DELFT UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering

Telecommunications and Traffic-Control Systems Group

Unslotted Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications

H.L.A. Le

Type:Thesis ReportSize:xiii + 147 + 33 pagesDate:September 5th, 1995

Graduation Professor: Mentors: Prof. Dr. R. Prasad Prof. Dr. R. Prasad Ir. J.A.M. Nijhof Ir. H.R.R. van Roosmalen A-664 November 1994 - October 1995

Assignment number: Period:

eriod.

Abstract

A hybrid Code Division Multiple Access / Inhibit Sense Multiple Access (CDMA/ISMA) protocol has been proposed as an effective multiple access scheme for Indoor Wireless Computer Communications. This new protocol combines the advantages of both CDMA and ISMA into one protocol. On the one hand the ISMA protocol introduces a limitation to the number of simultaneous accesses to the transmission channel. On the other hand the CDMA protocol introduces an improvement to the packet survival chance.

It is shown that the performance of the hybrid protocol is indeed better than CDMA only. In addition, code sharing can be applied to reduce hardware cost. The slotted hybrid CDMA/ISMA protocol using the *p-persistent* ISMA scheme has already been analysed. In this report, we investigate the hybrid protocol using the *unslotted non-persistent* ISMA scheme. The analysis is done for a star-connected multiple access wireless computer network. The performance comparison between the slotted and unslotted hybrid CDMA/ISMA protocol is evaluated in terms of throughput and delay using computer simulation and mathematical analysis.

Indexing Terms:

Code Division Multiple Access (CDMA), Inhibit Sense Multiple Access (ISMA), Spread Spectrum, Data Communications, Rayleigh Fading.

iv

SUMMARY

Compared to the fixed wired systems, Indoor Wireless Communication offers advantages such as time and cost saving, reduction or elimination of wiring and rewiring and increased flexibility and efficiency. However, the scarce electromagnetic spectrum and the hostile transmission medium oblige us to choose an appropriate multiple access protocol for the indoor wireless communication.

Herefore, a hybrid Code Division Multiple Access (CDMA) / Inhibit Sense Multiple Access (ISMA) protocol has been proposed. This protocol combines the advantages of both CDMA and ISMA into one protocol. The ISMA protocol limits the number of simultaneous transmissions of data packets while CDMA improves the survival chances of the transmitted packets. Furthermore, code sharing can be applied with a reasonable penalty in performance degradation.

We distinguish two types of hybrid protocols: slotted and unslotted. The slotted ppersistent hybrid CDMA/ISMA protocol has already been investigated and reported on. This work concentrates on the unslotted non-persistent hybrid CDMA/ISMA protocol.

The performance is evaluated in terms of throughput and delay in relationship to the offered traffic in a star-connected wireless computer network. The throughput is a measure for the efficiency of the protocol and the delay gives the average delay the data packets suffer before they are correctly received. We adopt two approaches to derive the performance of the hybrid protocol. First, we set up a computer simulation program and next, a mathematical model is derived. Then the results are compared.

As can be expected, comparison between the slotted and unslotted protocol shows that for low offered traffic the delay for the unslotted protocol is better than the slotted one. For high traffic it is the other way round. The hybrid protocol performs very well in case the propagation delay is not a concern. If the propagation delay is too high, the performance drops quickly. The results also show that code sharing is a good option for the hybrid protocol.

SAMENVATTING

Vergeleken met de kabelsystemen biedt binnenshuis draadloze communicatie voordelen zoals tijd- en kostenbesparing, reductie of eliminatie van bekabeling en herbekabeling en toename van flexibiliteit en efficiëntie. Echter, de schaarste van de elektromagnetische spectrum en de slechte transmissiemedium verplicht ons om een geschikte "multiple access" protocol voor de binnenhuis draadloze communicatie te vinden.

Hiervoor is een hybride Code Division Multiple Access (CDMA)/ Inhibit Sense Multiple Access (ISMA) protocol voorgesteld. Dit protocol combineert de voordelen van zowel CDMA als ISMA in één protocol. Het ISMA protocol beperkt het aantal gelijktijdige transmissies van datapakketten terwijl CDMA de overlevingskansen van de gezonden pakketten verbetert. Bovendien kan hergebruik van codes toegepast worden met een redelijke degradatie in de prestatie.

Wij onderscheiden twee types hybride protocollen: 'slotted' en 'unslotted'. De slotted p-persistent hybride CDMA/ISMA protocol was al onderzocht en gerapporteerd. Dit werk concentreert zich op de non-persistent hybride CDMA/ISMA protocol.

De prestatie is geëvalueerd in termen van doorstroming en vertraging in relatie tot het aangeboden verkeer in een draadloze sterverbindingsnetwerk. De doorstroming is een maat voor de efficiëntie van de protocol en de vertraging geeft de gemiddelde vertraging van de datapakketten voordat zij correct worden ontvangen. We gebruiken twee benaderingswijzen om de prestatie van de hybride protocol te achterhalen. We zetten eerst een computer simulatie programma op en daarna wordt een mathematische model afgeleid. Daarna worden de resultaten met elkaar vergeleken.

Zoals verwacht laat de vergelijking tussen de slotted en de unslotted protocol zien dat voor laag aangeboden verkeer, de vertraging voor de unslotted protocol beter is dan die van de slotted protocol. Voor hoog aangeboden verkeer is het net andersom. De hybride protocol presteert goed in het geval wanneer de propagatievertraging geen rol speelt. Als de propagatievertraging te hoog is, neemt de prestatie snel af. De resultaten laten ook zien dat hergebruik van codes een goede optie is voor hybride protocol.

PREFACE

This report is the results of my thesis work performed at the Department of Telecommunications and Traffic Control Systems, Faculty of Electrical Engineering, Delft University of Technology.

I would like to thank all my colleage students at the department for the friendly and nice atmosphere and the many joyable discussions during the breaks. Of course, this thesis work would not have been possible without the valuable advices, support and fruitful cooperation I have received from my mentors. This is why I specially like to thank R. Prasad, J.A.M. Nijhof and H.R.R. van Roosmalen.

I'd like to dedicate this report to my parents and Hong Hanh for their encouragements, motivation, support and guidance.

Huy Linh Anh Le

Delft, September 5th 1995

LIST OF SYMBOLS AND ABBREVIATIONS

a_k	Code waveform
A	Carrier signal amplitude
b_k	The <i>k</i> th user's information signal
BW _{RF}	RF bandwidth
с	Speed of light
С	Capacity in bits per second
d	Inhibit delay fraction
E_b	Energy per bit
f	Signal frequency
g_T	Transmitter gain
g_R	Receiver gain
G	Offered traffic
G_p	Processing gain
K	Number of users
Ka	Number of antennas
1	Distance between the transmitter and the receiver
L	Number of resolvable paths
L_p	Packet length
Μ	Order of diversity
n(t)	White Gaussian noise
N	Noise power
M_{max}	Maximum order of diversity
P_{tr}	Probability that a data packet is transmitted in case the channel is sensed
	free (only for slotted ISMA protocol)
P_R	Received power
P_T	Transmitted power
R _{data}	Data rate

r(t)	Received signal
$s_k(t)$	Transmitted signal
S	Signal power
t _c	Cycle time
T_b	Bit duration
T_c	Chip duration
T_{IA}	inter-arrival time
T_m	The rms delay spread
T_{pd}	Packet duration
W	Bandwidth in hertz
x	The number of arrivals during the inhibit delay fraction
α	Random delay parameter
β	Path gain
γ	Path phase
λ	Arrival rate
τ	Path delay
μ	Attenuation parameter
ω _c	Carrier frequency
AWGN	Additive White Gaussian Noise
CDF	Cumulative Distribution Function
CDMA	Code Division Multiple Access
CSMA	Carrier Sense Multiple Access
DPSK	Differential Phase Shift Keying
DS	Direct Sequence
FDMA	Frequency Division Multiple Access
FH	Frequency Hopping
ISMA	Inhibit Sense Multiple Access
IWC	Indoor Wireless Communications
LOS	Line Of Sight
PDF	Probability Density Function

ix

PN	Pseudo Noise
PRMA	Packet Reservation Multiple Access
PSD	Program Structure Diagram
RF	Radio Frequency
SDL	Specification and Description Language
SIR	Signal to Interference Ratio
TDMA	Time Division Multiple Access
TH	Time Hopping
TP	Transmission Period
IP	Idle Period

х

CONTENTS

ABSTRACTiv		
SUMMARYv		
SAMENVATTINGvi		
PREFACEvii		
LIST OF SYMBOLS AND ABBREVIATIONSviii		
1. INTRODUCTION1		
2. SYSTEM DESCRIPTION		
2.1 Multiple Access Protocols		
2.1.1 Contentionless Multiple Access Scheme		
2.1.2 Contention Multiple Access Scheme		
2.2 Fixed Assigned Contentionless Multiple Access Protocols7		
2.2.1 Frequency Division Multiple Access		
2.2.2 Time Division Multiple Access		
2.3 Code Division Multiple Access		
2.3.1 Processing Gain		
2.3.2 Properties of Spread Spectrum		
2.3.3 CDMA Techniques11		
2.4 Repeated Random Access Protocols		
2.4.1 The Slotted P-Persistent ISMA Protocol12		
2.4.2 The Unslotted Non-Persistent ISMA Protocol14		
2.5 Description of the Hybrid CDMA/ISMA Protocol		
2.6 Assumptions17		
2.6.1 Arrival Process		

2.6.2 Signal and Channel Characteristics	17
2.6.3 Network Configuration	
3. BIT ERROR RATE IN RAYLEIGH FADING CHANNEL	19
3.1 Transmitter Model	19
3.2 Channel Model	20
3.3 Receiver Model	22
4. COMPUTER SIMULATION	29
4.1 Simulation Techniques	29
4.2 Simulation Program	31
4.2.1 Main Structure	31
4.2.2 Event List	32
4.2.3 Event Types	32
4.3 The Negative Exponential Distribution	36
4.3.1 Proof of the Inverse Transform Method	
4.3.2 Generation of the Random Numbers for a Negative	
Exponential Distribution	
5. MATHEMATICAL ANALYSIS	41
5.1 Throughput	
5.2 Delay	50
5.3 Practical Implementation Aspect	
6. RESULTS	55
6.1 Comparison between Slotted and Unslotted Protocol	55
6.2 Comparison between Simulation and Mathematical Analysis	58
7. CONCLUSIONS AND RECOMMENDATIONS	69
7.1 Conclusions	69
7.2 Recommendations	70
REFERENCES	

APPENDIX A		
"Unslotted Hybrid CDMA/ISMA Protocol for Indoor Wireless		
Computer Communications"		
To be published in IEEE Proceedings of Third Symposium on		
Communications and Vehicular Technology in the Benelux		
APPENDIX B		
"Indoor Wireless Computer Communications using Unslotted		
Hybrid CDMA/ISMA Protocol"		
Submitted for participation in The International Conference on		
Communications (ICC '96)		
APPENDIX C		
"Unslotted Hybrid CDMA/ISMA Protocol for Indoor Wireless		
Computer Communications"		
Submitted for publication in IEEE Journal on Selected Areas of		
Communications		
APPENDIX D		

Listings of Computer Programs

1. INTRODUCTION

Indoor Wireless Communication (IWC) offers companies a flexibility not available in the fixed wired systems. For example it is not necessary to bring the network down for several minutes to add, remove or move the terminals to other places.

Suppose the company employs a group of representatives, who are most of the time in 'the field' for negotiation with customers and armed with powerful portable computers. When the representatives are back in the office, they can join in the network any time they like it to. In case of the fixed wired systems, it is necessary to perform a routine exercise like plugging and unplugging which can cause network failures because of bad connections. If the computers can communicate with each other wirelessly, they can participate in the network without problems concerning network connections. Time-consuming activities like rearranging the office to account for network connections is history. We can conclude that IWC offer increased flexibility and efficiency, reduction of wiring and down time of services and time and cost saving. Before we can make use of these advantages, we have to solve the problems of the scarce electromagnetic spectrum and the hostile transmission medium first.

Because the electromagnetic spectrum is scarce, it must be used as efficiently as possible. Therefore, it is necessary to solve the problem of the scarce bandwidth. Universities, research laboratories and industries are putting forward enormous effort to investigate new system concepts that can improve the spectrum efficiency. A very important subject with respect to the spectrum efficiency is the choice of a multiple access technique.

Indoor wireless office communication is our main research field. The office system we focus on consists of a building in which users work together in groups. The participants generate terminal traffic. Terminals communicate with each other by radio transmission using a random access protocol. Terminals might not detect each other's transmission in radio communications. It can easily happen that two users are hidden from each other by some obstacle, in which case they are not aware of each other's transmission and collisions can occur. This is called the *hidden terminal problem*. This results in severe performance degradation. The introduction of a central base station can alleviate this problem by instructing it to send a busy tone to all participating terminals to forbid new transmissions when a transmission is going on. Still, a situation can occur in which two or more terminals simultaneously start their transmission, resulting in a collision. However, a great reduction in the number of simultaneous transmissions is achieved by the introduction of a central base station. This concept is called ISMA [1]-[4].

If we could somehow increase the survival chances of colliding packets, we could improve protocol performance. The near-far effect is one way to achieve this. A packet may 'capture' the receiver if its received power is much stronger than its competitors. This can happen when terminals use the same transmission power, but are located at different distances from the receiver. Because the 'near terminals' have better performance compared to the 'far terminals' (due to the better survival chance of the packets), the near-far effect introduces an unfair element among the terminals. Perfect power control, in which the transmitted power of the terminals are adjusted such that their received powers are all equal, can eliminate the near-far effect described above. To increase the survival chances of colliding packets, we use the CDMA concept. Herefore, each terminal multiplies a specific code sequence to the data sequence before transmission. The receiver uses the same code sequence to gain back the data sequence. The receiver can lock into the first incoming packet and receive this packet correctly even when collisions occur. The performance of CDMA has been reported in a number of publications, e.g. [5]-[10].

The hybrid CDMA/ISMA protocol combines the advantages of both CDMA and ISMA into one protocol. The advantages of CDMA and ISMA are the improvement of the survival chance of data packets and the limitation of contention in the channel. Code sharing can also be applied. This is an important aspect because the number of distinct useful transmission codes is limited, especially for short code length.

In [11] and [12], the hybrid protocol combines Direct Sequence CDMA with *slotted* p-persistent ISMA. In this report, we investigate the performance of the hybrid CDMA/ISMA protocol using the *unslotted* non-persistent ISMA scheme. It is expected that for low traffic, the delay of the unslotted protocol improves because when a terminal has a data packet to send, it does not have to wait until the start of the next time slot. In addition, even when data packets collide, there is a probability that the data is received correctly due to the use of CDMA. This is the strength of the hybrid CDMA/ISMA protocol. By using the *unslotted non-persistent* ISMA scheme, it is expected that we take more advantage of this strength.

2

Fading is a result of the propagation of the transmitted signal through several paths. In this report, we model the channel as a Rayleigh fading channel. This assumption is valid if the received signal is composed of faded paths with more or less equal power. This means that a line of sight path is absent. For signal modulation, we employ Differential Phase Shift Keying (DPSK).

The performance of the hybrid CDMA/ISMA protocol is measured in terms of throughput and delay in relationship with the offered traffic. To this end, we follow two ways to derive the protocol's performance. First, a computer simulation program is set up. Second, we develop closed form formulas by mathematical analysis. To facilitate the calculations, we adopt a symmetrical star-connected network for our analysis.

This report is organised as follows. Chapter 2 gives the system description. The derivation of the DPSK bit error rate in Rayleigh fading channel is then given in Chapter 3. Chapter 4 describes the performance analysis of the hybrid protocol by simulation. The mathematical analysis then follows in Chapter 5. Subsequently, in Chapter 6, the results will be shown. Finally, conclusions and recommendations are drawn in Chapter 7.

Unslotted Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications

4

2. SYSTEM DESCRIPTION

This chapter introduces the system description as will be used in the analysis. Special attention will be given to CDMA and ISMA, the basic schemes for the hybrid CDMA/ISMA protocol. Also important assumptions with respect to the hybrid CDMA/ISMA protocol will be discussed. This chapter starts with a general discussion of the different types of multiple access protocols.

2.1 Multiple Access Protocols

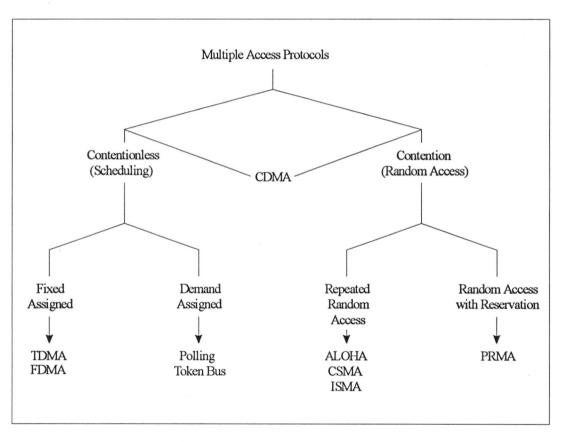


Fig. 2.1: Classification of the multiple access protocols

Fig. 2.1 depicts the classification of the multiple access protocols. Multiple access protocols can be divided in two classes, *contentionless* (or scheduling) and *contention* (or random

access) [13]. The contentionless class of protocols do not allow contention in the channel. The users are scheduled to transmit in an orderly manner. On the other hand, the contention class does allow contention in the channel. For this class of protocol, collisions can occur because users may access the channel at the same time. CDMA is a protocol which belongs to both classes. This protocol will be described in section 2.3.

2.1.1 Contentionless Multiple Access Scheme

Among the class of contentionless we reckon protocols which avoid the simultaneous access of the channel by two or more users. There are two ways to do this. The first *fixed assigned* method assigns to each user a fix amount of the channel's capacity, independent of his activity. If a user is not active, then his capacity is wasted because it can not be used by others. Time Division Multiple Access (TDMA) and Frequency Division Multiple Access (FDMA) belong to this method. *Demand assigned* is the second method in the class of contentionless multiple access protocols. This method only assigns channel's capacity to a user if he has something to transmit. We consider polling and token bus to be demand assigned methods.

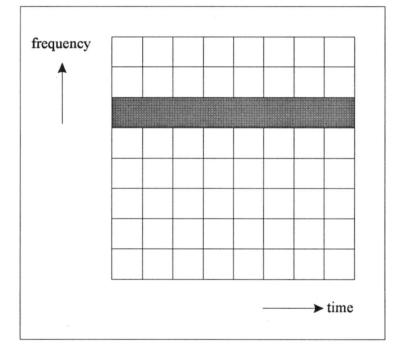
2.1.2 Contention Multiple Access Scheme

In contrast to the contentionless multiple access scheme where transmissions are organised orderly, transmissions of the class of contention multiple access protocols are not scheduled in an orderly manner. A user does not know about the intention of other users. At most, an ongoing transmission can be detected. This class of protocols can be further subdivided into two groups, *repeated random access* and *random access with reservation*. The repeated random access method uses retransmissions to resolve conflicts. We consider ALOHA, Carrier Sense Multiple Access (CSMA) and Inhibit Sense Multiple Access (ISMA) to belong to this class of protocols. With the random access with reservation scheme, only the first transmission suffers from contention. Once a user has access to the channel, capacity is reserved to this user and he is the only one who may use this. If this user is idle for some

time, then the capacity is reclaimed. Packet Reservation Multiple Access (PRMA) belongs to this type of protocol.

2.2 Fixed Assigned Contentionless Multiple Access Protocols

In this section, the fixed assigned contentionless multiple access protocols (TDMA and FDMA) will be discussed in more detail.



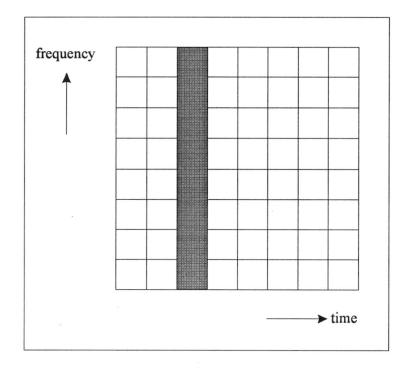
2.2.1 Frequency Division Multiple Access

Fig. 2.2: Principle of FDMA

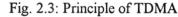
The classical method of providing multiple access capability is FDMA. In FDMA, each user assigned is a particular frequency band. An assigned frequency band can only be used by a particular user all the time and everybody else is excluded from this band (Fig. 2.2).

When all frequency bands are occupied, the system has reached its capacity and no further users may be added.

2.2.2 Time Division Multiple Access



Α more recent technique for providing multiple access is TDMA. TDMA. In the time is divided into equal frames and the frames on their turn are divided into equal time slots. Each user is assigned а particular time slot within time a frame, and during



this time slot, transmits a portion of a message by any standard digital technique. Each user is the sole 'owner' of his particular assigned time slot (Fig. 2.3). Time-division multiple access is an important application for satellite communications as well as ground-based digital communication systems. Again, however, when all time slots are occupied, the system is operating at capacity and no more users can be added.

2.3 Code Division Multiple Access

Code division multiple access (CDMA) is always accomplished by means of spread spectrum. In this system each user is assigned a particular code. Each user employs his own code to spread the signal into a wide band signal. Fig. 2.4 shows the transformation of the original data signal into a spread spectrum signal. The horizontal axis represents the frequency while the vertical axis depicts the power spectral density. At the receiver side, the same code is used to transform the wide band signal back to the original signal. The cross

correlation between users can be decreased by a proper choice of codes. If we manage to realise a low cross correlation, then we can bring down the interference between users [14]. In contrast to the previously mentioned methods of multiple access, code-division multiple access does not have any sharply defined system capacity. As the number of users increases, the signal-to-interference ratio (SIR) becomes smaller and there is a gradual degradation in performance until the SIR falls below a certain threshold. Thus the system can tolerate significant amounts of overload if the users are willing to tolerate poorer performance.

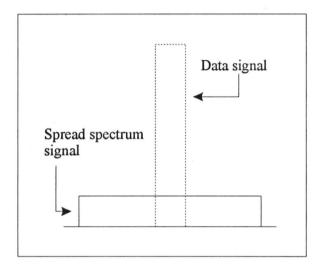


Fig. 2.4: Spreading of the data signal

An additional advantage of CDMA is that the messages intended for one user are not readily decodable by other users, because they may not know the proper codes or have the equipment for generating the appropriate reference signals. Thus there is a privacy feature that is not available in other multiple-access techniques.

The Shannon formula, which express the channel capacity, is the basis of spread spectrum technology:

$$C = W \cdot \log_2\left(1 + \frac{S}{N}\right) \tag{2.1}$$

where C =Capacity in bits per second,

W = Bandwidth in hertz,

S = Signal power,

N =Noise power.

2. SYSTEM DESCRIPTION

Equation (2.1) shows the trade-off between the transmitted power and the signal bandwidth. Spread spectrum is based on the increase of the signal bandwidth, thereby allowing the signal to noise ratio to decrease to achieve a certain channel capacity.

2.3.1 Processing Gain

The most commonly used quantity in describing or specifying spread spectrum systems is that of processing gain (G_p) :

$$G_p = \frac{BW_{RF}}{R_{data}}$$
(2.2)

Where the RF bandwidth (BW_{RF}) is the bandwidth of the transmitted spread spectrum signal and the information rate (R_{data}) is the data rate in the information baseband channel. So, the ratio between transmitted and original bandwidth is defined as the processing gain.

2.3.2 Properties of Spread Spectrum

An application of spread spectrum that is of particular interest in mobile communications is the ability of a wide-band signal to resist the effects of multipath fading. A property of multipath fading is that frequencies separated by only a few hundred kilohertz may fade essentially independently. Thus at any given time when the signal has a large bandwidth, only a small portion of the bandwidth will be in a fade. The average received signal power thus can be made more nearly constant than it would be for a narrow-band signal. This resistance to fading is an important consideration in the potential application of spread spectrum to mobile communication situations.

Another important element employed in the design of spread spectrum signals is pseudo-randomness. It makes the signals appear similar to random noise and difficult to demodulate by receivers other than the intended ones. A message may be hidden in the background noise by spreading its bandwidth with coding and transmitting the resultant signal at a low average power. Because of its low power level, the transmitted signal is said to be 'covert'. It has a low probability of being intercepted (detected) by a casual listener.

Message privacy may be obtained by superimposing a pseudo-random pattern on a transmitted message. The message can be demodulated by the intended receivers that know the pseudo-random pattern or key used at the transmitter but not by any other receivers that do not have knowledge of the key. Resuming the advantages spread spectrum offers we come to the following:

- Interference rejection;
- Anti-multipath fading (frequency diversity);
- Low probability of intercept;
- Secrecy;
- Multiple access capability.

2.3.3 CDMA Techniques

We can distinguish between three main types of spread spectrum techniques: Direct Sequence (DS), Frequency Hopping (FH) and Time Hopping (TH).

For the direct sequence CDMA method, a digital information stream is modulated by a binary code sequence (Fig. 2.5). The recovering of the signal at the receiver is done by correlating the received signal with the original code sequence used by the transmitter. In a frequency hopped spread spectrum communications system, the available channel bandwidth is subdivided into a large number of adjacent frequency bands. In any signalling interval, the transmitted signal occupies one or more of the available frequency bands. The selection of the frequency band(s) in each signalling interval is made pseudo-randomly according to the output from a pseudo noise (PN) generator. For the time hopping method, the time axis is divided into frames, which in turn are subdivided into time slots. The user transmits the data in a different time slot every frame, using all the available bandwidth.

