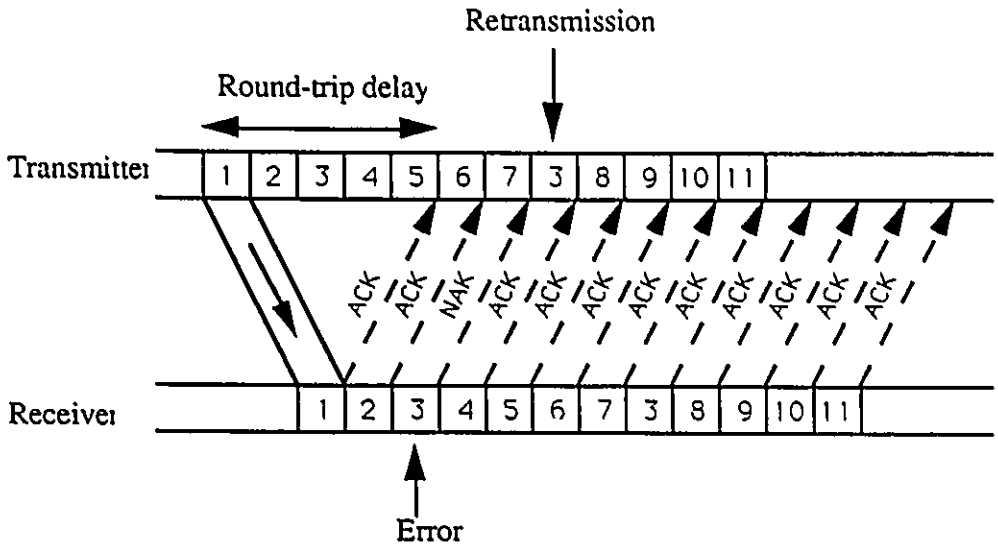


Figure 3.3: The Selective Repeat ARQ protocol.

### 3.2.2 FEC Schemes

In an FEC error control scheme, parity-check bits are appended to the message to help the receiver locate and correct the errors introduced by the channel. After error correction has been performed, the decoded word is delivered to the user. A decoding error is committed if the receiver either fails to detect the presence of errors or fails to determine the exact locations of the errors. The fact that the data is being delivered to the user regardless of whether it has been successfully decoded or not results in a lower degree of reliability compared to the ARQ schemes and contributes to the major drawback of FEC systems in data communications.

### 3.2.3 Hybrid ARQ Schemes

Drawbacks of the ARQ and FEC schemes can be overcome if the two basic error-control schemes are properly combined. A hybrid ARQ system consists of an FEC subsystem contained in an ARQ system. The function of the FEC code is to correct codewords which were received with a correctable number of errors in them. Retransmissions take place only in those less frequent cases where the message is severely damaged by the noise or by another kind of interference (collisions for example). Ideally, one can expect a high reliability as well as high throughput in a well designed hybrid system.

## 3.3 Error Control Coding over the LMSAT Channel

### 3.3.1 Choice of Error Control Scheme

As seen in Figure 2.5 and 2.6 in the previous chapter, our LMSAT channel is a particularly nasty environment in which to communicate, and, clearly, data transmission cannot be supported without the addition of some form of error control. In order to attain the level of reliability acceptable for data communication, an FEC scheme alone would require a very low code rate which would directly affect the throughput of the system. Conversely, an ARQ scheme would clearly result in a very poor performance in both delay and throughput because the probability for a message to be hit by a fade is very high and therefore a large number of retransmissions would be needed for every message.

The above arguments indicate that a logical choice for the error control scheme for our case is a hybrid ARQ scheme where messages that are lightly corrupted by the channel would be corrected whereas heavily damaged messages would have to be retransmitted [23] [17] [20] [25].

### 3.3.2 Choice of FEC Code

Two classes of codes that are strong candidates for the land-mobile satellite communications are the convolutional codes and the Reed-Solomon block codes [19]. The use of a convolutional code in such a bursty channel requires employing an interleaver in order to spread the errors and make the channel look like an AWGN channel. However, the maximum size of the interleaver cannot exceed the packet length and therefore might not be sufficient to spread the very long bursts caused by our channel. Instead of trying to randomize the channel, it is important to note that these bursts constitute an "a priori" information about the error patterns induced by the channel that can be put to use if an M-ary (M > 2) code is used instead of a binary code. When multibit symbols are transmitted in a bit-serial fashion, the erroneous bits are "trapped" in a relatively small number of symbols (because of the bursty nature of the channel) that can subsequently be corrected.

Reed-Solomon codes are powerful non-binary codes that have gained increasing interest in the last few years and are the ones that will be used throughout this thesis. An (n,k) Reed-Solomon code can correct up to $t = \lfloor \frac{n-k}{2} \rfloor$ symbol errors [27]. For instance, if a (255,125) RS code over the Galois field GF($2^8$) is used, up to 65 erroneous symbols can be corrected which correspond to $65 \times 8 = 520$ bits in a codeword of $255 \times 8 = 2040$ bits.

### 3.3.3 Symbol Interleaving

Since in RS codes the decoder complexity is a direct function of the code length as well as the number of bits per symbol, it is important to try to reduce the length of the code as much as possible. However, this reduction results in smaller codewords that can handle shorter bursts. Thus, the only way to use short code lengths in our case is to reduce the lengths of these bursts as well. This can be achieved by using a symbol interleaver as shown in Figure 3.4 [23].

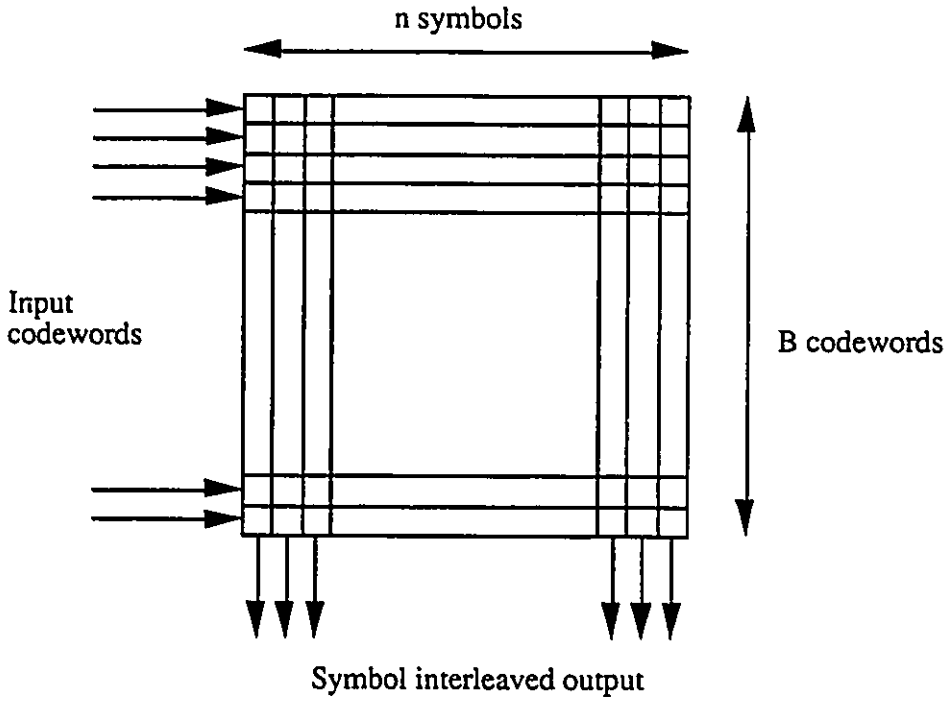


Figure 3.4: Symbol interleaving.

Codewords are read into rows of a matrix of memory elements, one symbol at a time, with each row containing one codeword. Once it is full, the contents of the matrix are read out by columns and sent to the transmitter. Figure 3.5 shows an example of a symbol interleaver used with an RS (7,3) code over GF(8). The depth of the interleaver is 6 and the length $3 \times 7 = 21$ (each symbol consists of 3 bits). The total length of the packet is then $21 \times 6 = 126$ bits. As we can see in this figure, a 30 bit error burst has been divided into small minibursts in each codeword, and, since at most two symbols per codeword have been affected, the packet will be correctly recovered by the decoder.

Note here that, with RS codes, symbol interleaving achieves a better performance

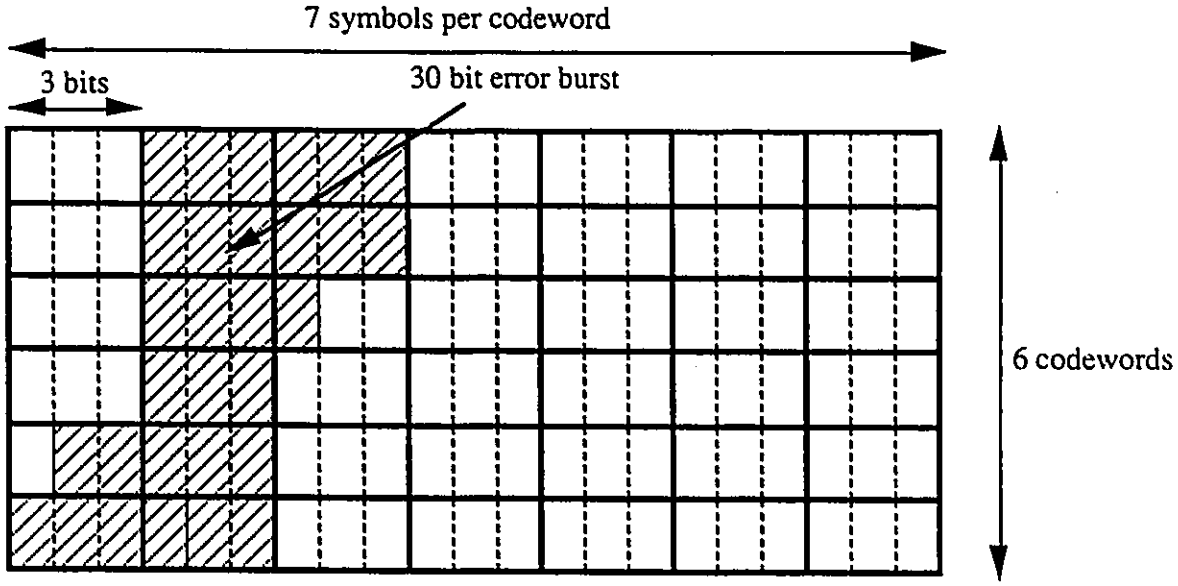


Figure 3.5: A 21 × 6 bits symbol interleaver.

than the usual bit interleaving. This can be illustrated in Figure 3.6 where a bit interleaver has been used and the same 30 bit error burst applied. As we can see, since the codewords contain more than two erroneous symbols, the decoder will fail to recover the packet in this case.

### 3.3.4 Proposed Hybrid ARQ Scheme

Unless indicated otherwise, the rest of this thesis assumes that the error-control scheme is as follows: when a message arrives at a station (user), parity-check bits are first appended to it. The block obtained is then encoded using a RS code which results in a number of codewords that are fed into a symbol interleaver as described earlier. A first transmission is then attempted taking into account the multiple access scheme in use. At the receiver end, the packet is first deinterleaved and then directed

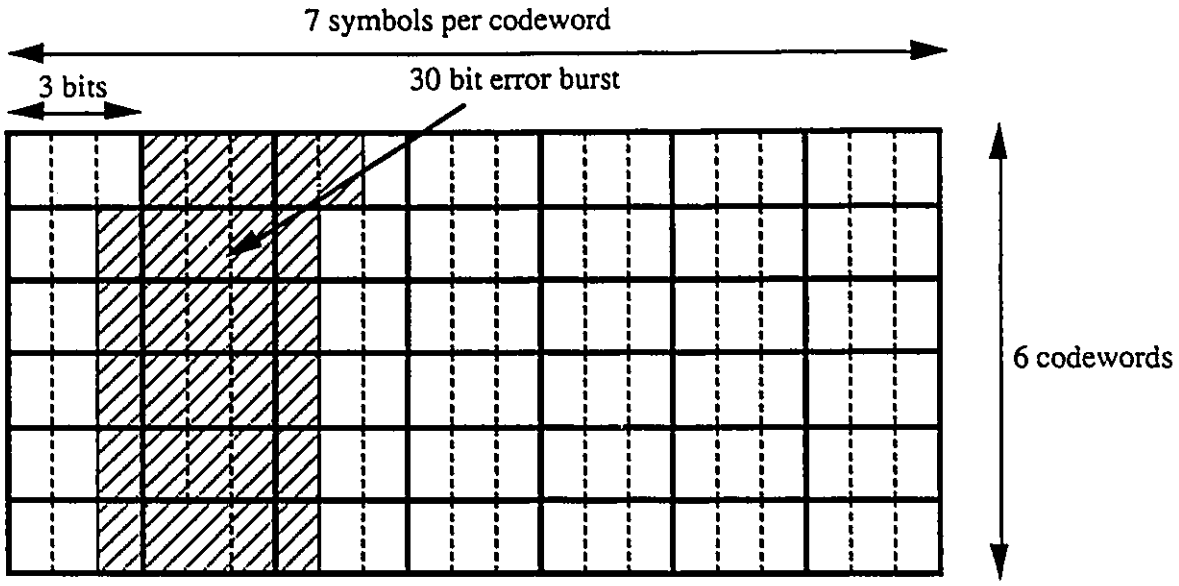


Figure 3.6: A 21 × 6 bit interleaver.

to a Reed-Solomon decoder. The validity of the decoded sequence is checked using the parity bits and, if no error is detected, an ACK is sent to the transmitter and the message is delivered to the user after the removal of the extra bits. In case of a detection of an error, the message is discarded and a NAK is sent to the transmitter asking for a retransmission. Retransmissions take place until the packet is ACK'ed. The loss of an ACK or a NAK is detected by a time out. As long as an acknowledgement for a packet has not been received, no other message can be transmitted by the user (stop-and-wait protocol). The reasons why this protocol is used and why the throughput of the system is not affected can be explained as follows.

- We are considering small mobile users that will not usually have more than one message to transmit at any one time.

- Unlike point to point communications, we are dealing here with a very large number of users sharing the same channel. Therefore, while one user might be waiting for an acknowledgement, the channel will not be idle because other users may access the channel so that no channel time is wasted and the throughput of the system is not affected by the stop and wait protocol.

## 3.4 Simulations and Results

In this section, we evaluate the performance of the coding scheme over our channel model. In order to measure the performance improvement due to coding, it is important to dissociate the coding problem from the multiple access problem which will be addressed in the next chapter. Therefore, for the rest of this chapter, the system model that is used is described below.

- The channel is dedicated to a single user that transmits packets continuously (this is equivalent to a stop and wait protocol with zero propagation delay and error free return channel).
- Before transmission, the messages go through an RS coder and a symbol interleaver as described earlier.
- After their passage through the channel, the packets are deinterleaved and decoded and the receiver computes the packet error rate (PER) that will be used to calculate the throughput of the system.
- The messages are assumed to be of 1000 information bits. The actual packet size depends on the rate of the code that is used. For instance, if a code rate of 0.5 is used, the length of the packet will be 2000 bits.
- The bit-rate is considered to be 4800 bps [9].

### 3.4.1 Throughput Evaluation

If no coding is used, the probability of a successful transmission of a packet is clearly very low. In fact, a simple analytical derivation reported in Appendix A shows that the system described above would result in a packet error rate of about 0.75 which means a normalized throughput (percentage of channel capacity effectively used) of around 25 %. Simulations were run to compute the throughput for different values of code rate and code length. The results are shown in Table 3.1.

The throughputs shown in the table are calculated as follows :

$$throughput = (1 - PER) \times R \tag{3.1}$$

where $R$ is the code rate of the RS code. As we can see from this table, using RS coding, a throughput of more than twice the throughput of an uncoded system can be achieved. The table also verifies that smaller code lengths can indeed achieve a performance comparable to that of longer code lengths if symbol interleaving is used. A non-interleaved system using a code over the Galois field GF (256) has a maximum throughput of around 58 % and so do the systems which use codes over the Galois fields GF (32), GF (64) and GF (128). We can note however that the maximum throughputs are not obtained for the same code rates.

### 3.4.2 Erasure Decoding

Despite this improvement over an uncoded system, such throughputs are not really satisfactory especially if one keeps in mind that the performance will inevitably decrease when the channel is not dedicated to a single user but shared by a very large number of them. This pushes us to explore the possibility of using erasure decoding in an attempt to increase the performance of the system.

An (n,k) RS code can correct a codeword if :

$$2t + e \leq n - k \tag{3.2}$$

Where $t$ is the number of symbols in error and $e$ is the number of symbols erased. If we assume that our receiver has the capability of monitoring the power of the received signal, we can design a system that puts to use this channel state information (CSI) in order to increase the error correcting capability of the Reed-Solomon code. When the CSI shows that the channel is in a shadowed state, it is preferrable to erase the bits received because they are unreliable. In doing so, the resulting word will contain only $e$ erasures and it will be corrected provided that :

$$e \leq n - k \tag{3.3}$$

Because all unreliable bits and not only erroneous bits are erased, the correcting capability of the RS code is slightly less than twice its capability when standard decoding is used. Table 3.2 shows the results obtained by simulation.

Figure 3.7 shows the improvement achieved by erasure decoding. The curves correspond to RS codes in GF (256) without interleaving.

Figure 3.8 shows the effect of the code rate on the throughput. We can see that a maximum throughput of about 0.68 are obtained for code rates around 0.8. We can also note once again that there is no need to use very high code lengths if symbol interleaving is used.

## 3.5 Conclusions

In this chapter, after a general description of different error control schemes that can be used, the reason for the choice of an hybrid ARQ scheme was discussed. Then the

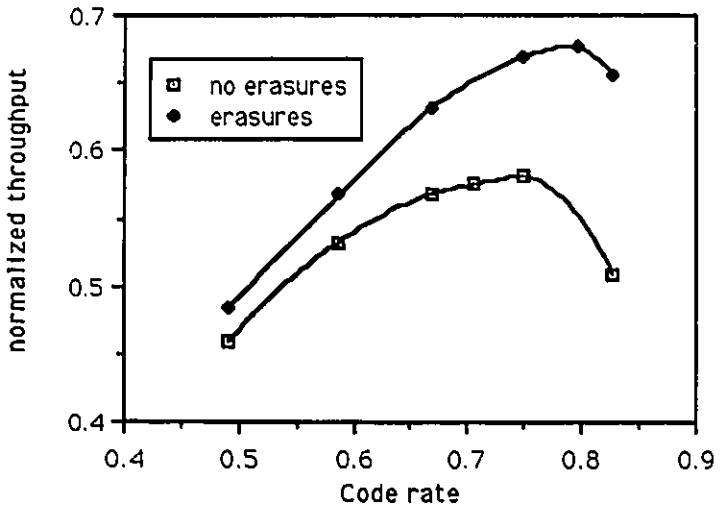


Figure 3.7: Throughput improvement due to erasure decoding in a single user system.

advantages of using Reed-Solomon codes for error correction were explained. We also showed that symbol interleaving permits to reduce the complexity of the decoder by allowing the use of short RS codes. Simulation results were obtained to evaluate the achievable reduction in code length. Finally, in an attempt to improve the overall performance of the system, erasure decoding which puts to use the channel information state was investigated and adopted after encouraging results were obtained by simulations.

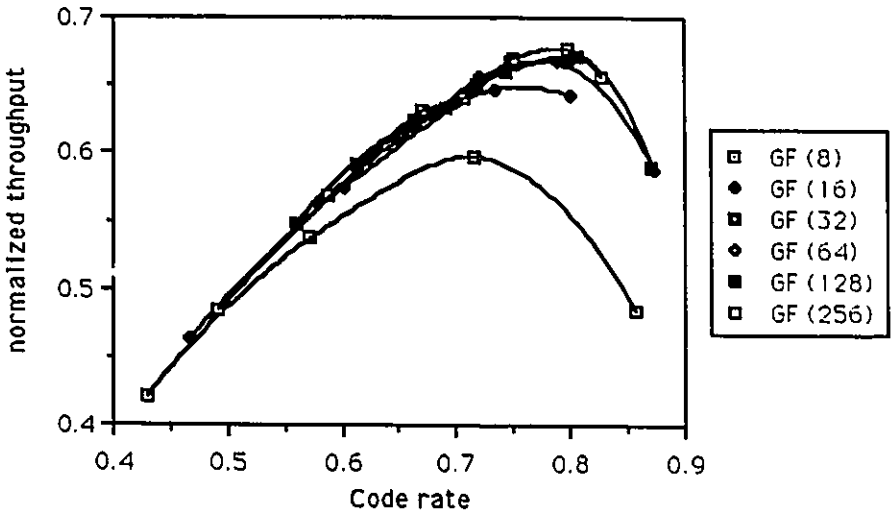


Figure 3.8: Effect of the code rate and the code length on the throughput of a single user system.

| n | k | Galois Field | Code rate | Interleaver (bits x bits) | Packet length (bits) | PER | Throughput |
|---|---|---|---|---|---|---|---|
| 7 | 5 | GF (8) | 0.71 | 21x66 | 1386 | 0.42 | 0.42 |
| 7 | 3 | GF (8) | 0.43 | 21x111 | 2331 | 0.17 | 0.35 |
| 15 | 11 | GF (16) | 0.73 | 60x23 | 1380 | 0.30 | 0.52 |
| 15 | 9 | GF (16) | 0.60 | 60x28 | 1680 | 0.17 | 0.50 |
| 15 | 7 | GF (16) | 0.47 | 60x36 | 2160 | 0.10 | 0.42 |
| 31 | 25 | GF (32) | 0.81 | 155x8 | 1240 | 0.37 | 0.50 |
| 31 | 23 | GF (32) | 0.74 | 155x9 | 1395 | 0.26 | 0.55 |
| 31 | 21 | GF (32) | 0.68 | 155x10 | 1550 | 0.14 | 0.59 |
| 31 | 19 | GF (32) | 0.61 | 155x11 | 1705 | 0.12 | 0.54 |
| 63 | 41 | GF (64) | 0.65 | 378x4 | 1512 | 0.13 | 0.56 |
| 57 | 33 | GF (64) | 0.58 | 342x5 | 1710 | 0.10 | 0.52 |
| 107 | 71 | GF (128) | 0.66 | 749x2 | 1498 | 0.14 | 0.57 |
| 127 | 71 | GF (128) | 0.56 | 889x2 | 1778 | 0.09 | 0.51 |
| 151 | 125 | GF (256) | 0.83 | none | 1208 | 0.38 | 0.51 |
| 157 | 125 | GF (256) | 0.80 | none | 1256 | 0.34 | 0.53 |
| 167 | 125 | GF (256) | 0.75 | none | 1336 | 0.22 | 0.58 |
| 177 | 125 | GF (256) | 0.71 | none | 1416 | 0.19 | 0.58 |
| 187 | 125 | GF (256) | 0.67 | none | 1496 | 0.15 | 0.57 |
| 213 | 125 | GF (256) | 0.59 | none | 1704 | 0.09 | 0.53 |
| 255 | 125 | GF (256) | 0.49 | none | 2040 | 0.06 | 0.46 |

Table 3.1: Results of the simulations for different codes and code rates.

| n | k | Galois Field | Code rate | Interleaver (bits x bits) | Packet length (bits) | PER | Throughput |
|---|---|---|---|---|---|---|---|
| 7 | 6 | GF (8) | 0.86 | 21x56 | 1176 | 0.43 | 0.48 |
| 7 | 5 | GF (8) | 0.71 | 21x66 | 1386 | 0.16 | 0.60 |
| 7 | 4 | GF (8) | 0.57 | 21x83 | 1743 | 0.06 | 0.54 |
| 7 | 3 | GF (8) | 0.43 | 21x111 | 2331 | 0.02 | 0.42 |
| 15 | 12 | GF (16) | 0.80 | 60x21 | 1260 | 0.20 | 0.64 |
| 15 | 11 | GF (16) | 0.73 | 60x23 | 1380 | 0.12 | 0.65 |
| 15 | 9 | GF (16) | 0.60 | 60x28 | 1680 | 0.04 | 0.57 |
| 15 | 7 | GF (16) | 0.47 | 60x36 | 2160 | 0.01 | 0.46 |
| 31 | 27 | GF (32) | 0.87 | 155x7 | 1085 | 0.32 | 0.59 |
| 31 | 25 | GF (32) | 0.81 | 155x8 | 1240 | 0.17 | 0.67 |
| 31 | 23 | GF (32) | 0.74 | 155x9 | 1395 | 0.11 | 0.66 |
| 31 | 21 | GF (32) | 0.68 | 155x10 | 1550 | 0.07 | 0.63 |
| 31 | 19 | GF (32) | 0.61 | 155x11 | 1705 | 0.04 | 0.59 |
| 63 | 55 | GF (64) | 0.87 | 378x3 | 1134 | 0.33 | 0.59 |
| 52 | 41 | GF (64) | 0.79 | 312x4 | 1248 | 0.15 | 0.67 |
| 57 | 41 | GF (64) | 0.72 | 342x4 | 1368 | 0.09 | 0.66 |
| 63 | 41 | GF (64) | 0.65 | 378x4 | 1512 | 0.06 | 0.61 |
| 57 | 33 | GF (64) | 0.58 | 342x5 | 1710 | 0.03 | 0.56 |
| 89 | 71 | GF (128) | 0.80 | 623x2 | 1246 | 0.16 | 0.67 |
| 95 | 71 | GF (128) | 0.75 | 665x2 | 1330 | 0.11 | 0.67 |
| 99 | 71 | GF (128) | 0.72 | 693x2 | 1386 | 0.09 | 0.65 |
| 103 | 71 | GF (128) | 0.69 | 721x2 | 1442 | 0.08 | 0.63 |
| 107 | 71 | GF (128) | 0.66 | 749x2 | 1498 | 0.06 | 0.62 |
| 127 | 71 | GF (128) | 0.56 | 889x2 | 1778 | 0.02 | 0.55 |
| 151 | 125 | GF (256) | 0.83 | none | 1208 | 0.21 | 0.66 |
| 157 | 125 | GF (256) | 0.80 | none | 1256 | 0.15 | 0.68 |
| 167 | 125 | GF (256) | 0.75 | none | 1336 | 0.11 | 0.67 |
| 177 | 125 | GF (256) | 0.71 | none | 1416 | 0.09 | 0.64 |
| 187 | 125 | GF (256) | 0.67 | none | 1496 | 0.06 | 0.63 |
| 213 | 125 | GF (256) | 0.59 | none | 1704 | 0.03 | 0.57 |
| 255 | 125 | GF (256) | 0.49 | none | 2040 | 0.01 | 0.48 |

Table 3.2: Results of the simulations for different codes and code rates when erasure decoding is used.

# Chapter 4

# MULTIPLE ACCESS

## 4.1 Introduction

In a land-mobile satellite network, all the mobiles share the same frequency band for the transmission of their information to the hub. Therefore, it is necessary to define a protocol which defines the rules for accessing the channel. The choice of this multiple access scheme has a direct influence on the complexity of the system and its performance. Ideally, an access scheme should be: easy to implement, result in a low average message delay and achieve a high throughput. In this chapter, we first give a general overview of different multiple access protocols. Then we focus on those protocols which can logically be used in the case of satellite data communications and select four different schemes in order to compare their performance over our LMSAT channel model. Each protocol is thus described separately in detail and results, obtained by analysis and/or computer simulations, about its performance are shown.