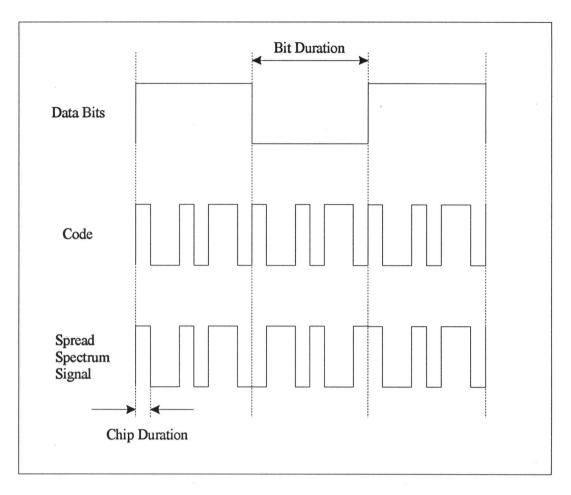


Fig. 2.5: Direct Sequence CDMA signal spreading

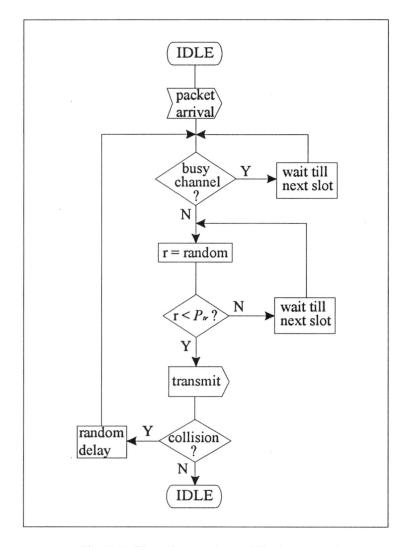
2.4 Repeated Random Access Protocols

ALOHA, CSMA and ISMA schemes belong to this type of repeated random access protocols. We only discuss the slotted p-persistent ISMA and the unslotted non-persistent ISMA protocol in this section.

2.4.1 The Slotted P-Persistent ISMA Protocol

The Specification and Description Language (SDL) diagram of the slotted p-persistent ISMA protocol is depicted in Fig. 2.6. Before transmitting the data packet to the receiver,

the terminal first listens to the channel to detect whether there is a busy tone transmitted by the base station. The base station broadcasts a busy tone to indicate that the channel is busy because there is a transmission going on.



If there is a busy tone, indicating that the channel is busy, the terminal postpones its transmission to the next time slot. Otherwise, if the channel is free, the terminal draws a number to see if it is permitted to send its data packet. The probability that a terminal succeeds is P_{tr} . (If $P_{tr}=0.1$ then the protocol is called 0.1-persistent ISMA). In case the terminal is unlucky, it waits till the next

Fig. 2.6: Slotted p-persistent ISMA protocol

time slot and draws a number again to see if it's permitted to send. This mechanism is built in to reduce the number of simultaneous transmissions.

If the terminal is permitted to send, it transmits its data packet to the receiver. It is possible that two or more terminals transmit in the same time slot. In this case a collision occurs and the data packets are damaged. The terminals then wait a random delay and start all over again (see also Fig. 2.6).

2.4.2 The Unslotted Non-Persistent ISMA Protocol

The SDL-diagram of the unslotted non-persistent ISMA protocol is shown in Fig. 2.7. This ISMA protocol is unslotted. There are two major differences between this unslotted and the previously in section 2.4.1 described slotted protocol.

The first difference is that the terminal waits a random delay when it senses that the channel is busy. Because the time axis is not divided into time slots, a random delay has to be introduced. After the random delay, terminals are allowed to sense the channel again to see if it's free.

The second difference is related to the first one. In the unslotted scheme, the terminals immediately transmit if the channel is free. The terminals already wait a random delay in case the channel is busy. So, if the channel is free, the number of terminals attempting to transmit is already reduced. It is not necessary to let the terminals draw a random number again to see if they may transmit.

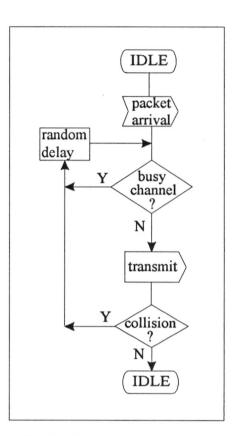


Fig. 2.7: Unslotted non-persistent ISMA

Although the number of collisions is greatly reduced with ISMA, collisions still can occur. In this unslotted ISMA scheme, collisions may occur due to multiple terminals transmitting data packets during an interval called the *inhibit delay fraction d*. This interval *d* is necessary to switch the busy tone from 'idle' to 'busy'. The reverse interval from 'busy' to 'idle' is denoted by *d'*. The inhibit delay fraction is normalised to the packet length, resulting in $0 \le d < 1$ (Fig. 2.8). In this report we assume that d'=d.

The analysis of the hybrid protocol using slotted p-persistent ISMA has already been performed ([11] and [12]). We now concentrate on the hybrid protocol using the unslotted

non-persistent ISMA scheme. In case of the unslotted protocol, we expect that for low traffic, the average packet delay can be reduced because terminals do not have to wait to the start of the new time slot.

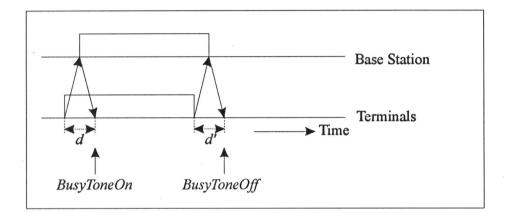


Fig. 2.8: Inhibit delay fraction d

2.5 Description of the Hybrid CDMA/ISMA Protocol

Users behind terminals generate data traffic, which is divided in data packets. The data packets arrive at terminals, who take care of the correct delivery. We assume that the arrivals of these packets are generated by a Poisson process. The state of the terminals can either be free or blocked. At the start, the terminals are in the free state. If a packet arrives at a free terminal, the terminal jumps into the blocked state. In the blocked state, the terminal takes care that the arriving data packet is serviced successfully. In the mean time, the blocked terminal ignores all incoming packets. This is a consequence of the assumption that the terminals don't have buffers for the incoming data packets. We assume that the layer above will handle this correctly. In the blocked state, the terminal uses the unslotted non-persistent ISMA protocol to service its data packets. If a data packet is received erroneously, then the ISMA scheme takes care that this problem is solved. Only when the data packet is received correctly, the terminal leaves the blocked state to jump into the free state. New data packets can then be handled.

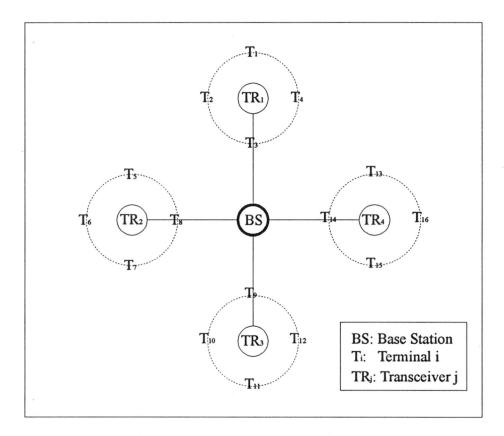


Fig. 2.9: 16-terminal network configuration

Fig. 2.9 depicts an example of the 16-terminal network configuration of the protocol. There is one central base station that is connected to several transceivers by wire. The base station controls the traffic flow with a busy tone that can be detected by all participated terminals. Because all terminals can detect the busy tone, the hidden terminal problem is solved. Several terminals together with one transceiver form a group. Wireless communication is considered to take place between the transceiver and the terminals. The terminals around each transceiver share the same code. In this way the number of codes can be reduced. When a data packet is ready for transmission, a terminal transmits the packet to its transceiver according to the chosen ISMA scheme.

After the arrival of the data packet, the transceiver then simply forwards this packet to the base station which forwards it to all terminals via the transceivers. The destination then receives the packet and decides whether the packet is errorless or not. In case the packet was erroneous, a retransmission must take place. The return channel is not included in our analysis because only the base station makes use of this return channel and therefore contention is not a concern.

2.6 Assumptions

It is impossible and mostly also not necessary to take all complexities of the real life system into account. That is why assumptions have to be made in order to be able to analyse a certain aspect of the system. It is also important to list the assumptions used to indicate under which conditions the generated result is valid. The assumptions we adopted in our work are listed here below. Unless stated otherwise, the values of the parameters we use to obtain the protocol's performance is also given in this section.

2.6.1 Arrival Process

The arrival of data packets at terminals is assumed to be generated by a Poisson process. This means that the inter-arrival times of the data packets at terminals are negative exponentially distributed with λ as parameter. Before a retransmission (caused by a collision or a busy channel) can take place, terminals have to undergo a random delay first (see Fig. 2.7). This random delay is also negative exponentially distributed with α as parameter. α is chosen to be 0.1 and the inhibit delay fraction d is 0.01. The negative exponential distribution will be discussed in Chapter 4.

2.6.2 Signal and Channel Characteristics

After Differential Phase Shift Keying (DPSK) modulation, the signal is spread with a Gold code. The length of this code is chosen to be 31 or 127. The signal is then transmitted, modulated on a 1.7 GHz carrier. The data rate is arbitrarily chosen to be 256.1024 b/s (0.26 Mb/s). The packet length is 64 bits. The delay spread is supposed to be 100 ns and the SNR is 20 dB. Perfect power control is assumed to assure that signals from terminals within the same group arrive at the transceiver with the same power.

The far field model [15] is chosen to describe the signal attenuation. Given the transmitted power P_T , the received power P_R can be expressed by the following equation:

$$P_R = \frac{g_T g_R}{\left(\frac{4\pi f l}{c}\right)^{\mu}} P_T$$
(2.3)

where g_T and g_R are respectively the transmitter and the receiver gains, f the signal frequency, l the distance between the transmitter and the receiver, c the speed of light and μ is the attenuation parameter. This attenuation parameter is chosen equal to 2, corresponding to free space propagation.

Fading is the result of the propagation of the transmitted signal through several paths. The channel is modelled as a Rayleigh fading channel. The Rayleigh fading channel model is valid when each path contributes the same amount of energy to the composite received signal. A Line of Sight (LOS) path is therefore assumed to be absent.

2.6.3 Network Configuration

An example of a 16-terminal network configuration that we adopted in this paper is shown in Fig. 2.9. The terminals are clustered around distributed transceivers at a fixed distance (5 m). The distributed transceivers are clustered around the base station at a fixed distance (30 m). Those distances are kept fixed throughout the analysis. The number of terminals within a transceiver group (also called group size) is chosen as a power of two. For a certain fixed number of participated terminals, we have to halve the group size if we want to double the number of codes (the number of codes also equals the number of transceivers). The comparison is fair in this manner, because when we want to investigate the effect of the number of codes on the performance, we have to keep the number of participating terminals fixed. The number of terminals is 32 unless stated otherwise.

3. BIT ERROR RATE IN RAYLEIGH FADING CHANNEL

The bit error rate for direct-sequence spread spectrum with Differential Phase Shift Keying (DPSK) modulation and diversity in Rayleigh fading channel is evaluated in this chapter. The transmitter, channel and receiver model will be presented in the subsequent sections and are similar to those in [16].

3.1 Transmitter Model

The number of active users in the indoor wireless communications system is denoted by K. Fig. 3.1 depicts the CDMA transmission model. User *i* applies code *i* to communicate with the base station.

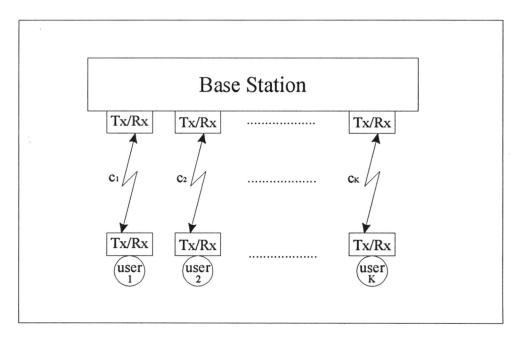


Fig. 3.1: CDMA transmission channel

The kth user's information signal b_k is the differentially encoded binary data sequence of rectangular pulses of width T_b . To spread each data bit, a code waveform a_k of N rectangular pulses of width T_c is used. In literature, a_k is also called the chip waveform. There is one period of code sequence per data bit resulting in $T_b = NT_c$. The chip and data waveforms for the kth user are denoted by $a_k(t)$ and $b_k(t)$ respectively and are given by:

$$a_{k}(t) = \sum_{i} a_{k}^{i} P_{T_{c}}(t - iT_{c}), \ a_{k}^{i} \in \{-1, 1\}$$
(3.1)

$$b_k(t) = \sum_{j=-\infty}^{\infty} b_k^j P_{T_b}(t - jT_b), \ b_k^j \in \{-1, 1\}$$
(3.2)

Here a_k^i represents the *i*th chip value of the *k*th user, b_k^j the *k*th user data bit at the *j*th timing interval and $a_k^i = a_k^{i+N}$ for all *i*; $P_W(t)$ is a rectangular pulse of unit height and width W.

Spreading is achieved by multiplying (or modulo-2 adding) the direct-sequence code to the data signal. The spread signal is then modulated onto the RF carrier signal $A\cos(\omega_c t + \theta_k)$. The carrier frequency is the same for all users and is denoted by ω_c , while the carrier phase for the *k*th user is θ_k and *A* is the carrier level (the energy per bit E_b equals $A^2T_b/2$). It is further assumed that $\omega_c T_b = 2l\pi$, in which *l* is an integer. The transmitted signal for the *k*th user $s_k(t)$ is given by:

$$s_k(t) = Aa_k(t)b_k(t)\cos(\omega_c t + \theta_k)$$
(3.3)

3.2 Channel Model

The emitted information reaches the destination not only via the line of sight (LOS) path but also via reflections by buildings and walls. In case the line of sight path dominates the other reflected signals, the sum of the signals at the receiver can be modeled by the Rician distribution. When all the signals are received with approximately the same power, the receiver power distribution can be modeled by the Rayleigh distribution (Fig. 3.2).

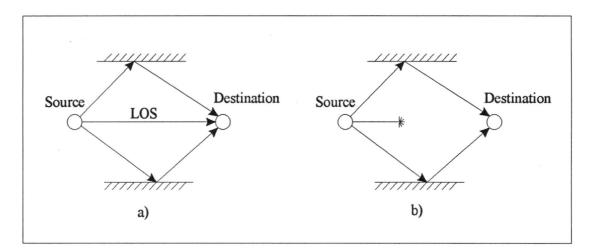


Fig 3.2: Fading channels a) Rice; b) Rayleigh

For our work we model our channel as a Rayleigh fading channel. The complex lowpass equivalent impulse response of the multipath Rayleigh fading channel for the link between the kth user and the base station (in our case it is the transceiver) is given by:

$$h_{k}(\tau) = \sum_{l=1}^{L} \beta_{lk} \delta(\tau - \tau_{lk}) e^{j\gamma lk}$$
(3.4)

where β is the path gain and is Rayleigh distributed, τ is the path delay and γ is the path phase. The index *lk* refers to the *l*th path of the *k*th user, and $j^2 = -1$. $\delta(.)$ is the Dirac delta function and *L* is the number of resolvable paths and is upper bounded by:

$$L = \left\lfloor \frac{T_m}{T_c} \right\rfloor + 1 \tag{3.5}$$

In which $\lfloor x \rfloor$ is the largest integer smaller than or equal to x [12], T_m is the rms delay spread and depends on the size and type of buildings. Reported values are between 20 and 50 ns for small and medium-size office buildings, between 30 and 300 ns for various factory environments, under 100 ns at several university buildings, less than 160 ns over 90% of the

3. BIT ERROR RATE IN RAYLEIGH FADING CHANNEL

area in a shielded building, less than 80 ns in an office building, under 120 ns in a large office building, and up to 200 ns in other large office buildings [17]. For our work we adopted 100 ns as the value for T_m .

We assume that the path delays τ_{lk} are independent random variables and uniformly distributed over $[0, T_b]$. Fading is the result of the propagation of the transmitted signal through several paths. The channel is modeled as a Rayleigh fading channel. The Rayleigh fading channel model is valid when each path contributes the same amount of energy to the composite received signal. A Line of Sight (LOS) path is therefore assumed to be absent (Fig. 3.2). So, the path gain β_{lk} is an independent Rayleigh random variable for each *l* and *k*. The Rayleigh distribution for β_{lk} is given as:

$$P_{\beta_{lk}}(r) = \frac{r}{\rho_{lk}} \exp\left(-\frac{r^2}{2\rho_{lk}}\right), \ r \ge 0$$
(3.6)

In which r is the signal amplitude and $\rho_{lk} = \frac{1}{2} E[\beta_{lk}^2]$ is the average scattered power.

3.3 Receiver Model

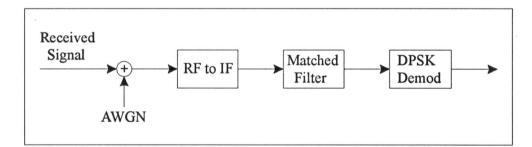


Fig. 3.3: Spread spectrum receiver using DPSK modulation

The receiver consists of a RF to IF converter, a matched filter and a DPSK demodulator (Fig. 3.3). We assume that the received signal r(t) is composed of the contributions of the different users, their different paths and additive white Gaussian noise (AWGN). The receiver input signal at the antenna for a certain user can be written as:

$$r(t) = \sum_{k=1}^{K} \sum_{l=1}^{L} A\beta_{lk} a_k (t - \tau_{lk}) b_k (t - \tau_{lk}) \cos(\omega_c t + \phi_{lk}) + n(t)$$
(3.7)

Here, n(t) is the white Gaussian noise with two-sided power spectral density $N_0/2$ [W/Hz]. We assume that the path phase ϕ_{lk} , given by $(\omega_c \tau_{lk} + \gamma_{lk} + \theta_k)$, is an independent random variable uniformly distributed over $[0, 2\pi]$. In terms of low-pass signals, r(t) and n(t) can also be expressed as:

$$r(t) = x(t)\cos(\omega_c t) - y(t)\sin(\omega_c t)$$
(3.8)

$$n(t) = n_c(t)\cos(\omega_c t) - n_s(t)\sin(\omega_c t)$$
(3.9)

Here, $n_c(t)$ and $n_s(t)$ are the lowpass equivalent components of n(t) and

$$x(t) = \sum_{k=1}^{K} \sum_{l=1}^{L} A\beta_{lk} a_k (t - \tau_{lk}) b_k (t - \tau_{lk}) \cos(\phi_{lk}) + n_c(t)$$
(3.10)

$$y(t) = \sum_{k=1}^{K} \sum_{l=1}^{L} A\beta_{lk} a_k (t - \tau_{lk}) b_k (t - \tau_{lk}) sin(\phi_{lk}) + n_s(t)$$
(3.11)

Selecting user 1 as the reference user, each component of the received signal is multiplied by the direct sequence code associated with user 1. The output (in-phase and quadrature component) of the matched filter of user 1 at the sampling instant $(t = T_b)$ is then given as:

$$g_{x}(T_{b}) = \sum_{k=1}^{K} \sum_{l=1}^{L} A\beta_{lk} \cos(\phi_{lk}) \Big[b_{k}^{-1} R_{1k}(\tau_{lk}) + b_{k}^{0} \hat{R}_{1k}(\tau_{lk}) \Big] + \eta$$
(3.12)

$$g_{y}(T_{b}) = \sum_{k=1}^{K} \sum_{l=1}^{L} A\beta_{lk} \sin(\phi_{lk}) \Big[b_{k}^{-1} R_{1k}(\tau_{lk}) + b_{k}^{0} \hat{R}_{1k}(\tau_{lk}) \Big] + \upsilon$$
(3.13)

3. BIT ERROR RATE IN RAYLEIGH FADING CHANNEL

Where g_x and g_y are the in-phase and the quadrature components, b_k^{-1} the previous and b_k^0 the current data bit. The noise samples η and υ are independent, zero-mean Gaussian random variables with identical variance $\sigma_n^2 = N_0 T_b$. Further,

$$R_{1k}(\tau) = \int_0^{\tau} a_k(t-\tau) a_1(t) dt$$
(3.14)

$$\hat{R}_{1k}(\tau) = \int_{\tau}^{T_b} a_k(t-\tau) a_1(t) dt$$
(3.15)

Let us assume without loss of generality that the receiver synchronizes to the *j*th path of user 1, so that $\tau_{j1} = 0$ and $\phi_{j1} = 0$. All other delay paths constitute interference and are then defined as relative according to this reference. The complex envelope of the signal $z_1 = g_x(T_b) + jg_y(T_b)$ at the current sampling instant is:

$$z_{1} = A\beta_{j1}T_{b}b_{1}^{0} + A\sum_{k=1}^{K} \left(b_{k}^{-1}X_{k} + b_{k}^{0}\hat{X}_{k}\right) + jA\sum_{k=1}^{K} \left(b_{k}^{-1}Y_{k} + b_{k}^{0}\hat{Y}_{k}\right) + \left(\eta_{1} + j\upsilon_{1}\right)$$
(3.16)

For the previous sampling instant, we have something similar

$$z_{-1} = A\beta_{j1}T_{b}b_{1}^{-1} + A\sum_{k=1}^{K} \left(b_{k}^{-2}X_{k} + b_{k}^{-1}\hat{X}_{k}\right) + jA\sum_{k=1}^{K} \left(b_{k}^{-2}Y_{k} + b_{k}^{-1}\hat{Y}_{k}\right) + \left(\eta_{2} + j\upsilon_{2}\right)$$
(3.17)

in which b_k^{-2} is the *k*th user data bit transmitted 2 bits intervals prior to b_k^0 and η_1 , η_2 , υ_1 and υ_2 are the noise variables independent of one another. Further,

$$X_{1} = \sum_{\substack{l=1\\l\neq j}}^{L} R_{11}(\tau_{l1}) \beta_{l1} \cos(\phi_{l1})$$
(3.18)

Unslotted Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications

$$\hat{X}_{1} = \sum_{\substack{l=1\\l\neq j}}^{L} \hat{R}_{11}(\tau_{l1}) \beta_{l1} \cos(\phi_{l1})$$
(3.19)

$$Y_{1} = \sum_{\substack{l=1\\l\neq j}}^{L} R_{11}(\tau_{l1}) \beta_{l1} \sin(\phi_{l1})$$
(3.20)

$$\hat{Y}_{1} = \sum_{\substack{l=1\\l\neq j}}^{L} \hat{R}_{11}(\tau_{l1}) \beta_{l1} \sin(\phi_{l1})$$
(3.21)

For $k \ge 2$,

$$X_{k} = \sum_{l=1}^{L} R_{lk}(\tau_{lk}) \beta_{lk} \cos(\phi_{lk})$$
(3.22)

$$\hat{X}_{k} = \sum_{l=1}^{L} \hat{R}_{lk}(\tau_{lk}) \beta_{lk} \cos(\phi_{lk})$$
(3.23)

$$Y_{k} = \sum_{l=1}^{L} R_{lk}(\tau_{lk}) \beta_{lk} \sin(\phi_{lk})$$
(3.24)

$$\hat{Y}_{k} = \sum_{l=1}^{L} \hat{R}_{lk}(\tau_{lk}) \beta_{lk} \sin(\phi_{lk})$$
(3.25)

The decision variable for DPSK demodulation is

$$\boldsymbol{\xi} = Re\left[\boldsymbol{z}_{1}\boldsymbol{z}_{-1}^{*}\right] \tag{3.26}$$

In which Re[x] denotes the real part of x and * the complex conjugation. Errors occur if the channel attenuation is large. If we somehow provide the receiver with several independently fading signal paths, the probability that all the signal components will fade simultaneously is

3. BIT ERROR RATE IN RAYLEIGH FADING CHANNEL

reduced considerably. This is the basis for diversity techniques. The selection diversity of order M is based on selecting the strongest of the decision variable:

$$\xi_{max}^{M} = max_{i=1,\dots,M}(\xi_i) \tag{3.27}$$

By using multiple antennas, the highest possible order of diversity, i.e. number of paths to choose from, can be increased to $M_{max}=K_aL$ where M_{max} is the maximum order of diversity and K_a is the number of antennas. Designate ξ_{max} as the decision variable obtained from demodulation of the strongest path. The bit error probability in case of selection diversity is defined as

$$P_{be}\underline{\Delta}P_r\left(\xi_{max} < 0 \middle| b_1^0 b_1^{-1} = 1\right)\underline{\Delta}P_r\left(\xi_{max} > 0 \middle| b_1^0 b_1^{-1} = -1\right)$$
(3.28)

If we assume that all path delays are given and β_{max} is correctly selected, the formula for the bit error probability is given by [16]:

$$P_{be|\beta_{max},\{\tau_{lk}\},L} = Q(a,b) - \frac{1}{2} \left(1 + \frac{\mu_{12}}{\sqrt{\mu_{1}\mu_{2}}} \right) I_0(a,b) exp\left(-\frac{a^2 + b^2}{2} \right)$$
(3.29)

where

Q(a,b) is the Marcum Q-function

$$a = \frac{m}{\sqrt{2}} \left| \frac{1}{\sqrt{\mu_1}} - \frac{1}{\sqrt{\mu_2}} \right|$$
(3.30)

$$b = \frac{m}{\sqrt{2}} \left| \frac{1}{\sqrt{\mu_1}} + \frac{1}{\sqrt{\mu_2}} \right|$$
(3.31)

$$m = A\beta_{max}T_b b_1^{\ 0} = A\beta_{max}T_b b_1^{\ -1}$$
(3.32)

$$\mu_{1} = A^{2}E\left[\sum_{k=1}^{K} X_{k}^{2} + \hat{X}_{k}^{2} + Y_{k}^{2} + \hat{Y}_{k}^{2} | \{\tau_{lk}\}, L\right] + 2A^{2}E\left[X_{1}\hat{X}_{1} + Y_{1}\hat{Y}_{1} | \{\tau_{lk}\}, L\right] + 2\sigma_{n}^{2}$$
(3.33)

$$\mu_{2} = A^{2}E\left[\sum_{k=1}^{K} X_{k}^{2} + \hat{X}_{k}^{2} + Y_{k}^{2} + \hat{Y}_{k}^{2} | \{\tau_{lk}\}, L\right] + 2\sigma_{n}^{2}$$
(3.34)

$$\mu_{12} = A^2 E \left[\sum_{k=1}^{K} \left(X_k \hat{X}_k + Y_k \hat{Y}_k \right) + \hat{X}_1^2 + \hat{Y}_1^2 | \{ \tau_{lk} \}, L \right]$$
(3.35)

The conditional expectations in the above expressions can be evaluated as follows:

$$E\left[X_{1}^{2}|\{\tau_{lk}\},L\right] = E\left[Y_{1}^{2}|\{\tau_{lk}\},L\right] = \sum_{\substack{l=1\\l\neq j}}^{L} R_{11}^{2}(\tau_{l1})\rho_{l1}$$
(3.36)

$$E\left[\hat{X}_{1}^{2}|\{\tau_{lk}\},L\right] = E\left[\hat{Y}_{1}^{2}|\{\tau_{lk}\},L\right] = \sum_{\substack{l=1\\l\neq j}}^{L}\hat{R}_{11}^{2}(\tau_{l1})\rho_{l1}$$
(3.37)

$$E\left[X_{1}\hat{X}_{1}|\{\tau_{lk}\},L\right] = E\left[Y_{1}\hat{Y}_{1}|\{\tau_{lk}\},L\right] = \sum_{l=1}^{L} R_{11}(\tau_{l1})\hat{R}_{11}(\tau_{l1})\rho_{l1}$$
(3.38)
$$I \neq j$$

$$E[X_k^2|\{\tau_{lk}\},L] = E[Y_k^2|\{\tau_{lk}\},L] = \sum_{l=1}^L R_{lk}^2(\tau_{lk})\rho_{lk}$$
(3.39)

$$E\left[\hat{X}_{k}^{2}|\{\tau_{lk}\},L\right] = E\left[\hat{Y}_{k}^{2}|\{\tau_{lk}\},L\right] = \sum_{l=1}^{L}\hat{R}_{lk}^{2}(\tau_{lk})\rho_{lk} \qquad ; k \ge 2 \qquad (3.40)$$

$$E\left[X_{k}\hat{X}_{k}|\{\tau_{lk}\},L\right] = E\left[Y_{k}\hat{Y}_{k}|\{\tau_{lk}\},L\right] = \sum_{l=1}^{L}R_{lk}(\tau_{lk})\hat{R}_{lk}(\tau_{lk})\rho_{lk} \quad ;k \ge 2 \quad (3.41)$$

4. COMPUTER SIMULATION

A simulation program has been created to analyse the protocol behaviour. In this chapter we describe how the computer simulation program was set up to measure the performance of the unslotted hybrid CDMA/ISMA protocol. The primary goal of this simulation program is to see whether the performance of the hybrid protocol using the *slotted p-persistent* scheme on the one hand and the *unslotted non-persistent* scheme on the other, differ from each other, and if so in what way. Knowledge in this matter give us meaningful insight that we can use to design our protocol appropriately. This chapter also includes the generation of Poisson data traffic.

4.1 Simulation Techniques

In general, if we want to use a computer simulation to analyse a system, the following steps can be distinguished. The dynamic behaviour of the system is studied by tracing various system states as a function of time and then collecting and analysing the system statistics. The events that change the system state are generated at different points in time, and the passage of time is represented by an internal clock which is incremented and maintained by the simulation program.

The simulation time can be advanced in two ways. The first method is the *interval-oriented* simulation (or the uniform time increment method) where the clock is advanced from time t to $(t + \Delta t)$ where Δt is a uniform fixed time increment. Fig. 4.1a depicts this mechanism. The second method is the *event-oriented simulation* (or the variable time increment method) where the clock is incremented from time t to the next event time t', whatever may be the value of t'. The state changes are made at event time t, the next event time t', and this process is continuously repeated. Thus, only events are represented explicitly in a simulation model and the periods between events are treated as inactive or insignificant and therefore consume no time even though the inter-event activities do consume time in the real world (Fig. 4.1b).

Obviously, method 1 detects the events that occur during the interval $(t, t + \Delta t)$ only at time $(t + \Delta t)$, thereby introducing errors in simulation. Another drawback of this method is that if the interval between two events is very large compared to Δt , then the simulator goes through several unproductive clock increments (during periods of inactivity) and the associated computing effort which will not bring about any change in system states. This fixed time increment method is suitable for the simulation of continuous systems and in particular systems with large numbers of state variables.

The second method involves sorting of event activation times and maintaining an *event list*. In our work we will employ the event-oriented simulation to analyse the unslotted hybrid CDMA/ISMA protocol.

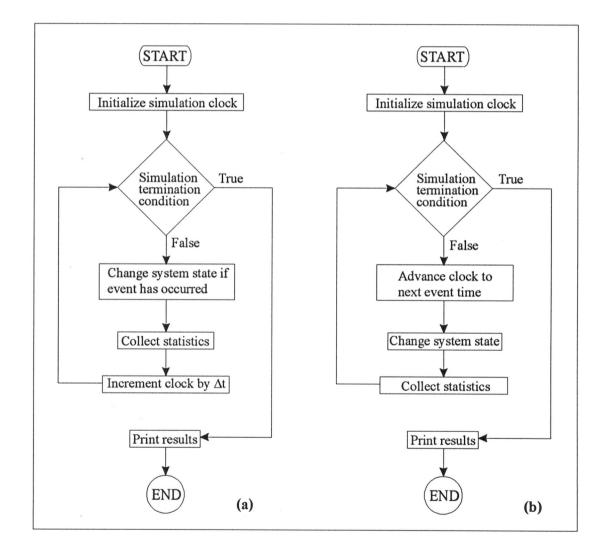


Fig. 4.1: (a) Interval-oriented simulation; (b) Event-oriented simulation

4.2 Simulation Program

This section describes the simulation program in terms of main program structure, event list and event types. The source code can be found in Appendix D.

4.2.1 Main Structure

The Program Structure Diagram (PSD) of the main program is depicted in Fig. 4.2. The simulation program simulates the working of the unslotted hybrid CDMA/ISMA protocol. First of all, the counters have to be reset to the proper values. *Clock* indicates the actual time while the simulation will last *EndTime*. While *Clock* has not reached *EndTime*, the simulation will loop. The loop is divided in four steps as shown in Fig. 4.2. In the first step, the program advances *Clock* to the next event time. This event is the first event to occur in future time. Because the event-oriented simulation is used, only events are represented explicitly in the simulation model. Depending on the event type, the program processes this event. This is the second step. The processing can generate other events or change the system state. In the third step, the generated events and the changed system states will be updated. Finally, statistics collection completes the loop. (The warm up procedure is not shown in Fig. 4.2).

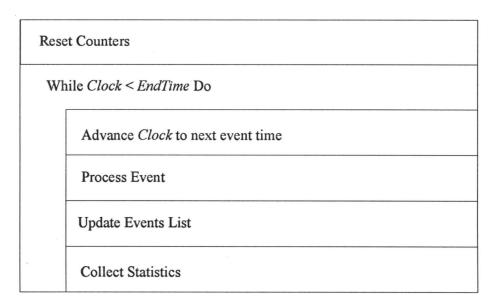


Fig. 4.2: PSD of main program

From Fig. 4.2 it emerges that events and event list play an important role in the main program structure. The whole program is based on the event list. The event list consists of events linked together in an organised manner. Here, organised means in an order in which the events occur. An example of an event list is shown in Fig. 4.3. The event list consists of nodes linked together to form a chain. In the event-oriented simulation this chain symbolises the time axis. Every node represents an event and consists of an event type, the time this event occur, other information and a tail. The tail consists of a pointer pointing to the next node. The last node points nowhere. This is illustrated with the ground symbol.

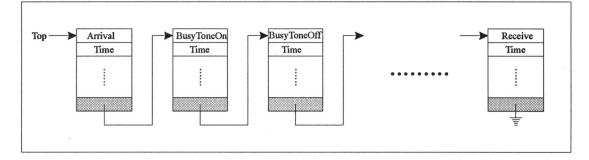


Fig. 4.3: Example of an event list

4.2.3 Event Types

There are five event types: *Arrival*, *Retransmission*, *BusyToneOn*, *BusyToneOff* and *Receive*. From Fig. 4.2 we see that after advancing the clock to the next event, the event will be processed depending on the event type. Those event types together with their corresponding processing are described here below.

Arrival - The event *Arrival* indicates the arrivals of newly arrived packets. This arrival process is modelled by the Poisson process. The inter-arrival times of the packets are negative exponentially distributed. Section 4.3 shows how the generation of this traffic stream is implemented in the simulation program.

The procedure *ProcessArrival* is illustrated in Fig. 4.4. If a new packet arrives, the next packet arrival is generated and put into the event list. This new *Arrival* event has to occur after the inter-arrival time T_{IA} , which depends on the implemented Poisson process with parameter λ . Because the terminals are assumed not be possess any buffers, a new packet only get serviced if this terminal is idle. Provided that this is the case, then the state of the terminal is set to BLOCKED, which means that newly arrived packets will be ignored. Then, collection of the arrival data is performed. *Clock* denotes the current simulation time of the internal clock of the program.

The program makes use of a global variable *BusyTone* that can takes on two values: ON or OFF. In case the *BusyTone* is ON, in which case the terminal is inhibited to send, then it has to wait a random delay before it can try again. Again, data collection must take place, this time it is the retransmission and delay data. If the *BusyTone* is OFF, then the terminal is allowed to send its data packet. First, the terminal state is altered to TRANSMIT. Then, after the inhibit delay fraction *d*, the *BusyTone* is set to ON. Subsequently, the event *Receive* is created. And finally, the delay data is collected for later processing.

Create the next Arrival event at time = $(Clock + T_{LA})$			
T Terminal = IDLE ? F			
Terminal := BLOCKED			
Collect arrival data			
T BusyTone = ON ? F			
Create event Retransmission at time = $(Clock + random delay)$	Terminal := TRANSMIT		
	Create event $BusyToneOn$ at time = ($Clock + d$)		
Collect retransmission data	Create event <i>Receive</i> at time = $(Clock + packet duration + d)$		
Collect delay data			

Fig. 4.4: PSD of procedure ProcessArrival

Retransmission - The event Retransmission indicates the arrivals of the retransmitted packets. The random delay the terminals have to wait before they are allowed to try again is also modelled by the Poisson process. The inter-arrival time of the packets is negative exponentially distributed with parameter α .

Fig. 4.5 depicts the procedure *ProcessRetransmission*. The terminal senses the channel upon the arrival of a retransmitted packet. If *BusyTone* is ON, then an another retransmission must be sent after a random delay. The collection of retransmission and delay data is then performed. If the *BusyTone* is OFF, then exactly the same steps as in *ProcessArrival* (after *BusyTone*=ON? is F) have to be taken.

T BusyTone = ON ? F		
Create event Retransmission at	Terminal := TRANSMIT	
time = ($Clock$ + random delay)	Create event $BusyToneOn$ at time = ($Clock + d$)	
Collect retransmission data	Create event <i>Receive</i> at time = (<i>Clock</i> + packet duration + <i>d</i>)	
Collect delay data		

Fig. 4.5: PSD of procedure ProcessRetransmission

BusyToneOn and BusyToneOff - The events BusyToneOn and BusyToneOff take care of the inhibit mechanism of the ISMA protocol. The global variable BusyTone contains the right state of the BusyTone at any clock instant, indicating whether the channel is free or not. Fig. 4.6 and 4.7 depict the procedures ProcessBusyToneOn and ProcessBusyToneOff.

BusyTone := **O**N

Fig. 4.6: ProcessBusyToneOn

BusyTone := OFF

Fig. 4.7: ProcessBusyToneOff

Receive - The event *Receive* indicates the moment that the terminal knows whether the transmitted packet is received correctly or it has to be retransmitted. The procedure *ProcessReceive* is shown in Fig. 4.8. Only the first incoming packet at the base station will be serviced. If the terminal is not the first sending terminal then it has to retransmit his packet again after a random delay because its packet is ignored by the base station. This is accomplished by the event *Retransmission*. Then retransmission and delay data is collected.

Т	Terminal=first sending terminal? F		
For i:=1 To all transmitting terminals		Terminal:=BLOCKED	
Terminal := INTERFERE		Create event Retransmission	
Calculate Bit Error Rate		at time=(Clock +random delay)	
Calculate Packet Success Probability Pps			
Create event BusyToneOff at ($Clock_{lp}$ + packet duration + d)		Collect retransmission data	
T random <p<sub>ps? F</p<sub>			
Terminal:=IDLE	Terminal:=BLOCKED		
Collect success data	Create event <i>Retransmission</i> at time=(<i>Clock</i> +random delay)		
	Collect retransmission data		
Collect delay data			

Fig. 4.8: ProcessReceive

If the terminal is the first sending terminal then its data packet is serviced by the base station. This does not mean that this packet is automatically received correctly by the destination. Due to the other terminals who transmit during the inhibit delay fraction (d), data packets collide which causes interference. Because the inhibit delay fraction is only a small fraction of the packet duration and the channel is stationary (=time invariant) (see also equation 3.4), we assume that the bit error rate does not alter during the data packet. Notice that this is the worst case situation we are calculating (Fig. 4.9). If the bit error rate is known, then the packet success probability is not difficult to calculate. Fig. 4.9 also shows where the *BusyTone* has to be turned OFF. This time instant is denoted in Fig. 4.8 with

(*Clock*_{*lp*} + packet duration + *d*). *Clock*_{*lp*} is the time instant that the last packet is transmitted during the inhibit delay fraction. If the packet is received correctly, then the terminal state is set to IDLE to allow other newly arrived packets to get serviced. Then, the data is collected. On the other hand, if the packet is received erroneously, then the terminal state is set to BLOCKED again and a retransmission is generated. Data collection finally completes the discussion of the five event types and their processing procedures.

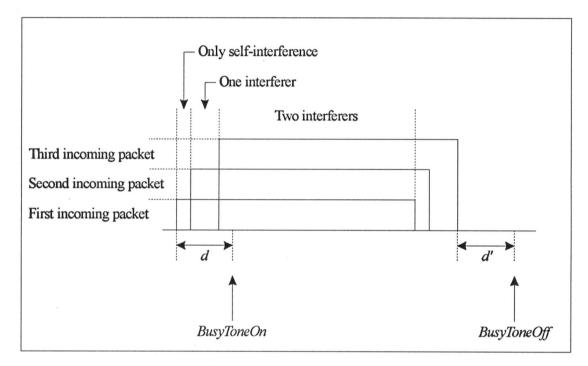


Fig. 4.9: Data packet collisions

4.3 The Negative Exponential Distribution

Arrivals of data packets at terminals are supposed to be generated by a Poisson process. The inter-arrival times of events of this Poisson process is negative exponentially distributed. The negative exponential distribution has a memoryless characteristic. The probability density function (PDF) and the cumulative distribution function (CDF) for the inter-arrival times are respectively given as:

$$f_T(t) = \begin{cases} \lambda \cdot e^{-\lambda t} & ; t \ge 0\\ 0 & ; t < 0 \end{cases}$$
(4.2)

$$F_T(t) = \begin{cases} 1 - e^{-\lambda t} & ; t \ge 0\\ 0 & ; t < 0 \end{cases}$$
(4.3)

In which t is the time it will passed till the next arrival of a packet at a terminal when on average there are λ arrivals per time unit [18]. The random variable T is then negative exponentially distributed with average and variance given as:

$$m_1 = \frac{1}{\lambda} \tag{4.4}$$

$$\sigma^2 = \frac{1}{\lambda^2} \tag{4.5}$$

Now we wish to generate a random variable T whose PDF and CDF are given in (4.2) and (4.3). Herefore, we make use of the so called 'Inverse Transform Method' [19]. This method functions as follows:

When the PDF $f_X(x)$ is given, we can calculate the CDF $F_X(x)$ by integration of $f_X(x)$. Otherwise, by differentiation of $F_X(x)$, we get $f_X(x)$. The value of $F_X(x)$ varies between zero and one. So, when we wish to generate a random variable whose PDF or CDF is given, we first have to generate a random number U between zero and one with a uniform distribution. This number U functions as a value for $F_X(x)$. We then have: $F_X(x) = U$. The random variable X we want to generate follows easily: $x = F_X^{-1}(U)$.

The above description is of course not a mathematical proof, but serves only as background for the inverse transform method. The mathematical proof follows now.

4.3.1 Proof of the Inverse Transform Method

Let X denote a random variable with CDF F_X and F_X^{-1} the inverse of F_X such that:

$$F_X^{-1}(y) = \inf \operatorname{imum}\{x \mid F_X(x) \ge y\}$$
 ; $0 < y < 1$ (4.6)

This means that $F_X^{-1}(y)$ is the *infimum* or smallest value of X for which $F_X(x) \ge y$ applies. First, we define the random variable Y as:

$$Y = F_X^{-1}(U)$$
 (4.7)

Then,

$$Pr(Y \le y) = Pr[F_X^{-1}(U) \le y]$$

= $Pr[U \le F_X(y)]$
= $F_X(y) = Pr(X \le y)$ (4.8)

So we can conclude that X and Y have the same distribution. Therefore, applying F_X^{-1} to an uniform distribution U produces an x from F_X . This result suggests the following computing algorithm:

Let U be uniformly distributed on (0,1). Then,

- 1. Generate U;
- 2. $x := F_X^{-1}(U);$
- 3. Deliver x

4.3.2 Generation of the Random Numbers for a Negative Exponential Distribution

Since the CDF of the negative exponential distribution exists explicitly as

$$F_T(t) = 1 - e^{-\lambda t} \tag{4.9}$$

and its inverse is easy to calculate

$$F_T^{-1}(t) = -\frac{1}{\lambda} \cdot \ln(1-t)$$
(4.10)

we can use the inverse transform method to generate the random numbers for the negative exponential distribution.

Algorithm for random numbers generation of $F_T(t)$:

Let U be uniformly distributed on (0,1), then

1. Generate U;

2.
$$t = F_T^{-1}(U) = -\frac{1}{\lambda} \cdot \ln(1-U)$$

3. Deliver t.

5. MATHEMATICAL ANALYSIS

The performance analysis of the hybrid unslotted non-persistent CDMA/ISMA will be explained from Fig. 5.1. The time axis in Fig. 5.1 is normalised to the packet duration. If a packet arrives and the terminal senses the channel free, then the packet is sent immediately. Suppose that this time instant is t. It takes d, the inhibit delay fraction, before the busy tone is heard by all terminals. This moment occurs at the same time at all terminals because of the symmetric network configuration. Any other packet arriving between t and t+d will sense the channel free (because the busy tone has not arrived yet) and will be transmitted resulting in a conflict. Because a CDMA scheme is also used, the conflict does not necessarily result in the loss of all packets. There is a probability that the first arriving packet can still be recovered successfully. If no other terminal transmits a packet during this period d, then no conflict occurs. We model the channel between the terminals and the corresponding receivers as a multipath Rayleigh-fading channel. So, when no conflicts occur, there is still a probability that the packet can not be recovered successfully.

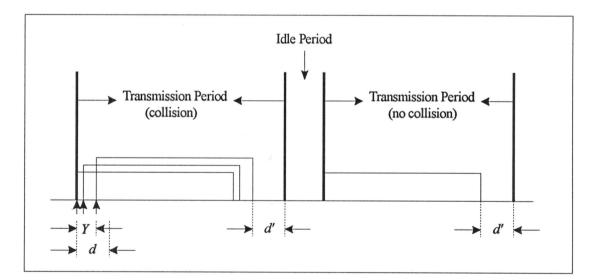


Fig. 5.1: Data cycle of the unslotted non-persistent ISMA protocol

5.1 Throughput

Let t+Y be the time of occurrence of the last packet arriving between t and t+d. This obviously means that Y must be between zero (the first transmitted packet is the only packet in the transmission period) and d (the end of the vulnerable period). The transmission of all packets arriving in (t, t+Y) will be completed at t+Y+1. As noted before, the channel is sensed unused only a period d later. So, any terminal becoming ready between t+d and t+Y+1+d will sense the channel busy and hence will reschedule its packet. The interval between t and t+Y+1+d is called a *transmission period* (*TP*). There will be *at most* one successful transmission during a *TP*. The *idle period* (*IP*) is defined as the period of time between two consecutive *TP*'s. A transmission period plus the following idle period constitute a *cycle*. Let \overline{TP} be the expected duration of the transmission period, \overline{I} the expected duration of the idle period, and the average cycle time can be written as [20]:

$$\overline{t_c} = \overline{TP} + \overline{I} \tag{5.1}$$

Let U denote the time during a cycle in which the inbound channel (towards base station) is used to carry a successful packet transmission and \overline{U} the corresponding average value, then we can write the throughput as:

$$S = \frac{\overline{U}}{\overline{t_c}}$$
(5.2)

The expected useful time \overline{U} can be computed as follows. When a packet is successful, the channel carries useful information for a duration of T_{pd} , the packet duration. In the unsuccessful case, no useful information is carried at all or, in other words:

$$U = \begin{cases} T_{pd} & ; Successful \ period \\ 0 & ; Unsuccessful \ period \end{cases}$$
(5.3)

If $P_{success}$ denotes the probability that a transmitted packet is successful then

$$\overline{U} = E[U] = T_{pd} \cdot P_{success} + 0 \cdot (1 - P_{success}) = T_{pd} \cdot P_{success}$$
(5.4)

As noted before, the time is normalised to T_{pd} (the packet duration), and therefore T_{pd} equals one. So, this gives us:

$$\overline{U} = P_{success} \tag{5.5}$$

To calculate the average idle period we make the assumption that the total rate at which users schedule new and retransmitted packets forms a Poisson process with parameter G. So, new and rescheduled packets arrive at a rate of G packets per unit time. Not all arrivals do result in a transmission. If a packet arrives and the terminal finds the channel in a busy state then the packet is rescheduled for transmission at a later time instant. In literature, G is also called the offered channel traffic.

If the packet arrival times are independent and exponentially distributed with a mean arrival rate of G packets per second, the probability of k arrivals in an interval of duration t is then a Poisson process given by $P_k(t)$ [18], where

$$P_k(t) = \frac{(Gt)^k \cdot e^{-Gt}}{k!}$$
(5.6)

For our work, the time is normalised to the packet duration (T_{pd}) . This means that G denotes the number of packet arrivals per *packet duration* and the time unit is also measured in *packet duration* instead of seconds. The probability of the idle time being greater than some value t is the probability that no packets are scheduled within a time interval of duration t and with the assumed Poisson packet scheduling process this probability becomes:

$$P(I \ge t) = P(no \ packet \ arrival \ in \ interval \ of \ duration \ t) = e^{-Gt}$$
(5.7)

Therefore the average value of I can be expressed as:

$$\overline{I} = \frac{1}{G} \tag{5.8}$$

5. MATHEMATICAL ANALYSIS

The average duration of a transmission period equals to:

$$\overline{TP} = 1 + \overline{Y} + d \tag{5.9}$$

where \overline{Y} is the expected value of Y. Since Y denotes the time at which the last interfering packet is scheduled, the probability of Y being smaller than some time y is the probability that no other packets (either new or retransmissions) are scheduled for transmission in an interval of duration d-y. With Poisson arrivals, the distribution for Y is:

$$F_{Y}(y) \underline{\Delta} Pr\{Y \le y\} = Pr\{no \text{ arrival occurs in an interval of length } d-y\}$$
$$= e^{-G(d-y)}$$
(5.10)

The average of Y is therefore given by:

$$\overline{Y} = d - \frac{1}{G} \left(1 - e^{-Gd} \right) \tag{5.11}$$

Combining the equations (5.1), (5.8), (5.9) and (5.11), we get:

$$\overline{t}_c = \overline{TP} + \overline{I} = 1 + \overline{Y} + d + \overline{I} = 1 + \left[d - \frac{1}{G}\left(1 - e^{-Gd}\right)\right] + d + \frac{1}{G}$$
(5.12)

$$S = \frac{\overline{U}}{\overline{t}_c} = \frac{P_{success}}{1 + \left[d - \frac{1}{G}\left(1 - e^{-Gd}\right)\right] + d + \frac{1}{G}} = \frac{G \cdot P_{success}}{G \cdot (1 + 2d) + e^{-Gd}}$$
(5.13)

The term $P_{success}$ in (5.7) is the only term that needs to be specified. Herefore we distinguish between four types of transmissions during a transmission period:

- 1. Successful transmission without conflict
- 2. Unsuccessful transmission without conflict
- 3. Successful transmission with conflict

4. Unsuccessful transmission with conflict

If the channel is used for the delivering of a successful packet during a TP then this is denoted by a successful transmission (situation 1 or 3). The difference between a transmission with or without conflict can be found in the number of packets transmissions during a TP. In case there is more than one packet transmission during a TP, we speak of a transmission with conflict. If there is only one packet transmission then the transmission is conflict-free. Keeping this in mind, we can divide $P_{success}$ up into two parts:

$$P_{success} = P_{success|noconflict} + P_{success|conflict}$$
(5.14)

In which $P_{success|noconflict}$ corresponds to situation 1 and $P_{success|conflict}$ to situation 3.

$$P_{success|noconflict} = e^{-Gd} P_{ps}$$
(5.15)

This is equal to the probability that no terminal transmits during the inhibit delay fraction d multiplied by the packet success probability P_{ps} . The multiplication with P_{ps} is necessary because of the assumption of the Rayleigh fading channel.

$$P_{ps} = \left(1 - P_{be|x=0}\right)^{L_p}$$
(5.16)

 $P_{be|x=0}$ is the bit error probability in Rayleigh fading channel using the CDMA scheme and is caused by self interference due to multipath. L_p denotes the packet length; in our model the packet is 64 bits long. The calculation of $P_{success|noconflict}$ is not very complicated. However, this is not the case for $P_{success|conflict}$ which is the probability that the first packet has arrived successfully given a collision (between two or more packets).

$$P_{success|conflict} = \sum_{x=1}^{K-1} P_x(d) \cdot P\{first \quad packet \quad OK|x\}$$
(5.17)

in which:

K is the number of active users in the system

 $P_x(d)$ is the probability of x arrivals during d for a Poisson distribution

x denotes the number of arrivals during d

Terminals within one group are supposed to be identical in every aspect. They have the same arrival rate, the same code and the same distance to their own transceiver. Notice that the distance between a terminal and the base station does not play a role. This can be explained as follows. If a terminal is allowed to transmit its data packet then it will do so using radio communication to deliver its packet to its transceiver. The transceiver on its turn forwards the received packet to the base station by wire. The base station only process the first incoming packet arriving by wire (see also Fig. 5.2). All other packets will be ignored by the base station.