## 4.2 Multiple Access Schemes

Multiple access techniques can be grouped into four different categories [30]:

1. Fixed assignment techniques.

2. Random access techniques.

3. Demand assignment techniques.

4. Mixed strategies.

Below, we briefly describe each of these categories and discuss their applicability to different traffic types and environments. The different multiple access schemes mentioned in the following constitute by no means an exhaustive list of all possible access strategies but are only given as examples for the description of each category.

### 4.2.1 Fixed Assignment Techniques

Fixed assignment techniques refer to preassigning portions of the channel bandwidth to each user. This assignment is fixed and does not vary as a function of the activity of the user at different instants. There are basic different access techniques of this type, each of them using a different way of partitioning the channel capacity.

1. Frequency division multiple access (FDMA): In this scheme, as shown in Figure 4.1, each user is assigned a frequency band for its exclusive use. Orthogonality is thus achieved in the frequency domain. This scheme is relatively simple to implement and is appropriate in the case of a stream-like traffic and when the capacity requirements of every user are well known. However, in a mesh network, interconnectivity is difficult to achieve because of the need to have several IF receivers.

2. Time division multiple access (TDMA): Here, the users have access to the entire channel bandwith but only during time slots which have been preassigned for each one of them as shown in Figure 4.2 (orthogonality in the time domain).

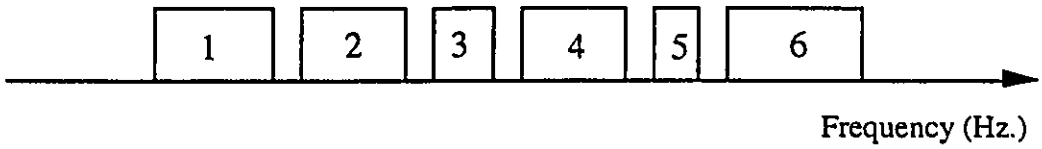


Figure 4.1: Frequency division multiple access (FDMA).

This scheme is also appropriate in the case of a stream-like traffic and well known capacity requirements for every user. Among the advantages of TDMA, we can cite:

- The allocation of the time slots can easily be modified to take into account the accomodation of new users or any changes in capacity requirements for existing users.
- Interconnectivity is easy to achieve since all the users transmit at the same frequency and no separate IF receivers are necessary.
- In the case of satellite communications, the intermodulation products resulting from multiple carriers is reduced considerably and, therefore, the satellite power resources can be used much more efficiently (no backoff is necessary and the full power of the satellite transponder is used).

However, TDMA assumes that all the stations share a global time reference, and this can be hard to achieve in some cases.

3. Code division multiple access (CDMA): This technique achieves quasi-orthogonality by use of different signalling codes for every transmitter. The intended receiver is equipped with a circuit that is "tuned" to the transmitter's code. This clearly results in a lack of flexibility and interconnectivity. However, CDMA has the advantage of allowing the use of the same frequency band and requiring no global time reference among the users.

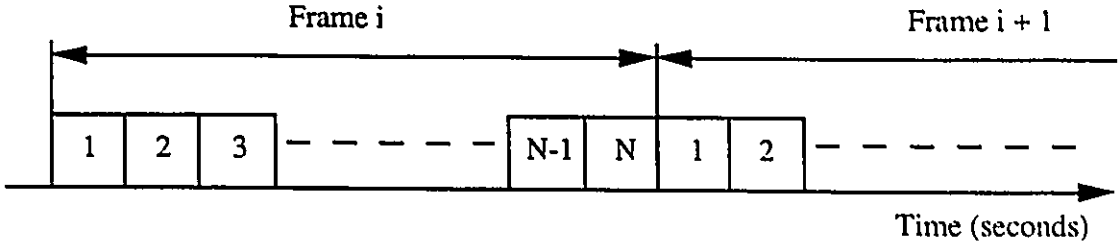


Figure 4.2: Time division multiple access (TDMA).

In the above description of the three protocols, we assumed fixed assignment (of frequencies, times slots and codes), but, clearly, as we shall see later, this assignment can be made on demand or randomly and other schemes will derive from these schemes.

When the user's traffic is highly bursty and low volume, fixed allocation can be very wasteful in channel capacity and other techniques should be employed.

### 4.2.2 Random Access Techniques

In applications involving messages that are short and transmitted infrequently, it may turn out that the chance of two or more sources requiring the channel simultaneously is quite low. In this case, it may be appropriate to let all users transmit randomly, at will. If a collision between messages of different sources occur, these messages must be retransmitted. Some schemes which fall in the category of random access techniques are described below.

1. ALOHA: This is the simplest scheme of its kind. The user is allowed to transmit at any time it desires as seen in Figure 4.3. After the transmission of a message, the user waits for an acknowledgement from the destination. If an ACK is not received within some appropriate time-out period, it assumes that a collision

has occured. To avoid repeated conflicts, the retransmissions take place after a random delay [37].

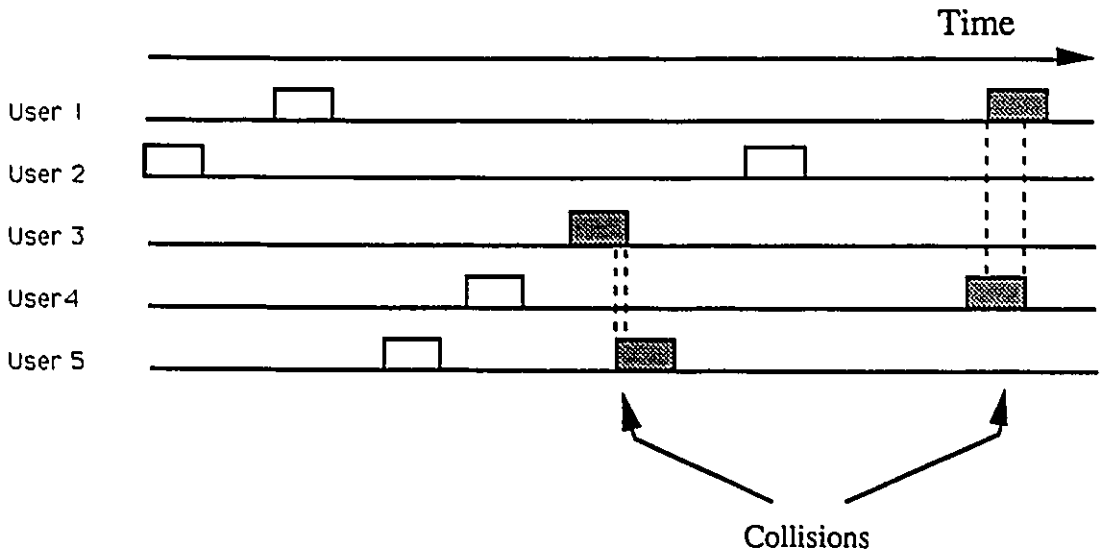


Figure 4.3: The ALOHA protocol.

2. Slotted ALOHA: In this scheme, the time is divided into slots of fixed duration equal to the time required to transmit a message as shown in Figure 4.4. When one user has a packet to transmit, it waits for the beginning of a slot before sending it. As in the previous scheme, the user then waits for an ACK and, in case of a collision, retransmits the message after a random number of slots. The advantage of this protocol compared to ALOHA is that it achieves a higher maximal throughput [37].

3. Carrier sense multiple access (CSMA): When the users are able to listen to the channel, they could inhibit the transmission of their message when the channel is busy and wait until it is idle before transmitting. If this carrier sensing method is used, collisions will only occur if two or more users have sensed the

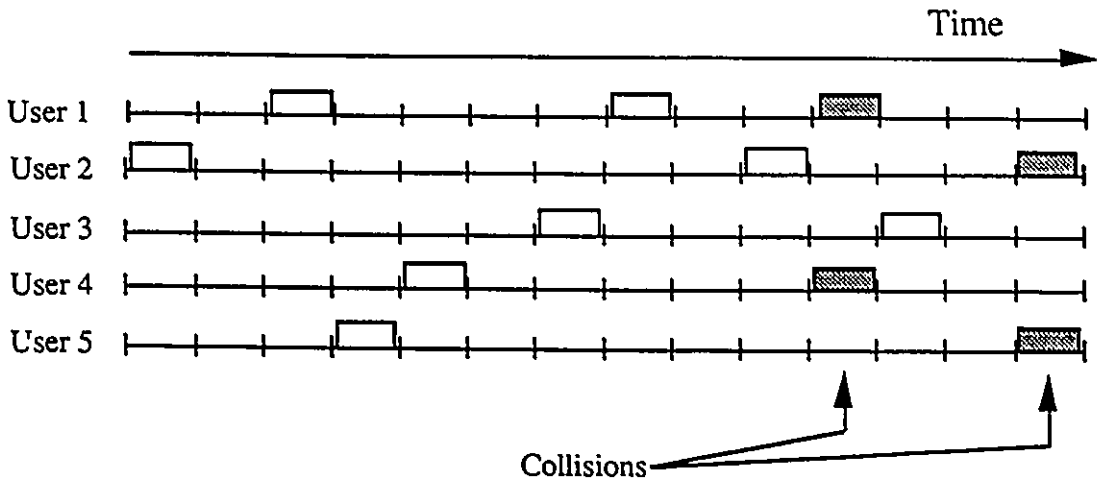


Figure 4.4: The slotted ALOHA protocol.

channel at the same time and begin the transmission of their data. When this happens, retransmission are rescheduled according to a randomly distributed delay. There exist many variations of this scheme which will not be covered here because of lack of space. However, it is important to note that when the propagation delay is not small, the ability for the users to sense the channel effectively is reduced and these schemes become quickly inefficient [32].

4. Random access in CDMA: In the case of bursty traffic, CDMA techniques can be employed in a random access mode. Since users do not transmit continuously, a larger number of them can be accomodated in the network. An example of such scheme is the frequency-hopped-spread-spectrum-multiple-access. In this scheme, the stations are allowed to transmit at will whenever they have a message to send. During the transmission, the frequency is changed according to some known or random pattern. If transmissions from two different users overlap in time, the actual portions of the messages where collision occurs both in time and frequency are likely to be short so that both messages can still be

recovered or 'captured' [41].

Random access techniques have a typical behavior: the throughput of such systems increases as the offered channel traffic increases but reaches a maximum value and then constantly decreases. This is due to the fact that the increase in channel traffic increases the number of collisions because of contention, this in turn increases the load on the channel and so forth. Such positive feedback causes instability and the throughput to decrease to very low values.

### 4.2.3 Demand Assignment Techniques

In demand assignment techniques, the transmission of messages is scheduled ahead of time so that collisions which waste channel capacity are avoided. The control of the channel can either be done by a unique station or by all users executing a distributed algorithm.

#### Centrally controlled demand assignment techniques

Examples of such schemes are given below.

1. Circuit oriented systems: In these schemes, the bandwith is divided into FDMA or TDMA subchannels which are assigned on demand. The allocation of a subchannel for a user remains for the duration of the message. Because of the setup times required for the assignment of bandwith, these schemes are only attractive for stream-type traffic like voice calls and long file transfers.

2. Polling systems: In this type of access strategy, a central controller interrogates all the users sequentially, one by one asking them to transmit any pending messages they have. The transmission of packets takes place on the inbound channel shown in Figure 4.5. A user with no messages to transmit indicates

so in a negative reply to the controller. Once all the stations have been given permission to transmit, a cycle is completed and a new cycle begins. There exists many variants of this scheme such as the tree search algorithm and others, however, these schemes are efficient only in those systems that satisfy the following conditions [32].

- The number of users is relatively small.
- The round-trip delay is negligible.
- The overhead due to polling messages is low.
- The communication links between the controller and the users are reliable.

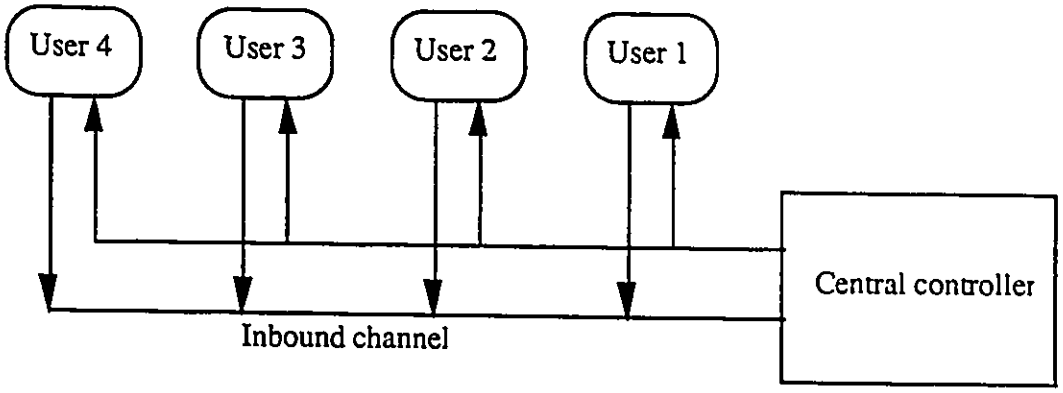


Figure 4.5: Polling systems.

3. Reservation schemes: In the polling schemes, it is the responsibility of the central controller to interrogate the stations and schedule their transmissions. On the contrary, in reservation schemes, it is the user who takes the initiative of asking the controller to assign a reserved time slot for its transmission. The job of the central controller is to manage the queue of requests that it receives and inform the demanding users of their allocated time. Obviously, since all users share the same inbound channel to the controller, it is necessary to define

an access scheme which decides how requests are to be transmitted. Fixed assignment and random access techniques are two possible choices although the latter would be more efficient in case of large number of bursty stations.

In reservation schemes, in order to avoid collisions between requests and messages, the channel is often split, by frequency division or time division, into two distinct subchannels, one for requests and the other one for messages.

In the case of a large population of bursty stations, a reservation scheme usually provides a higher throughput compared to a random access scheme at the cost, however, of larger delays in light traffic [31] [30] [42].

### Demand assignment with distributed control

The idea in this type of schemes is to let the users share some information regarding the demand on the channel and its usage so that an algorithm executed by each one of them independently will result in some sort of coordination in their action. We give two examples of such strategies.

1. Reservation-ALOHA: Here, every user is assumed to be able to hear the transmissions of all other users. Slots are organized into frames of fixed size. If a station has had a successful packet transmission in the previous frame, it gets to keep the slot. It loses the slot if it has nothing to transmit. Empty slots, including those involved in collision in the previous frame are available to all users on a random access basis. So, in this reservation scheme, the reservation is obtained by first transmitting a packet successfully. High throughput can be expected from this strategy in those cases where stations have long messages to transmit or transmit continuously [33].

2. Token ring: This access scheme requires that the stations be connected to each other in a ring topology as shown in Figure 4.6. All messages move around

the ring and are actively repeated by each station through which they pass. A circulating message is removed from the ring by the station that transmitted it. The access scheme consists of passing the access right, represented by a short packet called token, sequentially from station to station around the ring. Any station with a ready message waits for the control token, transmits its messages and then passes on the control token.

The token ring scheme described above, or any variant of it, is widely used in local area networks with a relatively small number of stations, fast reliable links and small propagation delays, however, such schemes are not suitable for satellite communications [32].

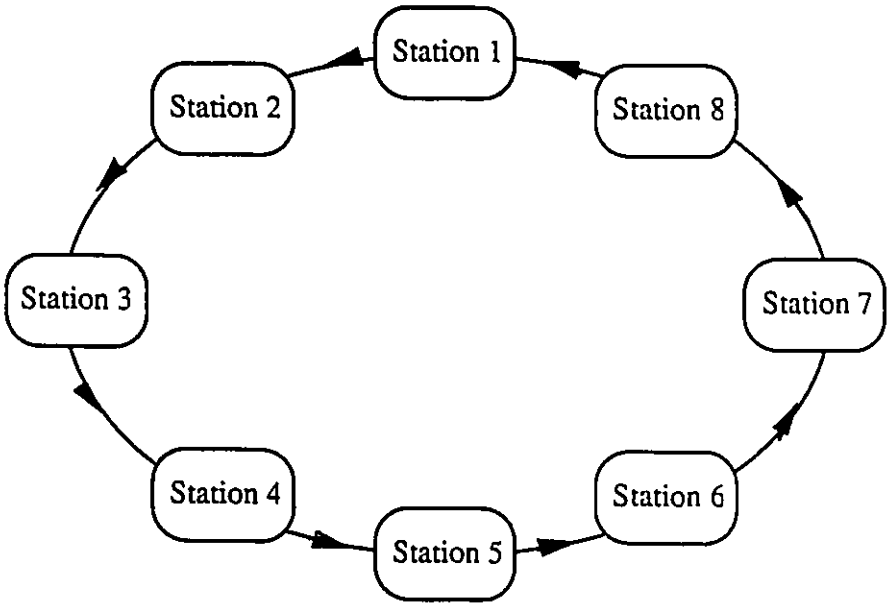


Figure 4.6: The ring topology.

### 4.2.4 Mixed Strategies

Because of the short delays achieved by random access techniques under light traffic and t[illegible] throughputs obtained by reservation schemes, it is logical to investigate the use of adaptive protocols which operate in a contention mode under light traffic and switch to reservation r[illegible] the offered traffic increases. Several such schemes have been proposed althou[illegible]ey tend to be rather complex in their implementation. SRUC (split reservation upon collision) [39] [33] and PODA (priority oriented demand assignment) [46] are two examples of access schemes that fall into this category.

## 4.3 Multiple Access for the Land-Mobile Satellite Data Networks

Although fixed assignment techniques have been widely used in satellite communications to transmit TV signals, voice traffic or large amount of data, they are obviously not adequate for our case where the traffic is coming from a large number of bursty users (mobiles).

Random access techniques are well suited for such kind of traffic and are interesting candidates for the LMSAT communications. Unfortunately, because of the large propagation delay, those random schemes that use carrier sensing are unusable.

In the demand assignment category of multiple access, we can exclude the use of schemes with distributed control because of the difficulty of any possible coordination among the mobiles (the signal transmitted by a mobile and reflected by the satellite transponder is too weak to be received by the antenna of another mobile because of its low gain). This problem does not arise in centrally controlled demand assignment schemes where the hub, with its large antenna, can play the role of the central controller or, more adequately in this case, the network management center (NMC). However, because of the large number of users and, once more, the propagation delay,

polling schemes should be excluded.

The access schemes that have been selected for a performance evaluation in this thesis are the slotted ALOHA and the frequency hopping multiple access from the random access family, AMAP (adaptive mobile access protocol) which is a centrally controlled demand assignment scheme and finally, a mixed strategy where we try to combine AMAP with the frequency hopping technique.

## 4.4 General assumptions and parameters

The following assumptions are made for the rest of this chapter.

1. Traffic comes from a large population of bursty users.
2. Messages are single packets of fixed size.
3. Messages are generated by the mobiles in a Poisson manner.
4. The rate at which messages are generated at the mobiles is the same for all the mobiles.
5. The transmission and processing time of the acknowledgements and assignment packets at the hub is negligible.
6. Undetected error probability is assumed to be zero.
7. The capture effect due to discrepancies in the received power of different transmissions is negligible.

Some of the parameters that will be used in the analysis are chosen as:

- Round trip propagation delay = 0.55 seconds.

- Packet size = 1000 information bits.
- Bit rate = 4800 bits per seconds.
- Guard time between succesive slots = 0 or 4 ms.

When computer simulation was used to derive the performance, the programs were run until at least 5000 packets were transmitted. This number was adopted after many trials which showed that running the simulations for longer periods would result in negligible differences in the results.

## 4.5 Slotted ALOHA

### 4.5.1 Description of the Protocol

As described earlier, the channel time in this protocol is divided into slots of fixed duration equal to the time required to transmit a packet. Users can only transmit messages at the beginning of a slot. An acknowledgement packet transmitted by the hub on a separate broadcast channel indicates to the user that the data was correctly received. If no ACK or a negative reply (NAK) is received, the user waits for a random number of slots before attempting to retransmit the data. The corruption of a packet is due to either a collision or noise in the channel (or both at the same time).

### 4.5.2 Performance Analysis

The performance of slotted ALOHA, using the set of parameters described earlier, is evaluated first in the case of ideal (error free) channels (both forward and return channel) and then using our LMSAT channel model.

#### Performance in an ideal channel

The performance of slotted ALOHA in an ideal channel has been extensively studied in the literature (e.g [38] or [32]). The reproduction of the basic steps of this analysis is necessary in order to derive the performance of the scheme in the LMSAT channel.

The channel traffic consists both of newly-generated packets and retransmitted packets. Let $\lambda$ be the rate of generation of packets at the mobiles. If N is the number of users and $\tau$ the slot duration, the total number of newly-generated packets during one slot is given by:

$$S = N\lambda\tau \text{ packets.} \tag{4.1}$$

The total traffic rate of packets attempting transmission over the channel, newly generated plus retransmitted ones, is then some number $\lambda' > \lambda$. The actual traffic intensity on the channel is thus a parameter G given by:

$$G = N\lambda'\tau \text{ packets per slot.} \tag{4.2}$$

We now make the assumption that the retransmitted messages, like the newly arrived ones, are Poisson distributed. This assumption has been proven to be valid when the random retransmission delay time is relatively long [38]. In this case, the total channel traffic is also Poisson distributed with parameter G. Then, since the probability of no transmission in one slot is $e^{-G}$, the probability for a packet transmitted by one user to collide is:

$$P = 1 - e^{-G} \tag{4.3}$$

Because this probability is also the probability for a packet to need retransmission, the total average number of retransmissions in a $\tau$ seconds interval is:

$$\text{Avg. no. of retransmissions in a } \tau \text{ sec. interval } = G(1 - e^{-G}) \tag{4.4}$$

But, by definition, we can write

$$G = S + \text{Avg. no. of retransmissions in a } \tau \text{ sec. interval.} \tag{4.5}$$

Hence

$$G = S + G(1 - e^{-G}) \tag{4.6}$$

Which becomes simply

$$S = Ge^{-G} \tag{4.7}$$

The equation above shows that S increases to a maximum value of 0.368 attained when the value of G is 1 and then decreases rapidly to zero as shown in Figure 4.7.

The average maximum number of packets that can be successfully transmitted during a $\tau$ seconds interval is then about 0.36 packets. In order to evaluate the

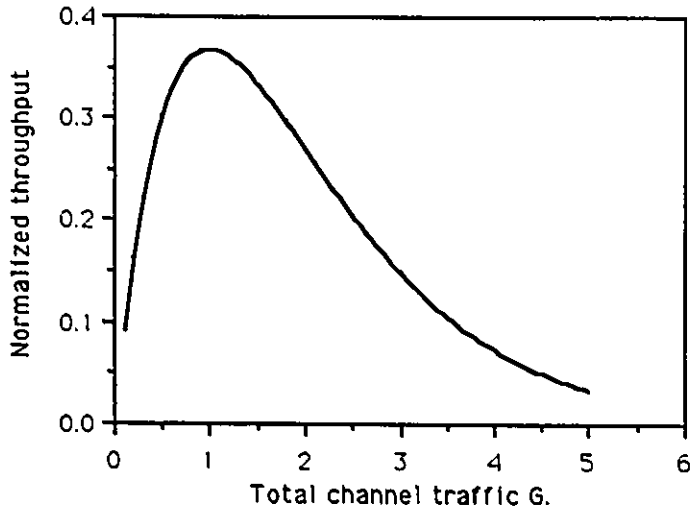


Figure 4.7: The throughput of slotted ALOHA in an ideal channel.

efficiency $\gamma$ of the access scheme, this number should be compared to 1 packet per slot that can be achieved by a theoretical perfect scheduling scheme. Thus, in this case, the efficiency, or, in other words, the normalized throughput of the scheme is given by:

$$\gamma = S = Ge^{-G} \tag{4.8}$$

In the case where guard times are used, slots become larger in size than the actual packets so that the number of packets per slot duration that can be transmitted with a perfect scheduling scheme is slightly more than just 1. Hence, the normalized throughput $\gamma$ is obtained by dividing S by this number. For instance, if a guard time of 4 ms is used, the duration of each slot is:

$$\frac{1000 \text{ bits}}{4800 \text{ bps}} + 4 \text{ ms} = 212.3 \text{ ms} \tag{4.9}$$

A perfect scheduling scheme can transmit

$$0.2123 \times 4800 = 1019 \text{ bits per slot} \tag{4.10}$$

Which represents $\frac{1019}{1000} = 1.019$ packets. The normalized throughput of slotted ALOHA is then obtained by dividing S by 1.019. Figure 4.8 shows the effect of the guard time on the efficiency of the scheme.

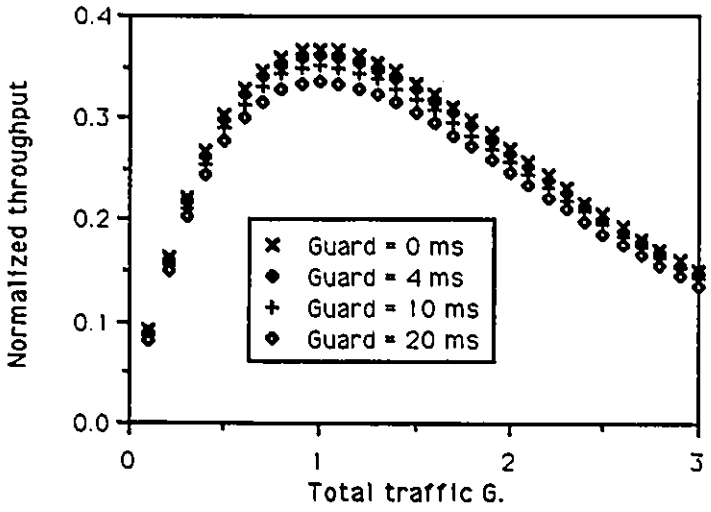


Figure 4.8: Effect of the guard time on the throughput of slotted ALOHA in an ideal channel.

On the other hand, we can show that the average packet delay, computed from the time the packet arrives at the mobile to the time an acknowledgement is received from the hub, is given by:

$$D = \tau[0.5 + 1 + \text{prop.} + E(0.5 + \text{prop.} + \frac{K+1}{2})] \text{ sec.} \qquad (4.11)$$

where:

- $\tau$ is the slot duration.
- prop. is the round trip propagation delay normalized to the slot duration $\tau$.
- $E$ is the average number of retransmissions per message.

- $K$ is the retransmission interval in number of slots.

The first 0.5 corresponds to the average time a user waits for the beginnning of a slot in order to transmit its packet. The following 1 is the transmission time of a message. Prop. is the time the user has to wait for an ACK or NAK. When a packet is to be retransmitted, half a slot on the average is again needed to wait for the beginning of a new slot and, finally, $\frac{K+1}{2}$ is the average number of slots required to retransmit the packet (transmission included). The average number of retransmission attempts per message obeys the equation

$$\frac{G}{S} = 1 + E \tag{4.12}$$

Hence

$$E = \frac{G}{S} - 1 = e^G - 1 \tag{4.13}$$

Thus, the average message delay becomes:

$$D = \tau[0.5 + 1 + \text{prop.} + (e^G - 1)(0.5 + \text{prop.} + \frac{K+1}{2})] \text{ sec.} \tag{4.14}$$

When guard times are used, $\tau$ represents the time required to transmit a packet plus the guard time.

Since the normalized throughput and the delay are both functions of traffic G, we can draw the throughput-delay performance of the scheme. The curves obtained are given in Figure 4.9 with a value of K chosen to be 15.