Suppose that the network configuration in Fig. 5.2 is deployed. T_i denotes terminal number i and TR_j the transceiver number j. The data packet transmitted by T_i is denoted by p_i . For example, we adopt the following scenario. T_1 starts sending its data packet p_1 at time instant t and T_7 a fraction later, at $(t + \Delta t)$ in which $\Delta t \le d$. The data packet p_1 arrives Δt earlier at its transceiver TR_1 than its competitor p_7 at TR_2 . Consequently, TR_1 forwards its received packet Δt earlier to the base station than TR_2 . The base station notices that TR_1 is the first active receiver and processes this packet. TR_2 , TR_3 and TR_4 are all ignored. Even though the data packet p_7 is ignored by the base station, this packet causes interference for p_1 . The radio transmitted packet p_7 not only arrives at TR_2 (its own receiver), but also at other receivers $(TR_1, TR_3 \text{ and } TR_4)$. Consequently TR_1 not only receives p_1 , but also the interfering packet p_7 . This interference is less severe than from the interference caused by T_2 , T_3 or T_4 . The explanation herefore is twofold. First, T_2 , T_3 and T_4 all have the same code as T_1 and second, the received interference power (at TR_1) of T_2 , T_3 or T_4 are higher than T_7 because of the shorter distance to TR_1 .

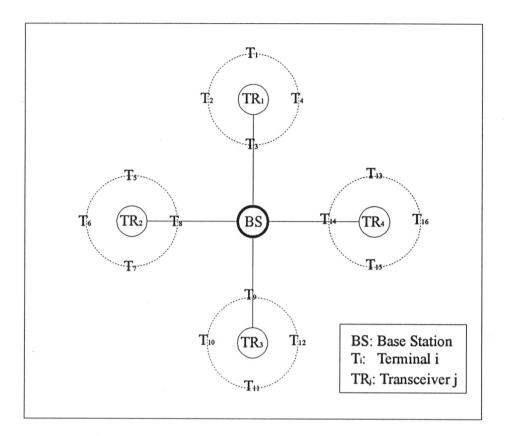


Fig. 5.2: Example of a 16-terminal network configuration

Without loss of generality, suppose that T_1 is the reference terminal. This terminal sends its data packet at an idle period first. Other terminals sending within the inhibit delay fraction interfere with the reception of the data packet p_1 . The bit error rate depends on the configuration of transmitting terminals. $P\{first \ packet \ OK|x\}$ can then be evaluated as:

$$P\{first \ packet \ OK | x\} = E \left[1 - P_{be|\vec{x}}\right]^{L_p}$$
(5.18)

E[.] denotes the expectation value, and \vec{x} a certain configuration of *x*. The total number of combinations for \vec{x} equals $\binom{K-1}{x}$, in which $\binom{K-1}{x} = \frac{(K-1)!}{x!(K-1-x)!}$.

Because we assume an uniform distribution for all possible configurations we can write (K is the total number of active users in the system):

$$E\left[1 - P_{be|\vec{x}}\right]^{L_p} = \frac{1}{\binom{K-1}{x}} \cdot \sum_{\vec{x}} \left(1 - P_{be|\vec{x}}\right)^{L_p}$$
(5.19)

$$P_{success|conflict} = \sum_{x=1}^{K-1} P_x(d) \cdot \frac{1}{\binom{K-1}{x}} \cdot \sum_{\vec{x}} \left(1 - P_{be|\vec{x}}\right)^{L_p}$$
(5.20)

$$P_{success} = e^{-Gd} \left(1 - P_{be|x=0} \right)^{L_p} + \sum_{x=1}^{K-1} P_x(d) \cdot \frac{1}{\binom{K-1}{x}} \cdot \sum_{\vec{x}} \left(1 - P_{be|\vec{x}} \right)^{L_p}$$
(5.21)

Combining the equations (5.13) and (5.21) finally gives:

$$S = \frac{G\left(e^{-Gd}\left(1 - P_{be|x=0}\right)^{L_{p}} + \sum_{x=1}^{K-1} P_{x}(d) \cdot \frac{1}{\binom{K-1}{x}} \cdot \sum_{\vec{x}} \left(1 - P_{be|\vec{x}}\right)^{L_{p}}\right)}{G(1+2d) + e^{-Gd}}$$
(5.22)

For large networks ($K \ge 32$), the calculation of the second term in the numerator can take long. If we take a closer look at this term we can distinguish two summations. For example, we take a 32 terminal network, and this simplified term looks like:

$$\sum_{x=1}^{31} \sum_{x} \dots$$
(5.23)

We see that the value of x determines the duration of the calculations. Obviously, it is not possible to derive equation (5.23) for x = 1 up to 31. The total number of combinations will be over 4 billions! Fortunately, this is also not necessary. We can limit x to a certain extent. To explain this we have to return to equation (5.17). From (5.17) we see that $P_{success/conflict}$ depends on the multiplication of $P_x(d)$ with $P\{first packet OK | x\}$. Fig. 5.3 and 5.4 depict

 $P_x(d)$ versus G with x and d as parameter respectively. $P_x(d)$ is derived from formula (5.6). For high values of x, $P_x(d)$ drops quickly. Because high values of x is identical to a high number of interferes, $P\{first \ packet \ OK|x\}$ also drops quickly for increasing values of x. Therefore, the contribution in formula (5.17) for high values of x is negligible. Consequently, we can limit x to a certain value. There is a trade off between the calculation time and the accuracy of the calculation.

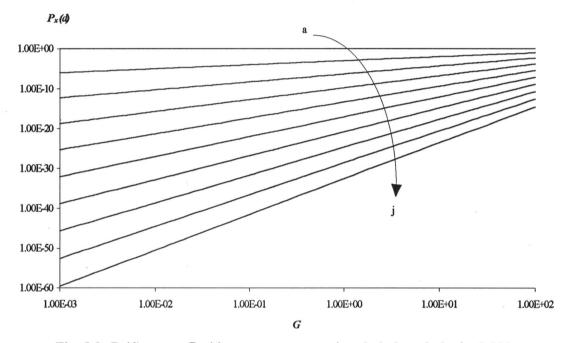


Fig. 5.3: $P_x(d)$ versus G with x as parameter; a...j: x=0, 1, 2, ..., 8, 9; d = 0.001

We adopt the following strategy for the derivation of the throughput for the hybrid protocol. The throughput will be derived by formula (5.22) for increasing limit value of x. After a certain value, the throughput with x do not differ much from the throughput with (x-1) and the calculation time will last too long. Depending on the achieved results we can choose a reasonable and considered limit value of x. Section 5.3 deals with this aspect in more detail.

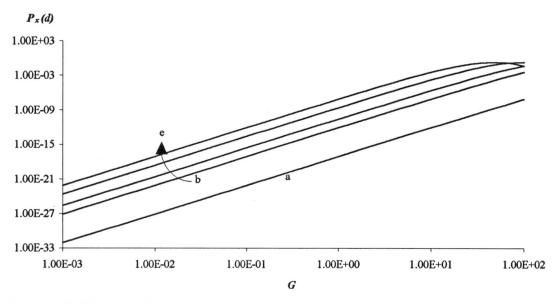


Fig. 5.4: $P_x(d)$ versus G with d as parameter; a..e: d=0.001, 0.01, 0.02, 0.05 and 0.1; x = 5

5.2 Delay

The average packet delay is defined as the duration between the transmission of the first bit till the correct reception of the last bit. For the calculation of the packet delay of the unslotted hybrid CDMA/ISMA protocol, we use the block diagram depicted in Fig. 5.5.

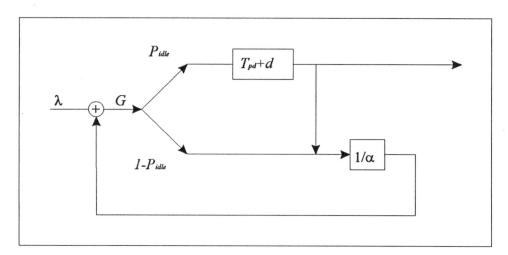


Fig. 5.5: Delay in the unslotted non-persistent hybrid CDMA/ISMA protocol

If a new packet arrives, the terminal immediately senses the channel to decide if it can transmit the packet. In case the channel is sensed busy (probability $1-P_{idle}$), the packet is rescheduled for a later time instant. This random delay process is assumed to be negative exponentially distributed with parameter α . The average random delay time is then $1/\alpha$. The average number of times the packet has to suffer this delay is $G(1-P_{idle})/S$ (see Fig. 5.5). S is the throughput as given by (5.22).

On the other hand, if the channel is sensed idle (probability P_{idle}), the packet will be transmitted immediately. This transmission will take a packet duration T_{pd} plus the inhibit delay fraction d before the knowledge about the outcome of the transmission is available (see also Fig. 5.5). So, the total delay until the terminal knows whether the transmission was successful or not is $(T_{pd} + d)$. The transmission of a data packet does not necessarily result in a successful transmission; there are two possibilities: a successful or a failed transmission. If the transmission was successful, the packet leaves the system. The delay this packet suffers is $(T_{pd} + d)$. If the transmission was a failure, the packet again has to wait a random

delay. In this case, the average delay is $\left(T_{pd} + d + \frac{1}{\alpha}\right)$ and the average number of schedulings is $\left(\frac{GP_{idle}}{S} - 1\right)$.

Combining the above results, the average delay is finally given by equation (5.24).

$$D = \left(\frac{GP_{idle}}{S} - 1\right) \left[T_{pd} + d + \frac{1}{\alpha}\right] + \frac{G(1 - P_{idle})}{S} \cdot \frac{1}{\alpha} + \left(T_{pd} + d\right)$$
(5.24)

Again, because T_{pd} equals to 1, (5.24) can be rewritten as:

$$D = \left(\frac{GP_{idle}}{S} - 1\right) \left[1 + d + \frac{1}{\alpha}\right] + \frac{G(1 - P_{idle})}{S} \cdot \frac{1}{\alpha} + (1 + d)$$
(5.25)

S can be found in formula (5.22) and P_{idle} still has to be calculated. The average probability of sensing the channel idle is the average probability that the channel is idle and this probability equals to the average idle time (\bar{I}) divided by the average cycle length (\bar{t}_c) .

$$P_{idle} = \frac{\overline{I}}{\overline{t}_c} = \frac{\frac{1}{G}}{\frac{1}{G} + \left\{1 + d - \frac{1}{G}\left(1 - e^{-Gd}\right) + d\right\}} = \frac{1}{G(1 + 2d) + e^{-Gd}}$$
(5.26)

5.3 Practical Implementation Aspect

Formula (5.22) can not be calculated exactly if we do not possess strong computing power. However, formula (5.22) allows us to approach the exact value very accurately in case we only have limited computing power.

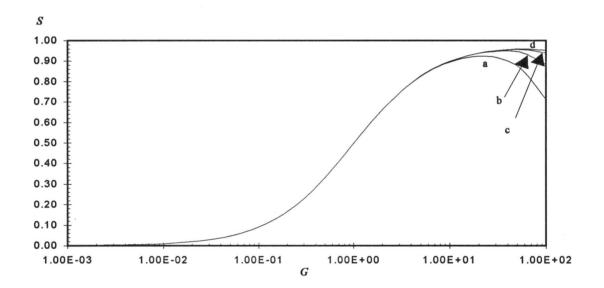


Fig. 5.6: Throughput versus offered traffic; a.d: x=1..4; $\alpha=0.1$; d=0.01; code length=31; SNR=20 dB; number of codes=8; number of terminals=32

The influence of x in formula (5.22) on the results is depicted in Fig. 5.6 and 5.7. As explained from the previous section, x denotes the number of arrivals during the inhibit delay

fraction d and determines for a great deal the calculation duration. From Fig. 5.6 and 5.7 in which the throughput and delay are depicted respectively, we can see that the protocol performance converges to a certain final level. (The delay time is normalised to the packet duration T_{pd}). The difference between x=3 and x=4 is hardly noticeable. We adopt x=4 as a reasonable value for our further analysis. However, for higher values of the offered traffic G, this difference increases. This can be explained by the fact that if the offered traffic increases, more packets arrive during the inhibit delay fraction.

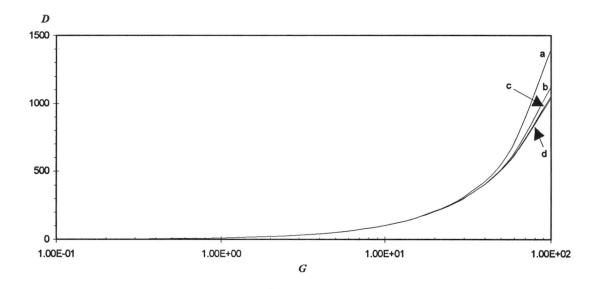


Fig. 5.7: Delay versus offered traffic; a..d: x=1..4; $\alpha=0.1$; d=0.01; code length=31; SNR=20dB; number of codes=8; number of terminals=32

The mathematical analysis in this chapter is nice in the sense that it allows us to make a trade off between accuracy and computing time. If we do not have strong computing power or computing time, we are also able to obtain the performance result of the hybrid protocol. However, we have to live with a less accurate result. On the other hand, if accuracy if necessary, then we have to pay for strong computing power.

6. RESULTS

This chapter presents the results obtained from simulation and mathematical analysis as explained in the previous chapters. The protocol performance has been measured in terms of throughput S and delay D in relationship with the offered traffic G. The throughput is a measure of the efficiency of the protocol and is defined as the fraction of time in which correct data packets are received. The delay is the time between the arrival of the first bit of a data packet at a terminal and its arrival of the last bit at the destination. The delay is measured in terms of the packet duration.

In section 1, we first compare the performance between the slotted and unslotted protocol using simulation. Then, in section 2, we discuss the comparison between the simulation and mathematical model for the unslotted protocol. The conditions under which the results are valid can be found in section 2.6. If a parameter is changed, then it will be stated explicitly.

6.1 Comparison between Slotted and Unslotted Protocol

The performance shown in this section is generated by simulation. Fig. 6.1a and b compare the performance of the slotted and unslotted hybrid protocol using computer simulation. In these figures, we also show the influence of P_{tr} on the performance of the slotted protocol. P_{tr} is one of the most important parameters for the slotted protocol and is described in paragraph 2.4.1 as the probability that the packet will actually be transmitted in case the channel is sensed *free*. As we can observe, for increasing value of P_{tr} , the throughput improves. The declaration for this observation can be found by the use of direct sequence CDMA. If only pure ISMA is used, than it is to be expected that for high traffic, the increase in P_{tr} leads to a degradation of the protocol's performance. High values of P_{tr} means a high number of simultaneouly transmitted data packets. Collision between those packets results in corruption of all data packets. However, because CDMA is also used, there is still a probability that the first incoming packet can be received correctly.

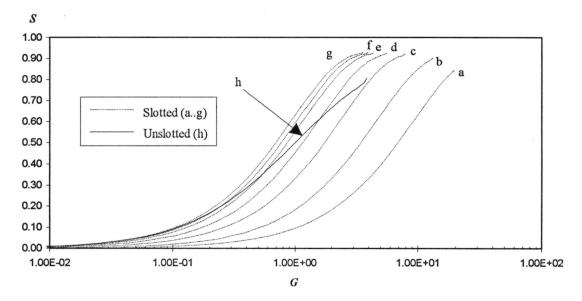


Fig. 6.1a: Throughput versus offered traffic using simulation of slotted (a..g: $P_{tr} = 0.1, 0.2, 0.4, 0.6, 0.8, 0.9$ and 1) and unslotted (h: d=0.01) hybrid protocol; 8 codes

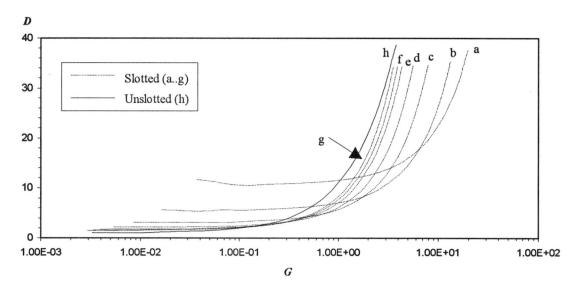


Fig. 6.1b: Delay versus offered traffic using simulation of slotted (a..g: $P_{tr} = 0.1, 0.2, 0.4, 0.6, 0.8, 0.9$ and 1) and unslotted (h: d=0.01) hybrid protocol; 8 codes

For the delay, there is one interesting observation we can make. If we look at low offered traffic, then the delay for P_{tr} is high. For example, the average delay for $P_{tr}=0.1$ is about 10 packet durations. This is not surprising because the probability of actually transmitting a

packet if the channel is sensed idle equals to 0.1. So, on average, a packet has to wait 10 packet durations before it can be transmitted. On the other hand, if we look at high offered traffic, the delay is lower for low values of P_{tr} . For example, take a look at the delay curves of $P_{tr}=0.1$ and 0.2 (curves a and b of Fig. 6.1b). For an offered traffic below 8 packets/(packet duration), the delay for $P_{tr}=0.1$ is higher than for $P_{tr}=0.2$. For an offered traffic beyond 8 packets/(packet duration) it is the other way round. So, if we have to choose a slotted scheme, then the region of the offered traffic in which we are, decides of value for the parameter P_{tr} . If the offered traffic is low, than a slotted ISMA protocol with a high value of P_{tr} is better.

The throughput of the unslotted protocol is better than the throughput of the slotted protocol for low values of the offered traffic. For increasing values of the offered traffic, the throughput of the unslotted protocol starts to drop below the throughput of the slotted protocol. This is because for high offered traffic, the slotted protocol organises its data stream better than the unslotted protocol. Packets are only allowed to be transmitted at the start of a time slot. Furthermore, the slotted protocol with low values of P_{tr} handles high offered traffic quite well, because it limits of the contention in the channel more profoundly. For low traffic, the delay of the unslotted protocol is, as expected, better than the slotted protocol. For high traffic, the situation is the other way round.

Fig. 6.2 shows the influence of code sharing on the performance of the slotted and unslotted protocol. P_{tr} plays an important role in the slotted scheme. This is the same for *d* with respect to the unslotted scheme. The slotted protocol is simulated for $P_{tr}=0.9$. As already noted in section 2.6, the inhibit delay fraction *d* equals to 0.01 for the unslotted protocol.

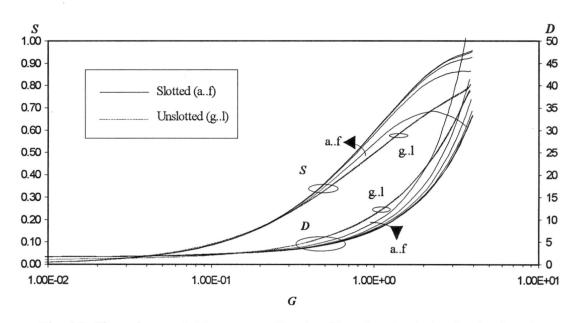


Fig. 6.2: Throughput and delay versus offered traffic using simulation for the slotted ($P_{tr}=0.9$) and unslotted (d=0.01) protocol, varying the number of codes a..f: g..l: number of codes: 1, 2, 4, 8, 16 and 32.

6.2 Comparison between Simulation and Mathematical Analysis

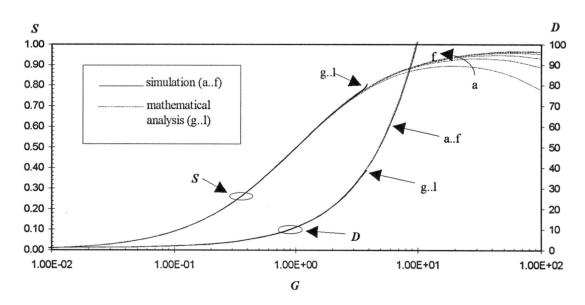


Fig. 6.3: Comparison between simulation and mathematical analysis for the unslotted protocol. a. f, g. l: 1, 2, 4, 8, 16, 32 codes

The comparison between the simulation and mathematical model for the unslotted protocol is given in Fig. 6.3. The performance of simulation does not exceed for $G \approx 4$ packets/(packet duration). This is because in case of simulation, the arrival rate at terminals is the input parameter and G is one of the output parameters. The results due to simulation and mathematical model are in good agreement. This graph also shows the effect of code sharing. For a 32-terminal network, it is sufficient to use only four codes. If we take a closer look at the simulation results, it seems like code sharing has no influence at all on the performance. Fig. 6.4a and b show the influence of the number of codes used on the performance of the unslotted protocol for poor parameters. In contrast to Fig. 6.3 where good parameters are chosen, Fig. 6.4a and b show that the number of codes used do influence protocol performance. The more codes, the better the protocol performs. However, code sharing is a very good option. The differences between 8 and 32 codes is much lower than the difference between 1 and 32 codes.

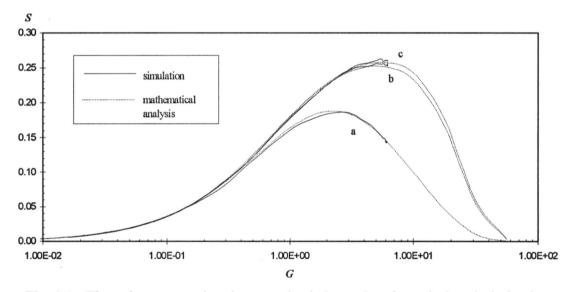


Fig. 6.4a: Throughput comparison between simulation and mathematical analysis for the unslotted protocol and varying the number of codes used; a..c: number of codes=1, 8 and 32; d=0.2; $\alpha=0.2$; SNR=6 dB; code length = 31

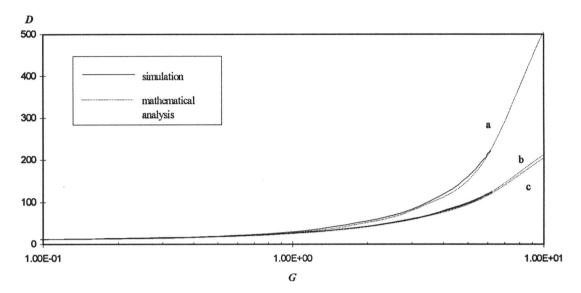


Fig. 6.4b: Delay comparison between simulation and mathematical analysis for the unslotted protocol and varying the number of codes used; a..e: number of codes=1, 8 and 32; d=0.2; $\alpha=0.2$; SNR=6 dB; code length = 31

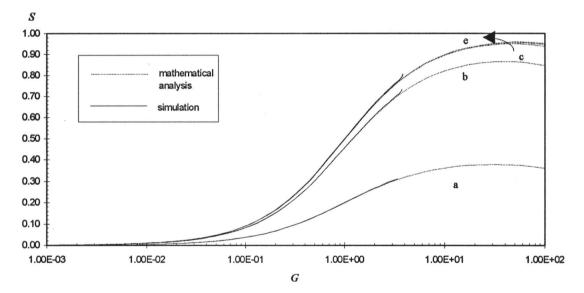


Fig. 6.5a: Throughput comparison between simulation and mathematical analysis for the unslotted protocol and varying SNR; a..e: SNR=6, 8, 10, 12 and 20 dB; number of codes=8

Fig. 6.5a and b show the effect of the signal to noise ratio (SNR) on the performance of the unslotted protocol. Of course, the throughput and delay improve for increasing SNR.

However, Fig. 6.5a and b also show that a SNR of 10 dB suffice. For values of SNR higher than 10 dB, the protocol performance also improves, but only for high values of the offered traffic and the improvement is only additional. Again, simulation and mathematical analysis agree very well with each other.

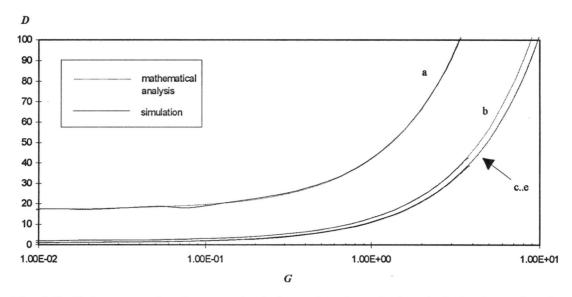


Fig. 6.5b: Delay comparison between simulation and mathematical analysis for the unslotted protocol and varying SNR; a..e: SNR=6, 8, 10, 12 and 20 dB; number of codes=8;

Fig. 6.6a and b show the effect of the inhibit delay fraction d on the performance of the unslotted protocol. For high values of d, the performance degrades tremendously. This is due to the increase of the number of data packets colliding during this period. This is why the ISMA protocol is not useful for environment in which the propagation delay is high.