### Performance over the LMSAT channel model

When the slotted ALOHA scheme is used over a non-ideal channel, retransmissions are needed due to three possible reasons:

1. Collisions with other packets.

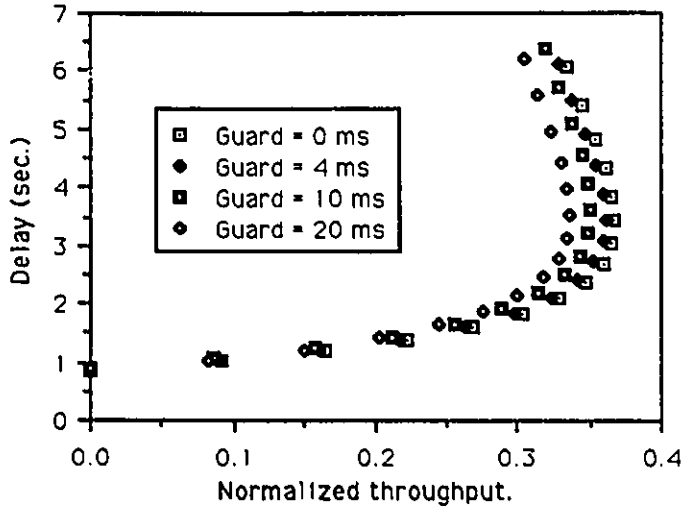


Figure 4.9: The delay throughput performance of slotted ALOHA in an ideal channel.

2. Errors on the return channel (mobile to hub).

3. Errors on the forward channel (hub to mobile) which results in the loss of an acknowledgement packet from the hub.

The probability of a successful transmission can be written as:

$$\text{Prob (success)} = P_1 \times P_2 \times P_3 \tag{4.15}$$

Where:

- $P_1$ is the probability that no collisions will occur.
- $P_2$ is the probability that the eventual errors in the information packet will be successfully corrected by the FEC decoder at the hub.
- $P_3$ is the probability that the ACK transmitted by the hub will successfully reach the mobile.

$P_1$ has already been calculated earlier and is equal to:

$$P_1 = e^{-G} \tag{4.16}$$

The values of packet error rates (PER) of different Reed-Solomon codes have been obtained in Chapter 3 and will be used here to calculate $P_2$ using the equation:

$$P_2 = 1 - PER \tag{4.17}$$

The value of $P_3$ is a function of the bit-rate on the forward channel, the coding used and the size of the acknowledgement packets. Therefore, we will evaluate the performance of the access scheme for three different values of $P_3$: 100, 95 and 90 percent.

The probability of success can then be written as:

$$\text{Prob (success)} = e^{-G} \times (1 - PER) \times P_3 \tag{4.18}$$

But this probability of success is also equal to S/G, the ratio of the number of successful packets per $\tau$ seconds to the total number of transmission attempts per $\tau$ sec. time. Hence,

$$\frac{S}{G} = e^{-G} \times (1 - PER) \times P_3 \tag{4.19}$$

so that

$$S = G\, e^{-G} \times (1 - PER) \times P_3 \text{ succ. packets / } \tau \text{ sec.} \tag{4.20}$$

If we are to compare the performance of the codes, expressing $S$ in terms of number of successful packets per slot time is not practical because the packet size is different for every code. It would be better then to express $S$ in number of successful packets per second knowing that all packets carry about the same amount of information. The equation above becomes:

$$S = \frac{G\, e^{-G} \times (1 - PER) \times P_3}{\tau} \text{ successful packets/sec.} \tag{4.21}$$

If the channel was ideal, the number of packets that could have been transmitted per second with a perfect synchronization among users is $4800/L$, where $L$ is the number of information bits per packet. Hence, the normalized throughput $\gamma$ of our channel is:

$$\gamma = \frac{S}{4800/L} = G\, e^{-G} \times (1 - PER) \times P_3 \times \frac{L}{4800 \times \tau} \tag{4.22}$$

Note that, when the guard time is zero, $\frac{L}{4800 \times \tau}$ is nothing but the code rate $R$ of the code that is being used. In Figure 4.10, $\gamma$ is plotted as a function of G for the RS (31,25) code in the Galois field GF(32) with no guard time and for three different values of $P_3$.

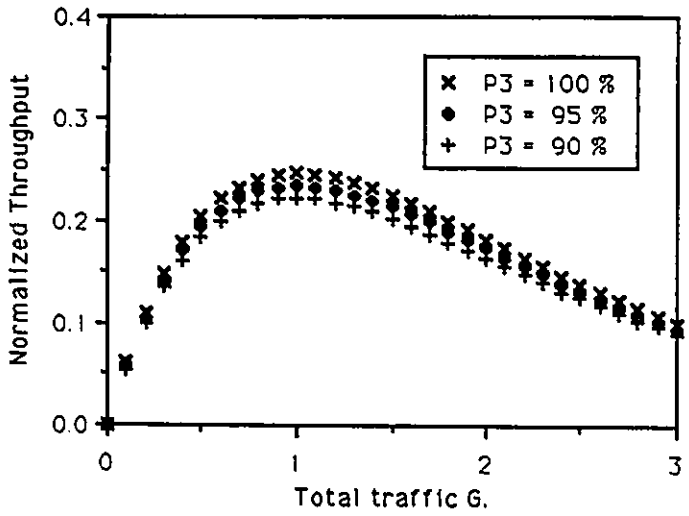


Figure 4.10: The throughput of slotted ALOHA over the LMSAT channel.

The equation for the average packet delay D is the same as earlier:

$$D = \tau[1.5 + \text{prop.} + E(0.5 + \text{prop.} + \frac{K+1}{2})] \text{ sec.} \tag{4.23}$$

But this time the average number of transmissions per packet is given by:

$$E = \frac{G}{S} - 1 = \frac{1}{e^{-G} \times (1 - PER) \times P_3} - 1 \tag{4.24}$$

Note that the following assumptions have been made here:

- The delay D is calculated from the time the packet is ready for transmission (i.e. encoding and interleaving are completed).
- The packet is de-interleaved during its reception so that no additional delay is taken into account.
- The decoding delay is negligible.

The assumptions above will also hold for the study of the other access schemes later in this chapter.

As in the case of the ideal channel, the equations for $\gamma$ and $D$ both as functions of G permit us to draw the throughput-delay performance of the access scheme. In Figure 4.11, we draw the curves corresponding to different code rates for codes chosen from the Galois field GF(32). The curves are drawn assuming no guard time, $P_3 = 1$, and K = 15. As mentioned earlier, the values for the packet error rates are taken from Table 3.2 in Chapter 3.

As we can see, the maximum throughput increases with the code rate up to a certain point around R = 0.8 and then decreases. This can be interpreted as follows:

- When the code rate is too high, the FEC code is unable to correct most of the packets because of its poor error correcting capability. Therefore, a high number of transmissions is needed for every packet which results in a low throughput.
- When the code rate is too small, the number of transmissions needed for every packet is small but the high redundancy in each one of them wastes most of the available throughput.

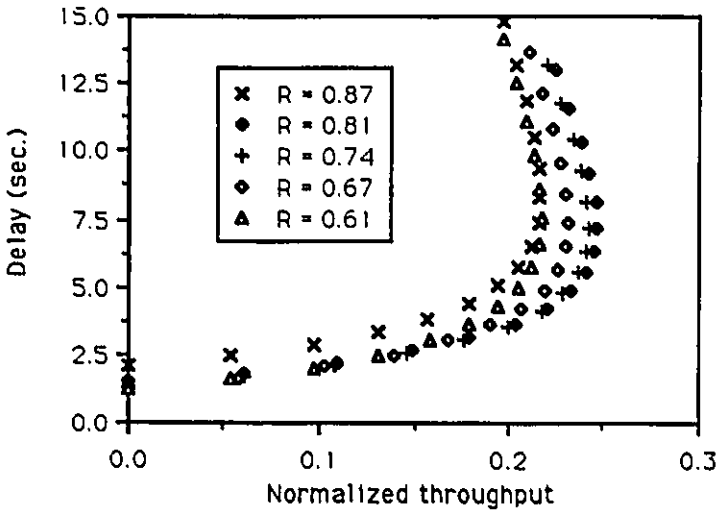


Figure 4.11: The effect of the code rate on the performance of slotted ALOHA in the LMSAT channel for RS codes over GF(32).

In Figure 4.12, the same curves are drawn but this time for codes in the Galois field GF(256) where no interleaving is necessary. We can see here again that the optimal value of the code rate is around 0.8. We can note also comparing Figure 4.11 to Figure 4.12 that no meaningful improvement is gained using longer codes providing that symbol interleaving is employed.

In Figure 4.13 the effect of the guard time is shown. The code used is the shortened RS (157,125) code from GF(256) which has a code rate of 0.8. Here also, the forward channel is assumed to be ideal ($P_3 = 1$). Finally, the same code is used in Figure 4.14 where the effect of $P_3$ is evaluated.

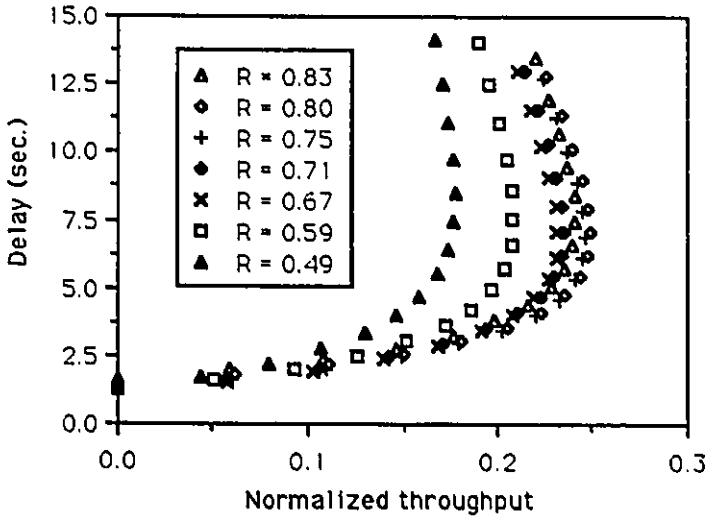


Figure 4.12: The effect of the code rate on the performance of slotted ALOHA in the LMSAT channel for RS codes over GF(256).

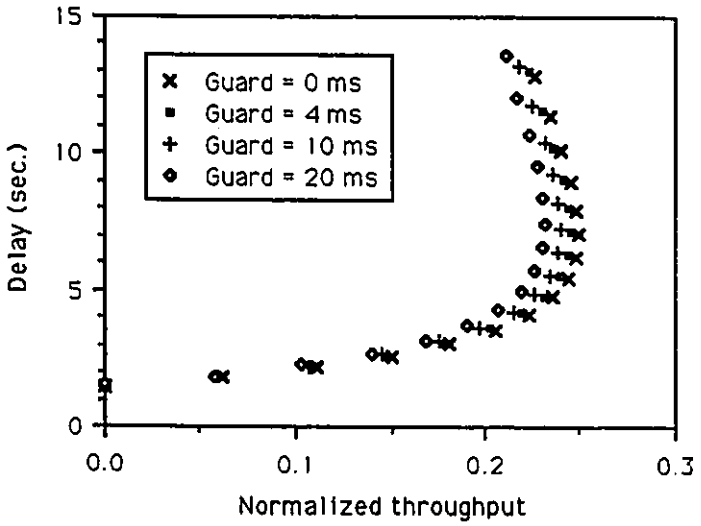


Figure 4.13: The effect of the guard time on the performance of slotted ALOHA in the LMSAT channel.

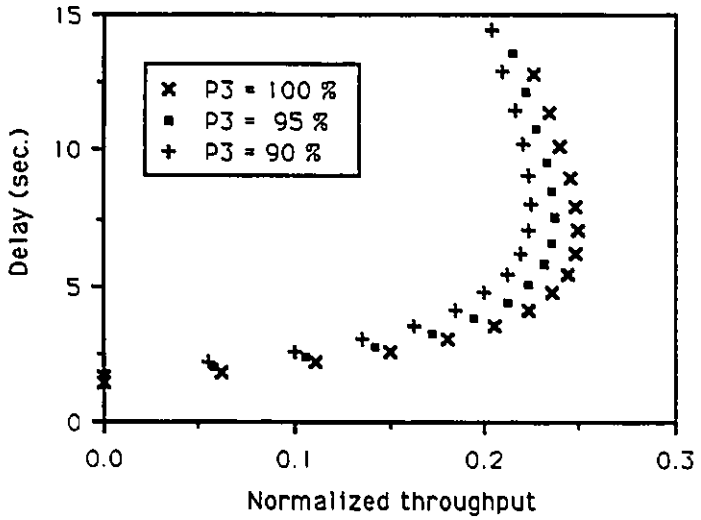


Figure 4.14: The effect of the packet error rate in the forward channel on the performance of slotted ALOHA in the LMSAT channel.

## 4.6 Frequency Hopping Multiple Access

### 4.6.1 Description of the Protocol

In ALOHA-type schemes, when two or more users transmit their packets simultaneously, a collision occurs and the destroyed packets have to be retransmitted. In frequency-hopping-multiple-access (FHMA), coding is used in order to recover from partial collisions so that fewer retransmission are needed and packets are delivered with shorter delays. The scheme requires a set of frequency channels $(f_1, f_2, ..., f_n)$. When a packet arrives at a user, it is first encoded using a powerful error correcting code. The resulting packet is then divided into $s$ small subpackets that are transmitted successively, each subpacket being transmitted on a chosen frequency. At the receiver, if the number of subpackets that have experienced collisions is small, the FEC code will correct the errors and the original packet will be recovered. Otherwise, a retransmission is required [40] [41]. Figure 4.15 shows the different steps needed for the transmission of a packet.

The choice of the frequency channel for every subpacket can either be randomly done or be fixed for each user or group of users. In the first case the scheme is called random-frequency-hopping-multiple-access (RFHMA). For the second case, the choice of the frequency patterns is usually done in such a way that if the transmission of one user overlaps with that of another user from a different group, at most one subpacket will collide. When the second user is from the same group, the previous condition still holds unless both transmisions start at the same time, in which case the packets mutually destroy each other completely. More details about how to construct such frequency patterns can be found in [41].

In synchronous FHMA the frequency channels are slotted and the subpackets are

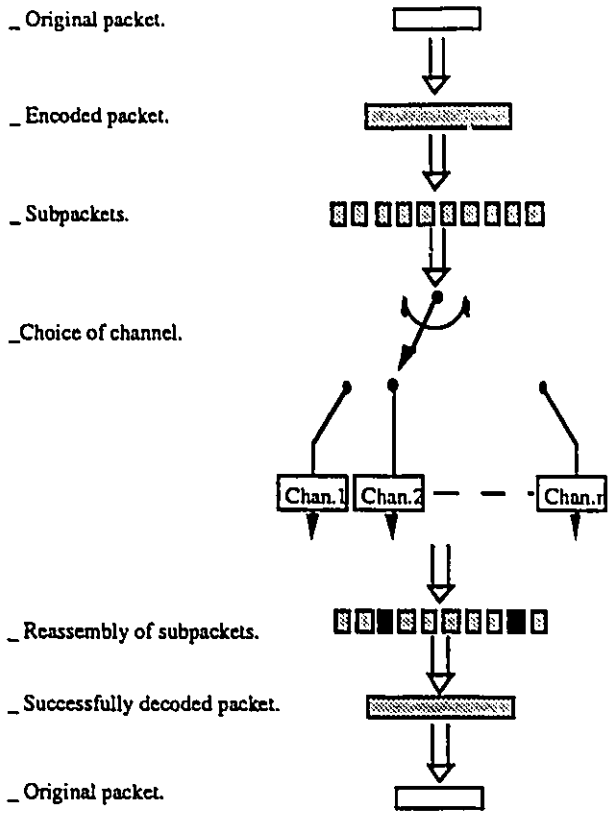


Figure 4.15: The different steps in frequency-hopping-multiple access.

to be transmitted within the slot limits as shown in Figure 4.16. In this case, as in slotted ALOHA, only full collisions can occur. The example in Figure 4.16 shows transmissions of two users in a system using five frequency channels ($F_1, F_2, ..., F_5$). Each packet has been divided into 10 subpackets (s=10) and we see that the sixth packet of the first user has collided with the fourth packet of the second. In asynchronous FHMA, the user can transmit whenever he is ready and partial subpacket collisions may result.

Instead of transmitting the different subpackets successively on different frequency channels, one could use a single frequency but transmit the subpackets not succes-

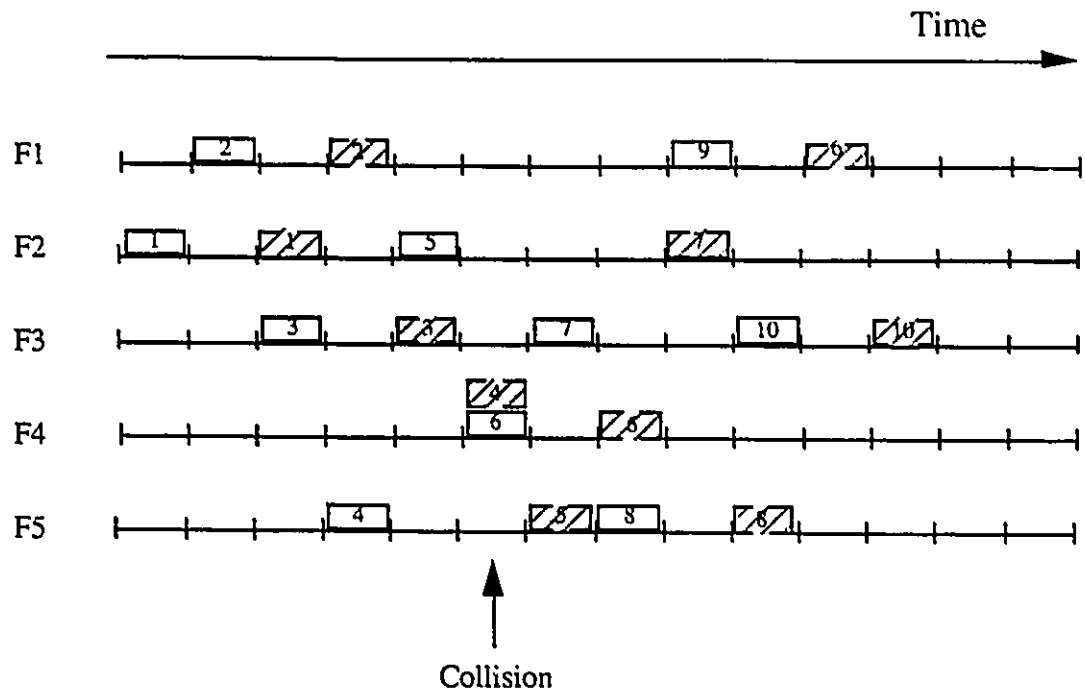


Figure 4.16: Synchronous RFHMA with n=5 and s=10.

sively but at time instants that are distant from each others. The scheme is then called time-hopping-multiple-access (THMA). As in FHMA, the time duration separating the subpackets can either follow a fixed pattern specific to each user or group of users or chosen randomly (RTHMA). Again in this case, both synchronous ( slotted channel) and asynchronous (unslotted channel) schemes are possible. In Figure 4.17, we show the transmission of two users using a synchronous THMA scheme with s=6 (6 subpackets per packet).

When only few frequency channels are available, FHMA cannot perform efficiently and it is possible then to construct an access scheme that combines FHMA and THMA. In this case, The transmission of the subpackets is not continuous but is decided as in the THMA schemes.

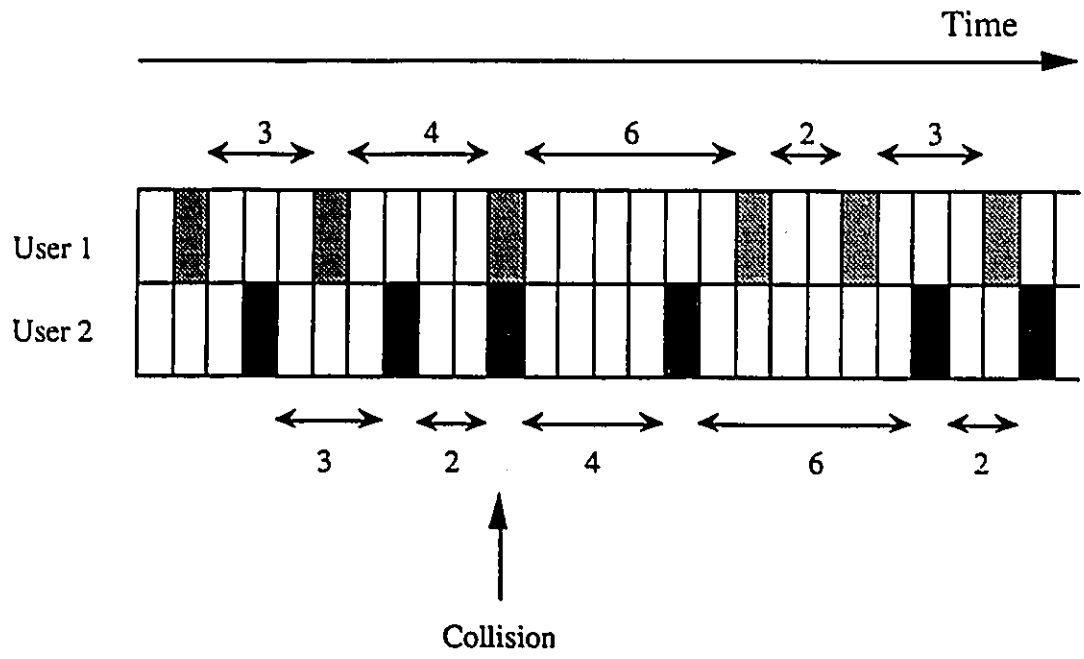


Figure 4.17: Synchronous RTHMA (s=6).

In the following, we only consider synchronous RFHMA. Reed-Solomon codes are used for the error correction. In the next section, we discuss in detail the coding mechanism chosen for this particular protocol.

### 4.6.2 Coding Mechanism

The data packet is first encoded by an (n,k) Reed-Solomon code over GF($2^q$). The resulting packet, which has $B$ codewords, is then fed into a long shift register that constitutes a symbol interleaver of $n$ columns and $B$ rows as shown in Figure 4.18.

Each cell on the figure contains one q-ary symbol. The columns of the interleaver are then grouped to form the $s$ subpackets. Each of these subpackets contains an

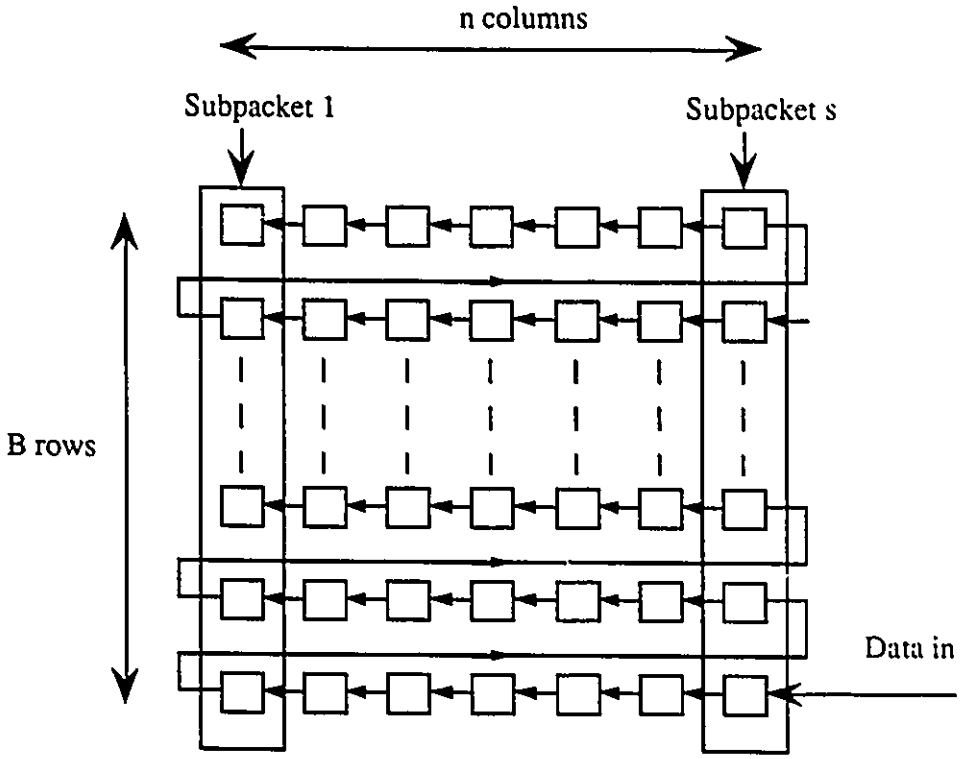


Figure 4.18: Symbol interleaving in RFHMA using Reed-Solomon coding..

integral number of symbols from each codeword. In the particular example of Figure 4.18 the subpackets have a single column. When a subpacket arrives in error at the receiver due to a collision or noise (this is detected by parity bits appended to the subpacket), the symbols that constitute this subpacket are erased. Thus, at the end, all the codewords will have the same number $e$ of erased symbols. The decoder will be able to recover the original packet if:

$$e \leq n - k \tag{4.25}$$

In the case of a decoding failure, no acknowledgement (or a negative acknowledgement) will be sent to the mobile which will then have to attempt a retransmission.

### 4.6.3 Performance Analysis

#### Performance in an ideal channel

The performance of the RFHMA multiple access is obtained by computer simulations. In Figure 4.19, we show the results for three different codes which are RS (15,5), RS (15,6) and RS (15,7). Every packet is divided into 15 subpackets, each one of them containing a single four bit symbol from each of the 15 codewords. The number of frequency channels has been chosen to be 15. The overhead due to the parity bits that are appended to the subpackets for error detection has been neglected. The reason for this is that if an error is not detected at the subpacket level, it will be detected at the packet level by the RS decoder. Thus, only a few number of parity bits (i.e 5 or 6 bits) is sufficient.

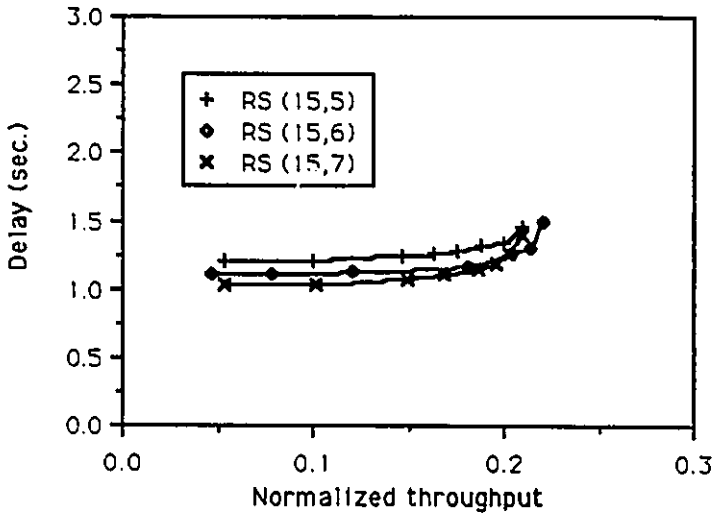


Figure 4.19: Effect of the code rate on the performance of synchronous RFHMA in an ideal channel.

The normalized throughput is calculated as follows. Let $t_{sim}$ be the simulation

time and $N$ the total number of packets that have been successfully transmitted during this time. The total number of 1000-bit messages that could have been transmitted using the 15 channels in the case of a perfect scheduling is:

$$15 \times \frac{t_{sim}}{1000/4800} \text{ packets.} \tag{4.26}$$

The normalized throughput is then

$$\gamma = \frac{N}{\frac{15 \times t_{sim}}{1000/4800}} \tag{4.27}$$

The value of the delays obtained can be easily verified in the case of light traffic. For example, for the (15,7) code, the size of the slots is 144 bits. The total delay in light traffic is, on the average equal to one half timeslot (to wait for the beginning of a slot) plus fifteen timeslots (there are fifteen subpackets) plus the round trip propagation delay. This equals to $\frac{15.5 \times 144}{4800} + 0.55 = 1.015$ seconds which corresponds to the value on the figure. The small difference noticed between the three curves can be explained by the fact that, when high code rates are used, the slot size is smaller and so is the average time the user has to wait for the beginning of a new slot. Besides this small difference, we can see that the performance of the three codes is not very different. This can be explained by the fact that, when the code rate is high, the subpackets are shorter in length compared to the case of a lower code rate, and thus have less chance to be hit by a fade, this is balanced by a greater correction capability in lower rate codes. What can be said about the scheme is that it performs well in low traffic and the packets are usually successfully received in the first attempt. However, at a certain point, the number of missing subpackets becomes greater than the error correction capability of the code and the performance degrades dramatically driving the system into an unstable state and resulting in infinite delays.

In Figure 4.20, we plot the performance of the RS (15,5) and RS (30,10). Both have the same code rate but for the second code we choose to divide the packet into

30 subpackets. As we can see, the performance is almost the same for both codes and thus, we can conclude that there is no need to use long and complex code lengths in this case.

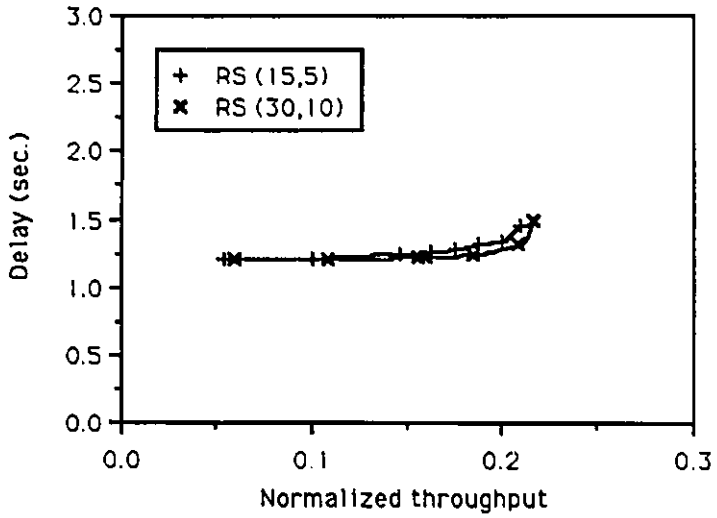


Figure 4.20: Effect of the code length on the performance of synchronous RFHMA in an ideal channel.

In Figure 4.21 we evaluate the effect of the guard time. The RS (15,5) code is used with three different values of the guard time: zero, 4 and 10 ms.

### Performance over the LMSAT channel

Fig 4.22 shows the performance of the RS (15,5) code when the LMSAT channel model is used instead of the ideal channel. We can see a significant degradation in the performance of the protocol.

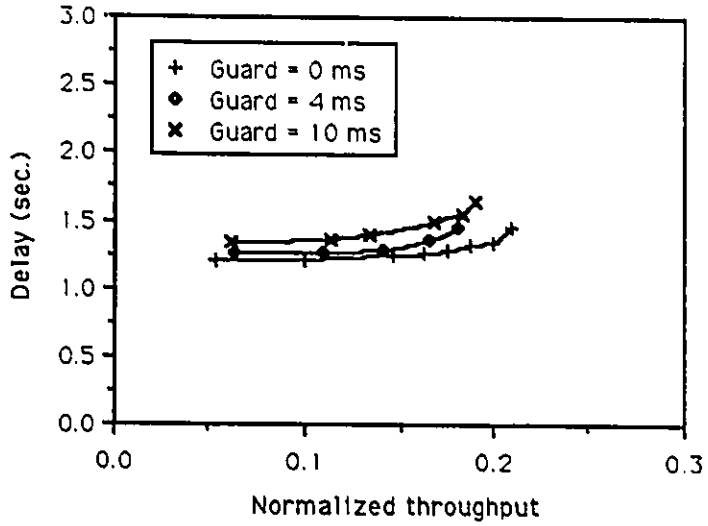


Figure 4.21: Effect of the guard time on the performance of synchronous RFHMA in an ideal channel.

In order to improve this performance, we try to make use of the channel state information (CSI) that we assume is available to the mobile by monitoring the received power on the forward channel. Now, the user will not be allowed to begin the transmission of a subpacket when he knows he is in a deep fade state. Because the subpackets are relatively short in length, there is a good chance in this case that the channel will stay in a non-fade state during their transmission. Figure 4.23 shows the improvement when this modification is made to the scheme.

Since shorter subpackets can lead to better results with this new scheme, we decided to run a simulation whith the RS (30,10) code where the message is divided into 30 short subpackets. The obtained result is shown in Figure 4.24.

It should be noted however that, because of the higher number of subpackets, the advantage seen in Figure 4.24 will definetely be reduced when the the guard time and

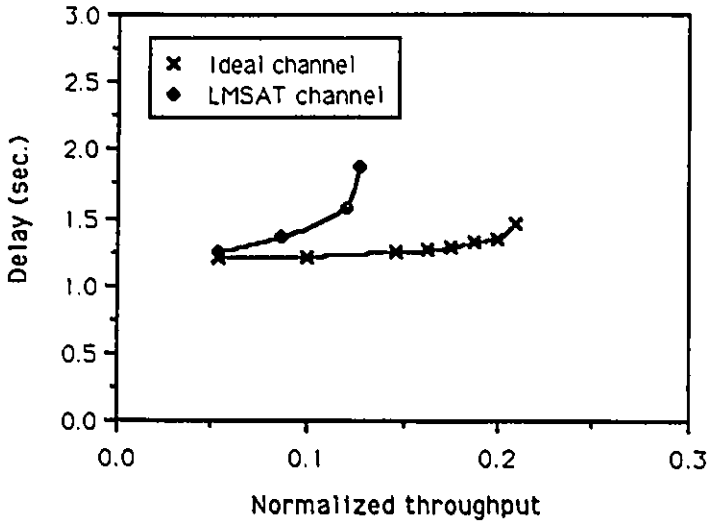


Figure 4.22: Performance of synchronous RFHMA in the LMSAT channel.

the overhead appended to each subpacket are taken into consideration. Figure 4.25 shows the effect of the guard time when the RS (30,10) code is used.

The comparison of the performance of RFHMA and slotted ALOHA is done after the analysis of all four schemes.

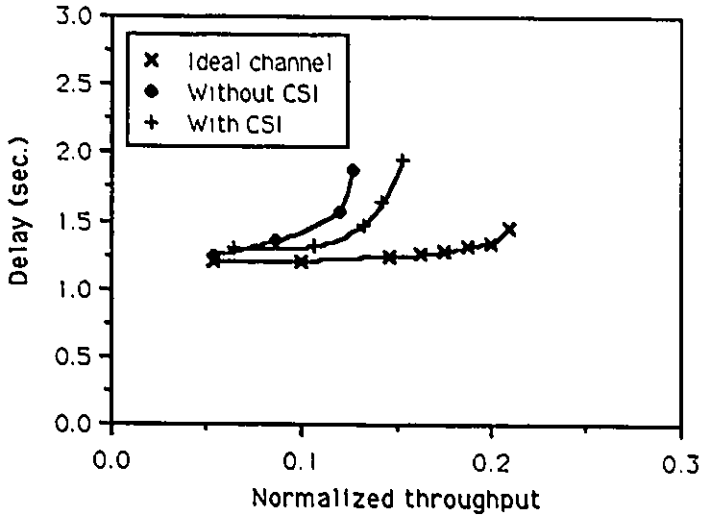


Figure 4.23: Using the channel state information to improve the performance of synchronous RFHMA in the LMSAT channel.

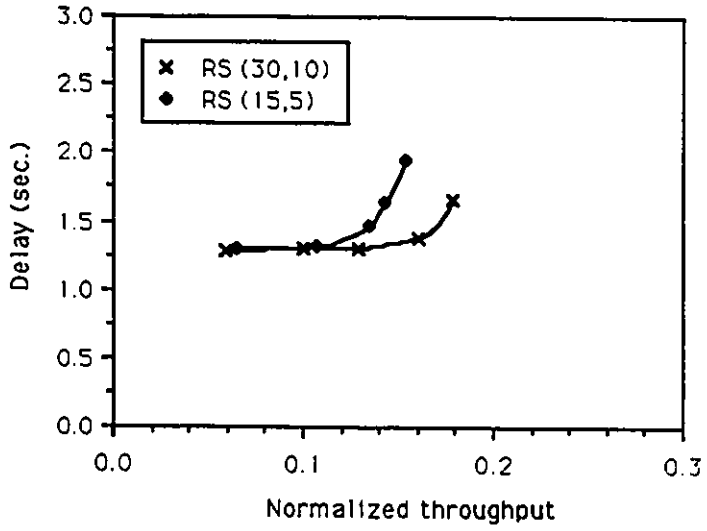


Figure 4.24: Effect of subpacket length on the performance of synchronous RFHMA using CSI in the LMSAT channel.

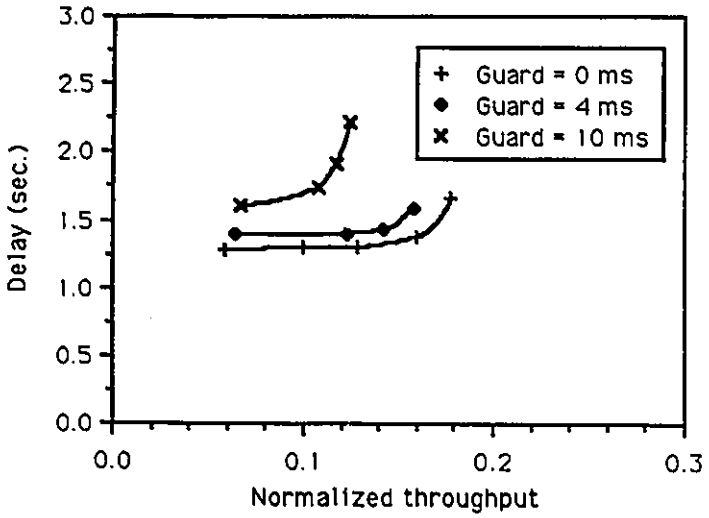


Figure 4.25: Effect of the guard time on the performance of synchronous RFHMA in the LMSAT channel using the RS (30,10) code.

## 4.7 Adaptive Mobile Access Protocol (AMAP)

### 4.7.1 Description of the Protocol

The adaptive mobile access protocol is a scheme that has been proposed by the Jet Propulsion Laboratory (JPL) for NASA's MSAT-X experiment. This scheme belongs to the family of centrally controlled demand access schemes. It is the hub that acts here as the central controller (CC) or the network management center (NMC). The protocol functions as follows.

The satellite channel bandwidth is frequency divided into $N$ equal size channels. Out of these $N$ channels, $N_r$ are reservation channels and the remaining $N_d$ are for data transfer. This channel partitioning is not fixed but is continuously updated according to the current traffic load in order to optimize system performance. The decision is made by the NMC which periodically informs the mobiles about the current channel assignment. When a mobile has a message to transmit, it first sends a reservation message on one of the $N_r$ reservation frequencies using slotted or unslotted ALOHA and waits for an answer from the hub. The reservation packet consists of an origin and destination addresses and the length of the message to be transmitted. If an acknowledgement is not received within a time out period, the mobile retransmits its request after a random delay. Upon receipt of this request, the NMC assigns to the user a time slot on the data channel with the smallest backlog. This information is sent to the user in an acknowledgement (or assignment) packet which contains the identity of the mobile, the id of the data channel to be used and a holding time $d_{ll}$. The mobile then initiates transmission on the specified frequency $d_{ll}$ seconds after the receipt of this acknowledgement from the central controller and waits for a positive acknowledgement. Again, the reservation process has to be repeated if no ACK is received within the time out period. The scheme is summarized in the flowchart given in Figure 4.26.

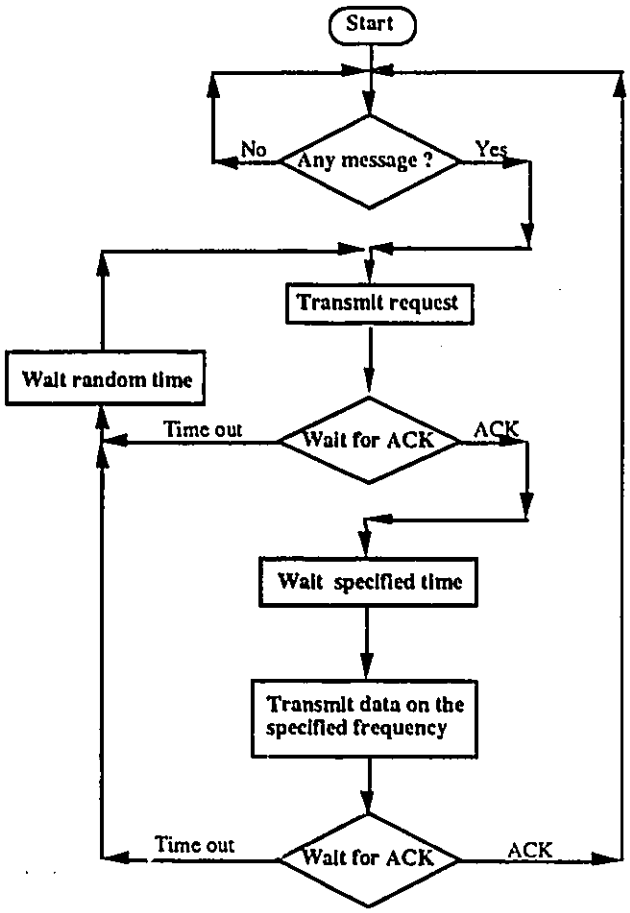


Figure 4.26: Flowchart describing the AMAP protocol.

In our study, we assume the following:

- The access scheme for the reservation channels is slotted ALOHA.
- As for the previous protocols, the size of the message is fixed and is 1000 information bits.
- The size of the request packets is 100 information bits.
- The total number of channels $N$ is 10.

### 4.7.2 Performance Analysis

In [42], the performance of AMAP in the case of ideal channels has been studied. The main idea in the analysis is to write the average total delay D as follows:

$$D = d_r + d_H + d_T + d_p \tag{4.28}$$

Where:

- $d_r$ is the average time needed to successfully make a reservation.
- $d_H$ is the average holding time defined above.
- $d_T$ is the time needed for the transmission of the message.
- $d_p$ is the round trip propagation delay (from mobile to hub to mobile).

The average reservation time $d_r$ is simply obtained from the analysis of the slotted ALOHA access scheme. In order to evaluate $d_H$, it is useful to picture the NMC as a queue which processes the requests coming from the mobiles as shown in Figure 4.27. The servers of this queue are nothing but the $N_d$ data channels and, because the arrival of the requests at the NMC can be approximated as a Poisson process, the system is equivalent to an M/G/s queue where s (the number of servers) is equal to $N_d$ in this case.

In the following we evaluate the performance of the protocol using computer simulations.

#### Performance in an ideal channel

In an ideal channel, no coding is necessary and the sizes of the reservation and the data packets are 100 and 1000 bits respectively. The results of the simulations are

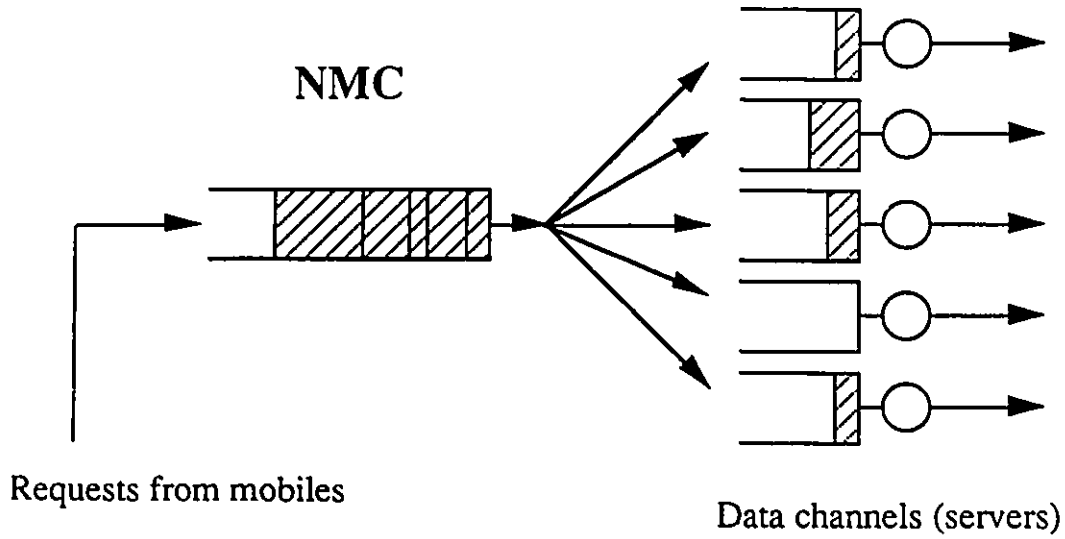


Figure 4.27: The equivalent M/G/s queue in the AMAP protocol.

shown in Fig 4.28. The notation $(x, y)$ represents the configuration with $x$ requests channels and $y$ data channels.

When the traffic is light, few collisions occur on the request channels and the global queue is almost always empty. Therefore the total delay in this case is roughly equal to the time needed to transmit both the request and data packets plus twice the round trip propagation delay which gives:

$$D \approx \frac{(100 + 1000)}{4800} + 2 \times 0.55 \approx 1.33 \text{ seconds.} \tag{4.29}$$

This result can be verified on the figure.

As the traffic increases, the number of successful requests increases and so is the queuing delay making the utilization of the data channels approach to 100%. When this happens, the normalized throughput reaches the value of $\frac{x}{x+y}$. We can verify in the figure that the $(5, 5)$ configuration achieves a maximal throughput of 50%, $(4, 6)$ a throughput of 60% and so on. In the case of the $(2, 8)$ configuration, the rate of

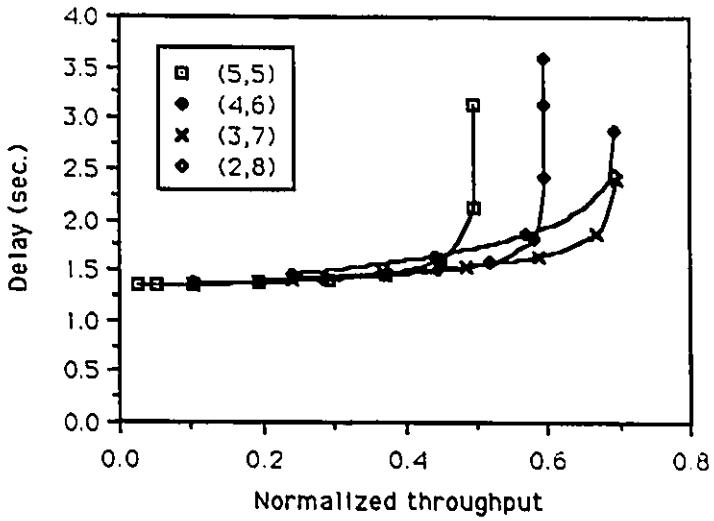


Figure 4.28: Performance of AMAP in an ideal channel as a function of the number of request and data channels.

successful requests is limited by the high probability of collision so that the utilization of the data channels does not reach its potential maximum and an 80% efficiency is not attained.

The results in Figure 4.28 also show how the adaptive scheme works. At light traffic, the number of request channels should be high in order to avoid collisions and keep the delay small. As the traffic increases, the data channels become saturated and it is necessary to shut down some request channels to reduce the number of reservations while at the same time increase the number of message channels to expedite clearing up the backlog. Note that the degradation in delay is not so great if the adaptivity of the protocol is not implemented and if the fixed $(3, 7)$ configuration is chosen. This might not be the case however if the size of the messages is variable or if the channels are not error free.

The simulation is repeated with values of 4 ms and 10 ms for the guard time and the results are shown in Figure 4.29. In this figure the final curves assuming an adaptive

scheme are shown. Note that the same guard times are used between slots in the reservation channels and between messages on the data channels.

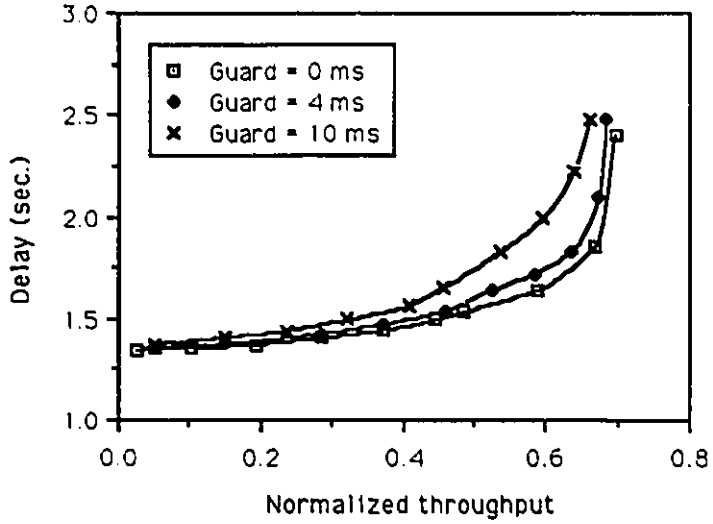


Figure 4.29: Performance of AMAP in an ideal channel as a function of the guard time.

### Performance in the LMSAT channel

When the channel is not ideal, the use of coding is mandatory in order to achieve good performance and reliable communication. Reed-Solomon codes will again be used here to encode the data packets. As concluded in Chapter 3, the use of an RS code with a long code length is equivalent to the use of one with a shorter code length if symbol interleaving is used (i.e. equal code rate will result in approximately equal packet error rate). Therefore, simulations will be run for different code rates but from only the GF(256) Galois field where interleaving is not necessary.

As for data packets, reservation packets should also be encoded because otherwise a packet error rate of about 20% results, which is clearly inacceptable. The problem

with these short request packets is that, because of their small size, the occurence of a fade during their transmissions severely damages them. Simulations show that the use of an RS(31,25) code bring the packet error rate down to 8.4% which is not bad when compared to the uncoded case. However, the length of the packets is now 155 bits instead of 100 bits. In the following discussion this code is assumed for the reservation packets.

In Figure 4.30, we show the result of the simulation for different channel assignment configurations. The code used is the RS(157,125) shortened code which has a code rate of 0.8. For this simulation, the return channel (from hub to mobiles) which carries the acknowledgements and the assignments has been assumed error free, the more realistic case where this channel is not ideal will be considered later. The first result that can be noticed in the figure is the decrease in the maximal throughput which has dropped from about 70% in the case of an ideal forward channel to around 45% now. We can also note that, as for the case of the ideal channel, the use of a fixed configuration with 3 request channels and 7 data channels will not lead to a significant performance degradation compared to the adaptive scheme. The remaining results in this section assume however that the adaptive scheme is implemented.

The curves in Figure 4.31 show the result of the simulations for three different code rates (for the data packets). As we can see, the range of optimum values is between 0.7 and 0.8. Smaller values of this code rate result in a high redundancy in the packets which reduces the efficiency of the scheme. The effect of the guard time is illustrated in Figure 4.32. In Figure 4.33, different values for the error rate of the acknowledgement and assignment packets on the forward channel are assumed and we can clearly see their effect.

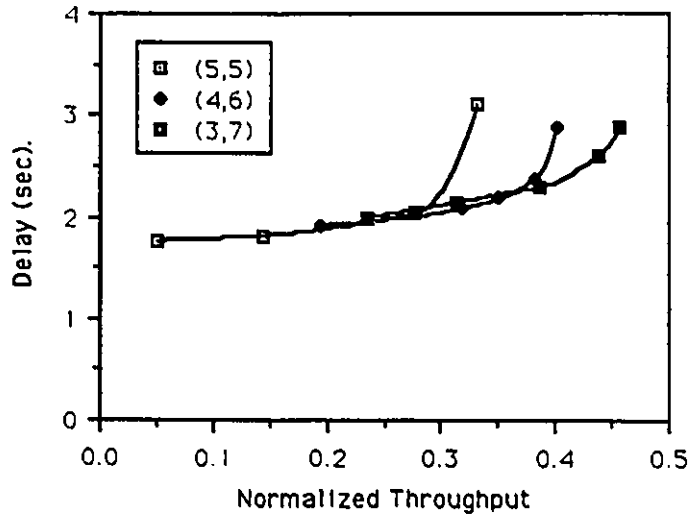


Figure 4.30: Performance of AMAP in the LMSAT channel using RS (157,125) code as a function of the number of request and data channels.

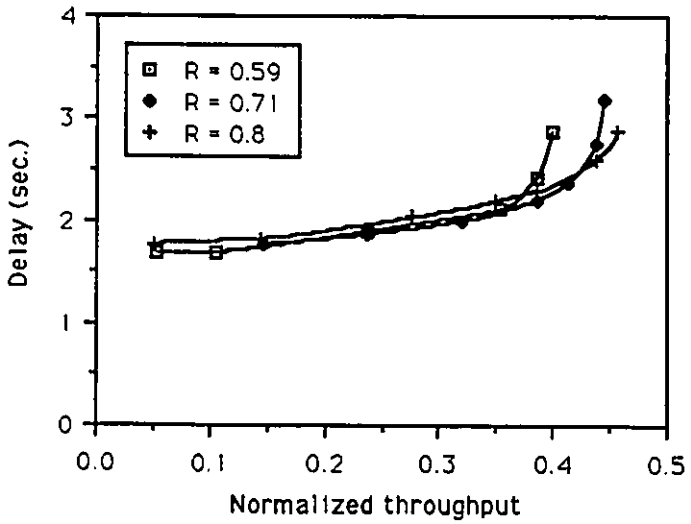


Figure 4.31: Effect of the code rate (in the data channels) on the performance of AMAP in the LMSAT channel.

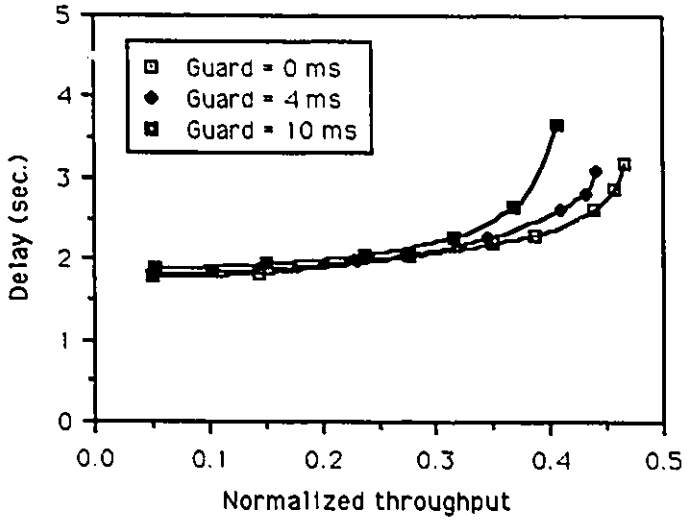


Figure 4.32: Effect of the guard time on the performance of AMAP in the LMSAT channel.