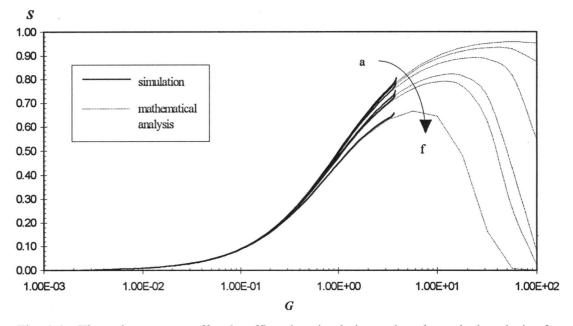


Fig. 6.6a: Throughput versus offered traffic using simulation and mathematical analysis of unslotted protocol, varying the inhibit delay fraction d. a..f: d=0.01, 0.02, 0.04, 0.08, 0.1 and 0.2; 8 codes

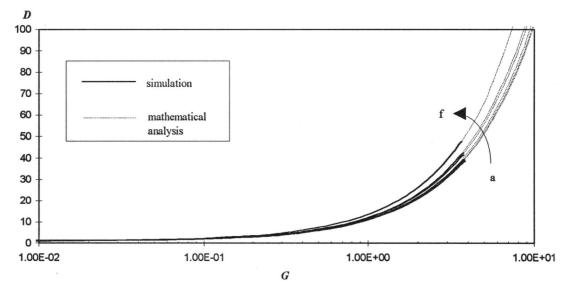


Fig. 6.6b: Delay versus offered traffic using simulation and mathematical analysis of unslotted protocol, varying the inhibit delay fraction d; a..f: d=0.01, 0.02, 0.04, 0.08, 0.1 and 0.2; 8 codes

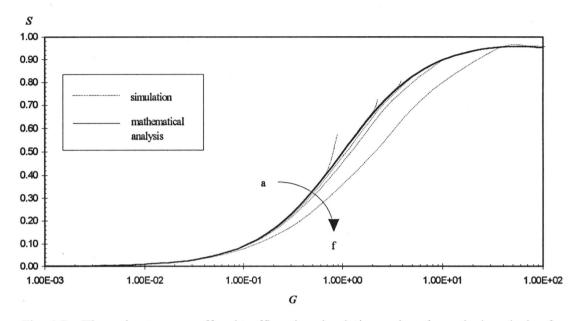


Fig. 6.7a: Throughput versus offered traffic using simulation and mathematical analysis of unslotted protocol, varying α , the random delay parameter; a..f: α =0.01, 0.05, 0.1, 0.5, 1 and 5; 8 codes

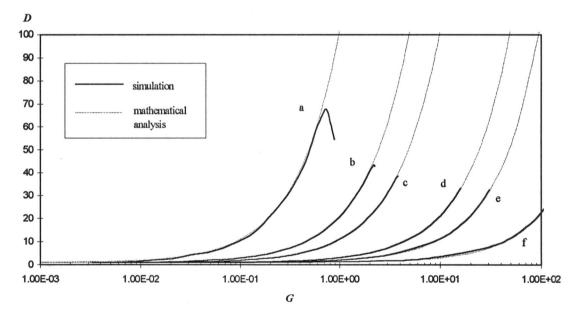


Fig. 6.7b: Delay versus offered traffic using simulation and mathematical analysis of unslotted protocol, varying α , the random delay parameter; a..f: α =0.01, 0.05, 0.1, 0.5, 1 and 5; 8 codes

Fig. 6.7a and b show the effect of α , the parameter for the random delay mechanism, on the performance of the unslotted protocol. This parameter is explained in paragraph 4.2.3. If a packet has to be retransmitted, than on average it has to wait $1/\alpha$ before it can try again. So, for low values of α , the delay increases tremendously. This can be observed in Fig. 6.7b. In case of the mathematical model, the throughput in Fig. 6.7a does not show much variation if the parameter α is changed. Fig. 6.7a and b also show strange bendings at the end of simulations. We try to depict this phenemenon more clearly by showing the delay versus throughput in Fig. 6.8.

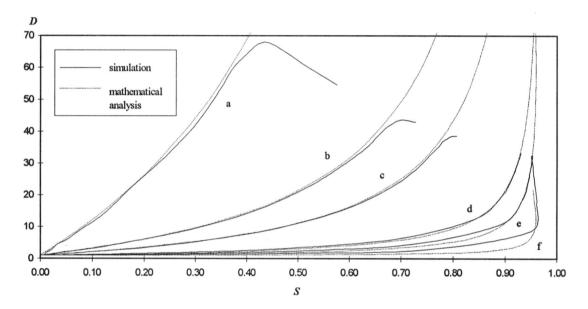


Fig. 6.8: Delay versus throughput using simulation and mathematical analysis of unslotted protocol, varying α , the random delay parameter; a..f: α =0.01, 0.05, 0.1, 0.5, 1 and 5; 8 codes

Fig. 6.8 depicts the delay-throughput characteristics of the unslotted protocol for varying values of α . If the throughput increases, then the delay also increases. For low values of α and especially at the end of the curves, the computer simulation does not agree with the mathematical analysis. Especially for curve a, where the delay even decreases tremendously at the end. It is very likely that this is caused by the abrupt program termination and the low value of α . A low value of α means that the random delay is large. This means that the collection of statistics has been done while most of the terminals still have to wait a random delay.

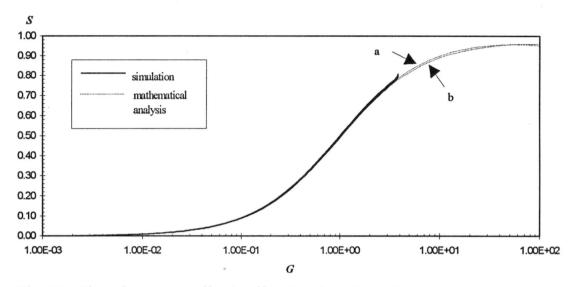


Fig. 6.9a: Throughput versus offered traffic using simulation and mathematical analysis of unslotted protocol, varying the code length; a..b: code length = 31 and 127; 8 codes

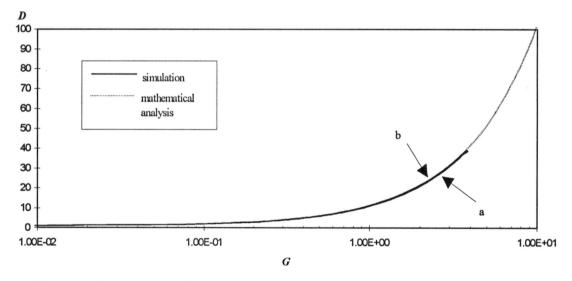


Fig. 6.9b: Delay versus offered traffic using simulation and mathematical analysis of unslotted protocol, varying the code length; a..b: code length = 31 and 127; 8 codes

Fig. 6.9a and b show the effect of code length on the performance of the unslotted protocol. At first sight, it is surprisingly to notice that there is hardly a difference between the performance if the code length is altered. We would expect that the performance of a longer code is better than a shorter one. This is the case for good parameters.

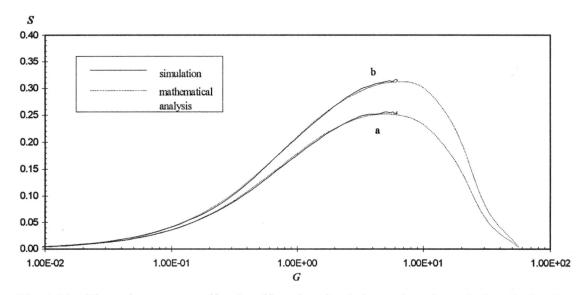


Fig. 6.10a: Throughput versus offered traffic using simulation and mathematical analysis of unslotted protocol, varying the code length; a..b: code length = 31 and 127; 8 codes; d=0.2;

α=0.2; SNR=6 dB

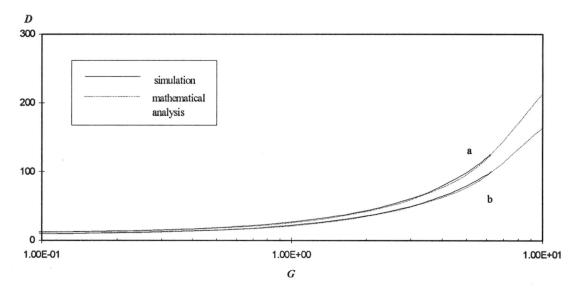


Fig. 6.10b: Delay versus offered traffic using simulation and mathematical analysis of unslotted protocol, varying the code length; a..b: code length = 31 and 127; 8 codes; d=0.2; $\alpha=0.2$; SNR=6 dB

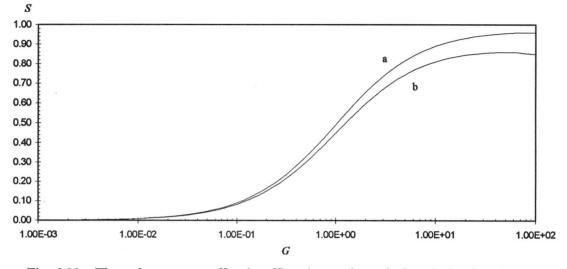


Fig. 6.11a: Throughput versus offered traffic using mathematical analysis of unslotted protocol, varying the code length; a: code length =127 and bit rate = 256*1024; b: code length = 31 and bit rate = 256*1024*4; 8 codes

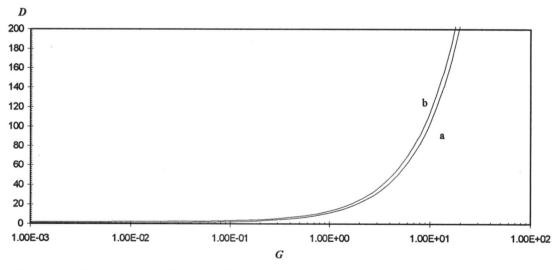


Fig. 6.11b: Delay versus offered traffic using mathematical analysis of unslotted protocol, varying the code length; a: code length =127 and bit rate = 256*1024; b: code length = 31 and bit rate = 256*1024*4; 8 codes

In case the parameters are poor, we do see differences in the performance if the code length is altered (Fig. 6.10a and b). Longer codes perform better than shorter codes because they

have better cross correlations characteristics. However, longer codes need a larger bandwidth. If we keep the bandwidth fixed, and we increase the code length, than we have to decrease the bit rate. This is illustrated in Fig. 6.11a and b. Here, we see that the performance is better in case the code length is longer.

7. CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

The unslotted hybrid CDMA/ISMA protocol has been investigated. The protocol performance is measured in terms of throughput S and delay D in relationship with the offered traffic G. Herefore, we have been able to derive a close form formula for the throughput and delay. Also a simulation program has been developed to compare the results. The following conclusions can be drawn from the previous chapters:

- First of all it can be concluded that the results from simulation and mathematical model agree very well with each other. The mathematical model is very suitable for analysing the hybrid protocol even when strong computing power is not available. This mathematical model allows a trade off between accuracy and computing power or computing time.
- Comparison between the slotted and unslotted protocol shows that for low offered traffic the delay for the unslotted protocol is better than the slotted protocol. For high traffic it is the other way round.
- The most important parameter for the slotted p-persistent scheme, P_{tr} , has a great influence on the protocol's performance. The choice of this parameter depends on the offered traffic the system generates.
- The hybrid protocol supports code sharing extremely well. For a 32 terminal network, only four codes suffice.
- Longer codes do not perform better than shorter codes if the system is under good conditions (the parameters are good). On the other hand, if the parameters are poor, then longer codes can indeed be used to improve the protocol's performance.

- The retransmission parameter α influences the average packet delay for a great deal.
- If the SNR increases, the performance also improves. However, beyond 10 dB, the performance hardly improves anymore.
- The inhibit delay fraction *d* plays an important role in the unslotted non-persistent ISMA scheme. It is concluded that the proposed unslotted hybrid protocol performs very well in case the propagation delay is not a concern (*d* is small). If the propagation delay is too high (*d* is large) the performance drops quickly.

7.2 Recommendations

- The system model should be extended with buffers. How does this feature influence the protocol's performance?
- Suppose the line of sight path is present. With other words, if the channel is modeled as a Rician fading channel, how does the protocol then perform? And does the Rayleigh fading channel describes our indoor channel sufficiently?
- The offered traffic is assumed to be a Poisson process. This is only valid if the number of terminals is large. The investigation of our system is only restricted to 32 terminals because we lack computer power and simulation time. Future research should pay attention to this matter.
- Error corrections can be included to improve the protocol's performance.
- The hybrid protocol composes of CDMA and ISMA. What about other combinations? How about CDMA and TDMA? We also can think of a protocol in which for low offered traffic, the protocol behaves like a random access protocol and for high offered traffic the protocol has a contentionless characteristic.

• Enhance the analysis of the protocol with the feedback channel and the acknowledgements to see how these aspects influence the protocol performance.

7. CONCLUSIONS AND RECOMMENDATIONS

Unslotted Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications

REFERENCES

[1] R. Prasad,

"Performance Analysis of Mobile Packet Radio Networks in Real Channels with Inhibit-Sense Multiple Access",

IEE Proc. I, Vol. 138, pp. 458-464, October 1991.

- I. Widipangestu, A.J. 't Jong and R. Prasad,
 "Capture Probability and Throughput Analysis of Slotted ALOHA and Unslotted np-ISMA in a Rician/Rayleigh Environment",
 IEEE Transactions on Vehicular Technology, Vol. 43, No. 3, pp. 457-465, August 1994.
- K.J. Zdunek, D.R. Ucci and J.L. Locicero,
 "Throughput of Nonpersistent Inhibit Sense Multiple Access with Capture", Electronics Letters, Vol. 25, No. 1, pp. 30-31, 5th January 1989.
- [4] R. Prasad, and C.Y. Liu,
 "Throughput Analysis of Some Mobile Packet Radio Protocols in Rician Fading Channels",
 IEE Proceedings-I, Vol. 139, No. 3, pp. 297-302, June 1992.
- [5] M.G. Jansen, and R. Prasad,
 "Capacity, Throughput, and Delay Analysis of a Cellular DS CDMA System With Imperfect Power Control and Imperfect Sectorization",
 IEEE Transactions on Vehicular Technology, Vol. 44, No. 1, pp. 67-75, February 1995.
- [6] R.A. Scholtz,"The Origins of Spread Spectrum Communications",

REFERENCES

IEEE Transactions on Communications, Vol. COM-30, No. 5, pp. 822-854, May 1982.

- [7] W.C.Y. Lee,
 "Overview of Cellular CDMA",
 IEEE Transactions on Vehicular Technology, Vol. 40, No. 2, pp. 291-302, May 1991.
- [8] M.B. Pursley,
 "Performance Evaluation of Phase Coded Spread Spectrum Multiple Access Communication - Part I: System Analysis",
 IEEE Transactions on Communications, Vol. COM-25, No. 8, pp. 795-799, August 1977.
- [9] R.L. Pickholtz, D.L. Schilling and L.B. Milstein "Theory of Spread Spectrum Communications - A Tutorial", IEEE Transactions on Communications, Vol. COM-30, No. 5, pp. 855-884, May 1982.
- [10] K.S. Gilhousen, I.M. Jacobs, R. Padovani, A.J. Viterbi, L.A. Weaver, Jr. and C.E.Wheatley III,
 "On the Capacity of a Cellular CDMA System",
 IEEE Transactions on Vehicular Technology, Vol. 40, No. 2, pp. 303-312, May 1991.
- [11] H. van Roosmalen, J. Nijhof and R. Prasad,
 "Performance Analysis of a Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications",
 IEEE Journal on Selected Areas in Communications, Vol. 12, No. 5, pp. 909-916, June 1994.

[12] H. van Roosmalen,

"Performance Analysis of a Hybrid ISMA/CDMA Protocol for Indoor Wireless Communication",

Thesis report, Delft University of Technology, the Netherlands, August 1992

- [13] C. van den Broek,
 "Multiple Access Techniques with a special interest in Code Division Multiple Access",
 AIO report Delft University of Technology, the Netherlands, January 1992
- [14] R. C. Dixon
 <u>Spread Spectrum Systems</u>
 John Wiley & Sons, 2nd edition, 1984
- [15] A. B. Carlson, <u>Communication systems</u>, Mc. Graw Hill Book Company, 3rd edition, 1988, pp. 92.
- [16] M. Kavehrad and B. Ramamurthi,
 "Direct -Sequence Spread Spectrum with DPSK Modulation and Diversity for Indoor Wireless Communications",
 IEEE Transactions on Communications, Vol. COM-35, No. 2, pp. 224-236, February 1987.
- [17] H. Hashemi,
 "The Indoor Radio Propagation Channel"
 Proceedings of the IEEE, VOL. 81, No. 7, pp. 943-966, Juli 1993

F. Schoute, <u>Prestatie-Analyse van Telecommunicatiesystemen</u>, Kluwer Technische Boeken B.V., 1989

[19] I. Mitrani,

Simulation Techniques for Discrete Event Systems, Cambridge: Cambridge University Press, 1982

[20] L. Kleinrock and F.A. Tobagi,

"Packet Switching in Radio Channels: Part I- Carrier Sense Multiple Access and their Throughput-Delay Characteristics",

IEEE Trans. on Commun., Vol. COM-23, No. 12, pp. 1400-1416, December 1975.

APPENDIX A:

"Unslotted Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications"

Proceedings IEEE Third Symposium on Communications and Vehicular Technology in the Benelux, October 25-26 1995, Eindhoven, The Netherlands, pp. 68-75

Unslotted Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications

Huy Linh Anh Le, Student Member, IEEE* Huub van Roosmalen** Jos Nijhof, Member, IEEE* Ramjee Prasad, Senior Member, IEEE*

*Telecommunications and Traffic Control Systems Group Delft University of Technology P.O. Box 5031, 2600 GA Delft The Netherlands <u>Tel.</u>: +31 15-78.23.86 (before 10-10-'95) +31 15-278.24.17 (after 10-10-'95) <u>Fax</u>: +31 15-78.17.74 E-Mail: Huy@Octopus.et.tudelft.nl

**Holland Institute of Traffic Technology B.V. P.O. Box 245 7300 AE Apeldoorn The Netherlands Fax: +31 55-432553

ABSTRACT - A hybrid Code Division Multiple Access / Inhibit Sense Multiple Access (CDMA/ISMA) protocol has been proposed as an effective multiple access scheme for Indoor Wireless Computer Communications. This new protocol combines the advantages of both CDMA and ISMA into one protocol. On the one hand the ISMA protocol introduces a limitation to the number of simultaneous accesses to the transmission channel. On the other hand the CDMA protocol introduces an improvement to the packet survival chance. Slotted hybrid CDMA/ISMA protocol has been reported in [1]. It is shown that the performance of the hybrid protocol is indeed better than CDMA only. In addition, code sharing can be applied to reduce hardware cost. This paper presents the performance analysis of the unslotted CDMA/ISMA protocol in order to take more advantage of the strength of the hybrid protocol. The performance comparison between the slotted and unslotted hybrid CDMA/ISMA protocol is evaluated in terms of throughput and delay using computer simulation and mathematical analysis.

I. INTRODUCTION

Indoor wireless office communication is our main research field. The office system we focus on consists of a building in which users work together in groups. The participants generate terminal traffic. Terminals communicate with each other by radio transmission using a random access protocol. Terminals might not detect each other's transmission in radio communications. It can easily happen that two users are hidden from each other by some obstacle, in which case severe performance degradation results. This is called the hidden terminal problem. The introduction of a central base station can alleviate this problem by instructing it to send a busy tone to all participating terminals to forbid new transmissions when a transmission is going on. Still, a situation can occur in which two or more terminals simultaneously start their transmission, resulting in a collision. However, a great reduction in the number of simultaneous transmissions is achieved by the introduction of a central base station. This concept is called ISMA [1]-[4].

If we could somehow increase the survival chances of colliding packets, we could improve protocol performance. The near-far effect is one way to achieve this. A packet may 'capture' the receiver if it is much stronger than its competitors. This can happen when terminals use the same transmission power, but at different distances from the receiver. Because the 'near terminals' have better performance compared to the 'far terminals' (due to the better survival chance of the packets), the near-far effect introduces an unfair element among the terminals. Perfect power control, in which the transmitted power of the terminals are adjusted such that their received powers are all equal, can eliminate the near-far effect described above. The performance of CDMA has been reported in a number of publications, e.g. [5]-[10].

The hybrid CDMA/ISMA protocol combines the advantages of both CDMA and ISMA into one protocol. The advantages of CDMA and ISMA are the improvement of the survival chance of packets and the limitation of contention in the channel. Code sharing can also be applied. This is an important aspect because the number of distinct useful transmission codes is limited, especially for short code length.

In [11] the hybrid protocol combines Direct Sequence CDMA with *slotted* p-persistent ISMA. In this paper, we have investigated the performance of the hybrid CDMA/ISMA protocol using the *unslotted* nonpersistent ISMA scheme. It is expected that for low traffic, the delay of the unslotted protocol will improve because when a terminal has a data packet to send, it does not have to wait until the start of the next time slot. In addition, even when data packets collide, there is a probability that the data is received correctly due to the use of CDMA. This is the strength of the hybrid CDMA/ISMA protocol. By using the *unslotted nonpersistent* ISMA scheme, it is expected that we take more advantage of this strength.

This paper is organized as follows. Section II gives the protocol description of the unslotted hybrid CDMA/ISMA protocol. The performance analysis is then given in Section III. Section IV shows the assumptions used. Results are discussed in Section V. Finally, conclusions can be found in Section VI.

II. PROTOCOL DESCRIPTION

Fig. 1 depicts an example of the 16-terminal network configuration of the protocol. There is one central base station that is connected to several transceivers by wire. The base station controls the traffic flow with a busy tone that can be detected by all participating terminals. Because all terminals can detect the busy tone, the hidden terminal problem is solved. Several terminals together with one transceiver form a group. Wireless communication is considered to take place between the transceiver and the terminals. The terminals around each transceiver share the same code. In this way the number of codes can be reduced. When a data packet is ready for transmission, a terminal transmits the packet to its transceiver according to the chosen ISMA scheme.

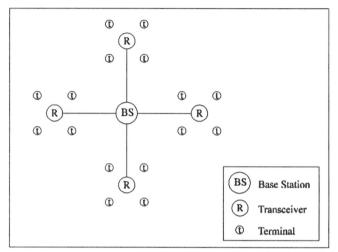


Fig. 1: 16-terminal network configuration

After the arrival of the data packet, the transceiver then simply forwards this packet to the base station which forwards it to all terminals via the transceivers. The destination then receives the packet and decides whether the packet is errorless or not. In case the packet was erroneous, a retransmission must take place. The return channel is not included in our analysis because only the base station makes use of this return channel and therefore contention is not a concern. The state of the terminals can either be free or blocked. At the beginning, the terminals are in the free state. If a packet arrives at a free terminal, the terminal jumps into the blocked state. In the blocked state, the terminal takes care that the arriving data packet is serviced successfully. In the mean time the blocked terminal ignores all incoming packets. This is a consequence of the assumption that the terminals do not have buffers for the incoming data packets.

The unslotted non-persistent ISMA protocol

Users behind terminals generate data packets. The data packets arrive at terminals, which will take care of the correct delivery of the packets. We assume that the arrivals of these packets are generated by a Poisson process. The SDL diagram of the unslotted nonpersistent ISMA protocol is shown in Fig. 2. Before transmitting the data packet to the receiver, the terminal first listens to the channel to detect whether there is a busy tone going on. The base station broadcasts a busy tone to signal that the channel is busy because there is a transmission in progress. If the channel is busy, the terminal waits a random delay before it can try again. Otherwise, the packet will be transmitted immediately. If a collision occurs from the transmission, the collided packets have to wait a random delay before they are allowed to try again (Fig. 2).

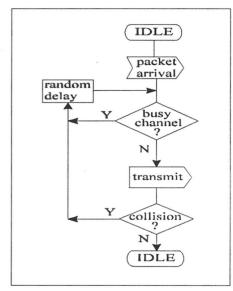


Fig. 2: Unslotted non-persistent ISMA

Although the number of collisions is greatly reduced with ISMA, collisions still can occur. In this unslotted ISMA scheme, collisions may occur due to multiple terminals transmitting packets during an interval called the *inhibit delay fraction d*. This interval *d* is necessary to switch from 'idle' to 'busy'. The reverse interval from 'busy' to 'idle' is denoted by *d'*. The inhibit delay fraction is normalized to the packet length, resulting in $0 \le d < 1$ (Fig. 3). In this paper we assume d = d'.

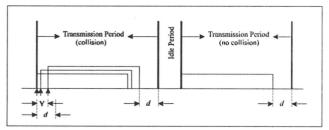


Fig. 3: Data Cycle

III. PERFORMANCE ANALYSIS

The protocol performance has been measured in terms of throughput S and delay D in relationship with the offered traffic G. The throughput measures the efficiency of the protocol and is defined as the fraction of time in which correct data packets are received. The delay is the time

between the arrival of the first bit of a data packet at a terminal and its arrival of the last bit at the destination.

A. Throughput

The performance analysis of the hybrid unslotted nonpersistent CDMA/ISMA will be explained from Fig. 3. The time axis in Fig. 3 is normalized to the packet duration. If a packet arrives and the terminal senses the channel free then the packet is sent immediately. Suppose that this time instant is t. It takes d, the inhibit delay fraction, before a busy tone is heard by all terminals. This moment occurs at the same time at all terminals because of the symmetric network configuration. Any other packet arriving between t and t+d will sense the channel free (because the busy tone has not arrived yet) and will be transmitted resulting in a conflict. Because a CDMA scheme is also used, the conflict does not necessarily result in the loss of all packets. There is a probability that the first arriving packet can still be recovered successfully. If no other terminal transmits a packet during this period d, then no conflict occurs. We model the channel between the terminals and the corresponding transceivers as a multipath Rayleigh-fading channel. So, when no conflicts occur, there is still a probability that the packet *cannot* be recovered successfully.

Let t+Y be the time of occurrence of the last packet arriving between t and t+d. This obviously means that Y must be between zero (the first transmitted packet is the only packet in the transmission period) and d (the end of the vulnerable period). The transmission of all packets arriving in (t, t+Y) will be completed at t+Y+1. As noted before, the channel is sensed unused only a period d later. So, any terminal becoming ready between t+dand t+Y+I+d will sense the channel busy and hence will reschedule its packet. The interval between t and t+Y+l+d is called a transmission period (TP). There will be at most one successful transmission during a TP. The *idle period* (IP) is defined as the period of time between two consecutive TP's. A transmission period plus the following idle period constitute a cycle. Let TP be the expected duration of the transmission period, \overline{I} the expected duration of the idle period, and the average cycle time can be written as:

$$\overline{t_c} = \overline{TP} + \overline{I} \tag{1}$$

Let U denote the time during a cycle in which the inbound channel (towards base station) is used to carry a successful packet transmission and \overline{U} the corresponding average value, then we can write the throughput as [12]:

$$S = \frac{\overline{U}}{\overline{t_c}} \tag{2}$$

The expected useful time \overline{U} can easily be computed. When a packet is successful, the channel carries useful information for a duration of T_{pd} , the packet duration. In the unsuccessful case no useful information is carried at all or, in other words:

$$U = \begin{cases} T_{pd} & Successful \ period \\ 0 & Unsuccessfull \ period \end{cases}$$
(3)

If $P_{success}$ denotes the probability that a transmitted packet is successful then

$$\overline{U} = T_{pd} \cdot P_{success} \tag{4}$$

As noted before, the time is normalized to T_{pd} (the packet duration), and therefore T_{pd} equals to one. So, this gives us:

$$\overline{U} = P_{success} \tag{5}$$

To calculate the average idle period we make the assumption that the total rate at which users schedule new and retransmitted packets forms a Poisson process with parameter G. So, new and rescheduled packets arrive at a rate of G packets per unit time. In literature, G is also called the offered channel traffic.

The probability of the idle time being greater than some value t is the probability that no packets are scheduled within a time interval of duration t and with the assumed Poisson packet scheduling process this probability becomes e^{-Gt} . Therefore the average value of I can be expressed as:

$$\overline{I} = \frac{1}{G} \tag{6}$$

The average duration of a transmission period equals to:

$$\overline{TP} = 1 + \overline{Y} + d \tag{7}$$

where \overline{Y} is the expected value of Y. Since Y denotes the time at which the last interfering packet is scheduled, the probability of Y being smaller than some time y is the probability that no other packets (either new or retransmissions) are scheduled for transmission in an interval of duration d-y. With Poisson arrivals, the distribution for Y is $e^{-G(d-y)}$. The average of Y is therefore given by:

$$\overline{Y} = d - \frac{1}{G} \left(1 - e^{-Gd} \right) \tag{8}$$

Applying the formulas obtained above we get:

$$\overline{t}_{c} = \overline{TP} + \overline{I} = 1 + \overline{Y} + d + \overline{I}$$
$$= 1 + \left[d - \frac{1}{G} \left(1 - e^{-Gd} \right) \right] + d + \frac{1}{G}$$
(9)

$$S = \frac{\overline{U}}{\overline{t}_c} = \frac{P_{success}}{1 + \left[d - \frac{1}{G}\left(1 - e^{-Gd}\right)\right] + d + \frac{1}{G}}$$
$$= \frac{G \cdot P_{success}}{G(1 + 2d) + e^{-Gd}}$$
(10)

The term $P_{success}$ in (10) is the only term that needs to be specified. Herefore, we distinguish between four types of transmissions during a transmission period:

- 1. Successful transmission without conflict
- 2. Unsuccessful transmission without conflict
- 3. Successful transmission with conflict
- 4. Unsuccessful transmission with conflict

If the channel is used for the delivering of a successful packet during a TP then this is denoted by a successful transmission (situation 1 or 3). The difference between a transmission with or without conflict can be found in the number of packet transmissions during a TP. In case there is more than one packet transmission during a TP, we speak of a transmission with conflict. If there is only one packet transmission then the transmission is conflict-free. Keeping this in mind, we can divide $P_{success}$ into two parts:

$$P_{success} = P_{success|noconflict} + P_{success|conflict}$$
(11)

In which $P_{success|noconflict}$ corresponds to situation 1 and $P_{success|conflict}$ to situation 3.

$$P_{success|noconflict} = e^{-Gd} P_{ps}$$
(12)

This is equal to the probability that no terminal transmits during the inhibit delay fraction d multiplied by the packet success probability P_{ps} . The multiplication with P_{ps} is necessary because of the assumption of the Rayleigh fading channel.

$$P_{ps} = \left(1 - P_{be|x=0}\right)^{L_p} \tag{13}$$

 $P_{be|x=0}$ is the bit error probability in Rayleigh fading channel using the CDMA scheme and is caused by self interference due to multipath. L_p denotes the packet length. We assume that during the packet duration, the bit error rate remains constant. The calculation of $P_{success|noconflict}$ is not very complicated. However, this is not the case for $P_{success|conflict}$ which is the probability that the first packet has arrived successfully given a collision (between two or more packets).

$$P_{success|conflict} = \sum_{x=1}^{K-1} P_x(d) \cdot P\{first_packet_OK|x\}$$
(14)

in which:

K is the number of active users in the system;

 $P_x(d)$ is the probability of x arrivals during d for a Poisson distribution;

x denotes the number of arrivals during d.

Consider a reference terminal T_R . This terminal sends its data packet at an idle period first. Other terminals sending within the inhibit delay fraction interfere with the reception at this reference terminal T_R . The bit error rate depends on the configuration. So, if we try to take the different configurations into account, we have to calculate the bit error probability for all possible configurations and then average.

$$P\{first_packet_OK|x\} = E\left[1 - P_{be|\vec{x}}\right]^{L_p}$$
(15)

E[.] denotes the expectation value, and \vec{x} a certain configuration of *x*. The total number of combinations for \vec{x} equals $\binom{K-1}{x}$. Because we assume an uniform

distribution for all possible configurations we can write (K is the total number of active users in the system):

$$E\left[1-P_{be|\vec{x}}\right]^{L_p} = \frac{1}{\binom{K-1}{x}} \cdot \sum_{\vec{x}} \left(1-P_{be|\vec{x}}\right)^{L_p}$$
(16)
$$P_{success|conflict} = \sum_{x=1}^{K-1} P_x(d) \cdot \frac{1}{\binom{K-1}{x}} \cdot \sum_{\vec{x}} \left(1-P_{be|\vec{x}}\right)^{L_p}$$
(17)

The formula for the bit error probability with DPSK modulation in Rayleigh fading channel is given by Kavehrad and Ramamurthi (formula 13a in [13]) in

which the signal to noise ratio (SNR) is taken to be 20 dB. Combining the above results, the throughput can easily be calculated.

B. Delay

The average packet delay is defined as the duration between the transmission of the first bit till the correct reception of the last bit. For the calculation of the packet delay of the unslotted hybrid CDMA/ISMA protocol, we use the block diagram depicted in Fig. 4.

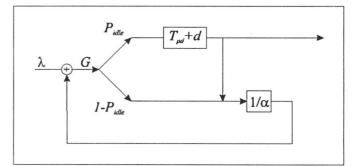


Fig. 4: Delay in the unslotted non-persistent hybrid CDMA/ISMA protocol

If a new packet arrives, the terminal immediately senses the channel to decide if it can transmit the packet. In case the channel is sensed busy (probability $1-P_{idle}$), the packet is rescheduled for a later time instant. This random delay process is assumed to be negative exponentially distributed with parameter α . The average random delay time is then $1/\alpha$. The average number of times the packet has to suffer this delay is $G(1-P_{idle})/S$. In which G, S and P_{idle} are defined as before.

On the other hand, if the channel is sensed idle (probability P_{idle}), the packet will be transmitted immediately. This transmission will take a packet duration T_{pd} plus the inhibit delay fraction d before the knowledge about the outcome of the transmission is available (see also Fig. 3). So, the total delay until the terminal knows if the transmission was successful or not is $(T_{pd} + d)$. The transmission of a data packet does not necessarily result in a successful transmission; there are two possibilities: a successful or a failed transmission. If the transmission was successful, the packet leaves the system. The delay this packet suffers is $(T_{pd} + d)$. If the transmission was a failure, the packet again has to wait a random delay. In this case, the average delay is $\left(T_{pd} + d + \frac{1}{\alpha}\right)$ and the average number of schedulings is $\left(\frac{GP_{idle}}{S}-1\right)$. Combining the above results, the average delay is finally given by formula (18).

$$D = \left(\frac{GP_{idle}}{S} - 1\right) \left[T_{pd} + d + \frac{1}{\alpha}\right] + \frac{G(1 - P_{idle})}{S} \cdot \frac{1}{\alpha} + (T_{pd} + d)$$
(18)

IV. ASSUMPTIONS

After DPSK modulation the signal is spread with a Gold code. The length of these codes are chosen to be 31. The signal is then transmitted, modulated on a 1.7 GHz carrier. The data rate is arbitrarily chosen to be 256.1024 b/s (0.26 Mb/s). The packet length is 64 bits. The delay spread is supposed to be 100 ns. Perfect power control is assumed to assure that signals from terminals within the same group arrive at the transceiver with the same power.

The far field model [14] is chosen to describe the signal attenuation. Given the transmitted power P_T , the received power P_R can be expressed by the following equation:

$$P_R = \frac{g_T g_R}{\left(\frac{4\pi f l}{c}\right)^{\alpha}} P_T \tag{19}$$

where g_T and g_R are respectively the transmitter and the receiver gains, f the signal frequency, l the distance between the transmitter and the receiver, c the speed of light and α is the attenuation parameter. This attenuation parameter is chosen equal to 2, corresponding to free space propagation.

Fading is the result of the propagation of the transmitted signal through several paths. The channel is modeled as a Rayleigh fading channel. The Rayleigh fading channel model is valid when each path contributes the same amount of energy to the composite received signal. A Line of Sight (LOS) path is therefore assumed to be absent.

An example of a 16-terminal network configuration that we adopted in this paper is shown in Fig. 2. The terminals are clustered around distributed transceivers at a fixed distance (5 m). The distributed transceivers are clustered around the base station at a fixed distance (30 m). Those distances are kept fixed for all simulations. The number of terminals within a transceiver group (also called group size) is chosen as a power of two. For a certain fixed number of participated terminals, we have to halve the group size if we want to double the number of codes (the number of codes is also the number of transceivers). The comparison is fair in this manner, because when we want to investigate the effect of the number of codes on the performance, we

V. RESULTS

Fig. 5 a and b depict the simulation comparison between the slotted and unslotted hybrid protocol.

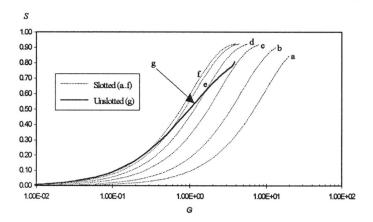


Fig. 5a: Throughput versus offered traffic using simulation of slotted (a..f: $P_{tr} = 0.1$, 0.2, 0.4, 0.6, 0.8, 0.9) and unslotted (g: d=0.01) hybrid protocol; 8 codes; $\alpha=0.1$

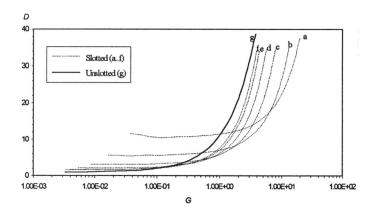


Fig. 5b: Delay versus offered traffic using simulation of slotted (a..f: $P_{tr} = 0.1, 0.2, 0.4, 0.6, 0.8, 0.9$) and unslotted (g: d=0.01) hybrid protocol; 8 codes; $\alpha=0.1$

In [11] P_{tr} is defined for the slotted protocol as the probability that the packet will actually be transmitted in case the channel is sensed *free*. For low traffic, the delay of the unslotted protocol is, as expected, better than the slotted protocol. For high traffic the situation is the other way round. The slotted protocol does not differ much from the unslotted protocol for high values of P_{tr} .

Fig. 6 represents the effect of d on the performance of the unslotted protocol. For high values of d, the performance degrades tremendously. This is due to the increase of the number of data packets colliding during

this period. This is why the ISMA protocol is not useful for environment in which the propagation delay is high.

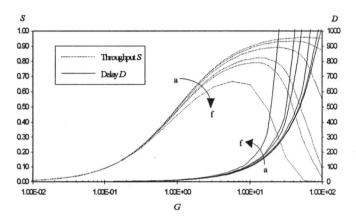


Fig. 6: Throughput and delay versus offered traffic using mathematical analysis of unslotted protocol, varying the inhibit delay fraction d. a..f: d=0.01, 0.02, 0.04, 0.08, 0.1 and 0.2; 8 codes; $\alpha=0.1$

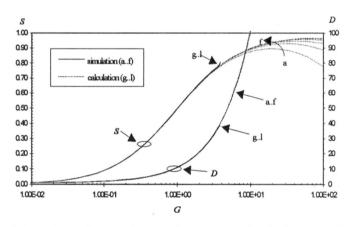


Fig. 7: Comparison between simulation and mathematical analysis for the unslotted protocol. a..f, g..l: 1, 2, 4, 8, 16, 32 codes; d=0.01; $\alpha=0.1$

The comparison between the simulation and mathematical model for the unslotted protocol is given in Fig. 7. In case of simulation, the arrival rate at terminals is the input parameter and G is one of the output parameters. In addition, due to the assumption that terminals does not have buffers the performance of simulation does not exceed $G \approx 4$ packets/(packet duration). The simulation and mathematical model does quite agree with each other. This graph also shows the effect of code sharing. For a 32-terminal network, it is sufficient to use only four codes.

VI. CONCLUSIONS

The unslotted hybrid CDMA/ISMA protocol has been investigated. The protocol performance is measured in terms of throughput S and delay D in relationship with

the offered traffic G. Herefore, we have been able to derive a close form formula for the throughput and delay. Also a simulation program has been developed to compare the results. The results from simulation and mathematical model does quite agree with each other.

Comparison between the slotted and unslotted protocol shows that for low offered traffic the delay for the unslotted protocol is better than the slotted protocol. For high traffic it is the other way round.

Furthermore it is concluded that the proposed hybrid protocol performs very well in case the propagation delay is not a concern. If the propagation delay is too high the performance drops quickly.

REFERENCES

- R. Prasad,
 "Performance Analysis of Mobile Packet Radio Networks in Real Channels with Inhibit-Sense Multiple Access",
 IEE Proc. I, Vol. 138, pp. 458-464, October 1991.
- I. Widipangestu, A.J. 't Jong and R. Prasad, "Capture Probability and Throughput Analysis of Slotted ALOHA and Unslotted np-ISMA in a Rician/Rayleigh Environment", IEEE Transactions on Vehicular Technology, Vol. 43, No. 3, pp. 457-465, August 1994.
- K.J. Zdunek, D.R. Ucci and J.L. Locicero,
 "Throughput of Nonpersistent Inhibit Sense Multiple Access with Capture",
 Electronics Letters, Vol. 25, No. 1, pp. 30-31, 5th January 1989.
- [4] R. Prasad, and C.Y. Liu,
 "Throughput Analysis of Some Mobile Packet Radio Protocols in Rician Fading Channels", IEE Proceedings-I, Vol. 139, No. 3, pp. 297-302, June 1992.
- [5] M.G. Jansen, and R. Prasad,
 "Capacity, Throughput, and Delay Analysis of a Cellular DS CDMA System With Imperfect Power Control and Imperfect Sectorization", IEEE Transactions on Vehicular Technology, Vol. 44, No. 1, pp. 67-75, February 1995.
- [6] R.A. Scholtz,
 "The Origins of Spread Spectrum Communications",
 IEEE Transactions on Communications, Vol. COM-30, No. 5, pp. 822-854, May 1982.

- [7] W.C.Y. Lee,
 "Overview of Cellular CDMA",
 IEEE Transactions on Vehicular Technology,
 Vol. 40, No. 2, pp. 291-302, May 1991.
- [8] M.B. Pursley,
 "Performance Evaluation of Phase Coded Spread Spectrum Multiple Access Communication - Part I: System Analysis", IEEE Transactions on Communications, Vol. COM-25, No. 8, pp. 795-799, August 1977.
- [9] R.L. Pickholtz, D.L. Schilling and L.B. Milstein "Theory of Spread Spectrum Communications -A Tutorial", IEEE Transactions on Communications, Vol. COM-30, No. 5, pp. 855-884, May 1982.
- K.S. Gilhousen, I.M. Jacobs, R. Padovani, A.J. Viterbi, L.A. Weaver, Jr. and C.E.Wheatley III, "On the Capacity of a Cellular CDMA System", IEEE Transactions on Vehicular Technology, Vol. 40, No. 2, pp. 303-312, May 1991.

- [11] H. van Roosmalen, J. Nijhof and R. Prasad, "Performance Analysis of а Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications", IEEE Journal on Selected Areas in Communications, Vol. 12, No. 5, pp. 909-916, June 1994.
- [12] L. Kleinrock and F.A. Tobagi,
 "Packet Switching in Radio Channels: Part I-Carrier Sense Multiple Access and their Throughput-Delay Characteristics",
 IEEE Trans. on Commun., Vol. COM-23, No. 12, pp. 1400-1416, December 1975.
- [13] M. Kavehrad and B. Ramamurthi,
 "Direct -Sequence Spread Spectrum with DPSK Modulation and Diversity for Indoor Wireless Communications",
 IEEE Transactions on Communications, Vol. COM-35, No. 2, pp. 224-236, February 1987.
- [14] A. B. Carlson, *Communication systems*, Mc. Graw Hill Book Company, 3rd edition, 1988, pp. 92.

APPENDIX B:

"Indoor Wireless Computer Communications using Unslotted Hybrid CDMA/ISMA Protocol"

Submitted for participation in the International Conference on Communications (ICC '96).

Indoor Wireless Computer Communications using Unslotted Hybrid CDMA/ISMA Protocol

Ramjee Prasad, Senior Member, IEEE Huy Linh Anh Le, Student Member, IEEE Huub van Roosmalen Jos Nijhof, Member, IEEE

Presenter

Should the paper be accepted, then Prof. Dr. R. Prasad will attend the conference and also present the paper.

Technical Subject Areas:

- Computer Communications
- Data Communications
- Broadband Access and Delivery Systems

Abstract

A hybrid Code Division Multiple Access / Inhibit Sense Multiple Access (CDMA/ISMA) protocol has been proposed as an effective multiple access scheme for Indoor Wireless Computer Communications. This new protocol combines the advantages of both CDMA and ISMA into one protocol. On the one hand the ISMA protocol introduces a limitation to the number of simultaneous accesses to the transmission channel. On the other hand the CDMA protocol introduces an improvement to the packet survival chance. It is shown that the performance of the hybrid protocol is superior to the performance of CDMA only. In addition, code sharing can be applied to reduce hardware cost. To take more advantage of the strength of the hybrid protocol, we have investigated the *unslotted non-persistent* ISMA scheme instead of the *slotted p-persistent* ISMA scheme. The analysis is done for a star-connected multiple access radio network. The performance comparison between the slotted and unslotted hybrid CDMA/ISMA protocol is evaluated in terms of throughput and delay using computer simulation and mathematical analysis.

I. INTRODUCTION

Indoor wireless communication has proved to be a very attractive means of computer/data communication in an office environment. The office system we focus on consists of a building in which users work together in groups. The participants generate terminal traffic. Terminals communicate with each other by radio transmission using a random access protocol. Terminals might not detect each other's transmission in radio communications. It can easily happen that two users are hidden from each other by some obstacle, in which case severe performance degradation results. This is called the *hidden terminal problem*. The introduction of a central base station can alleviate this problem by instructing it to send a busy tone to all participating terminals to forbid new transmissions when a transmission is going on. Still, a situation can occur in which two or more terminals simultaneously start their transmission, resulting in a collision. However, a great reduction in the number of simultaneous transmissions is achieved by the introduction of a central base station. This concept is called ISMA [1]-

If we could somehow increase the survival chances of colliding packets, we could improve protocol performance. The near-far effect is one way to achieve this. A packet may 'capture' the receiver if it is much stronger than its competitors. This can happen when terminals use the same transmission power, but at different distances from the receiver. Because the 'near terminals' have better performance compared to the 'far terminals' (due to the better survival chance of the packets), the near-far effect introduces an unfair element among the terminals. Perfect power control, in which the transmitted power of the terminals are adjusted such that their received powers are all equal, can eliminate the near-far effect described above. The performance of CDMA has been reported in a number of publications, e.g. [5]-[10].

The hybrid CDMA/ISMA protocol combines the advantages of both CDMA and ISMA into one protocol. The advantages of CDMA and ISMA are the improvement of the survival chance of packets and the limitation of contention in the channel. Code sharing can also be applied. This is an important aspect because the number of distinct useful transmission codes is limited, especially for short code length.

In [11] the hybrid protocol combines Direct Sequence CDMA with *slotted* p-persistent ISMA. In this paper, we have investigated the performance of the hybrid CDMA/ISMA protocol using the *unslotted* non-persistent ISMA scheme. It is expected that for low traffic, the delay of the unslotted protocol will improve because when a terminal has a data packet to send, it does not have to wait until the start of the next time slot. In addition, even when data packets collide, there is a probability that the data is received correctly due to the use of CDMA. This is the strength of the hybrid CDMA/ISMA protocol. By using the *unslotted non-persistent* ISMA scheme, it is expected that we take more advantage of this strength.

This paper is organized as follows. Section II gives the protocol description of the unslotted hybrid CDMA/ISMA protocol. The performance analysis is then given in Section III. Section IV shows the assumptions used. Results are discussed in Section V. Finally, conclusions are given in Section VI.

II. PROTOCOL DESCRIPTION

Fig. 1 depicts an example of the 16-terminal network configuration of the protocol. There is one central base station that is connected to several transceivers by wire. The base station controls the traffic flow with a busy tone that can be detected by all participating terminals. Because all terminals can detect the busy tone, the hidden terminal problem is solved. Several terminals together with one transceiver form a group. Wireless communication is considered to take place between the transceiver and the terminals. The terminals around each transceiver share the same code. In this way the number of codes can be

reduced. When a data packet is ready for transmission, a terminal transmits the packet to its transceiver according to the chosen ISMA scheme.

After the arrival of the data packet, the transceiver then simply forwards this packet to the base station which forwards it to all terminals via the transceivers. The destination then receives the packet and decides whether the packet is errorless or not. In case the packet was erroneous, a retransmission must take place. The return channel is not included in our analysis because only the base station makes use of this return channel and therefore contention is not a concern. The state of the terminals can either be free or blocked. At the beginning, the terminals are in the free state. If a packet arrives at a free terminal, the terminal jumps into the blocked state. In the blocked state, the terminal takes care that the arriving data packet is serviced successfully. In the mean time the blocked terminal ignores all incoming packets. This is a consequence of the assumption that the terminals do not have buffers for the incoming data packets.

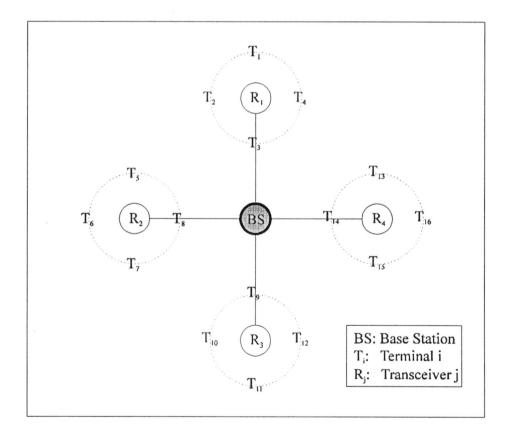


Fig. 1: 16-terminal network configuration

The unslotted non-persistent ISMA protocol

Users behind terminals generate data packets. The data packets arrive at terminals, which will take care of the correct delivery of the packets. We assume that the arrivals of these packets are generated by a Poisson process. The Specification and Description Language (SDL) diagram of the unslotted non-persistent ISMA protocol is shown in Fig. 2. Before transmitting the data packet to the receiver, the terminal first listens to the channel to detect whether there is a busy tone going on. The base station broadcasts a busy tone to signal that the channel is busy because there is a transmission in progress. If the channel is busy, the terminal waits a random delay before it can try again. Otherwise, the packet will be transmitted immediately. If a collision occurs from the transmission, the collided packets have to wait a random delay before they are allowed to try again (Fig. 2).

Although the number of collisions is greatly reduced with ISMA, collisions still can occur. In this unslotted ISMA scheme, collisions may occur due to multiple terminals transmitting packets during an interval called the *inhibit delay fraction* d. This interval d is necessary to switch from 'idle' to 'busy'. The reverse interval from 'busy' to 'idle' is denoted by d'. The inhibit delay fraction is normalized to the packet length, resulting in $0 \le d < 1$ (Fig. 3). In this paper we assume d' = d.

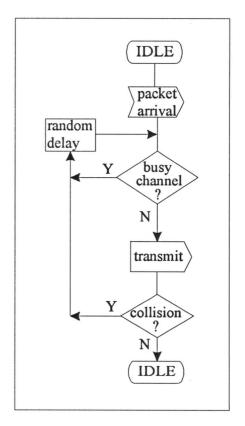


Fig. 2: Unslotted non-persistent

ISMA

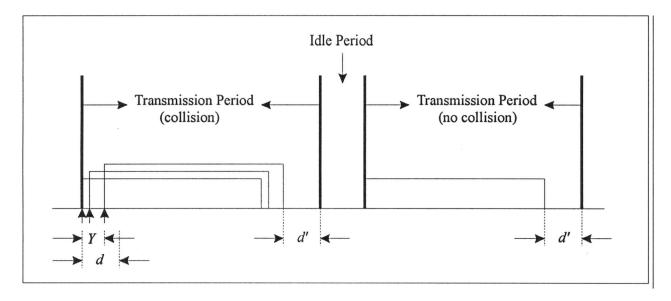


Fig. 3: Data Cycle

III. PERFORMANCE ANALYSIS

The protocol performance has been measured in terms of throughput S and delay D in relationship with the offered traffic G. The throughput measures the efficiency of the protocol and is defined as the fraction of time in which correct data packets are received. The delay is the time between the arrival of the first bit of a data packet at a terminal and its arrival of the last bit at the destination.