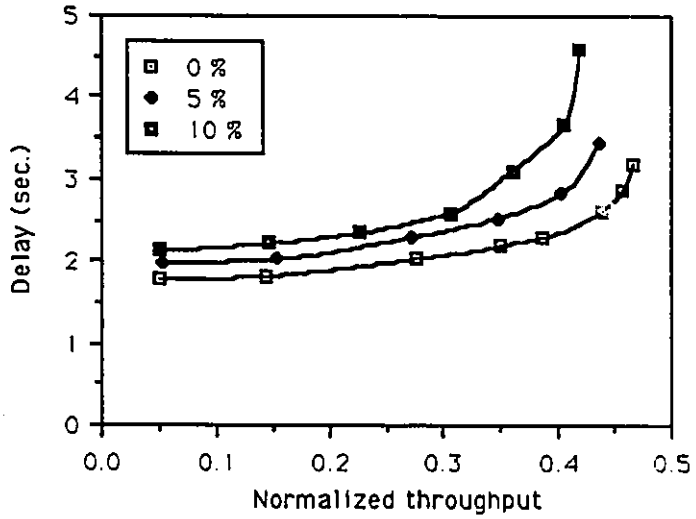


Figure 4.33: Effect of the PER on the forward channel on the performance of AMAP in the LMSAT channel.

## 4.8 Combined AMAP-RFHMA protocol

### 4.8.1 Description of the protocol

Examining the results of the two previous schemes, we notice that RFHMA achieves a low delay but limited throughput whereas the AMAP protocol gives a good throughput but higher delays. One can then think of combining these two protocols in an attempt to create a new scheme that can achieve short delays and high throughput. The scheme that is tried here functions as follows.

The channel bandwidth is divided into $N$ equal size channels. These $N$ channels are then separated into two groups. The first group consists of $N_1$ channels and the second one of $N_2$ channels. When a message arrives at a user, it is first encoded and divided into subpackets. The user then sends the subpackets employing RFHMA using the first $N_1$ channels. If an acknowledgement is received, the message has been received at the other end in a short delay. Otherwise, if at least one subpacket is received correctly, the hub recognizes the id of the mobile (this id is assumed to be appended to every subpacket) and reserves for this user a time slot on one of the $N_2$ reservation channels as in the AMAP protocol. In the following, we call the first $N_1$ channels the RFHMA channels and the second $N_2$ channels the AMAP channels.

### 4.8.2 Performance Analysis

#### Performance in an ideal channel

The performance of this scheme has been obtained by computer simulations. In the first simulation, the channel is assumed to be error free. The code used for the transmission using RFHMA is the RS(30,10) code with interleaving as explained in Section 4.6. No coding is necessary for the transmission on the AMAP channels since they are free of error and of collisions. The results of the simulation are shown in

Figure 4.34.

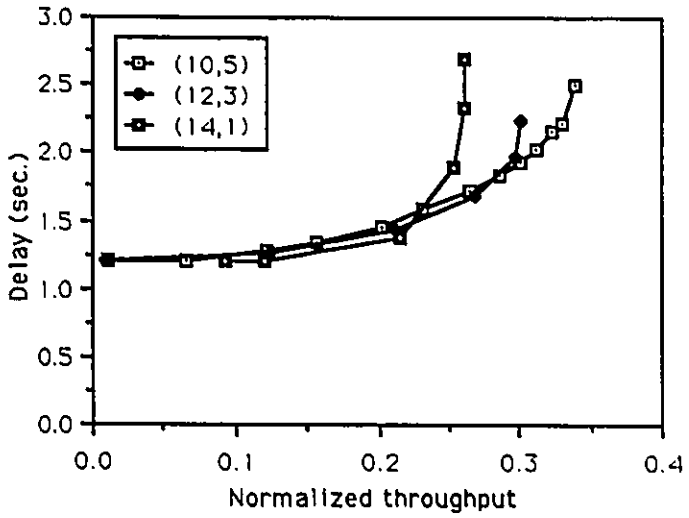


Figure 4.34: Performance of AMAP-RFHMA in an ideal channel.

On this figure, the notation $(x, y)$ represents the configuration where $N_1 = x$ and $N_2 = y$. Looking at this figure, we can make the following remarks.

- When a small number of AMAP channels is used, most of the packets require only one transmission attempt since the number of RFHMA channels is high. However, at one point, when the channels used in random access attain their maximal throughput, the number of packets that need retransmisions grow rapidly which result in the saturation of the few AMAP channels and thus in a rapid increase in the queuing delay and therefore in the overall average delay.

- When a configuration with a high number of AMAP channels is used, the RFHMA channels will saturate at an earlier stage but, because of the larger number of AMAP channels, the increase in the queuing delay will be smoother than in the previous case. The higher throughput can be explained by the fact

that more channels (the AMAP channels) are used in an efficient way in this case.

### Performance in the LMSAT channel

The same simulation is run in the case of the LMSAT channel. In this case, the RS(31,23) code is used for the transmission over the reservation channels since they are no longer error free. The results are shown in Figure 4.35.

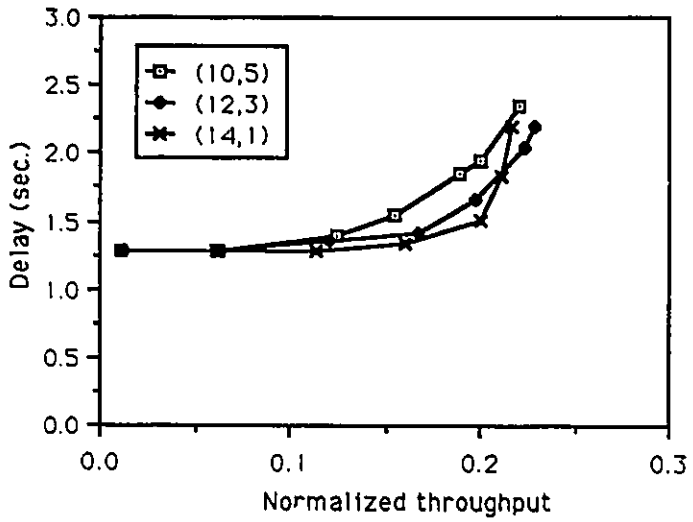


Figure 4.35: Performance of AMAP-RFHMA in the LMSAT channel.

As we can see, the configuration (10,5) with the high number of AMAP channels have somewhat lost the advantage it had over the other configurations. This can be explained by the fact that the efficiency of the transmission over the AMAP channels has been affected by:

1. The coding used over these channels (the code rate is around 75%).

2. The residual packet error rate which is around 11%.

In conclusion, we can say that this combination of RFHMA and AMAP does not yield a throughput comparable to that of AMAP but has extended the efficiency of RFHMA because of the removal of the traffic generated by retransmissions from the randomly accessed channels.

## 4.9 Performance Comparison

In this section we make a comparison of the different access schemes studied above. We first compare the performance of the schemes assuming error free channels and no guard time. The curves corresponding to the four protocols are shown in Figure 4.36.

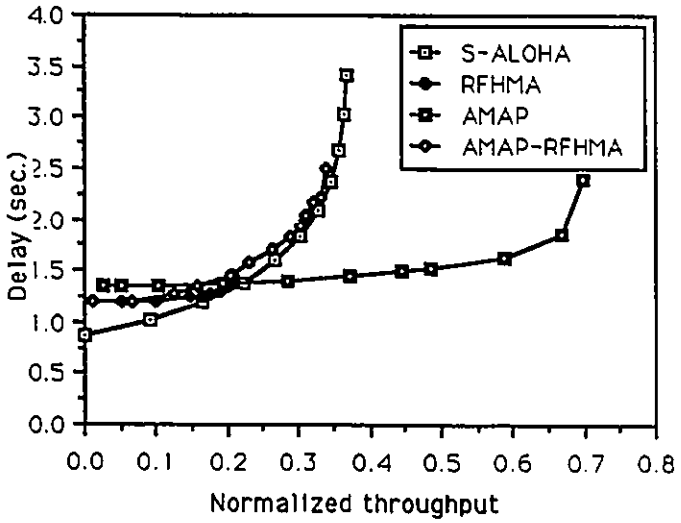


Figure 4.36: Performance comparison of the four access schemes in an ideal channel.

As we can see, it is the slotted ALOHA protocol that has the shortest delays in low traffic and the AMAP scheme that achieves the highest efficiency. The two other

protocols (RFHMA and AMAP-RFHMA) do not show any particular advantage. The difference in delays between these schemes and slotted ALOHA is explained by the fact that, because of coding, the packets are three times longer in the frequency hopping schemes and thus need more time to be transmitted.

The performance of the schemes in the LMSAT channel is shown in Figure 4.37. Again here, no guard time is taken into consideration and the acknowledgement and assignment channel (from hub to mobiles) is assumed error free. It can be seen that the performance of the slotted ALOHA protocol has been severely affected whereas the frequency hopping schemes retain their short delays in light traffic because of their powerful coding. It should be noted here that the curves for RFHMA and AMAP-RFHMA are the ones where the channel state information (CSI) is used, i.e. the mobile does not begin the transmission of a subpacket if the propagation conditions are not good. The use of the same method for slotted ALOHA and AMAP would not result in a significant improvement because the packets are big in size and waiting for good propagation conditions before transmission would not reduce significantly the probability of encountering a fade later on.

The curves plotted in Figure 4.38 show the performance of the schemes in the LMSAT channel with a guard time of 4 ms and a packet error rate of 5% for the acknowledgements and assignments sent by the hub to the mobiles.

In conclusion, we can make the following remarks:

- The slotted ALOHA scheme is simple to implement but relatively inefficient and the use of more elaborate schemes can indeed improve the performance of the system.
- Random frequency hopping achieves the shortest delays in light traffic but is

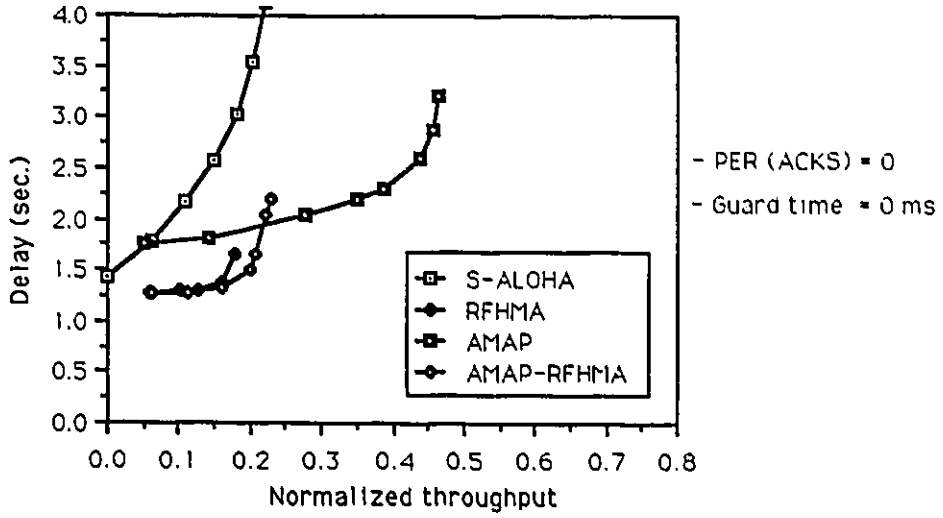


Figure 4.37: Performance comparison of the four access schemes in the LMSAT channel assuming no guard time and error free forward channel.

limited in throughput and is sensitive to the value of the guard time. It should also be said that this scheme requires a high degree of complexity in coding and interleaving.

- The performance of AMAP shows a significant improvement in channel utilization over the random access schemes, its main disadvantage is the associated longer delays even in low traffic.

- The results of the protocol that combines AMAP and RFHMA show an improvement of the efficiency of RFHMA but this improvement may not be sufficient to justify the added complexity of the implementation of such a scheme. An alternative method would be to access the channels in RFHMA at low traffic and when the traffic exceeds a certain threshold switch to the AMAP protocol.

It is important to remind here that the results shown in this chapter assume that the messages are fixed in size. Obviously, this is not true in reality and therefore the efficiency of schemes that use fixed size slots will degrade because of the need to

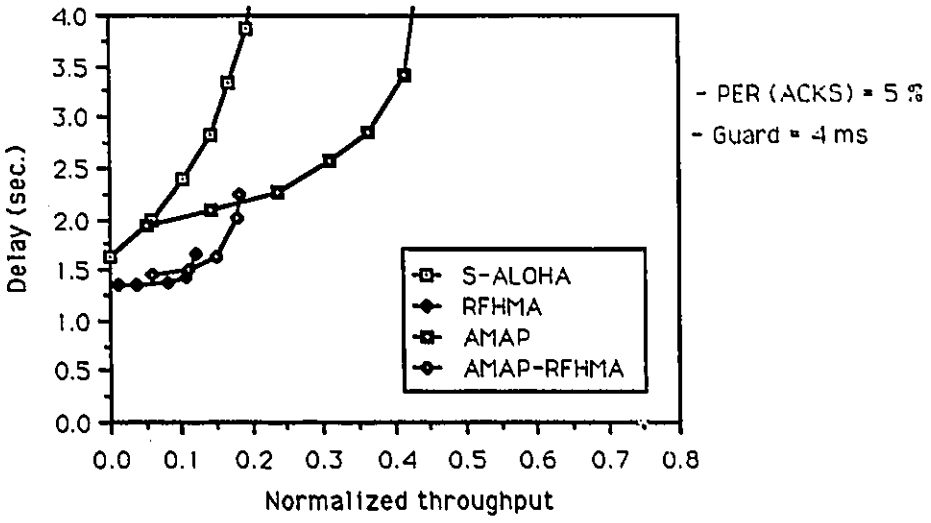


Figure 4.38: Performance comparison of the four access protocols in the LMSAT channel considering a 4 ms guard time and a 5% packet error rate on the forward channel.

fill portions of the packets with blanks. Another consequence of this is the existence of multi-packets messages for which the delay will be considerably different from the curves shown in this chapter. Here also, a reservation scheme such has AMAP has an advantage because of the possibility of modifying the scheme so that the duration of the reserved time slots matches the size of the messages. In this case, the request packets should also carry the size of the message to be transmitted.

Also, for the AMAP protocol, it is relatively easy to integrate voice communications in the system. The request packets should then specify the nature of the connection desired (voice or data). In case of a voice connection two voice channels are assigned to allow a two-way conversation to be established. The modified scheme is called integrated adaptive mobile access protocol (I-AMAP) and is described in [43].

## 4.10 Conclusion

This chapter dealt with the problem of multiple access resulting from the sharing of the same channel by the mobiles for their communication with the hub station. We first described the different categories of access schemes and discussed their applicability to different environments. Then we focused on the schemes that can be used in the case of satellite communications and chose some of them for further investigation. Each of these schemes was analyzed in detail and computer simulations were run to obtain their performance and efficiency. The effect of parameters such as the code rate, the guard time and the error rate on the acknowledgement channel was also evaluated for each scheme. A performance comparison was then carried out expressing the advantages and disadvantages of the different protocols.

# Chapter 5

# CONCLUSIONS

## 5.1 Summary of the thesis

In the literature, error control coding and multiple access protocols are usually examined independently from each other. In discussing error control schemes in noisy channels, point to point stream traffic is most often assumed and average bit error rates are evaluated. When analyzing the performance of multiple access protocols, the usual trend is to assume error free channels or a given packet error rate. However, both problems are interrelated and have a direct effect on the efficiency of the system. In this thesis we have analyzed the performance of some access schemes for land mobile satellite data networks taking into consideration the effect of coding.

In Chapter 2, we discussed the channel model which is adopted for the computer simulations throughout the thesis and which is derived from experimental measurements done by the German Aerospace Research Establishment (DFVLR) and the European Space Agency (ESA).

In Chapter 3, the coding scheme was analyzed and the following conclusions made:

1. A hybrid ARQ error control scheme that combines FEC and ARQ schemes is required in order to achieve the reliability needed for data communications

without an excessive reduction in throughput.

2. Because of their burst error correcting capabilies and their need for a lesser degree of interleaving compared to convolutional codes, Reed-Solomon codes have been found effective for forward error correction.

3. When no interleaving is used, very long and complex RS codes are needed to combat the deep fades resulting from shadowing.

4. Computer simulations showed that the use of symbol interleavers (which are more appropriate than bit interleavers in conjunction with RS codes) allows a significant reduction in code length and, therefore, simpler decoders can be employed.

5. Channel state information (CSI), which can be obtained by monitoring the level of the received signal power, can be put to use in order to improve the performance of the system. If the symbols received during deep fades are erased rather than decoded, the correcting capabilities of the RS codes is augmented and, as shown in our simulation results, a substantial gain in efficiency can be obtained.

In Chapter 4, we first presented a brief overview of the different categories of multiple access protocols and discussed their applicability to satellite communications. Four different schemes were then selected for computer simulations, these schemes are: slotted ALOHA, frequency hopping, adaptive mobile access protocol (AMAP) and a scheme that combines AMAP and frequency hopping. The delay-throughput performance of each protocol was obtained for the case of an error free channel and for the land-mobile satellite channel model. The coding rate was adjusted for each scheme in order to maximize its performance. The effect of the guard time and the error rate on the acknowledgement channel were also studied. At the end of the chapter, we compared and discussed the performance of the protocols under similar conditions.

It was found that the reservation protocol (AMAP) has a substantial throughput advantage over the other schemes that were analyzed and that the performance of the slotted ALOHA protocol is poor in land mobile satellite channels.

## 5.2 Suggestions for further research

The channel model that we have assumed in this thesis is a simplified binary channel where shadowing is considered predominant. The effect of random errors due to other types of noise, like the thermal noise, will reduce the error correcting capability of the RS codes that we have used. One could investigate then the possibility of dealing with these random errors using a concatenation of Reed-Solomon and convolutional codes or another coding scheme.

In our study, we have also assumed that the information messages were fixed in size. When this is not the case, the performance of the schemes obviously changes. Since some protocols may be affected more than others it could be interesting to analyze the effect of the message length and its distribution on the efficiency of the schemes.

Another interesting study would be to analyze the performance of the direct-sequence spread spectrum technique and to compare it to the performance of the schemes that were studied in this thesis.

# Appendix A

# Probability of error in an uncoded system

In the following we derive analytically the probability of a successful transmission of a packet using our binary land-mobile satellite channel model. This analysis considers a system with no coding and/or interleaving and through a dedicated channel (no multiple access).

For the packet to be received successfully, the following conditions should be verified:

1. The transmission of the packet begins in a non fade interval.
2. The interfade interval should last at least the time required to transmit the whole packet.

The probability of packet error is then:

$$PER = 1 - Prob(1) \times Prob(2) \tag{A.1}$$

- The first probability is equal to the average non-fade probability and is 0.9 in this case since the channel has an overall fading probability of 10 %.

- The second probability depends on the number of tests (to see if a bursts occur or not) that are done during the transmission of the packet. Since the tests are done every 10 ms, this number is equal to $\lfloor \tau/10 \rfloor$ or $\lceil \tau/10 \rceil$ depending on the arrival time of a packet in a 10 ms interval ($\tau$ is the packet transmission time in msec). For example, if the packet length is 1000 bits and the bit rate is 4800 bps, the value of $\tau$ is 1000/4800 = 208.3 msec, then, the number of negative tests (no burst decision) necessary is either 20 or 21. As seen in A.2 and A.1, 20 successive tests are sufficient if the packet arrives in the first 8.3 ms (210-208.3) of a 10 ms interval and 21 tests otherwise, i.e. if the packet arrives in the last 1.7 ms (208.3-200) of the interval.

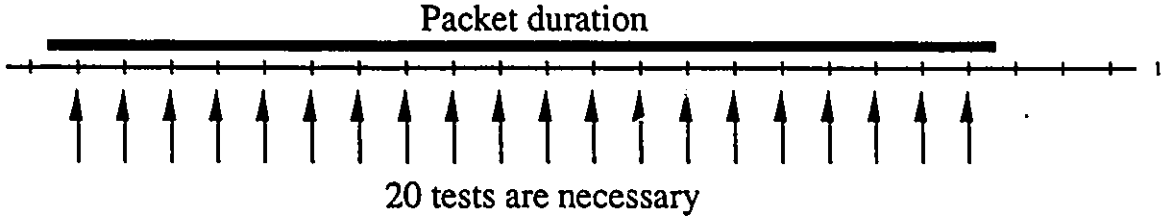


Figure A.1: When the packet arrives during the first 8.3 ms of an interval only 20 tests are necessary.

Since the arrival time of a packet can be considered uniformly distributed in an interval, we can write:

$$Prob(2) = \frac{8.3}{10}(1-p)^{20} + \frac{1.7}{10}(1-p)^{21} \qquad (A.2)$$

Where $1-p$ is the probability of a negative decision (no burst). Since the value of $p$ in our case is $p = 5.944\ 10^{-2}$, we get $Prob(2) = 0.29$.

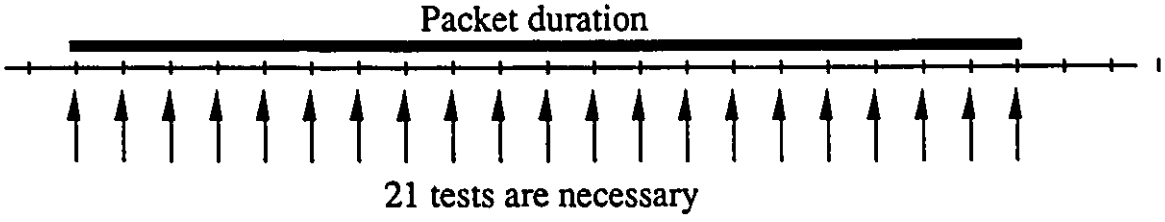


Figure A.2: When the packet arrives during the last 1.7 ms of an interval 21 tests are necessary.

The final probability of packet error is then:

$$PER = 1 - Prob(1) \times Prob(2) = 1 - 0.9 \times 0.29 \approx 0.74 \qquad \text{(A.3)}$$

This result has also been verified by simulation. The computation of the packet error rate when coding and interleaving are used is more complex and the results will only be obtained by computer simulation.

# Appendix B

# Simulation Programs

The computer programs were written in QNAP2 which is a discrete event simulation language adapted to queuing networks. Additional information about this language can be found in [47]. The listings of the following programs are provided:

- "PER-SINT.QNP": this program evaluates the packet error rates obtained in Table 3.1 Chapter 3, when RS coding and symbol interleaving are used.
- "PER-SINT-ERAS.QNP": this program provided the values of the packet error rate listed in Table 3.2 in Chapter 3.
- "RFHMA.QNP": this program simulates a frequency hopping scheme.
- "RFHMA-CSI.QNP": this program simulates a RFHMA scheme where the channel state information (CSI) is used.
- "AMAP.QNP": this program simulates the adaptive mobile access protocol.
- "AMAP-FADING.QNP": this program simultes an AMAP scheme in the ON-OFF fading channel with multi-packet messages.
- "AMAP-RFHMA.QNP": this program simulates the combined AMAP-RFHMA protocol.

```
/CONTROL/ OPTION = NSOURCE;

& THE NAME OF THIS PROGRAM IS PER_SINT.QNP
&*************************************************************************
& This program calculates the packet error rate in an ON-OFF fading
& channel with AWGN. Reed-Solomon codes are used for error correction.
& Binary block codes like (HAMMING, GOLAY...) can also be selected.
& The sequence of bits is symbol-interleaved and encoded before
& transmission. If an ideal channel is wanted, BER should be set to zero
& and so is the INIT parameter for the FADING queues.
&*************************************************************************

/DECLARE/
INTEGER
                n             = 151,        & n of the block code
                k             = 125,        & k of the block code
                q             = 8,          & nbr of bits per symbol (RS)
                bloc_len      = n*q,        & interleaving block length
                depth         = 1,          & interleaving depth
                pack_len=bloc_len*depth,    & length of the encoded sequence
                info_len=pack_len*k/n,      & length of an info. packet in bits
                m,i,j,l,                    & counters
                t             = 3,          & only if binary block codes are used
                temp (pack_len),            & scratch buffer space
                rnd_num,                    & random number generated
                err_bits,                   & number of err bits in symbol
                err_smbl,                   & number of err symbols in codeword
                tot_err,                    & nbr of packts received in error
                tot_pkts,                   & number of packets txd
                suc_pkts;                   & number of successful packets

REAL
                bit_rate = 4800,            & bit rate
                bit_time = 1/bit_rate,      & bit time
                sim_time = 1000,            & simulation time
                fade_len,                   & total fading time in sec.
                BER      = 0.0,             & bit error rate due to AWGN
                burst,                      & burst length in sec.
                thruput;                    & throughput of the system

BOOLEAN
                fade,                       & indicates if there is fade or no
                success;                    & indicates if the pkt has been
                                            & succesfully transmitted

CUSTOMER INTEGER
                packtin   (info_len),       & information sequence
                packtenc  (pack_len),       & encoded  sequence
                packttxd  (pack_len),       & txd  sequence
                packtrxd  (pack_len),       & rxd sequence
                packtout  (info_len);       & decoded  sequence

QUEUE
                FADING,                     & fading generator
                PACKGEN;                    & source of packets


&*************************************************************
&       G E N E R A T I O N   O F   F A D E S .   &----------------------

/STATION/ NAME       = FADING;
          TYPE       = SERVER;
          INIT       = 1;
          SERVICE    = BEGIN
                         WHILE TRUE DO
                         BEGIN
```

```
                              i       := 0;
                              fade    := FALSE;
                              WHILE (RINT (1,1800) < 1693) DO
                                   i := i + 1;
                              CST (i*0.01);
                              fade := TRUE;
                              rnd_num := RINT (1,100);
                              IF rnd_num < 81 THEN
                                      burst := 0.01
                                 ELSE
                                    IF rnd_num < 91 THEN
                                       burst := 0.02
                                    ELSE
                                       IF rnd_num < 96 THEN
                                          burst := 0.04
                                       ELSE
                                          IF rnd_num < 100 THEN
                                             burst := 0.1
                                          ELSE
                                             burst := 0.2;
                              fade_len := fade_len + burst;
                              CST (burst);
                          END;
                       END;
      TRANSIT     = FADING;

&*****************************************************************
&               P A C K E T    G E N E R A T O R .
&-----------------------------------------------------------------

$MACRO generator
BEGIN
     packtin := -1 REPEAT info_len;
END;
$END

&*****************************************************************
&                         E N C O D E R .
&-----------------------------------------------------------------

$MACRO encoder
BEGIN
&FOR i :=1 STEP 1 UNTIL pack_len DO
&  packtenc (i) := i;
      packtenc := -1 REPEAT pack_len;
END;
$END


&*****************************************************************
&                      I N T E R L E A V E R .
&-----------------------------------------------------------------

$MACRO intrlvr
BEGIN
&     FOR j := 0 STEP 1 UNTIL depth-1 DO
&         FOR i := 0 STEP 1 UNTIL n -1 DO
&            FOR l := 1 STEP 1 UNTIL q DO
&               packttxd (j*q+i*depth*q+l) := packtenc (j*bloc_len+i*q+l);
      packttxd := packtenc;          & no need to interleave for real
                                     & since packet is all zeroes.
END;
$END
```

```
&*************************************************************
&                        T R A N S M I T T E R .
&-------------------------------------------------------------

$MACRO xmitter
BEGIN
    FOR i := 1 STEP 1 UNTIL pack_len DO
    BEGIN
        IF fade THEN
           packtrxd (i) := 2*RINT(0,1) - 1;
        IF NOT (DRAW(1-BER)) THEN
           packtrxd (i) := - packtrxd (i);
        CST (bit_time);
    END;
END;
$END

&*************************************************************
&                    D E I N T E R L E A V E R  .
&-------------------------------------------------------------

$MACRO dintrlvr
BEGIN
    temp  := packtrxd;
    FOR j := 0 STEP 1 UNTIL depth-1 DO
        FOR i := 0 STEP 1 UNTIL n -1 DO
           FOR l := 1 STEP 1 UNTIL q DO
               packtrxd (j*bloc_len+i*q+l) := temp (j*q+i*depth*q+l);
END;
$END

&*************************************************************
&                      R S   D E C O D E R.
&-------------------------------------------------------------

$MACRO rsdcoder
BEGIN
     tot_pkts := tot_pkts + 1;
     success  := TRUE;
     FOR m := 0 STEP n*q UNTIL pack_len - n*q DO
     BEGIN
          err_smbl := 0;
          FOR j := 0 STEP q UNTIL (n-1)*q DO
          BEGIN
               err_bits := 0;
               FOR i := 1 STEP 1 UNTIL q DO
               BEGIN
                    IF packtrxd (m+j+i) <> -1 THEN
                         err_bits := err_bits + 1;
               END;
               IF err_bits <> 0 THEN
                  err_smbl := err_smbl + 1;
          END;
&    PRINT(" number of symbols in error : ",err_smbl);
          IF err_smbl > (n-k)/2 THEN
               success := FALSE;
     END;
     IF success THEN
     BEGIN
          suc_pkts := suc_pkts + 1;
&          PRINT ("Packet ",suc_pkts,"received OK...TIME : ",TIME);
     END;
END;
$END
```

```
&*************************************************************
&                          D E C O D E R .
&-------------------------------------------------------------

& This decoder can be used for binary block codes like HAMMING
& and GOLAY codes. The parameter 't' should be set to the number
& of correctable bits.

$MACRO decoder
BEGIN
     tot_pkts := tot_pkts + 1;
     success  := TRUE;
     m        := 0;
     WHILE (m*n < pack_len) AND (success) DO
     BEGIN
         err_bits := 0;
         FOR i := 1 STEP 1 UNTIL n DO
         BEGIN
             IF packtenc (m*n+i) <> packtrxd (m*n+i) THEN
                 err_bits := err_bits + 1;
         END;
         IF err_bits > t THEN
             success := FALSE;
         m := m + 1;
     END;
     IF success THEN
     BEGIN
         suc_pkts := suc_pkts + 1;
&         PRINT ("receiving packet ",suc_pkts," TIME :",TIME);
     END;
END;
$END

&*************************************************************
&              S O U R C E   O F   P A C K E T S .
&-------------------------------------------------------------

/STATION/ NAME        = PACKGEN;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
&                             $generator
&                             $encoder
&                             $intrlvr
packtrxd := -1 REPEAT pack_len;
                            $xmitter
                            $dintrlvr
                            $rsdcoder
&                            $decoder
                          END;
          TRANSIT     = PACKGEN;

&*************************************************************
&             C O N T R O L     P R O G R A M .
&-------------------------------------------------------------

/CONTROL/ TMAX = sim_time;

/EXEC/
BEGIN
&pack_len := depth*bloc_len;
&info_len := pack_len*k/n;
&  PRINT ("BEGINNING OF INITIALIZATION...");
   fade_len := 0;
   tot_pkts := 0;
   suc_pkts := 0;
```

```
&  PRINT ("BEGINNING OF SIMULATION...");
   SIMUL;
   PRINT ("% no fade          = ",100 - fade_len * 100.0 / TIME);
   PRINT ("nbr of succ. pkts  = ",suc_pkts);
   PRINT ("tot nbr of pkts    = ",tot_pkts);
   PRINT ("packet error rate  = ",REALINT(tot_pkts-suc_pkts)/tot_pkts);
&   PRINT ("traffic             = ",tot_pkts * pack_len / bit_rate / TIME);
&   PRINT ("throughput          = ",suc_pkts * pack_len / bit_rate / TIME);
PRINT ("infobits = ",info_len/1000.0," Kbits");
PRINT ("packlen  = ",pack_len/1000.0," Kbits");
END;
&*****************************************************************
/END/
```

```
/CONTROL/ OPTION = NSOURCE;

& THE NAME OF THIS PROGRAM IS PER_SINT_ERAS.QNP
&*****************************************************************
& This program calculates the packet error rate in an ON-OFF fading
& channel with AWGN. Reed-Solomon codes are used for error correction.The
& sequence of bits is symbol-interleaved and
& encoded before transmission, UNRELIABLE symbols are ERASED.
&*****************************************************************

/DECLARE/
INTEGER
                n             = 31,          & n of the block code
                k             = 19,          & k of the block code
                q             = 5,           & nbr of bits per symbol (RS)
                bloc_len      = n*q,         & interleaving block length
                depth         = 1,           & interleaving depth
                pack_len=bloc_len*depth,     & length of the encoded sequence
                info_len=pack_len*k/n,       & length of an info. packet in bits
                m,i,j,l,                     & counters
                t             = 3,           & only for binary block cdes
                temp (pack_len),             & scratch buffer space
                rnd_num,                     & rnd nbr  for burst length computation
                ers_bits,                    & number of bits erased from rxd sumbol
                err_bits,                    & number of bits in error
                ers_smbl,                    & number of symbol erased in rxd packet
                err_smbl,                    & number of symbols in error
                tot_err,                     & nbr of packts received in error
                tot_pkts,                    & number of packets txd
                suc_pkts;                    & number of successful packets

REAL
                bit_rate = 4800,             & bit rate
                bit_time = 1/bit_rate,       & bit time
                sim_time = 150,              & simulation time
                fade_len,                    & total fading time in sec.
                BER      = 0.0,              & bit error rate
                burst,                       & burst length in sec.
                thruput;                     & throughput of the system

BOOLEAN
                fade,                        & indicates if there is fade or no
                success;                     & indicates if the pkt has been
                                             & succesfully transmitted

CUSTOMER INTEGER
                packtin    (info_len),       & information sequence
                packtenc   (pack_len),       & encoded  sequence
                packttxd   (pack_len),       & txd  sequence
                packtrxd   (pack_len),       & rxd sequence
                packtout   (info_len);       & decoded  sequence

QUEUE
                FADING,                      & fading generator
                PACKGEN;                     & source of packets


&*************************************************************
&              F A D I N G     I N S T A N T S .
&-------------------------------------------------------------

/STATION/ NAME        = FADING;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
                          WHILE TRUE DO
```

```
                        BEGIN
                           i      := 0;
                           fade   := FALSE;
&                           WHILE (RINT (1,1800) < 1750) DO
                           WHILE (RINT (1,1800) < 1693) DO
                                 i := i + 1;
                           CST (i*0.01);
                           fade := TRUE;
                           rnd_num := RINT (1,100);
                           IF rnd_num < 81 THEN
                                   burst := 0.01
                                ELSE
                                   IF rnd_num < 91 THEN
                                      burst := 0.02
                                   ELSE
                                      IF rnd_num < 96 THEN
                                         burst := 0.04
                                      ELSE
                                         IF rnd_num < 100 THEN
                                            burst := 0.1
                                         ELSE
                                            burst := 0.2;
                           fade_len := fade_len + burst;
                           CST (burst);
                         END;
                      END;
          TRANSIT    = FADING;


&*************************************************************
&                  G E N E R A T O R .
&-------------------------------------------------------------

$MACRO generator
BEGIN
     packtin := -1 REPEAT info_len;
END;
$END

&*************************************************************
&                        E N C O D E R .
&-------------------------------------------------------------

$MACRO encoder
BEGIN
&FOR i :=1 STEP 1 UNTIL pack_len DO
&  packtenc (i) := i;
      packtenc := -1 REPEAT pack_len;
END;
$END


&*************************************************************
&                     I N T E R L E A V E R .
&-------------------------------------------------------------

$MACRO intrlvr
BEGIN
&    FOR j := 0 STEP 1 UNTIL depth-1 DO
&        FOR i := 0 STEP 1 UNTIL n -1 DO
&           FOR l := 1 STEP 1 UNTIL q DO
&              packttxd (j*q+i*depth*q+l) := packtenc (j*bloc_len+i*q+l);
      packttxd := packtenc;          & no need to interleave for real
                                     & since the packet is all zeroes.
END;
```

```
$END

&************************************************************
&                        T R A N S M I T T E R .
&------------------------------------------------------------

$MACRO xmitter
BEGIN
    packtrxd     := packttxd;
    FOR i := 1 STEP 1 UNTIL pack_len DO
    BEGIN
        IF fade THEN
           packtrxd (i) := 0;        & bit will be erased if fading
        IF NOT (DRAW(1-BER)) THEN
           packtrxd (i) := 1;        & bit will be in error if due to AWGN
        CST (bit_time);
    END;
END;
$END

&************************************************************
&                    D E I N T E R L E A V E R  .
&------------------------------------------------------------

$MACRO dintrlvr
BEGIN
    temp  := packtrxd;
    FOR j := 0 STEP 1 UNTIL depth-1 DO
        FOR i := 0 STEP 1 UNTIL n -1 DO
           FOR l := 1 STEP 1 UNTIL q DO
               packtrxd (j*bloc_len+i*q+l) := temp (j*q+i*depth*q+l);
END;
$END

&************************************************************
&                       R S   D E C O D E R.
&------------------------------------------------------------

$MACRO rsdcoder
BEGIN
     tot_pkts := tot_pkts + 1;
     success  := TRUE;
     FOR m := 0 STEP n*q UNTIL pack_len - n*q DO
     BEGIN
          ers_smbl := 0;
          err_smbl := 0;
          FOR j := 0 STEP q UNTIL (n-1)*q DO
          BEGIN
               ers_bits := 0;
               err_bits := 0;
               FOR i := 1 STEP 1 UNTIL q DO
               BEGIN
                    IF packtrxd (m+j+i) <> -1 THEN
                    BEGIN
                        IF packtrxd (m+j+i) = 0 THEN
                            ers_bits := ers_bits + 1
                        ELSE
                            err_bits := err_bits + 1;
                    END;
               END;
               IF ers_bits <> 0 THEN
                  ers_smbl := ers_smbl + 1
               ELSE
                  IF err_bits <> 0 THEN
                     err_smbl := err_smbl + 1;
```

```
          END;
&    PRINT(" number of symbols in error : ",err_smbl);
&    PRINT(" number of symbols erased   : ",ers_smbl);
          IF ers_smbl + 2*err_smbl > (n-k) THEN
               success := FALSE;
     END;
     IF success THEN
     BEGIN
          suc_pkts := suc_pkts + 1;
&          PRINT ("Packet ",suc_pkts,"received OK...TIME : ",TIME);
     END;
END;
$END

&************************************************************
&                        D E C O D E R .
&------------------------------------------------------------


& this routine assume standard decoding (not erasure)

$MACRO decoder
BEGIN
     tot_pkts := tot_pkts + 1;
     success  := TRUE;
     m        := 0;
     WHILE (m*n < pack_len) AND (success) DO
     BEGIN
         err_bits := 0;
         FOR i := 1 STEP 1 UNTIL n DO
         BEGIN
             IF packtenc (m*n+i) <> packtrxd (m*n+i) THEN
                 err_bits := err_bits + 1;
         END;
         IF err_bits > t THEN
             success := FALSE;
         m := m + 1;
     END;
     IF success THEN
     BEGIN
         suc_pkts := suc_pkts + 1;
&          PRINT ("receiving packet ",suc_pkts," TIME :",TIME);
     END;
END;
$END

&************************************************************
&              S O U R C E   O F   P A C K E T S  .
&------------------------------------------------------------

/STATION/ NAME        = PACKGEN;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
&                             $generator
&                             $encoder
&                             $intrlvr
packttxd := -1 REPEAT pack_len;
                            $xmitter
                            $dintrlvr
                            $rsdcoder
&                            $decoder
                        END;
          TRANSIT     = PACKGEN;

&************************************************************
```

```
&            C O N T R O L    P R O G R A M .
&-------------------------------------------------------------

/CONTROL/ TMAX = sim_time;

/EXEC/
BEGIN
&pack_len := depth*bloc_len;
&info_len := pack_len*k/n;
&  PRINT ("BEGINNING OF INITIALIZATION...");
   fade_len := 0;
   tot_pkts := 0;
   suc_pkts := 0;
&  PRINT ("BEGINNING OF SIMULATION...");
   SIMUL;
   PRINT ("% no fade           = ",100 - fade_len * 100.0 / TIME);
   PRINT ("nbr of succ. pkts   = ",suc_pkts);
   PRINT ("tot nbr of pkts     = ",tot_pkts);
   PRINT ("packet error rate   = ",REALINT(tot_pkts-suc_pkts)/tot_pkts);
&   PRINT ("traffic              = ",tot_pkts * pack_len / bit_rate / TIME);
&   PRINT ("throughput           = ",suc_pkts * pack_len / bit_rate / TIME);
PRINT ("infobits = ",info_len/1000.0," Kbits");
PRINT ("packlen  = ",pack_len/1000.0," Kbits");
END;
&*************************************************************
/END/
```

```
/CONTROL/ OPTION = NSOURCE;
          OPTION = NRESULT;

& THE NAME OF THIS PROGRAM IS RFHMA.QNP FOR RANDOM FREQUENCY HOPPING
& MULTIPLE ACCESS.
&*****************************************************************
& This program simulates the random (slow) frequency hopping multiple
& access on the ON-OFF fading channel. Reed-Solomon codes are used for
& error correction. Each packet is divided into a number of subpackets
& after encoding. Subpackets are then transmitted on different frequencies
& which are randomly chosen. The subpackets that are not received correctly
& are erased completely. An ideal channel can be selected by setting BER to
& zero as well as the INIT parameter in the FADING queue.
&*****************************************************************

/DECLARE/
INTEGER
            n             = 30,           & n of the RS code
            k             = 10,           & k of the RS code
            q             = 5,            & nbr of bits per symbol
            sbpacsym      = 1,            & nbr of symbols in subpac from each CW
            nbsbpacs      = n/sbpacsym,   & number of subpackets per packet
            bloc_len      = n*q,          & interleaving block length
            depth         = 20,           & interleaving depth
            sbpaclen      = sbpacsym*depth*q, & nbr of bits in sub_packets
            pack_len = bloc_len*depth,    & length of the encoded sequence
            info_len = pack_len*k/n,      & length of an info. packet in bits
            err_sbpc,                     & number of subpackets in error
            err_bits,                     & number of bits in error
            err_smbl,                     & number of symbols in error
            rnd_num,                      & random number
            user_nbr      = 100,          & number of users
            backoff       = 15,           & retransmission length
            nbr_chan      = 15,           & number of channels
            slct_chn (user_nbr),          & selcted  channel
            txing (nbr_chan),             & user who transmitted last
            m,                            & counter
            slot_no       = 0,            & slot counter
            tot_msgs      = 0,            & total number of messages
            tot_pkts      = 0,            & number of packets txd
            suc_pkts      = 0;            & number of successful packets

REAL
            bit_rate = 4800,            & bit rate
            bit_time = 1/bit_rate,      & bit time
            sim_time = 100,             & simulation time
            gurd_tim = 0.0008,          & guard time
            slot_tim = ((sbpaclen)/bit_rate+gurd_tim),   & slot duration
            BER      = 0.0,             & AWGN bit error rate
            arv_rate = 0.15,            & packets arrival rate per second
            pr_delay = 0.55,            & propagation delay
            tot_del,                    & total delay
            avg_tx,                     & average number of  tx per message
            thruput,                    & throughput of the system
            fade_len = 0.0,             & percentage of fading
            burst    (user_nbr);        & fade burst length

BOOLEAN
            success (user_nbr),     & indicates if the pkt has been
                                    & succesfully transmitted
            wayclear (user_nbr),    & no fade and beginning of a slot
            collisio (nbr_chan),    & collision in slot or not
            fade     (user_nbr);    & fade or no fade

CUSTOMER INTEGER
            packtrxd  (pack_len), & rxd information sequence
```

```
                owner,                  & user who generated the msg
                nbr_tx;                 & number of data transfer attempts

CUSTOMER REAL
                time_in,                & time message has been generated
                time_out;               & time customer got out of system


QUEUE INTEGER
                i,j,l,i0,i1,i2,id;      & counters

QUEUE
                TIMER,                  & time slotter
                USER    (user_nbr),     & users stations
                FADING  (user_nbr),     & fading statistics
                DELAY;                  & propagation delay

FLAG
                new_slot;               & flag indicating beginning
                                        & of a new slot
&---------------------------------------------------------------
&               F A D I N G     I N S T A N T S .
&---------------------------------------------------------------

/STATION/ NAME        = FADING;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
                          WHILE TRUE DO
                          BEGIN
                             i           := 0;
                             fade (id)   := FALSE;
                             WHILE (RINT (1,1800) < 1693) DO
                                 i := i + 1;
                             CST (i*0.01);
                             fade (id) := TRUE;
                             rnd_num := RINT (1,100);
                             IF rnd_num < 81 THEN
                                    burst (id) := 0.01
                                ELSE
                                   IF rnd_num < 91 THEN
                                      burst (id) := 0.02
                                   ELSE
                                      IF rnd_num < 96 THEN
                                         burst (id) := 0.04
                                      ELSE
                                         IF rnd_num < 100 THEN
                                            burst (id) := 0.1
                                         ELSE
                                            burst (id) := 0.2;
                             fade_len := fade_len + burst (id);
                             CST (burst (id));
                          END;
                        END;
          TRANSIT     = FADING (id);


&---------------------------------------------------------------
&                         T I M E R .
&---------------------------------------------------------------
& This routine defines the time slots and detect the collisions.

/STATION/ NAME        = TIMER;
          TYPE        = SERVER;
          INIT        = 1;
```

```
          SERVICE     = BEGIN
                              WHILE TRUE DO
                              BEGIN
                                   RESET (new_slot);
                                   CST (slot_tim/2);
                                   FOR i := 1 STEP 1 UNTIL nbr_chan DO
                                   BEGIN
                                       IF txing (i) > 1 THEN
                                       BEGIN
                                        PRINT("***COLLISION*** in channel",i);
                                          collisio (i) := TRUE;
                                       END
                                       ELSE
                                         collisio (i) := FALSE;
                                       txing (i) := 0;
                                   END;
                                   slot_no := slot_no + 1;
                                   CST (slot_tim/2);
                                   SET (new_slot);
                                   CST (0);
                              END;
                         END;

&-------------------------------------------------------------
&                          U S E R S .
&-------------------------------------------------------------
&*************************************************************
&                         WAIT NEW SLOT.
&-------------------------------------------------------------

$MACRO waitslot
BEGIN
 WAIT(new_slot);
END;
$END

&*************************************************************
&                           TRANSMIT SUBPACKET.
&-------------------------------------------------------------

$MACRO xmitsbpac
BEGIN
    slct_chn (id)         := RINT (1,nbr_chan);
    txing (slct_chn(id)) := txing (slct_chn(id)) + 1;
    FOR i := 1 STEP 1 UNTIL sbpaclen DO
    BEGIN
        IF fade (id) THEN
              packtrxd (j*sbpaclen+i):= RINT(0,1);
        IF DRAW (BER) THEN
              packtrxd (j*sbpaclen+i):= 1;
        CST (bit_time);
    END;
    IF collisio (slct_chn(id)) THEN
    BEGIN
        FOR i:=1 STEP 1 UNTIL sbpaclen DO
            packtrxd (j*sbpaclen+i):= RINT(0,1);
    END;
END;
$END



&*************************************************************
&                           RS DECODER .
&-------------------------------------------------------------
```

```
$MACRO rsdcoder
BEGIN
    err_sbpc := 0;
    FOR i1 := 0 STEP 1 UNTIL nbsbpacs - 1 DO
    BEGIN
        err_bits := 0;
        FOR i2 := 1 STEP 1 UNTIL sbpaclen DO
        BEGIN
            IF packtrxd (i1*sbpaclen+i2) <> 0 THEN
               err_bits := err_bits + 1;
        END;
        IF err_bits <> 0 THEN
           err_sbpc := err_sbpc + 1;
    END;
    IF err_sbpc*sbpacsym <= n-k THEN
        success (id):= TRUE
    ELSE
        UNIFORM (0,backoff*slot_tim);
END;
$END
&*************************************************************
&                        U S E R S .
&-------------------------------------------------------------

/STATION/ NAME        = USER;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
                            EXP (1/arv_rate);
                            owner         := id;
                            time_in       := TIME;
                            nbr_tx        := 0;
                            success (id)  := FALSE;
                            WHILE NOT (success (id)) DO
                            BEGIN
                                nbr_tx := nbr_tx + 1;
                                j := 0;
                                packtrxd := 0 REPEAT pack_len;
                                WHILE (j < nbsbpacs) DO
                                BEGIN
                                    $waitslot
                                    $xmitsbpac
                                    j := j + 1;
&PRINT ("used slot #",slot_no," channel #",slct_chn(id));
                                END;
                                $rsdcoder
                                CST (pr_delay);
                            END;
                            suc_pkts := suc_pkts + 1;
                            avg_tx   := avg_tx   + nbr_tx;
                            tot_del  := tot_del  + TIME - time_in;
                            PRINT ("MESSAGE #",suc_pkts," nbr tx:",nbr_tx,
                                  " Delay:",TIME-time_in," TIME : ",TIME);
                        END;
          TRANSIT     = USER (id);

&*************************************************************
&               C O N T R O L     P R O G R A M  .
&-------------------------------------------------------------

/CONTROL/ TMAX = sim_time;

/EXEC/
BEGIN
   PRINT ("BEGINNING OF INITIALIZATION...");
   FOR m    := 1 STEP 1 UNTIL user_nbr DO
```

```
   BEGIN
       USER (m).id     := m;
       FADING (m).id   := m;
   END;
   tot_pkts := 0;
   suc_pkts := 0;
   tot_del  := 0.0;
   avg_tx   := 0.0;
   PRINT ("BEGINNING OF SIMULATION...");
   SIMUL;
   PRINT ("PARAMETERS:");
   PRINT ("-----------");
   PRINT ("RS (",n,",",k,") code.");
   PRINT (info_len/1000.0," INFO. Kbits  ",pack_len/1000.0,
          " Kbits in   total.");
   PRINT (nbsbpacs," subpackets of ",sbpaclen," bits each.");
   PRINT ("Depth = ",depth," number of channels = ",nbr_chan);
   PRINT ("GENERAL PERFORMANCE :");
   PRINT ("--------------------");
   PRINT ("number of messages transmitted = ",suc_pkts);
   PRINT ("thruput  = ",suc_pkts/TIME," packets / sec.");
   PRINT ("NORMALIZED THROUGHPUT = ",suc_pkts/TIME/nbr_chan*
          info_len/bit_rate);
   PRINT ("avg. number of tx per msg      = ",REALINT(avg_tx)/suc_pkts);
   PRINT ("avg. total delay               = ",tot_del/suc_pkts," sec");
END;
&************************************************************
/END/
```

```
/CONTROL/ OPTION = NSOURCE;
          OPTION = NRESULT;

& THE NAME OF THIS PROGRAM IS RFHMA_CSI.QNP FOR RANDOM FREQUENCY HOPPING
& MULTIPLE ACCESS WITH THE USE OF THE CHANNEL STATE INFORMATION.
&*****************************************************************
& This program is identical to rfhma.qnp except that the user does not
& transmit a subpacket if the propagation conditions are not good (if
& there is a fade).
&*****************************************************************

/DECLARE/
INTEGER
            n             = 30,           & n of the RS code
            k             = 10,           & k of the RS code
            q             = 5,            & nbr of bits per symbol
            sbpacsym      = 1,            & nbr of symbols in subpac from each CW
            nbsbpacs      = n/sbpacsym,   & number of subpackets per packet
            bloc_len      = n*q,          & interleaving block length
            depth         = 20,           & interleaving depth
            sbpaclen      = sbpacsym*depth*q, & nbr of bits in sub_packets
            pack_len = bloc_len*depth,    & length of the encoded sequence
            info_len = pack_len*k/n,      & length of an info. packet in bits
            err_sbpc,                     & number of subpackets in error
            err_bits,                     & number of bits in error
            err_smbl,                     & number of symbols in error
            rnd_num,                      & random number
            user_nbr      = 100,          & number of users
            backoff       = 15,           & retransmission length
            nbr_chan      = 15,           & number of channels
            slct_chn (user_nbr),          & selcted  channel
            txing (nbr_chan),             & user who transmitted last
            m,                            & counter
            slot_no       = 0,            & slot counter
            tot_msgs      = 0,            & total number of messages
            tot_pkts      = 0,            & number of packets txd
            suc_pkts      = 0;            & number of successful packets

REAL
            bit_rate = 4800,             & bit rate
            bit_time = 1/bit_rate,       & bit time
            sim_time = 100,              & simulation time
            gurd_tim = 0.0008,           & guard time
            slot_tim = ((sbpaclen)/bit_rate+gurd_tim),   & slot duration
            BER      = 0.0,              & AWGN bit error rate
            arv_rate = 0.15,             & packets arrival rate per second
            pr_delay = 0.55,             & propagation delay
            tot_del,                     & total delay
            avg_tx,                      & average number of  tx per message
            thruput,                     & throughput of the system
            fade_len = 0.0,              & percentage of fading
            burst    (user_nbr);         & fade burst length

BOOLEAN
            success (user_nbr),     & indicates if the pkt has been
                                    & succesfully transmitted
            wayclear (user_nbr),    & no fade and beginning of a slot
            collisio (nbr_chan),    & collision in slot or not
            fade     (user_nbr);    & fade or no fade

CUSTOMER INTEGER
            packtrxd  (pack_len), & rxd information sequence
            owner,                & user who generated the msg
            nbr_tx;               & number of data transfer attempts

CUSTOMER REAL
```

```
                time_in,                & time message has been generated
                time_out;               & time customer got out of system


QUEUE INTEGER
                i,j,l,i0,i1,i2,id;      & counters

QUEUE
                TIMER,                  & time slotter
                USER    (user_nbr),     & users stations
                FADING  (user_nbr),     & fading statistics
                DELAY;                  & propagation delay

FLAG
                new_slot;               & flag indicating beginning
                                        & of a new slot
&---------------------------------------------------------------
&               F A D I N G     I N S T A N T S .
&---------------------------------------------------------------

/STATION/ NAME          = FADING;
          TYPE          = SERVER;
          INIT          = 1;
          SERVICE       = BEGIN
                            WHILE TRUE DO
                            BEGIN
                               i            := 0;
                               fade (id)    := FALSE;
                               WHILE (RINT (1,1800) < 1693) DO
                                    i := i + 1;
                               CST (i*0.01);
                               fade (id) := TRUE;
                               rnd_num := RINT (1,100);
                               IF rnd_num < 81 THEN
                                       burst (id) := 0.01
                                  ELSE
                                     IF rnd_num < 91 THEN
                                        burst (id) := 0.02
                                     ELSE
                                        IF rnd_num < 96 THEN
                                           burst (id) := 0.04
                                        ELSE
                                           IF rnd_num < 100 THEN
                                              burst (id) := 0.1
                                           ELSE
                                              burst (id) := 0.2;
                               fade_len := fade_len + burst (id);
                               CST (burst (id));
                            END;
                          END;
          TRANSIT       = FADING (id);


&---------------------------------------------------------------
&                       T I M E R .
&---------------------------------------------------------------
& This routine defines the time slots and detect the collisions.

/STATION/ NAME          = TIMER;
          TYPE          = SERVER;
          INIT          = 1;
          SERVICE       = BEGIN
                               WHILE TRUE DO
                               BEGIN
                                    RESET (new_slot);
```

```
                                    CST (slot_tim/2);
                                    FOR i := 1 STEP 1 UNTIL nbr_chan DO
                                    BEGIN
                                        IF txing (i) > 1 THEN
                                        BEGIN
                                         PRINT("***COLLISION*** in channel",i);
                                          collisio (i) := TRUE;
                                        END
                                        ELSE
                                          collisio (i) := FALSE;
                                        txing (i) := 0;
                                    END;
                                    slot_no := slot_no + 1;
                                    CST (slot_tim/2);
                                    SET (new_slot);
                                    CST (0);
                            END;
                    END;

&-------------------------------------------------------------
&                          U S E R S .
&-------------------------------------------------------------
&*************************************************************
&              WAIT NEW SLOT AND GOOD CONDITIONS (using CSI) .
&-------------------------------------------------------------

$MACRO waitslot
BEGIN
    wayclear (id) := FALSE;
    WHILE NOT (wayclear (id)) DO
    BEGIN
        WAIT (new_slot);
        IF  fade (id) THEN
              CST (slot_tim /2)
        ELSE
              wayclear (id) := TRUE;
    END;
END;
$END

&*************************************************************
&                          TRANSMIT SUBPACKET.
&-------------------------------------------------------------

$MACRO xmitsbpac
BEGIN
    slct_chn (id)         := RINT (1,nbr_chan);
    txing (slct_chn(id)) := txing (slct_chn(id)) + 1;
    FOR i := 1 STEP 1 UNTIL sbpaclen DO
    BEGIN
        IF fade (id) THEN
              packtrxd (j*sbpaclen+i):= RINT(0,1);
        IF DRAW (BER) THEN
              packtrxd (j*sbpaclen+i):= 1;
        CST (bit_time);
    END;
    IF collisio (slct_chn(id)) THEN
    BEGIN
        FOR i:=1 STEP 1 UNTIL sbpaclen DO
            packtrxd (j*sbpaclen+i):= RINT(0,1);
    END;
END;
$END
```

```
&*****************************************************************
&                          RS DECODER .
&-----------------------------------------------------------------

$MACRO rsdcoder
BEGIN
    err_sbpc := 0;
    FOR i1 := 0 STEP 1 UNTIL nbsbpacs - 1 DO
    BEGIN
        err_bits := 0;
        FOR i2 := 1 STEP 1 UNTIL sbpaclen DO
        BEGIN
            IF packtrxd (i1*sbpaclen+i2) <> 0 THEN
               err_bits := err_bits + 1;
        END;
        IF err_bits <> 0 THEN
           err_sbpc := err_sbpc + 1;
    END;
    IF err_sbpc*sbpacsym <= n-k THEN
        success (id):= TRUE
    ELSE
        UNIFORM (0,backoff*slot_tim);
END;
$END
&*****************************************************************
&                        U S E R S .
&-----------------------------------------------------------------

/STATION/ NAME        = USER;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
                            EXP (1/arv_rate);
                            owner        := id;
                            time_in      := TIME;
                            nbr_tx       := 0;
                            success (id) := FALSE;
                            WHILE NOT (success (id)) DO
                            BEGIN
                                nbr_tx := nbr_tx + 1;
                                j := 0;
                                packtrxd := 0 REPEAT pack_len;
                                WHILE (j < nbsbpacs) DO
                                BEGIN
                                    $waitslot
                                    $xmitsbpac
                                    j := j + 1;
&PRINT ("used slot #",slot_no," channel #",slct_chn(id));
                                END;
                                $rsdcoder
                                CST (pr_delay);
                            END;
                            suc_pkts := suc_pkts + 1;
                            avg_tx   := avg_tx   + nbr_tx;
                            tot_del  := tot_del  + TIME - time_in;
                            PRINT ("MESSAGE #",suc_pkts," nbr tx:",nbr_tx,
                                   " Delay:",TIME-time_in," TIME : ",TIME);
                       END;
          TRANSIT     = USER (id);

&*****************************************************************
&              C O N T R O L     P R O G R A M .
&-----------------------------------------------------------------

/CONTROL/ TMAX = sim_time;
```

```
/EXEC/
BEGIN
   PRINT ("BEGINNING OF INITIALIZATION...");
   FOR m     := 1 STEP 1 UNTIL user_nbr DO
   BEGIN
       USER (m).id     := m;
       FADING (m).id   := m;
   END;
   tot_pkts := 0;
   suc_pkts := 0;
   tot_del  := 0.0;
   avg_tx   := 0.0;
   PRINT ("BEGINNING OF SIMULATION...");
   SIMUL;
   PRINT ("PARAMETERS:");
   PRINT ("-----------");
   PRINT ("RS (",n,",",k,") code.");
   PRINT (info_len/1000.0," INFO. Kbits  ",pack_len/1000.0,
          " Kbits in   total.");
   PRINT (nbsbpacs," subpackets of ",sbpaclen," bits each.");
   PRINT ("Depth = ",depth," number of channels = ",nbr_chan);
   PRINT ("GENERAL PERFORMANCE :");
   PRINT ("---------------------");
   PRINT ("number of messages transmitted = ",suc_pkts);
   PRINT ("thruput  = ",suc_pkts/TIME," packets / sec.");
   PRINT ("NORMALIZED THROUGHPUT = ",suc_pkts/TIME/nbr_chan*
          info_len/bit_rate);
   PRINT ("avg. number of tx per msg      = ",REALINT(avg_tx)/suc_pkts);
   PRINT ("avg. total delay               = ",tot_del/suc_pkts," sec");
END;
&************************************************************
/END/
```

```
/CONTROL/ OPTION = NSOURCE;
          OPTION = NRESULT;

& THE NAME OF THIS PROGRAM IS AMAP.QNP
&*****************************************************************
& This program simulates an AMAP (adaptive mobile access protocol)
& system given the packet error rate resulting from fading and AWGN.
& The request channels operate in a slotted ALOHA fashion. The PER
& for the request (reservation) packets should also be given to the program.
& the PER the acknowledgements should also be provided (zero if ideal
& ACK channel). Multiple packet messages are possible by setting the maximum
& number of packets per message (maxsize). The number of packets will then be
& uniformly distributed between 1 and maxsize. It should be noted however
& that in the case of multipacket messages and fading channels this program
& gives approx results since it assumes that errors in successive packets are &