A. Throughput

The performance analysis of the hybrid unslotted non-persistent CDMA/ISMA will be explained from Fig. 3. The time axis in Fig. 3 is normalized to the packet duration. If a packet arrives and the terminal senses the channel free, then the packet is sent immediately. Suppose that this time instant is t. It takes d, the inhibit delay fraction, before a busy tone is heard by all terminals. This moment occurs at the same time at all terminals because of the symmetric network configuration. Any other packet arriving between t and t+d will sense the channel free (because the busy tone has not arrived yet) and will be

transmitted resulting in a conflict. Because a CDMA scheme is also used, the conflict does not necessarily result in the loss of all packets. There is a probability that the first arriving packet can still be recovered successfully. If no other terminal transmits a packet during this period *d*, then no conflict occurs. We model the channel between the terminals and the corresponding transceivers as a multipath Rayleigh-fading channel. So, when no conflicts occur, there is still a probability that the packet *cannot* be recovered successfully.

Let t+Y be the time of occurrence of the last packet arriving between t and t+d. This obviously means that Y must be between zero (the first transmitted packet is the only packet in the transmission period) and d (the end of the vulnerable period). The transmission of all packets arriving in (t, t+Y) will be completed at t+Y+1. As noted before, the channel is sensed unused only a period d later. So, any terminal becoming ready between t+d and t+Y+1+d will sense the channel busy and hence will reschedule its packet. The interval between t and t+Y+1+d is called a *transmission period* (*TP*). There will be *at most* one successful transmission during a *TP*. The *idle period* (*IP*) is defined as the period of time between two consecutive *TP*'s. A transmission period plus the following idle period constitute a *cycle*. Let \overline{TP} be the expected duration of the transmission period, \overline{I} the expected duration of the idle period, and the average cycle time can be written as:

$$\overline{t_c} = \overline{TP} + \overline{I}$$
(1)

Let U denote the time during a cycle in which the inbound channel (towards base station) is used to carry a successful packet transmission and \overline{U} the corresponding average value, then we can write the throughput as [12]:

$$S = \frac{\overline{U}}{t_c} \tag{2}$$

The expected useful time \overline{U} can easily be computed. When a packet is successful, the channel carries useful information for a duration of T_{pd} , the packet duration. In the unsuccessful case no useful information is carried at all or, in other words:

$$U = \begin{cases} T_{pd} & Successful \ period \\ 0 & Unsuccessful \ period \end{cases}$$
(3)

If $P_{success}$ denotes the probability that a transmitted packet is successful then

$$U = T_{pd} \cdot P_{success} \tag{4}$$

As noted before, the time is normalized to T_{pd} (the packet duration), and therefore T_{pd} equals to one. So, this gives us:

$$\overline{U} = P_{success} \tag{5}$$

To calculate the average idle period we make the assumption that the total rate at which users schedule new and retransmitted packets forms a Poisson process with parameter G. So, new and rescheduled packets arrive at a rate of G packets per unit time. In literature, G is also called the offered channel traffic. If the packet arrival times are independent and exponentially distributed with a mean arrival rate of G packets per second, the probability of k arrivals in an interval of duration t is then a Poisson process given by $P_k(t)$ where

$$P_k(t) = \frac{\left(Gt\right)^k \cdot e^{-Gt}}{k!} \tag{6}$$

The probability of the idle time being greater than some value t is the probability that no packets are scheduled within a time interval of duration t and with the assumed Poisson packet scheduling process this probability becomes e^{-Gt} . Therefore the average value of I can be expressed as:

$$\overline{I} = \frac{1}{G} \tag{7}$$

The average duration of a transmission period equals to:

$$\overline{TP} = 1 + \overline{Y} + d \tag{8}$$

where \overline{Y} is the expected value of Y. Since Y denotes the time at which the last interfering packet is scheduled, the probability of Y being smaller than some time y is the probability that no other packets (either new or retransmissions) are scheduled for transmission in an interval of duration d-y. With Poisson arrivals, the distribution for Y is $e^{-G(d-y)}$. The average of Y is therefore given by:

$$\overline{Y} = d - \frac{1}{G} \left(1 - e^{-Gd} \right) \tag{9}$$

Applying the equations (8) and (9), we get:

$$\overline{t}_c = \overline{TP} + \overline{I} = 1 + \overline{Y} + d + \overline{I} = 1 + \left[d - \frac{1}{G}\left(1 - e^{-Gd}\right)\right] + d + \frac{1}{G}$$
(10)

Using (5) and (10), the throughput can be given as:

$$S = \frac{\overline{U}}{\overline{t}_c} = \frac{P_{success}}{1 + \left[d - \frac{1}{G}\left(1 - e^{-Gd}\right)\right] + d + \frac{1}{G}} = \frac{G \cdot P_{success}}{G\left(1 + 2d\right) + e^{-Gd}}$$
(11)

The term $P_{success}$ in (10) is the only term that needs to be specified. Herefore, we distinguish between four types of transmissions during a transmission period:

- 1. Successful transmission without conflict
- 2. Unsuccessful transmission without conflict
- 3. Successful transmission with conflict
- 4. Unsuccessful transmission with conflict

If the channel is used for the delivering of a successful packet during a TP then this is denoted by a successful transmission (situation 1 or 3). The difference between a transmission with or without conflict can be found in the number of packet transmissions during a TP. In case there is more than one packet transmission during a TP, we speak of a transmission with conflict. If there is only one packet transmission then the transmission is conflict-free. Keeping this in mind, we can divide $P_{success}$ into two parts:

$$P_{success} = P_{success|noconflict} + P_{success|conflict}$$
(12)

In which $P_{success|noconflict}$ corresponds to situation 1 and $P_{success|conflict}$ to situation 3.

$$P_{success|noconflict} = e^{-Gd} P_{ps}$$
(13)

This is equal to the probability that no terminal transmits during the inhibit delay fraction d multiplied by the packet success probability P_{ps} . The multiplication with P_{ps} is necessary because of the assumption of the Rayleigh fading channel.

$$P_{ps} = \left(1 - P_{be|x=0}\right)^{L_p} \tag{14}$$

 $P_{be|x=0}$ is the bit error probability in Rayleigh fading channel using the CDMA scheme and is caused by self interference due to multipath. L_p denotes the packet length. We assume that during the packet duration, the bit error rate remains constant. The calculation of $P_{success|noconflict}$ is not very complicated. However, this is not the case for $P_{success|conflict}$ which is the probability that the first packet has arrived successfully given a collision (between two or more packets).

$$P_{success|conflict} = \sum_{x=1}^{K-1} P_x(d) \cdot P\{first_packet_OK|x\}$$
(15)

in which:

K is the number of active users in the system;

x denotes the number of arrivals during d;

 $P_x(d)$ is the probability of x arrivals during d for a Poisson distribution and is given by (6).

Consider a reference terminal T_R . This terminal sends its data packet at an idle period first. Other terminals sending within the inhibit delay fraction interfere with the reception at this reference terminal T_R . The bit error rate depends on the configuration. So, if we try to take the different configurations into account, we have to calculate the bit error probability for all possible configurations and then average.

$$P\{first_packet_OK|x\} = E\left[1 - P_{be|\vec{x}}\right]^{L_p}$$
(16)

E[.] denotes the expectation value, and \vec{x} a certain configuration of *x*. The total number of combinations for \vec{x} equals $\binom{K-1}{x}$.

Because we assume an uniform distribution for all possible configurations we can write (K is the total number of active users in the system):

$$E\left[1-P_{be|\vec{x}}\right]^{L_p} = \frac{1}{\binom{K-1}{x}} \cdot \sum_{\vec{x}} \left(1-P_{be|\vec{x}}\right)^{L_p}$$
(17)

$$P_{success|conflict} = \sum_{x=1}^{K-1} P_x(d) \cdot \frac{1}{\binom{K-1}{x}} \cdot \sum_{\vec{x}} \left(1 - P_{be|\vec{x}}\right)^{L_p}$$
(18)

$$P_{success} = e^{-Gd} \left(1 - P_{be|x=0} \right)^{L_p} + \sum_{x=1}^{K-1} P_x(d) \cdot \frac{1}{\binom{K-1}{x}} \cdot \sum_{\vec{x}} \left(1 - P_{be|\vec{x}} \right)^{L_p}$$
(19)

The formula for the bit error probability with DPSK modulation in Rayleigh fading channel is given by Kavehrad and Ramamurthi (formula 13a in [13]) in which the signal to noise ratio (SNR) is taken to be 20 dB. Combining the above results, the throughput can easily be calculated as:

$$S = \frac{G\left(e^{-Gd}\left(1 - P_{be|x=0}\right)^{L_{p}} + \sum_{x=1}^{K-1} P_{x}(d) \cdot \frac{1}{\binom{K-1}{x}} \cdot \sum_{\vec{x}} \left(1 - P_{be|\vec{x}}\right)^{L_{p}}\right)}{G(1+2d) + e^{-Gd}}$$
(20)

B. Delay

The average packet delay is defined as the duration between the transmission of the first bit till the correct reception of the last bit. For the calculation of the packet delay of the unslotted hybrid CDMA/ISMA protocol, we use the block diagram depicted in Fig. 4.

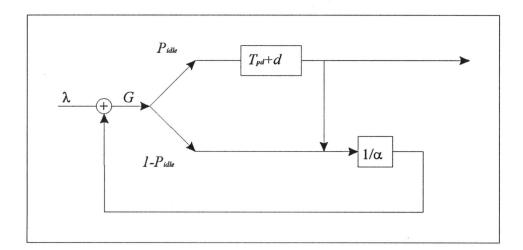


Fig. 4: Delay in the unslotted non-persistent hybrid CDMA/ISMA protocol

If a new packet arrives, the terminal immediately senses the channel to decide if it can transmit the packet. In case the channel is sensed busy (probability $1-P_{idle}$), the packet is rescheduled for a later time instant. This random delay process is assumed to be negative exponentially distributed with parameter α . The average random delay time is then $1/\alpha$. The average number of times the packet has to suffer this delay is $G(1-P_{idle})/S$.

On the other hand, if the channel is sensed idle (probability P_{idle}), the packet will be transmitted immediately. This transmission will take a packet duration T_{pd} plus the inhibit delay fraction d before the knowledge about the outcome of the transmission is available (see also Fig. 3). So, the total delay until the terminal knows if the transmission was successful or not is $(T_{pd} + d)$. The transmission of a data packet does not necessarily result in a successful transmission; there are two possibilities: a successful or a failed transmission. If the transmission was successful, the packet leaves the system. The delay this packet suffers is $(T_{pd} + d)$. If the transmission was a failure, the packet again has to wait a random delay. In this case, the average delay is $(T_{pd} + d + \frac{1}{\alpha})$ and the average number of schedulings

is $\left(\frac{GP_{idle}}{S} - 1\right)$. Combining the above results, the average delay is finally given by:

$$D = \left(\frac{GP_{idle}}{S} - 1\right) \left[T_{pd} + d + \frac{1}{\alpha}\right] + \frac{G\left(1 - P_{idle}\right)}{S} \cdot \frac{1}{\alpha} + \left(T_{pd} + d\right)$$
(21)

S can be found in formula (20) and P_{idle} still has to be calculated. The average probability of sensing the channel idle is the average probability that the channel is idle and this probability is equal to the average idle time (\overline{I}) divided by the average cycle length (\overline{t}_c).

$$P_{idle} = \frac{\overline{I}}{\overline{t}_c} = \frac{\frac{1}{G}}{\frac{1}{G} + \left\{1 + d - \frac{1}{G}\left(1 - e^{-Gd}\right) + d\right\}} = \frac{1}{G\left(1 + 2d\right) + e^{-Gd}}$$
(22)

IV. COMPUTER SIMULATION

In this section we describe how the computer simulation program was set up to measure the performance of the unslotted hybrid CDMA/ISMA protocol.

A. Representation of Time

In general, if we want to use a computer simulation to analyse a system, the following steps can be distinguished. The dynamic behaviour of the system is studied by tracing various system states as a function of time and then collecting and analysing the system statistics. The events that change the

system state are generated at different points in time, and the passage of time is represented by an internal clock which is incremented and maintained by the simulation program.

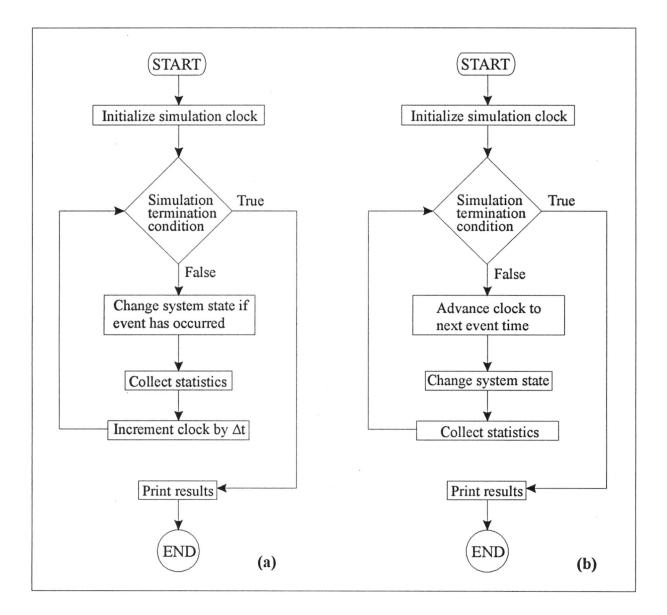


Fig. 5: (a) Interval-oriented simulation; (b) Event-oriented simulation

The simulation time can be advanced in two ways. The first method is the *interval-oriented simulation* (or the uniform time increment method) where the clock is advanced from time t to $(t + \Delta t)$ where Δt is a uniform fixed time increment. Fig. 5a depicts this mechanism. The second method is the *event-oriented simulation* (or the variable time increment method) where the clock is incremented from time t

to the next event time t', whatever may be the value of t'. The state changes are made at event time t, the next event time t', and this process is continuously repeated. Thus, only events are represented explicitly in a simulation model and the periods between events are treated as inactive or insignificant and therefore consume no time even though the interevent activities do consume time in the real world (Fig. 5b).

Obviously, method 1 detects the events that occur during the interval $(t, t + \Delta t)$ only at time $(t + \Delta t)$, thereby introducing errors in simulation. Another drawback of this method is that if the interval between two events is very large compared to Δt , then the simulator goes through several unproductive clock increments (during periods of inactivity) and the associated computing effort which will not bring about any change in system states. This fixed time increment method is suitable for the simulation of continuous systems and in particular systems with large numbers of state variables.

The second method involves sorting of event activation times and maintaining *events list*. In our work we will use the event-oriented simulation (Figure 5b) to simulate the unslotted hybrid CDMA/ISMA protocol.

B. Main Structure

The Program Structure Diagram (PSD) of the main program is depicted in Fig. 6. The simulation program simulates the working of the unslotted hybrid CDMA/ISMA protocol. First of all, the counters have to be reset to the proper values. *Clock* indicates the actual time while the simulation will last *EndTime*. While *Clock* has not achieved *EndTime*, the simulation will loop. The loop is divided in four steps as shown in Fig. 6. In the first step, the program advances *Clock* to the next event time. This event is the first event to occur in future time. Because the event-oriented simulation is used, only events are represented explicitly in the simulation model. Depending on the event type, the program processes this event. This is the second step. The processing can generate other events or change the system state. In the third step, the generated events and the changed system state will be updated. Finally, statistics collection completes the loop.

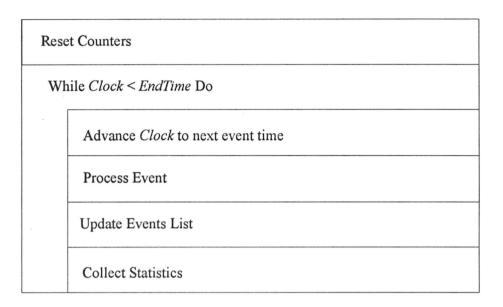


Fig. 6: PSD of main program

C. Event Lists

From Fig. 6 it emerges that events and event list play an important role in the main program structure.

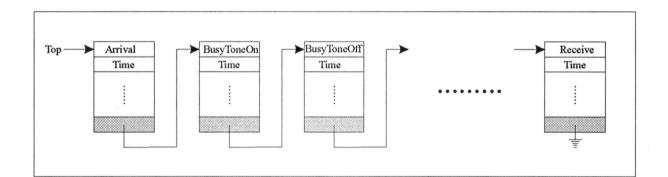


Fig. 7: Example of an event list

The whole program is based on the event list. The event list consists of events linked together in an organised manner. Here, organised means in an order in which the events occur. An example of an event list is shown in Fig. 7. The event list consists of nodes linked together to form a chain. In the event oriented simulation this chain symbolizes the time axis. Every node represents an event and consists of

an *EventType*, the time this event occur, a tail and other information. The tail consists of a pointer pointing to the next node. The last node points to nowhere. This is illustrated with the ground symbol.

IV. ASSUMPTIONS

After DPSK modulation the signal is spread with a Gold code. The length of these codes are chosen to be 31. The signal is then transmitted, modulated on a 1.7 GHz carrier. The data rate is arbitrarily chosen to be 256.1024 b/s (0.26 Mb/s). The packet length is 64 bits. The delay spread is supposed to be 100 ns. Perfect power control is assumed to assure that signals from terminals within the same group arrive at the transceiver with the same power.

The far field model [14] is chosen to describe the signal attenuation. Given the transmitted power P_T , the received power P_R can be expressed by the following equation:

$$P_R = \frac{g_T g_R}{\left(\frac{4\pi f l}{c}\right)^{\alpha}} P_T \tag{19}$$

where g_T and g_R are respectively the transmitter and the receiver gains, f the signal frequency, l the distance between the transmitter and the receiver, c the speed of light and α is the attenuation parameter. This attenuation parameter is chosen equal to 2, corresponding to free space propagation.

Fading is the result of the propagation of the transmitted signal through several paths. The channel is modeled as a Rayleigh fading channel. The Rayleigh fading channel model is valid when each path contributes the same amount of energy to the composite received signal. A Line of Sight (LOS) path is therefore assumed to be absent.

An example of a 16-terminal network configuration that we adopted in this paper is shown in Fig. 2. The terminals are clustered around distributed transceivers at a fixed distance (5 m). The distributed transceivers are clustered around the base station at a fixed distance (30 m). Those distances are kept fixed for all simulations. The number of terminals within a transceiver group (also called group size) is chosen as a power of two. For a certain fixed number of participated terminals, we have to halve the group size if we want to double the number of codes (the number of codes is also the number of transceivers). The comparison is fair in this manner, because when we want to investigate the effect of the number of codes on the performance, we have to keep the number of participated terminals fixed. The number of terminals is 32 unless stated otherwise.

V. RESULTS

Fig. 5 a and b compare the performance of the slotted and unslotted hybrid protocol using computer simulation.

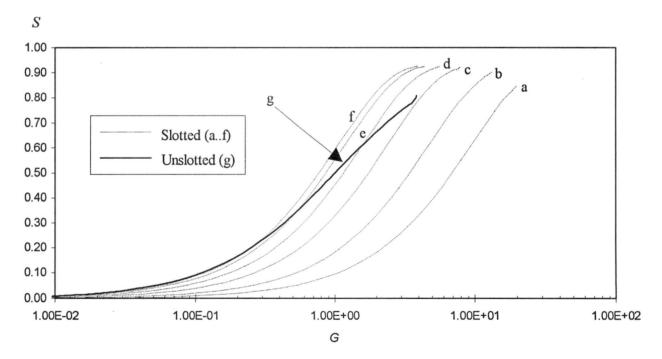


Fig. 5a: Throughput versus offered traffic using simulation of slotted (a..f: $P_{tr} = 0.1, 0.2, 0.4, 0.6, 0.8,$

0.9) and unslotted (g: d=0.01) hybrid protocol; 8 codes; $\alpha=0.1$

In [1] P_{tr} is defined for the slotted protocol as the probability that the packet will actually be transmitted in case the channel is sensed *free*. For low traffic, the delay of the unslotted protocol is, as expected, better than the slotted protocol. For high traffic the situation is the other way round. The slotted protocol does not differ much from the unslotted protocol for high values of P_{tr} ,

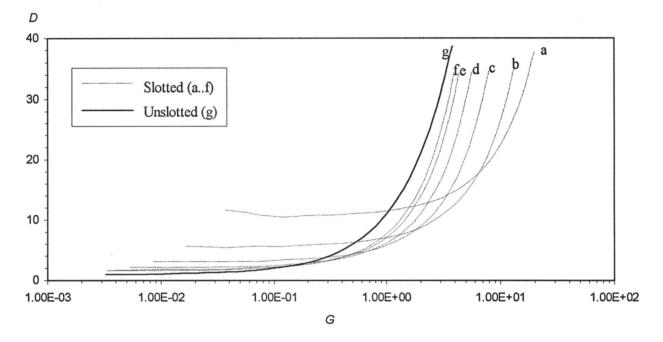


Fig. 5b: Delay versus offered traffic using simulation of slotted (a..f: $P_{tr} = 0.1, 0.2, 0.4, 0.6, 0.8, 0.9$) and unslotted (g: d=0.01) hybrid protocol; 8 codes; $\alpha=0.1$

Fig. 6 shows the effect of the inhibit delay fraction d on the performance of the unslotted protocol. For high values of d, the performance degrades tremendously. This is due to the increase of the number of data packets colliding during this period. This is why the ISMA protocol is not useful for environment in which the propagation delay is high.

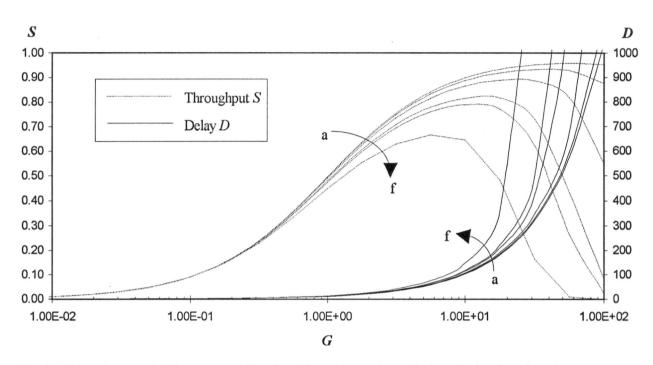


Fig. 6: Throughput and delay versus offered traffic using mathematical analysis of unslotted protocol, varying the inhibit delay fraction d. a. f: d=0.01, 0.02, 0.04, 0.08, 0.1 and 0.2; 8 codes; $\alpha=0.1$

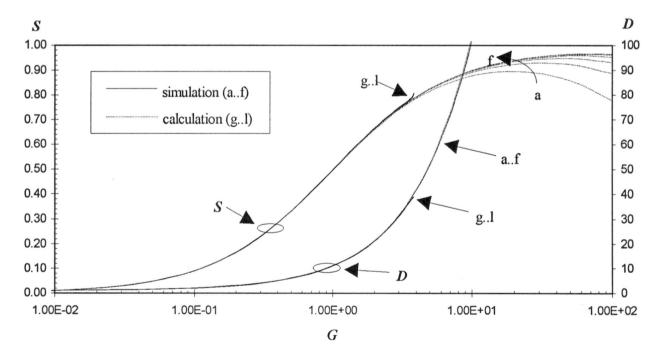


Fig. 7: Comparison between simulation and mathematical analysis for the unslotted protocol. a..f, g..l:
1, 2, 4, 8, 16, 32 codes; d=0.01; α=0.1

The comparison between the simulation and mathematical model for the unslotted protocol is given in Fig. 7. In case of simulation, the arrival rate at terminals is the input parameter and G is one of the output parameters. In addition, due to the assumption that terminals do not have buffers the performance of simulation does not exceed for $G \approx 4$ packets/(packet duration). The results due to simulation and mathematical model are in good agreement. This graph also shows the effect of code sharing. For a 32-terminal network, it is sufficient to use only four codes.

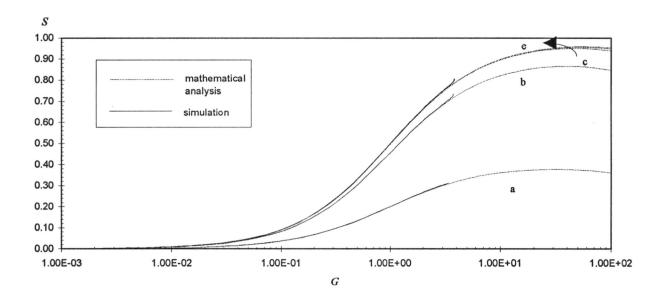


Fig. 8: Comparison between simulation and mathematical analysis for the unslotted protocol and varying SNR; a..e: SNR=6, 8, 10, 12 and 20 dB; number of codes=8; d=0.01; α=0.1

Fig. 8 shows the effect of the signal to noise ratio (SNR) on the throughput of the unslotted protocol. Again, simulation and mathematical analysis agree very well with each other. Of course, the throughput improves for increasing SNR. However, Fig. 8 also shows that a SNR of 12 dB suffice. For values of SNR higher than 12 dB, the protocol performance also improves, but only for high values of the offered traffic and the improvement is only additional.

VI. CONCLUSIONS

The unslotted hybrid CDMA/ISMA protocol has been investigated. The protocol performance is measured in terms of throughput S and delay D in relationship with the offered traffic G. Herefore, we have been able to derive a close form formula for the throughput and delay. Also a simulation program has been developed to compare the results. The results from simulation and mathematical agree very well with each other.

Comparison between the slotted and unslotted protocol shows that for low offered traffic the delay for the unslotted protocol is better than the slotted protocol. For high traffic it is the other way round. Furthermore it is concluded that the proposed hybrid protocol performs very well in case the propagation delay is not a concern. If the propagation delay is too high the performance drops quickly. The results also show that the hybrid protocol supports code sharing extremely good. For a 32 terminal network, only four codes suffice.