& case.
&*****************************************************************

/DECLARE/
INTEGER
                user_nbr     = 250,      & number of users
                nbrqstch     = 2,        & number of request channels
                nbdatach     = 8,        & number of data channels
                rsv_len      = 155,      & reservation packet length
                pack_len     = 1704,     & length of a packet after encoding
                info_len     = 1000,     & information bits in packet
                K            = 15,       & retransmission length
                maxsize      = 1,        & max. nbr of pkts in message
                txing (nbrqstch),        & user who transmitted last
                rqstch (user_nbr),       & selcted request channel
                m,                       & counter
                q_size (nbdatach),       & queuing size for data ech data channel
                tot_rqst,                & total number of rqst pckts
                suc_rqst,                & total number of successful request
                tot_msgs,                & total number of messages
                tot_pkts,                & number of packets txd
                suc_pkts;                & number of successful packets

REAL
                bit_rate = 4800,         & bit rate
                bit_time = 1/bit_rate,   & bit time
                sim_time = 500,          & simulation time
                gurd_tim = 0.004,        & guard time
                slot_tim = (rsv_len/bit_rate+gurd_tim),   & slot duration
                PER_rqst = 8.375E-2,     & rqst packet error rate (fading+noise)
                PER_rqak = 0.1,          & PER for the ack of requests
                PER_data = 3.32E-2,      & data packet error rate (fading+noise)
                PER_dtak = 0.1,          & PER for the ack of data packets
                arv_rate = 0.09,         & packets arrival rate per second
                pr_delay = 0.55,         & propagation delay
                rqst_t0  = 0.55,         & time out before new request
                data_t0  = 0.55,         & time out before new request
                time_mem (user_nbr),     & memorizes the time
                rqst_del,                & request delay
                tot_del,                 & total delay
                avg_qdel,                & average queuing delay
                avg_rqst,                & average number of requests per message
                avg_tran,                & average number of data tx per message
                tot_tx,                  & total number of transfer attempts
                thruput;                 & throughput of the system

BOOLEAN
                success (user_nbr);      & indicates if the pkt has been
                                         & succesfully transmitted

CUSTOMER INTEGER
```

```
                owner,                  & user who generated the msg
                nbr_rqst,               & number of request attempts
                nbr_tx,                 & number of data transfer attempts
                msg_size,               & message size in number of packets
                msg_rem;                & number of pkts remaining

CUSTOMER REAL
                time_in,                & time message has been generated
                tim_in_q,               & time message got in data channel queue
                time_out;               & time customer got out of system


QUEUE INTEGER
                i,j,id;                 & counters

QUEUE
                TIMER,                  & time slotter
                USER    (user_nbr),     & users stations
                REQUEST (user_nbr),     & request management
                DATACH  (nbdatach),     & data channels
                DELAY;                  & propagation delay

FLAG
                new_slot;               & flag indicating beginning
                                        & of a new slot

&---------------------------------------------------------
&                       T I M E R .
&---------------------------------------------------------

/STATION/ NAME        = TIMER;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
                            WHILE TRUE DO
                            BEGIN
                                 RESET (new_slot);
                                 CST (slot_tim);
                                 txing := 0 REPEAT nbrqstch;
                                 SET (new_slot);
                                 CST (0);
                            END;
                        END;

&---------------------------------------------------------
&                       U S E R S .
&---------------------------------------------------------

/STATION/ NAME        = USER;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
                           EXP (1/arv_rate);
                           owner        := id;
                           time_in      := TIME;
                           nbr_rqst     := 0;
                           nbr_tx       := 0;
                           msg_size     := RINT (1,maxsize);
                           msg_rem      := msg_size;
                        END;
          TRANSIT     = REQUEST (id);

&---------------------------------------------------------
&                     R E Q U E S T S .
&---------------------------------------------------------
```

```
/STATION/ NAME        = REQUEST;
          TYPE        = SERVER;
          SERVICE     = BEGIN
                            time_mem (id) := TIME;
                            success (id)  := FALSE;
                            WHILE NOT (success(id)) DO
                            BEGIN
                                WAIT (new_slot);
                                rqstch (id)          := RINT (1,nbrqstch);
                                txing (rqstch (id)) := txing (rqstch (id))+ 1;
                                CST (slot_tim/2);
                                IF (txing(rqstch(id))=1)
                                    AND (DRAW((1-PER_rqst)*(1-PER_rqak))) THEN
                                        success (id) := TRUE;
                                WAIT (new_slot);
                                nbr_rqst := nbr_rqst + 1;
                                tot_rqst := tot_rqst + 1;
                                IF success (id) THEN
                                BEGIN
                                      CST  (pr_delay);
                                      suc_rqst := suc_rqst + 1;
                                      rqst_del := rqst_del + TIME - time_mem (id).
                                END ELSE
                                BEGIN
                                      CST  (rqst_t0);
                                      UNIFORM (0,K*slot_tim);
                                END;
                            END;
                            i := 1;
                            FOR j := 1 STEP 1 UNTIL nbdatach DO
                                IF q_size (j) < q_size (i) THEN i := j;
                            q_size (i) := q_size (i) + msg_rem;
                            tim_in_q   := TIME;
                            TRANSIT (DATACH(i));
                       END;

&*************************************************************
&        D A T A    T R A N S F E R   C H A N N E L S    .
&-------------------------------------------------------------

/STATION/ NAME      = DATACH;
          TYPE      = SERVER;
          SCHED     = FIFO;
          SERVICE   = BEGIN
                          avg_qdel := avg_qdel + TIME - tim_in_q;
                          tot_tx   := tot_tx + 1;
                          nbr_tx   := nbr_tx + 1;
                          CST (gurd_tim);
                          j := 0;
                          FOR i := 1 STEP 1 UNTIL msg_rem DO
                          BEGIN
                              CST (pack_len / bit_rate );
                              tot_pkts    := tot_pkts + 1;
                              q_size (id) := q_size (id) - 1;
                              IF DRAW ((1-PER_data)*(1-PER_dtak)) THEN
                              BEGIN
                                  j := j+1;
                                  suc_pkts := suc_pkts + 1;
                              END;
                          END;
                          msg_rem := msg_rem - j;
                      END;
          TRANSIT   = DELAY;

&*************************************************************
```

```
&                          D E L A Y .
&-------------------------------------------------------------

/STATION/ NAME     = DELAY;
          TYPE     = INFINITE;
          SERVICE  = BEGIN
                        IF msg_rem <> 0 THEN
                        BEGIN
                           CST (data_t0);
                           TRANSIT (REQUEST (owner));
                        END ELSE
                        BEGIN
                           CST (pr_delay);
                           tot_msgs := tot_msgs + 1;
                           tot_del  := tot_del + TIME - time_in;
                           avg_rqst := avg_rqst + nbr_rqst;
                           avg_tran := avg_tran + nbr_tx;
&PRINT ("message # ",tot_msgs," REQUESTS :",nbr_rqst," NBR TX :",nbr_tx,
&"  Delay :",TIME-time_in,"    TIME : ",TIME);
                           TRANSIT (USER (owner));
                        END;
                     END;

&*************************************************************
&               C O N T R O L    P R O G R A M  .
&-------------------------------------------------------------

/CONTROL/ TMAX = sim_time;

/EXEC/
BEGIN
   PRINT ("BEGINNING OF INITIALIZATION...");
   FOR m    := 1 STEP 1 UNTIL user_nbr DO
   BEGIN
       USER (m).id    := m;
       REQUEST(m).id  := m;
   END;
   FOR m    := 1 STEP 1 UNTIL nbdatach DO
   BEGIN
       DATACH (m).id    := m;
   END;
   q_size   := 0 REPEAT nbdatach;
   tot_msgs := 0;
   tot_tx   := 0;
   tot_rqst := 0;
   suc_rqst := 0;
   tot_pkts := 0;
   suc_pkts := 0;
   rqst_del := 0.0;
   tot_del  := 0.0;
   avg_qdel := 0.0;
   avg_rqst := 0.0;
   avg_tran := 0.0;
   PRINT ("BEGINNING OF SIMULATION...");
   SIMUL;
   PRINT ("REQUEST CHANNELS PERFORMANCE :");
   PRINT ("-----------------------------");
   PRINT ("avg. rqst delay                          = ",rqst_del/suc_rqst," sec.");
   PRINT ("avg. nbr of rqsts before suc rqst  = ",REALINT(tot_rqst)/suc_rqst);
   PRINT ("request PER (collisions + error)   = ",
                                        REALINT(tot_rqst-suc_rqst)/tot_rqst);
   PRINT ("traffic (rqst)= ",REALINT(tot_rqst)*slot_tim/nbrqstch/TIME);
   PRINT ("thruput (rqst)= ",REALINT(suc_rqst)*slot_tim/nbrqstch/TIME);
   PRINT (" ");

   PRINT ("DATA CHANNELS PERFORMANCE :");
```

```
   PRINT ("---------------------------");
   PRINT ("avg. queuing delay = ",avg_qdel/tot_tx," sec.");
   PRINT ("traffic(data)= ",REALINT(tot_pkts)*pack_len/nbdatach/bit_rate/TIME);
   PRINT ("thruput(data)= ",REALINT(suc_pkts)*pack_len/nbdatach/bit_rate/TIME);
   PRINT (" ");

   PRINT ("GENERAL PERFORMANCE :");
   PRINT ("--------------------");
   PRINT ("number of messages transmitted = ",tot_msgs);
   PRINT ("number of packets  transmitted = ",suc_pkts);
   PRINT ("avg. number of requests / msg  = ",avg_rqst/tot_msgs);
   PRINT ("avg. number of tx per msg      = ",avg_tran/tot_msgs);
   PRINT ("avg. total delay               = ",tot_del/tot_msgs," sec");
   PRINT ("Normalized throughput =",
            (suc_pkts/TIME/(nbdatach+nbrqstch))/(bit_rate/info_len));
END;
&*****************************************************************
/END/
```

```
/CONTROL/ OPTION = NSOURCE;
          OPTION = NRESULT;

& THE NAME OF THIS PROGRAM IS AMAP_FADING.QNP
&*************************************************************************
& This program simulates an AMAP system in a fading channel. Reed-Solomon
& coding and symbol interleaving are used.
& The PER for the request packets and ACK packets should be provided to the
& program. This program should be used instead of 'AMAP.QNP' when the messages
& have multiple packets and an ON-OFF cfading channel is used.
&*************************************************************************

/DECLARE/
INTEGER
            n             = 31,           & n of the block code
            k             = 25,           & k of the block code
            q             = 5,            & nbr of bits per symbol (RS)
            bloc_len      = 155,          & interleaving block length
            depth         = 4,            & interleaving depth
            maxsize       = 1,            & max nbr of pkts in message
            pack_len = bloc_len*depth,    & length of the encoded sequence
            temp (pack_len),              & scratch buffer space
            err_bits,                     & number of bits in error
            ers_bits,                     & number of bits erased
            err_smbl,                     & number of symbols in error
            ers_smbl,                     & number of symbols erased
            rnd_num,                      & random number
            user_nbr      = 250,          & number of users
            nbrqstch      = 3,            & number of request channels
            nbdatach      = 7,            & number of data channels
            rsv_len       = 155,          & reservation packet length
            backoff       = 15,           & retransmission length
            txing (nbrqstch),             & user who transmitted last
            rqstch (user_nbr),            & selcted request channel
            m,                            & counter
            q_size (nbdatach),            & queuing size for data ech data channel
            tot_rqst      = 0,            & total number of rqst pckts
            suc_rqst      = 0,            & total number of successful request
            tot_msgs      = 0,            & total number of messages
            tot_pkts      = 0,            & number of packets txd
            suc_pkts      = 0;            & number of successful packets

REAL
            bit_rate = 4800,              & bit rate
            bit_time = 1/bit_rate,        & bit time
            sim_time = 2500,               & simulation time
            gurd_tim = 0.0,               & guard time
            slot_tim = (rsv_len/bit_rate+gurd_tim),   & slot duration
            BER      = 0.0,               & AWGN bit error rate
            PER_rqst = 0.05,     & rqst packet error rate (fading + noise)
            PER_rqak = 0.05,     & PER for the ack of requests
            PER_dtak = 0.0,     & PER for the ack of data packets
            arv_rate = 0.01,              & packets arrival rate per second
            pr_delay = 0.55,              & propagation delay
            rqst_t0  = 0.55,              & time out before new request
            data_t0  = 0.55,              & time out before new request
            time_mem (user_nbr),          & memorizes the time
            rqst_del,                     & request delay
            tot_del,                      & total delay
            avg_qdel,                     & average queuing delay
            avg_rqst,                     & average number of requests per message
            avg_tran,                     & average number of data tx per message
            tot_tx,                       & total number of transfer attempts
            thruput,                      & throughput of the system
            fade_len = 0.0,               & percentage of fading
            burst    (user_nbr);          & fade burst length
```

```
BOOLEAN
                success (user_nbr),     & indicates if the pkt has been
                                        & succesfully transmitted
                ok,                     & packet received ok or not
                fade    (user_nbr);     & fade or no fade

CUSTOMER INTEGER
                packtin   (pack_len),   & information sequence
                owner,                  & user who generated the msg
                nbr_rqst,               & number of request attempts
                nbr_tx,                 & number of data transfer attempts
                msg_size,               & message size in number of packets
                msg_rem;                & number of pkts remaining

CUSTOMER REAL
                time_in,                & time message has been generated
                tim_in_q,               & time message got in data channel queue
                time_out;               & time customer got out of system


QUEUE INTEGER
                i,j,l,i0,i1,i2,id;              & counters

QUEUE
                TIMER,                  & time slotter
                USER    (user_nbr),     & users stations
                REQUEST (user_nbr),     & request management
                DATACH  (nbdatach),     & data channels
                FADING  (user_nbr),     & fading statistics
                DELAY;                  & propagation delay

FLAG
                new_slot;               & flag indicating beginning
                                        & of a new slot
&---------------------------------------------------------------
&               F A D I N G     I N S T A N T S .
&---------------------------------------------------------------

/STATION/ NAME       = FADING;
          TYPE       = SERVER;
          INIT       = 1;
          SERVICE    = BEGIN
                         WHILE TRUE DO
                         BEGIN
                            i           := 0;
                            fade (id)   := FALSE;
&                            WHILE (RINT (1,1800) < 1750) DO
                            WHILE (RINT (1,1800) < 1693) DO
                                 i := i + 1;
                            CST (i*0.01);
                            fade (id) := TRUE;
                            rnd_num := RINT (1,100);
                            IF rnd_num < 81 THEN
                                    burst (id) := 0.01
                                 ELSE
                                    IF rnd_num < 91 THEN
                                       burst (id) := 0.02
                                    ELSE
                                       IF rnd_num < 96 THEN
                                          burst (id) := 0.04
                                       ELSE
                                          IF rnd_num < 100 THEN
                                             burst (id) := 0.1
                                          ELSE
                                             burst (id) := 0.2;
```

```
                        fade_len := fade_len + burst (id);
                        CST (burst (id));
                      END;
                    END;
          TRANSIT     = FADING (id);


&-------------------------------------------------------------
&                          T I M E R .
&-------------------------------------------------------------

/STATION/ NAME        = TIMER;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
                          WHILE TRUE DO
                          BEGIN
                              RESET (new_slot);
                              CST (slot_tim);
                              txing := 0 REPEAT nbrqstch;
                              SET (new_slot);
                              CST (0);
                          END;
                        END;

&-------------------------------------------------------------
&                          U S E R S .
&-------------------------------------------------------------

/STATION/ NAME        = USER;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
                           EXP (1/arv_rate);
                           owner         := id;
                           time_in       := TIME;
                           nbr_rqst      := 0;
                           nbr_tx        := 0;
                           msg_size      := RINT (1,maxsize);
                           msg_rem       := msg_size;
                        END;
          TRANSIT     = REQUEST (id);


&-------------------------------------------------------------
&                        R E Q U E S T S .
&-------------------------------------------------------------

/STATION/ NAME        = REQUEST;
          TYPE        = SERVER;
          SERVICE     = BEGIN
                           time_mem (id) := TIME;
                           success (id)  := FALSE;
                           WHILE NOT (success(id)) DO
                           BEGIN
                               WAIT (new_slot);
                               rqstch (id)          := RINT (1,nbrqstch);
                               txing (rqstch (id)) := txing (rqstch (id))+ 1;
                               CST (slot_tim/2);
                               IF (txing(rqstch(id))=1)
                                   AND (DRAW((1-PER_rqst)*(1-PER_rqak))) THEN
                                        success (id) := TRUE;
                               WAIT (new_slot);
                               nbr_rqst := nbr_rqst + 1;
                               tot_rqst := tot_rqst + 1;
```

```
                    IF success (id) THEN
                    BEGIN
                         CST  (pr_delay);
                         suc_rqst := suc_rqst + 1;
                         rqst_del := rqst_del + TIME - time_mem (id)
                    END ELSE
                    BEGIN
                         CST  (rqst_t0);
                         UNIFORM (0,backoff*slot_tim);
                    END;
               END;
               i := 1;
               FOR j := 1 STEP 1 UNTIL nbdatach DO
                    IF q_size (j) < q_size (i) THEN i := j;
               q_size (i) := q_size (i) + msg_rem;
               tim_in_q   := TIME;
               TRANSIT (DATACH(i));
          END;

&*****************************************************************
&                              TRANSMITTER.
&-----------------------------------------------------------------

$MACRO xmit
BEGIN
     packtin  := -1  REPEAT  pack_len;
     FOR i := 1 STEP 1 UNTIL pack_len DO
     BEGIN
          IF fade (id) THEN
               packtin (i) := 0;    & bit will be erased if fading
          IF DRAW (BER) THEN
               packtin (i) := 1;    & bit will be in error if due to AWGN
          CST (bit_time);
     END;
END;
$END

&*****************************************************************
&                      DEINTERLEAVER.
&-----------------------------------------------------------------
$MACRO dintrlvr
BEGIN
   temp  := packtin;
   FOR i0 := 0 STEP 1 UNTIL depth-1 DO
        FOR i1 := 0 STEP 1 UNTIL n -1 DO
           FOR i2 := 1 STEP 1 UNTIL q DO
               packtin (i0*bloc_len+i1*q+i2) := temp (i0*q+i1*depth*q+i2);
END;
$END
&*****************************************************************
&                      RS-DECODER.
&-----------------------------------------------------------------
$MACRO rsdcoder
BEGIN
     ok     := TRUE;
     FOR i0 := 0 STEP n*q UNTIL pack_len - n*q DO
     BEGIN
          ers_smbl := 0;
          err_smbl := 0;
          FOR i1   := 0 STEP q UNTIL (n-1)*q DO
          BEGIN
               ers_bits := 0;
               err_bits := 0;
               FOR i2 := 1 STEP 1 UNTIL q DO
               BEGIN
```

```
                        IF packtin (i0+i1+i2) <> -1 THEN
                        BEGIN
                            IF packtin (i0+i1+i2) = 0 THEN
                                ers_bits := ers_bits + 1
                            ELSE
                                err_bits := err_bits + 1;
                        END;
                    END;
                    IF ers_bits <> 0 THEN
                       ers_smbl := ers_smbl + 1
                    ELSE
                       IF err_bits <> 0 THEN
                          err_smbl := err_smbl + 1;
               END;
               IF ers_smbl + 2*err_smbl > (n-k) THEN
                    ok := FALSE;
          END;
          tot_pkts    := tot_pkts + 1;
          q_size (id) := q_size (id) - 1;
          IF  (ok) AND DRAW (1-PER_dtak) THEN
          BEGIN
               j := j+1;
               suc_pkts := suc_pkts + 1;
          END;
END;
$END
&************************************************************
&        D A T A    T R A N S F E R   C H A N N E L S   .
&------------------------------------------------------------

/STATION/ NAME      = DATACH;
          TYPE      = SERVER;
          SCHED     = FIFO;
          SERVICE   = BEGIN
                          avg_qdel := avg_qdel + TIME - tim_in_q;
                          tot_tx   := tot_tx + 1;
                          nbr_tx   := nbr_tx + 1;
                          CST (gurd_tim);
                          j := 0;
                          FOR l := 1 STEP 1 UNTIL msg_rem DO
                          BEGIN
                               $xmit
                               $dintrlvr
                               $rsdcoder
                          END;
                          msg_rem := msg_rem - j;
                      END;

          TRANSIT   = DELAY;

&************************************************************
&                        D E L A Y .
&------------------------------------------------------------

/STATION/ NAME      = DELAY;
          TYPE      = INFINITE;
          SERVICE   = BEGIN
                          IF msg_rem <> 0 THEN
                          BEGIN
                             CST (data_t0);
                             TRANSIT (REQUEST (owner));
                          END ELSE
                          BEGIN
                             CST (pr_delay);
                             tot_msgs := tot_msgs + 1;
                             tot_del  := tot_del + TIME - time_in;
```

```
                        avg_rqst := avg_rqst + nbr_rqst;
                        avg_tran := avg_tran + nbr_tx;
                        PRINT (" message # ",tot_msgs,"    TIME : ",TIME);
                        TRANSIT (USER (owner));
                    END;
                END;

&*************************************************************
&              C O N T R O L     P R O G R A M  .
&-------------------------------------------------------------

/CONTROL/ TMAX = sim_time;

/EXEC/
BEGIN
   PRINT ("BEGINNING OF INITIALIZATION...");
   FOR m     := 1 STEP 1 UNTIL user_nbr DO
   BEGIN
       USER (m).id     := m;
       REQUEST(m).id   := m;
       FADING (m).id   := m;
   END;
   FOR m     := 1 STEP 1 UNTIL nbdatach DO
   BEGIN
       DATACH (m).id     := m;
   END;
   q_size    := 0 REPEAT nbdatach;
   tot_msgs  := 0;
   tot_tx    := 0;
   tot_rqst  := 0;
   suc_rqst  := 0;
   tot_pkts  := 0;
   suc_pkts  := 0;
   rqst_del  := 0.0;
   tot_del   := 0.0;
   avg_qdel  := 0.0;
   avg_rqst  := 0.0;
   avg_tran  := 0.0;
   PRINT ("BEGINNING OF SIMULATION...");
   SIMUL;
   PRINT ("REQUEST CHANNELS PERFORMANCE :");
   PRINT ("-----------------------------");
   PRINT ("avg. rqst delay                          = ",rqst_del/suc_rqst," sec.");
   PRINT ("avg. nbr of rqsts before suc rqst  = ",REALINT(tot_rqst)/suc_rqst);
   PRINT ("request PER (collisions + error)    = ",
                                        REALINT(tot_rqst-suc_rqst)/tot_rqst);
   PRINT ("traffic (rqst)= ",REALINT(tot_rqst)*rsv_len/nbrqstch/bit_rate/TIME);
   PRINT ("thruput (rqst)= ",REALINT(suc_rqst)*rsv_len/nbrqstch/bit_rate/TIME);
   PRINT (" S't'          = ",suc_rqst*slot_tim/TIME);
   PRINT (" ");

   PRINT ("DATA CHANNELS PERFORMANCE :");
   PRINT ("--------------------------");
   PRINT ("avg. queuing delay = ",avg_qdel/tot_tx," sec.");
   PRINT ("traffic(data)= ",REALINT(tot_pkts)*pack_len/nbdatach/bit_rate/TIME);
   PRINT ("thruput(data)= ",REALINT(suc_pkts)*pack_len/nbdatach/bit_rate/TIME);
   PRINT (" ");

   PRINT ("GENERAL PERFORMANCE :");
   PRINT ("--------------------");
   PRINT ("number of messages transmitted = ",tot_msgs);
   PRINT ("normalized arrival rate   S t' = ",tot_msgs*pack_len/bit_rate/TIME);
   PRINT ("avg. number of requests / msg  = ",avg_rqst/tot_msgs);
   PRINT ("avg. number of tx per msg      = ",avg_tran/tot_msgs);
   PRINT ("avg. total delay               = ",tot_del/tot_msgs," sec");
   PRINT ("normalized average delay       =",
```

```
                        tot_del/tot_msgs*bit_rate/pack_len);
   PRINT ("throughput = ",suc_pkts*pack_len/bit_rate/(nbdatach+nbrqstch)/TIME);
END;
&*************************************************************
/END/
```

```
/CONTROL/ OPTION = NSOURCE;
          OPTION = NRESULT;

& THE NAME OF THIS PROGRAM IS AMAP-RFHMA.QNP
&*****************************************************************
& This program simulates a combined random frequency hopping multiple
& access and AMAP system given the packet error rate
& resulting from fading and AWGN. The delays for coding, interleaving,
& deinterleaving and decoding should be given to the program. The output
& of the program are : throughput, avg. delay, traffic...etc...
&*****************************************************************

/DECLARE/
INTEGER
          n           = 15,          & n of the block code
          k           = 09,          & k of the block code
          q           = 4,           & nbr of bits per symbol (RS)
          sbpacsym    = 1,           & nbr of symbols in subpac from each CW
          nbsbpacs    = n/sbpacsym,  & number of subpackets per packet
          bloc_len    = n*q,         & interleaving block length
          depth       = 28,          & interleaving depth
          sbpaclen    = sbpacsym*depth*q, & nbr of bits in sub_packets


          pack_ln1 = bloc_len*depth, & length of the encoded sequence
          pack_ln2 = 1416,           &
          info_len = 1000,           & length of an info. packet in bits
          err_sbpc,                  & number of subpackets in error
          err_bits,                  & number of bits in error
          err_smbl,                  & number of symbols in error
          rnd_num,                   & random number
          user_nbr    = 100,         & number of users
          nbrqstch    = 10,           & number of request channels
          nbdatach    = 5,           & number of data channels
          slct_chn (user_nbr),       & selcted  channel
          maxsize     = 1,           & max. nbr of pkts in message
          txing (nbrqstch),          & user who transmitted last
          m,                         & counter
          q_size (nbdatach),         & queuing size for each data channel
          tot_tx1,                   & total number of rqst pckts
          tot_tx2,                   & total number of rqst pckts
          slot_no     = 0,           & slot counter
          suc_tx1,                   & total number of successful tx1
          suc_tx2,                   & total number of successful tx2
          tot_msgs,                  & total number of messages
          tot_pkts;                  & number of packets txd

REAL
          bit_rate = 4800,           & bit rate
          bit_time = 1/bit_rate,     & bit time
          sim_time = 40,           & simulation time
          gurd_tim = 0.004,          & guard time
          slot_tim = (sbpaclen/bit_rate+gurd_tim),   & slot duration
          PER_rqak = 0.0,            & PER for the ack of requests
          PER_data = 9.25E-2,        & data packet error rate (fading+noise)
          PER_dtak = 0.0,            & PER for the ack of data packets
          BER      = 0.0,            & bit error rate
          arv_rate = 0.20,           & packets arrival rate per second
          pr_delay = 0.55,           & propagation delay
          rqst_t0  = 0.55,           & time out before new request
          data_t0  = 0.55,           & time out before new request
          time_mem (user_nbr),       & memorizes the time
          tot_del,                   & total delay
          avg_qdel,                  & average queuing delay
          avg_tx1,                   & average number of requests per message
          avg_tx2,                   & average number of data tx per message
```

```
                tot_tx,                 & total number of transfer attempts
                fade_len = 0.0,         & percentage of fading
                burst    (user_nbr),    & fade burst length
                thruput;                & throughput of the system

BOOLEAN
                wayclear (user_nbr),    & no fade and beginning of a slot
                collisio (nbrqstch),    & collision in slot or not
                fade     (user_nbr),    & fade or no fade

                assigned (user_nbr),    & if an ACK or assign has been rxd
                success (user_nbr);     & indicates if the pkt has been
                                        & succesfully transmitted

CUSTOMER INTEGER
                packtrxd  (pack_lnl),   & rxd information sequence
                owner,                  & user who generated the msg
                nbr_rqst,               & number of request attempts
                nbr_tx1,                & number of data transfer attempts
                nbr_tx2,                & number of data transfer attempts
                msg_size,               & message size in number of packets
                msg_rem;                & number of pkts remaining

CUSTOMER REAL
                time_in,                & time message has been generated
                tim_in_q,               & time message got in data channel queue
                time_out;               & time customer got out of system


QUEUE INTEGER
                i,j,l,i0,i1,i2,id;      & counters

QUEUE
                TIMER,                  & time slotter
                FADING   (user_nbr),    & fading statistics
                USER     (user_nbr),    & users stations
                REQUEST  (user_nbr),    & request management
                DATACH   (nbdatach),    & data channels
                DELAY;                  & propagation delay

FLAG
                new_slot;               & flag indicating beginning
                                        & of a new slot

&-----------------------------------------------------------
&                        T I M E R .
&-----------------------------------------------------------

/STATION/ NAME       = TIMER;
          TYPE       = SERVER;
          INIT       = 1;
          SERVICE    = BEGIN
                         WHILE TRUE DO
                         BEGIN
                              RESET (new_slot);
                              CST (slot_tim/2);
                              FOR i := 1 STEP 1 UNTIL nbrqstch DO
                              BEGIN
                                  IF txing (i) > 1 THEN
                                  BEGIN
                                   PRINT("***COLLISION*** in channel",i);
                                     collisio (i) := TRUE;
                                  END
                                  ELSE
                                    collisio (i) := FALSE;
                                  txing (i) := 0;
```

```
                                END;
                                slot_no := slot_no + 1;
                                CST (slot_tim/2);
                                SET (new_slot);
                                CST (0);
                            END;
                        END;
&-------------------------------------------------------------
&              F A D I N G     I N S T A N T S .
&-------------------------------------------------------------

/STATION/ NAME        = FADING;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
                          WHILE TRUE DO
                          BEGIN
                             i             := 0;
                             fade (id)     := FALSE;
                             WHILE (RINT (1,1800) < 1693) DO
                                   i := i + 1;
                             CST (i*0.01);
                             fade (id) := TRUE;
                             rnd_num := RINT (1,100);
                             IF rnd_num < 81 THEN
                                    burst (id) := 0.01
                                ELSE
                                   IF rnd_num < 91 THEN
                                      burst (id) := 0.02
                                   ELSE
                                      IF rnd_num < 96 THEN
                                         burst (id) := 0.04
                                      ELSE
                                         IF rnd_num < 100 THEN
                                            burst (id) := 0.1
                                         ELSE
                                            burst (id) := 0.2;
                             fade_len := fade_len + burst (id);
                             CST (burst (id));
                          END;
                        END;
          TRANSIT     = FADING (id);


&-------------------------------------------------------------
&                        U S E R S .
&-------------------------------------------------------------

/STATION/ NAME        = USER;
          TYPE        = SERVER;
          INIT        = 1;
          SERVICE     = BEGIN
                           EXP (1/arv_rate);
                           owner         := id;
                           time_in       := TIME;
                           nbr_tx1       := 0;
                           nbr_tx2       := 0;
                           msg_size      := RINT (1,maxsize);
                           msg_rem       := msg_size;
                        END;
          TRANSIT     = REQUEST (id);


&-------------------------------------------------------------
&                      R E Q U E S T S .
```

```
&-----------------------------------------------------------

&***********************************************************
&             WAIT NEW SLOT AND GOOD CONDITIONS  .
&-----------------------------------------------------------

$MACRO waitslot
BEGIN
    wayclear (id) := FALSE;
    WHILE NOT (wayclear (id)) DO
    BEGIN
        WAIT (new_slot);
        IF  fade (id) THEN
              CST (slot_tim /2)
        ELSE
              wayclear (id) := TRUE;
    END;

& IF NO CSI AVAILABLE SKIP THE PREVIOUS BLOCK AND JUST EXECUTE THE FOLLOWING
& COMMAND :
& WAIT(new_slot);

END;
$END

&***********************************************************
&                          TRANSMIT SUBPACKET.
&-----------------------------------------------------------

$MACRO xmitsbpac
BEGIN
    slct_chn (id) := RINT (1,nbrqstch);
    txing (slct_chn(id)) := txing (slct_chn(id)) + 1;
    FOR i := 1 STEP 1 UNTIL sbpaclen DO
    BEGIN
        IF fade (id) THEN
              packtrxd (j*sbpaclen+i) := RINT(0,1);
        IF DRAW (BER) THEN
              packtrxd (j*sbpaclen+i) := 1;
        CST (bit_time);
    END;
    IF collisio (slct_chn(id)) THEN
    BEGIN
        FOR i:=1 STEP 1 UNTIL sbpaclen DO
            packtrxd (j*sbpaclen+i):= RINT(0,1);
    END;
END;
$END

&***********************************************************
&                          DECODER 1.
&-----------------------------------------------------------

$MACRO dcoder1
BEGIN
    err_sbpc := 0;
    FOR i1 := 0 STEP 1 UNTIL nbsbpacs - 1 DO
    BEGIN
        err_bits := 0;
        FOR i2 := 1 STEP 1 UNTIL sbpaclen DO
        BEGIN
            IF packtrxd (i1*sbpaclen+i2) <> 0 THEN
               err_bits := err_bits + 1;
        END;
        IF err_bits <> 0 THEN
           err_sbpc := err_sbpc + 1;
```

```
    END;
PRINT ("symbols in error :",err_sbpc*sbpacsym);
    IF err_sbpc*sbpacsym <= n-k THEN
        success (id) := TRUE
    ELSE
        success (id) := FALSE;

END;
$END
&-----------------------------------------------------------------
/STATION/ NAME       = REQUEST;
          TYPE       = SERVER;
          SERVICE    = BEGIN
                           assigned (id) := FALSE;
                           WHILE NOT (assigned (id)) DO
                           BEGIN
                              nbr_tx1  := nbr_tx1 + 1;
                              tot_tx1  := tot_tx1 + 1;
                              packtrxd := 0 REPEAT pack_ln1;
                              FOR j    := 0 STEP 1 UNTIL nbsbpacs - 1 DO
                              BEGIN
                                 $waitslot
                                 $xmitsbpac
&PRINT ("used slot #",slot_no," channel #",slct_chn(id));
                              END;
                              $dcoder1
                              IF DRAW (1-PER_rqak) THEN
                                  assigned (id) := TRUE
                              ELSE
                                  CST (pr_delay);
                           END;

                           IF success (id) THEN
                           BEGIN
                              suc_tx1 := suc_tx1 + 1;
                              msg_rem := msg_rem - 1;
                           END;
                           IF msg_rem = 0 THEN
                              TRANSIT (DELAY)
                           ELSE
                           BEGIN
                              CST (pr_delay);
                              i := 1;
                              FOR j := 1 STEP 1 UNTIL nbdatach DO
                                 IF q_size (j) < q_size (i) THEN i := j;
                              q_size (i) := q_size (i) + msg_rem;
                              tim_in_q   := TIME;
                              TRANSIT (DATACH(i));
                           END;
                       END;

&*****************************************************************
&        D A T A    T R A N S F E R   C H A N N E L S    .
&-----------------------------------------------------------------

/STATION/ NAME     = DATACH;
          TYPE     = SERVER;
          SCHED    = FIFO;
          SERVICE  = BEGIN
                           avg_qdel := avg_qdel + TIME - tim_in_q;
                           tot_tx   := tot_tx   + 1;
                           nbr_tx2  := nbr_tx2  + 1;
                           CST (gurd_tim);
                           j := 0;
                           FOR i := 1 STEP 1 UNTIL msg_rem DO
                           BEGIN
```

```
                            CST (pack_ln2 / bit_rate );
                            tot_tx2      := tot_tx2 + 1;
                            q_size (id) := q_size (id) - 1;
                            IF DRAW ((1-PER_data)*(1-PER_dtak)) THEN
                            BEGIN
                                j := j+1;
                                suc_tx2 := suc_tx2 + 1;
                            END;
                        END;
                        msg_rem := msg_rem - j;
                    END;
          TRANSIT  = DELAY;

&*************************************************************
&                        D E L A Y .
&-------------------------------------------------------------

/STATION/ NAME     = DELAY;
          TYPE     = INFINITE;
          SERVICE  = BEGIN
                        IF msg_rem <> 0 THEN
                        BEGIN
                           CST (data_t0);
                           TRANSIT (REQUEST (owner));
                        END ELSE
                        BEGIN
                           CST (pr_delay);
                           tot_msgs := tot_msgs + 1;
                           tot_pkts := tot_pkts + 1;
                           tot_del  := tot_del  + TIME - time_in;
                           avg_tx1  := avg_tx1  + nbr_tx1;
                           avg_tx2 := avg_tx2   + nbr_tx2;
PRINT ("message # ",tot_msgs,"  NBR TX1 :",nbr_tx1,"  NBR TX2 :",nbr_tx2,
"  Delay :",TIME-time_in,"    TIME : ",TIME);
                           TRANSIT (USER (owner));
                        END;
                    END;

&*************************************************************
&               C O N T R O L     P R O G R A M  .
&-------------------------------------------------------------

/CONTROL/ TMAX = sim_time;

/EXEC/
BEGIN
   PRINT ("BEGINNING OF INITIALIZATION...");
   FOR m     := 1 STEP 1 UNTIL user_nbr DO
   BEGIN
       USER (m).id     := m;
       REQUEST(m).id   := m;
       FADING (m).id   := m;
   END;
   FOR m     := 1 STEP 1 UNTIL nbdatach DO
   BEGIN
       DATACH (m).id     := m;
   END;
   q_size    := 0 REPEAT nbdatach;
   tot_msgs  := 0;
   tot_pkts  := 0;
   tot_tx2   := 0;
   tot_tx1   := 0;
   suc_tx1   := 0;
   suc_tx2   := 0;
   tot_del   := 0.0;
   avg_qdel  := 0.0;
```

```
   avg_tx1  := 0.0;
   avg_tx2  := 0.0;
   PRINT ("BEGINNING OF SIMULATION...");
   SIMUL;
   PRINT ("fade = ",fade_len*100/TIME/user_nbr," %");
   PRINT ("REQUEST CHANNELS PERFORMANCE :");
   PRINT ("-----------------------------");
   PRINT ("avg. nbr of rqsts before suc rqst  = ",REALINT(tot_tx1)/suc_tx1);
   PRINT ("request PER (collisions + error)   = ",
                                        REALINT(tot_tx1-suc_tx1)/tot_tx1);
   PRINT ("traffic (rqst)= ",REALINT(tot_tx1*nbsbpacs)*slot_tim/nbrqstch/TIME);
   PRINT ("thruput (rqst)= ",REALINT(suc_tx1*nbsbpacs)*slot_tim/nbrqstch/TIME);
   PRINT (" ");

   PRINT ("DATA CHANNELS PERFORMANCE :");
   PRINT ("--------------------------");
   PRINT ("avg. queuing delay = ",avg_qdel/tot_tx," sec.");
   PRINT ("traffic(data)= ",REALINT(tot_tx2)*pack_ln2/nbdatach/bit_rate/TIME);
   PRINT ("thruput(data)= ",REALINT(suc_tx2)*pack_ln2/nbdatach/bit_rate/TIME);
   PRINT (" ");

   PRINT ("GENERAL PERFORMANCE :");
   PRINT ("--------------------");
   PRINT ("number of messages transmitted = ",tot_msgs);
   PRINT ("number of packets  transmitted = ",tot_pkts);
   PRINT ("avg. number of tx1 per msg     = ",avg_tx1/tot_msgs);
   PRINT ("avg. number of tx2 per msg     = ",avg_tx2/tot_msgs);
   PRINT ("avg. total delay               = ",tot_del/tot_msgs," sec");
   PRINT ("Normalized throughput =",
            (tot_pkts/TIME/(nbdatach+nbrqstch))/(bit_rate/info_len));
END;
&************************************************************
/END/
```

# Bibliography


[1] C.E. Mahle, G. Hyde and T. Inukai "Satellite scenarios and technology for the 1990's", IEEE journal on selected areas in communications, Vol. SAC-5, No. 4, pp. 556-570, May 1987.

[2] J.N. Pelton, W.W. Wu "The challenge of 21$^{st}$ century satellite communications: INTELSAT enters the second millennium", IEEE journal on selected areas in communications, Vol. SAC-5, No. 4, pp. 592-601, May 1987.

[3] P. Bartholome "The future of satellite communications in Europe", IEEE journal on selected areas in communications, Vol. SAC-5, No. 4, pp. 615-623, May 1987.

[4] A. Ghais, G. Berzins and D. Wright "INMARSAT and the future of mobile satellite services", IEEE journal on selected areas in communications, Vol. SAC-5, No. 4, pp. 556-570, May 1987.

[5] M. Wagg "MOBILESAT, Australia's Own", Proceedings of the second International Mobile Satellite Conference IMSC'90, Ottawa, Ontario, pp. 3-7, June 1990.

[6] I.M. Jacobs, A. Salmasi, K.S. Gilhousen, L.A. Weaver and T.J. Bernard "A second anniversary operational review of the OmniTRACS, the first two-way mobile Ku-band satellite communications system", Proceedings of the second International Mobile Satellite Conference IMSC'90, Ottawa, Ontario, pp. 13-18, June 1990.

[7] M. Wachira "Domestic Mobile Satellite Systems in North America", Proceedings of the second International Mobile Satellite Conference IMSC'90, Ottawa, Ontario, pp. 3-7, June 1990.

[8] W.B. Garner "The American Mobile Satellite System", Proceedings of the second International Mobile Satellite Conference IMSC'90, Ottawa, Ontario, pp. 28-31, June 1990.

[9] G.K. Noreen "MSAT: Mobile Communications Throughout North America" Proc. Vehicular Technology Conference, pp. 557-562, 1989.

[10] K.A. Norton, L.E. Vogler, W.V. Mansfield and P.J. Short, "The probability distribution of a constant vector plus a Rayleigh distributed vector", Proc. of the IRE, pp. 1354-1361, October 1955.

[11] Erich Lutz and Ernst Plochinger, "Generating Rice processes with given spectral properties", IEEE transactions on vehicular technology, Vol. VT-34, No.4, pp. 178-181, November 1985.

[12] C. Loo, "A statistical model for a land-mobile satellite link", IEEE transactions on vehicular technology, Vol. VT-34, pp. 122-127, August 1985.

[13] C. Loo "Measurements and models of a mobile satellite link with applications", Proc. Globecom 85, New Orleans, L.A., December 1985.

[14] C. Loo, E.E. Matt, J.S. Butterworth and M. Dufour, "Measurements and modelling of a land-mobile satellite signal statistics", 1986 Vehicular technology conference, Dallas, Texas, May 1986.

[15] Y. Hase " Fade/non-fade duration characteristics of land mobile satellite communication link" Globecom 1990, san Diego, pp. 110-114.

[16] Standard-M Technical note TN/1/3"Simplified Land Mobile Burst Channel Model For Standard-M", August 1989.

[17] B. Wicker "Hybrid-ARQ Reed-Solomon coding in an adaptive rate system", Proc. International conference on communications, pp. 1383-1387, 1989.

[18] D. Cygan "Analytical evaluation of average bit error rate for the land mobile satellite channel", International journal of satellite communications, vol. 7, pp. 99-102, 1989.

[19] J. Hagenauer and E. Lutz "Forward error correction coding for fading compensation in mobile satellite channels", IEEE journal on selected areas in communications, Vol. SAC-5, No. 2, pp. 215-225, February 1987.

[20] Erich Lutz "Simulation of FEC/ARQ data transmission using stored land mobile satellite channels" Proc. Vehicular Technology Conference, pp. 109-115, 1986.

[21] R. Fantacci "Efficient ARQ schemes for error control of satellite channels" International journal of satellite communications, vol. 7, pp. 193-200, 1989.

[22] K.H.H. Wong, L. Hanzo and R. Steel " Channel coding for satellite mobile channels" International journal of satellite communications, vol. 7, pp. 143-163, 1989.

[23] S.B. Wicker "High-reliability data transfer over the land mobile radio channel using interleaved hybrid-ARQ error control", IEEE transactions on vehicular technology, Vol. 39, N0. 1, February 1990.

[24] C.K. Siew and D.J. Goodman "Packet data transmission over mobile radio channels", IEEE transactions on vehicular technology, vol. 38, No. 2, pp. 95-101, May 1989.

[25] I.C. Ferebee, D.J. Tait and D.H. Taylor "An ARQ/FEC coding scheme for land-mobile communication", International journal of satellite communication, vol. 7, pp. 219-224, 1989.

[26] D.J. Costello and M.J. Miller "Automatic repeat request error control schemes", IEEE communications magazine, Vol. 22, No. 12, pp. 5-17,December 1984.

[27] D.J. Costello and S. Lin "Error control coding: fundamentals and applications." Prentice-Hall editor, 1983, ISBN 0-13-283796.

[28] K. Mokrani and S.S. Soliman "Concatenated codes over fading dispersive channels", Proc. International Conference on Communications, pp. 1378-1382, 1989.

[29] A. Brine, P.G. Farrell and R.A. Harris "Low complexity concatenated coding schemes for digital satellite communications", International journal of satellite communications, vol.7, pp. 209-217, 1989.

[30] F.A. Tobagi "Multiaccess protocols in packet communication systems", IEEE transactions on communications, Vol.COM-28, N0. 4, April 1980.

[31] D. Minoli and W. Nakakine "A taxonomy and comparison of random access protocols for computer networks", Data communication and

computer networks, North-Holland Publishing Company, pp. 187-206, 1981.

[32] M. Shwartz "Telecommunication networks: protocols, modeling and analysis", Addison-Wesley Publishing Company, 1987, ISBN 0-201-16423-X.

[33] S. Tasaka "Multiple-access protocols for satellite packet communication networks: a performance comparison", Proceedings of the IEEE, vol. 72, No. 11, November 1984.

[34] C. Wolejsza, D. Taylor, M. Grossman and W.P. Osborne "Multiple access protocols for data communications via VSAT networks", IEEE communications magazine, Vol. 25, No. 7, pp. 30-39, July 1987.

[35] K. Joseph and D. Raychaudhuri "Simulation models for performance evaluation of satellite multiple access protocols", IEEE journal on selected areas in communications, vol. 6, No. 1, pp. 210-222, January 1988.

[36] D. Raychaudhuri, K. Joseph "Channel access protocols for Ku-band VSAT networks: a comparative evaluation", IEEE communication magazine, pp. 34-44, vol. 26, No. 5, May 1988.

[37] L. Kleinrock and S.S. Lam "Packet switching in a multiple access broadcast channel: performance evaluation", IEEE transactions on communications, vol. COM-23, pp. 410-423, April 1975.

[38] S.S. Lam, "Packet switching in a multi-access broadcast channel", Ph.D. dissertation, Dept. of Computer Science, UCLA, April 1974.

[39] J. Borgonovo and L. Fratter "SRUC: a technique for packet transmission on multiple access channels", Proceedings of the International Conference on Computer Communications, Kyoto, Japan, pp. 601-607, September 1978.

[40] M.B. Pursley and S.D. Sandgerg "Delay and throughput for three transmission schemes in packet radio networks", IEEE transactions on communications, Vol.37, No. 12, pp. 1264-1274, December 1989.

[41] A.W. Lam and D.V. Sarwate "Time-Hopping and Frequency-Hopping Multiple-Access Packet Communications", IEEE transactions on communications, Vol.38, N0. 6, pp. 875-888, April 1990.

[42] V.O.K. Li and T-Y. Yan "Adaptive mobile access protocol (AMAP) for the message service of a land mobile satellite experiment (MSAT-X)", IEEE journal on selected areas in communications, Vol. SAC-2, pp. 621-627, July 1984.

[43] V.O.K. Li and T-Y. Yan "An integrated voice and data multiple-access scheme for a land-mobile satellite system", Proceedings of the IEEE, Vol. 72, No. 11, pp. 1611-1619, November 1984.

[44] V.O.K. Li and T-Y. Yan "A reliable pipeline protocol for the message service of the land-mobile satellite experiment", IEEE journal on selected areas in communications, Vol. SAC-5, No. 4, pp. 637-647, May 1987.

[45] L.C. Palmer and P.Y. Chang "Simulation of a random-access-with-notification protocol for VSAT applications", Comsat technical review, vol. 18, No. 1, pp. 21-53, Spring 1988.

[46] I. Jacobs et al. "CPODA a demand assigned protocol for satnet", Proceedings 5th data communication symposium, Snowbird, Utah, September 1977.

[47] Qnap2 reference manual, Bull and INRIA, 1984.