Thus it can be concluded that unslotted hybrid CDMA/ISMA protocol will be highly suitable for the indoor wireless computer communication.

REFERENCES

[1] R. Prasad,

"Performance Analysis of Mobile Packet Radio Networks in Real Channels with Inhibit-Sense Multiple Access",

IEE Proc. I, Vol. 138, pp. 458-464, October 1991.

[2] I. Widipangestu, A.J. 't Jong and R. Prasad,

"Capture Probability and Throughput Analysis of Slotted ALOHA and Unslotted np-ISMA in a Rician/Rayleigh Environment",

IEEE Transactions on Vehicular Technology, Vol. 43, No. 3, pp. 457-465, August 1994.

- K.J. Zdunek, D.R. Ucci and J.L. Locicero,
 "Throughput of Nonpersistent Inhibit Sense Multiple Access with Capture", Electronics Letters, Vol. 25, No. 1, pp. 30-31, 5th January 1989.
- [4] R. Prasad, and C.Y. Liu,
 "Throughput Analysis of Some Mobile Packet Radio Protocols in Rician Fading Channels",
 IEE Proceedings-I, Vol. 139, No. 3, pp. 297-302, June 1992.
- [5] M.G. Jansen, and R. Prasad,

"Capacity, Throughput, and Delay Analysis of a Cellular DS CDMA System With Imperfect Power Control and Imperfect Sectorization",

IEEE Transactions on Vehicular Technology, Vol. 44, No. 1, pp. 67-75, February 1995.

[6] R.A. Scholtz,

"The Origins of Spread-Spectrum Communications",

IEEE Transactions on Communications, Vol. COM-30, No. 5, pp. 822-854, May 1982.

[7] W.C.Y. Lee,

"Overview of Cellular CDMA",

IEEE Transactions on Vehicular Technology, Vol. 40, No. 2, pp. 291-302, May 1991.

[8] M.B. Pursley,

"Performance Evaluation of Phase Coded Spread Spectrum Multiple Access Communication -Part I: System Analysis",

IEEE Transactions on Communications, Vol. COM-25, No. 8, pp. 795-799, August 1977.

- [9] R.L. Pickholtz, D.L. Schilling and L.B. Milstein
 "Theory of Spread Spectrum Communications A Tutorial",
 IEEE Transactions on Communications, Vol. COM-30, No. 5, pp. 855-884, May 1982.
- K.S. Gilhousen, I.M. Jacobs, R. Padovani, A.J. Viterbi, L.A. Weaver, Jr. and C.E.Wheatley III,
 "On the Capacity of a Cellular CDMA System",
 IEEE Transactions on Vehicular Technology, Vol. 40, No. 2, pp. 303-312, May 1991.
- [11] H. van Roosmalen, J. Nijhof and R. Prasad,
 "Performance Analysis of a Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications",
 IEEE Journal on Selected Areas in Communications, Vol. 12, No. 5, pp. 909-916, June 1994.
- [12] L. Kleinrock and F.A. Tobagi,
 "Packet Switching in Radio Channels: Part I- Carrier Sense Multiple Access and their Throughput-Delay Characteristics",
 IEEE Trans. on Commun., Vol. COM-23, No. 12, pp. 1400-1416, December 1975.
- [13] M. Kavehrad and B. Ramamurthi,
 "Direct -Sequence Spread Spectrum with DPSK Modulation and Diversity for Indoor Wireless Communications",

IEEE Transactions on Communications, Vol. COM-35, No. 2, pp. 224-236, February 1987.

[14] A. B. Carlson,

Communication Systems,

Mc. Graw Hill Book Company, 3rd edition, 1988, pp. 92.

APPENDIX C:

"Unslotted Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications"

Submitted for publication in IEEE Journal on Selected Areas of Communications.

Unslotted Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications

Huy Linh Anh Le, Student Member, IEEE Huub van Roosmalen Jos Nijhof, Member, IEEE Ramjee Prasad, Senior Member, IEEE

Telecommunications and Traffic Control Systems Group Delft University of Technology P.O. Box 5031, 2600 GA Delft The Netherlands

Abstract

A hybrid Code Division Multiple Access / Inhibit Sense Multiple Access (CDMA/ISMA) protocol has been proposed as an effective protocol for Indoor Wireless Computer Communications. This new protocol combines the advantages of both CDMA and ISMA into one protocol. On the one hand the ISMA protocol introduces a limitation to the number of simultaneous accesses to the transmission channel. On the other hand the CDMA protocol introduces an improvement to the packet survival chance. It is shown that the performance of the hybrid protocol is indeed better than CDMA only. In addition, code sharing can be applied to reduce hardware cost. To take more advantage of the strength of the hybrid protocol, we have investigated the *unslotted non-persistent* ISMA scheme instead of the *slotted p-persistent* ISMA scheme. The performance comparison between the slotted and unslotted hybrid CDMA/ISMA protocol is evaluated in terms of throughput and delay using computer simulation.

I. INTRODUCTION

On the one hand the Inhibit Sense Multiple Access (ISMA) protocol introduces a limitation to the number of simultaneous accesses to the transmission channel. On the other hand the Code Division Multiple Access (CDMA) protocol introduces an improvement to the packet survival chance. It is therefore to be expected that the combination of the strengths of the two protocols could improve performance. This concept is described below.

Indoor wireless office communication is our main research field. The office system we focus on consists of a building in which users work together in groups. The participants generate terminal traffic. Terminals communicate with each other by radio transmission using a random access protocol. The information is lost when two or more terminals transmit simultaneously, i.e. a collision occurs. Terminals might not detect each other's transmission in radio communications. It can easily happen that two users are hidden from each other by some obstacle, in which case severe performance degradation results. This is called the *hidden terminal problem*. The introduction of a central base station can alleviate this problem by instructing it to send a busy tone to all participating terminals to forbid new transmissions when a transmission is going on. Still, a situation can occur in which two or more terminals simultaneously start their transmission, resulting in a collision. However, a great reduction in the number of simultaneous transmissions is achieved by the introduction of a central base station. This concept is called the Inhibit Sense Multiple Access (ISMA) [1]-[4].

If we could somehow increase the survival chances of colliding packets, we could improve protocol performance. The near-far effect is one way to achieve this. A packet may 'capture' the receiver if it is much stronger than its competitors. This can happen when terminals use the same transmission power, but at different distances from the receiver. Because the 'near terminals' have better performance compared to the 'far terminals' (due to the better survival chance of the packets), the near-far effect introduces an unfair element among the terminals. Perfect power control, in which the transmitted power of the terminals are adjusted such that their received powers are all equal, can eliminate the near-far effect described above. The performance of CDMA has been reported in a number of publications, e.g. [5]-[10].

A hybrid CDMA/ISMA protocol has been proposed in [11]. This new protocol combines the advantages of both CDMA and ISMA into one protocol. The advantages of CDMA and ISMA are the improvement of the survival chance of packets and the limitation of contention in the channel. A less complex receiver structure is needed. More importantly, the number of distinct useful transmission codes is limited, especially for short code length.

In [11] the hybrid protocol combines Direct Sequence CDMA with *slotted* p-persistent ISMA. In this paper, we have investigated the performance of the hybrid CDMA/ISMA protocol using the *unslotted* non-persistent ISMA scheme. It is expected that for low traffic, the delay of the unslotted protocol will improve because when a terminal has a data packet to send, it does not have to wait until the start of the next time slot. In addition, even when data packets collide, there is a probability that the data is received correctly due to the use of CDMA. This is the strength of the hybrid CDMA/ISMA protocol. By using the *unslotted non-persistent* ISMA scheme, it is expected that we take more advantage of this strength.

This paper is organized as follows. Section II gives the protocol description of both the slotted and unslotted hybrid ISMA/CDMA protocol. Assumptions are given in Section III. Computer simulations and results are discussed in Section IV. Finally, conclusions can be found in Section V.

II. PROTOCOL DESCRIPTION

Figure 1 depicts the network structure of the protocol. There is one central base station that is connected to several receivers by wire. The base station controls the traffic flow with a busy tone that can be detected by all participating terminals. Because all terminals can detect the busy tone, the hidden terminal problem is solved. Several terminals together with one receiver form a group. Wireless communication is considered to take place between the receiver and the terminals. The terminals around each receiver share the same code. In this way the number of codes can be reduced. In this paper we assume that the several groups have the same groupsize; the distance from the central base station to the several receivers is the same and the distance between the terminals and the receiver within one group is also the same. When a data packet is ready for transmission, a terminal transmits the packet to its receiver according to the chosen ISMA scheme. Two ISMA schemes are important for us and have been discussed later on: *slotted* p-persistent ISMA protocol and *unslotted* non-persistent ISMA protocol.

After the arrival of the data packet, the receiver then simply forwards this packet to the base station which broadcasts it to all terminals. The destination then receives the packet and decides whether the packet is errorless or not. In case the packet was erroneous, a retransmission must take place. The return channel is not included in our analysis because only the base station makes use of this return channel and therefore contention is not a concern. The state of the terminals can either be free or blocked. At the beginning, the terminals are in the free state. If a packet arrives at a free terminal, the terminal jumps into the blocked state. In the blocked state, the terminal takes care that the arriving data packet is serviced successfully. In the mean time the blocked terminal ignores all incoming packets. This is a consequence of the assumption that the terminals do not have buffers for the incoming data packets. The arrivals of data packets at terminals are assumed to be generated by a Poisson process. This means that the inter-arrival times of data packets at terminals are negative exponentially distributed with P_{Arr} as parameter (P_{Arr} indicates the average number of arrivals per terminal per time unit).

A. The slotted p-persistent ISMA protocol

Users behind terminals generate data packets. The data packets arrive at terminals, which will take care of the correct delivery of the packets. We assume that the arrivals of these packets are generated by a Poisson process. The Specification and Description Language (SDL) like diagram of the slotted p-persistent ISMA protocol is depicted in Figure 2. In this case, the terminal uses the slotted p-persistent ISMA protocol to service its data packet when it is in the blocked state. Before transmitting the data packet to the receiver, the terminal first listens to the channel to hear whether there is a busy tone transmitted by the base station. The base station broadcasts a busy tone to signal that the channel is busy because there is a transmission in progress. If there is a busy tone, indicating that the channel is busy, the terminal postpones its transmission to the next time slot. Otherwise, if the channel is free, the terminal performs a random binary experiment to determine whether it will send its data packet. If the terminal decides not to transmit, it waits until the next time slot and draws a number again to see if it is permitted to send. This mechanism is built in to reduce the number of simultaneous transmissions. Simultaneous transmissions cause collisions and must be reduced as much as possible.

If the terminal is permitted to send, it transmits its data packet to the receiver. It is possible that two or more terminals transmit in the same time slot. In this case a collision occurs. Because we also make use of CDMA, a collision does not necessarily result in the loss of the data packets. There is a probability that the receiver can lock onto one of the data streams and this data packet can be received properly. In case the data packet is received erroneously, the terminal then waits for a random period and starts all over again (see also Figure 2). On the other hand, if no collision occurs, the terminal assumes that the packet has been received correctly and leaves the blocked state to jump into the free state. The random delay for retransmissions is set to zero in our experiment because the random binary experiment already introduces a delay after a collision.

B. The unslotted non-persistent ISMA protocol

The SDL-like diagram of the unslotted non-persistent ISMA protocol is shown in Figure 3. This ISMA protocol is unslotted. There are two major differences between this unslotted and the previously described slotted protocol. The first difference is that the terminal waits a random period when it senses that the channel is busy. Because the time axis is not divided into time slots, terminals have to wait a random period. After the random delay, the terminals are allowed to sense the channel again to see if it

is free. The random period is modeled to be negative exponentially distributed. The second difference is related to the first one. In the unslotted scheme, the terminals immediately transmit if the channel is free. As a consequence of the first difference mentioned above, the terminals already wait a random period in case the channel is busy. So, if the channel is free the number of terminals attempting to transmit is already reduced. It is not necessary to let the terminals draw a random number to see if they may transmit. Although the number of collisions is greatly reduced with ISMA, collisions can still occur. In this unslotted ISMA scheme, collisions may occur due to multiple terminals transmitting packets during an interval called the *inhibit delay fraction* (d). This interval d is necessary to switch from 'idle' to 'busy'. The reverse interval from 'busy' to 'idle' is denoted by d'. The inhibit delay fraction is normalized to the packet length, resulting in $0 \le d < 1$ (Figure 4). In this paper we assume d = d'.

III. ASSUMPTIONS

A. Transmitter and Receiver

After DPSK modulation the signal is spread with a Gold code. The length of these codes are chosen to be 31 and 127 to evaluate the influence of code length. The signal is then transmitted, modulated on a 1.7 GHz carrier. The data rate is arbitrarily chosen to be 256.1024 b/s (0.26 Mb/s). The packet length is 64 bits. Perfect power control is assumed to assure that signals from terminals within the same group arrive at the receiver with the same power.

We consider one antenna for each receiver. So, only spread spectrum diversity is taken into account. The number of resolvable paths L can be found with the following formula [12]:

$$L = \left\lfloor \frac{T_m}{T_c} \right\rfloor + 1 \tag{1}$$

where T_m is the RMS delay spread, T_c is the chip duration and $\lfloor x \rfloor$ denotes the largest integer smaller than or equal to x. The relationship between T_c the chip duration and T_b the bit duration is given by the code length N:

$$T_c = \frac{T_b}{N} \tag{2}$$

We have a number of resolvable paths due to spread spectrum. The maximum value for T_m has been measured by Saleh and Valenzuela and is 100 ns [13]. For our work we assume T_m to be 100 ns. For N = 31 this gives one resolvable path, and for N = 127 this gives us four resolvable paths.

B. Signal Attenuation Model

The far field model is chosen to describe the signal attenuation [14]. Given the transmitted power P_T , the received power P_R can be expressed by the following equation:

$$P_{R} = \frac{g_{T}g_{R}}{\left(\frac{4\pi fl}{c}\right)^{\alpha}}P_{T}$$
(3)

where g_T and g_R are respectively the transmitter and the receiver gains, f the signal frequency, l the distance between the transmitter and the receiver, c the speed of light and α is the attenuation parameter. This attenuation parameter is chosen equal to 2, corresponding to free space propagation.

C. Rayleigh Fading Channel

Fading is the result of the propagation of the transmitted signal through several paths. The channel is modeled as a Rayleigh fading channel. The Rayleigh fading channel model is valid when each path contributes the same amount of energy to the composite received signal. A Line of Sight (LOS) path is therefore assumed to be absent. The formula for the bit error probability is given by Kavehrad and Ramamurthi [15]:

$$P_{be|\beta_{\max},\{\tau_{lk}\},L} = Q(a,b) - \frac{1}{2} \left(1 + \frac{\mu_{12}}{\sqrt{\mu_1 \mu_2}} \right) I_0(a,b) \exp\left(-\frac{a^2 + b^2}{2}\right)$$
(4.1)

where

Q(a,b) is the Marcum Q-function

$$a = \frac{m}{\sqrt{2}} \left| \frac{1}{\sqrt{\mu_1}} - \frac{1}{\sqrt{\mu_2}} \right|, \text{ and } b = \frac{m}{\sqrt{2}} \left| \frac{1}{\sqrt{\mu_1}} + \frac{1}{\sqrt{\mu_2}} \right|$$
(4.2)

$$m = A\beta_{\max} T_b b_1^{\ 0} = A\beta_{\max} T_b b_1^{-1}$$
(4.3)

$$\mu_{1} = A^{2}E\left[\sum_{k=1}^{Nt} X_{k}^{2} + \hat{X}_{k}^{2} + Y_{k}^{2} + \hat{Y}_{k}^{2} | \{\tau_{lk}\}, L\right] + 2A^{2}E\left[X_{1}\hat{X}_{1} + Y_{1}\hat{Y}_{1} | \{\tau_{lk}\}, L\right] + 2\sigma_{n}^{2}$$

$$(4.4)$$

$$\mu_{2} = A^{2} E \left[\sum_{k=1}^{Nt} X_{k}^{2} + \hat{X}_{k}^{2} + Y_{k}^{2} + \hat{Y}_{k}^{2} | \{\tau_{lk}\}, L \right] + 2\sigma_{n}^{2}$$
(4.5)

$$\mu_{12} = A^2 E \left[\sum_{k=1}^{Nt} \left(X_k \hat{X}_k + Y_k \hat{Y}_k \right) + \hat{X}_1^2 + \hat{Y}_1^2 | \{ \tau_{1k} \}, L \right]$$
(4.6)

In which σ_n^2 is the thermal noise power; *A* is the transmitted signal amplitude; *Nt* denotes the number of active users; *L* is the number of paths; β_{lk} and τ_{lk} are the *l*th path gain and delay respectively, for the *k*th user; β_{max} denotes the largest value of β_{l1} . The conditional expectations in the above expressions can be evaluated as follows:

$$E[X_1^2|\{\tau_{lk}\},L] = E[Y_1^2|\{\tau_{lk}\},L] = \sum_{\substack{l=1\\l\neq j}}^{L} R_{11}^2(\tau_{l1})\rho_{l1}$$
(5.1)

$$E[\hat{X}_{1}^{2}|\{\tau_{lk}\},L] = E[\hat{Y}_{1}^{2}|\{\tau_{lk}\},L] = \sum_{\substack{l=1\\l\neq j}}^{L} \hat{R}_{11}^{2}(\tau_{l1})\rho_{l1}$$
(5.2)

$$E[X_1\hat{X}_1|\{\tau_{lk}\},L] = E[Y_1\hat{Y}_1|\{\tau_{lk}\},L] = \sum_{l=1\atop l\neq j}^{L} R_{11}(\tau_{l1})\hat{R}_{11}(\tau_{l1})\rho_{l1}$$
(5.3)

$$E[X_{k}^{2}|\{\tau_{lk}\},L] = E[Y_{k}^{2}|\{\tau_{lk}\},L] = \sum_{l=1}^{L} R_{lk}^{2}(\tau_{lk})\rho_{lk}$$
(5.4)

$$E[\hat{X}_{k}^{2}|\{\tau_{lk}\},L] = E[\hat{Y}_{k}^{2}|\{\tau_{lk}\},L] = \sum_{l=1}^{L}\hat{R}_{lk}^{2}(\tau_{lk})\rho_{lk} \qquad ;k \ge 2$$
(5.5)

APPENDIX C

$$E[X_k \hat{X}_k | \{\tau_{lk}\}, L] = E[Y_k \hat{Y}_k | \{\tau_{lk}\}, L] = \sum_{l=1}^{L} R_{lk}(\tau_{lk}) \hat{R}_{lk}(\tau_{lk}) \rho_{lk} \qquad ; k \ge 2$$
(5.6)

In which

$$R_{1k}(\tau) = \int_0^\tau a_k(t-\tau)a_1(t)dt$$
(5.7)

$$\hat{R}_{1k}(\tau) = \int_{\tau}^{T} a_k(t-\tau) a_1(t) dt$$
(5.8)

$$a_{k}(t) = \sum_{i} a_{k}^{i} P_{T_{c}}(t - iT_{c})$$
(5.9)

$$b_k(t) = \sum_{j=-\infty}^{\infty} b_k^j P_T(t - jT_b)$$
(5.10)

and
$$\rho_{lk} = \frac{1}{2} E \left[\beta_{lk}^2 \right]$$
 (5.11)

D. Network Configuration

An example of a 16-terminal network configuration that we adopted in this paper is shown in Figure 5. The terminals are clustered around distributed receivers at a fixed distance (5 m). The distributed receivers are clustered around the base station at a fixed distance (30 m). Those distances are kept fixed for all simulations. The number of terminals within a receiver group (also called group size) is chosen as a power of two. For a certain fixed number of participated terminals, we have to halve the group size if we want to double the number of codes (the number of codes is also the number of receivers). The comparison is fair in this manner, because when we want to investigate the effect of the number of codes on the performance, we have to keep the number of participated terminals fixed.

IV. COMPUTER SIMULATION AND RESULTS

In this section the performance of the slotted p-persistent and the unslotted non-persistent ISMA scheme is given. The performance analysis for the slotted scheme has been evaluated by computer simulation and Markov model in [11]. In this paper we concentrate on the performance comparison between the slotted and unslotted hybrid CDMA/ISMA protocol through computer simulation.

A1. Representation of Time

As a first approach to analyse the *unslotted* hybrid CDMA/ISMA protocol, a computer simulation program will be set up. The dynamic behaviour of the system is studied by tracing various system states as a function of time and then collecting and analysing the system statistics. The events that change the system state are generated at different points in time, and the passage of time is represented by an internal clock which is incremented and maintained by the simulation program.

The simulation time can be advanced in two ways. The first method is the *interval-oriented* simulation (or the uniform time increment method) where the clock is advanced from time t to $(t + \Delta t)$ where Δt is a uniform fixed time increment. Figure 6a depicts this mechanism. The second method is the *event-oriented simulation* (or the variable time increment method) where the clock is incremented from time t to the next event time t', whatever may be the value of t'. The state changes are made at event time t, the next event time t', and this process is continuously repeated. Thus, only events are represented explicitly in a simulation model and the periods between events are treated as inactive or insignificant and therefore consume no time even though the interevent activites do consume time in the real world (Figure 6b).

Obviously, method 1 detects the events that occur during the interval $(t, t + \Delta t)$ only at time $(t + \Delta t)$, thereby introducing errors in simulation. Another drawback of this method is that if the interval between two events is very large compared to Δt , then the simulator goes through several unproductive clock increments (during periods of inactivity) and the associated computing effort which will not bring about any change in system states. This fixed time increment method is suitable for the simulation of continuous systems and in particular systems with large numbers of state variables.

The second method involves sorting of event activation times and maintaining *events list*. In our work we will use the event-oriented simulation (Figure 6b) to simulate the hybrid unslotted CDMA/ISMA protocol.

A2. Main Structure

The Program Structure Diagram (PSD) of the main program is depicted in Figure 7. The simulation program simulates the working of the hybrid unslotted CDMA/ISMA protocol. First of all, the counters have to be reset to the proper values. *Clock* indicates the actual time while the simulation will last *EndTime*. While *Clock* has not achieved *EndTime*, the simulation will loop. The loop is divided in four steps as shown in Figure 7. In the first step, the program advances *Clock* to the next event time. This event is the first event to occur in future time. Because the event-oriented simulation is used, only events are represented explicitly in the simulation model. Depending on the event type, the program processes this event. This is the second step. The processing can generate other events or change the system state. In the third step, the generated events and the changed system state will be updated. Finally, statistics collection completes the loop.

A3. Event Lists

From Figure 7 it emerges that events and event lists play an important role in the main program structure. The whole program is based on the event lists. The event lists consist of events linked together in an organised manner. Here, organised means in an order in which the events occur. An example of the event list is shown in Figure 8. The event lists consist of nodes linked together to form a chain. In the event oriented simulation this chain symbolizes the time axis. Every node represents an event and consists of an *EventType*, the time this event occur, a tail and other information. The tail consists of a pointer pointing to the next node. The last node points to nowhere. This is illustrated with the ground symbol.

B. Performance Measurements

The protocol performance has been measured in terms of throughput S and delay D in relationship with the arrival rate at terminals P_{Arr} . Note that in [11] the physical traffic is used instead of P_{Arr} (see also Figures 2 and 3 for the physical traffic reference). The physical traffic indicates the number of terminals that actually transmit data packets to the receiver.

The throughput S measures the efficiency of the protocol and is defined as the fraction of time in which correct data packets are received. The delay D is the time between the arrival of the first bit of a data packet at a terminal and its arrival of the last bit at the destination.

First, the performance inherent to the unslotted non-persistent scheme will be presented. After that, we present the performance comparison between the slotted and unslotted scheme.

C. Performance of the unslotted scheme

As discussed before, the inhibit delay fraction (d) is one of the parameters that influences the protocol behavior. By varying this inhibit delay fraction and measuring the performance, Figure 9 and Figure 10 are obtained. Figure 9 depicts the throughput of the unslotted scheme as a function of the arrival rate at terminals for different values of the inhibit delay fraction. This inhibit delay fraction d is varied between 0.01 to 0.2 (note that d is normalized and can vary between zero and one). For low arrival rates, the throughput curves hardly differ. This can be explained by the fact that for low arrival rates, the probability that more than one packet arrives within the inhibit delay fraction is very low. On the other hand, for high traffic, the throughput drops for high value of d. The throughput degradation is due to the collisions of data packets. Because for high traffic, more packets arrive within the inhibit delay fraction if d is too large.

The same explanation applies to the delay curves of the unslotted scheme which is shown in Figure 10. In this figure, the delay is depicted versus the arrival rate at terminals for different values of the inhibit delay fraction. The delay is normalized in the number of packet durations for comparison purposes with the slotted protocol later on. As can be expected from Figure 9, the delays for high traffic get worse for high values of the inhibit delay fraction d.

Another parameter which influences the performance of the unslotted hybrid protocol is the random delay that can be found in Figure 3. Before a retransmission (caused by a collision or a busy channel) can take place, terminals have to undergo a random delay first. This is to prevent too many terminals from retransmitting their data packets at the same time. We model this random process to be negative exponentially distributed with λ as parameter. λ represents the mean number of arrivals per packet duration (the time is normalized in packet duration for comparison purposes) and influences the period the data packet has to wait before it can be retransmitted. The average time a packet has to wait before it can be retransmitted. The average time a packet has to wait before it can be retransmitted. The average time a packet has to wait before it can be retransmitted. The average time a packet has to wait before it can be retransmitted. The average time a packet has to wait before it can be retransmitted. The average time a packet has to wait before it can be retransmitted. The average time a packet has to wait before it can be retransmitted. The average time a packet has to wait before it can be retransmitted. The average time a packet has to wait before it can be retransmitted is then given by: E[random delay]= $1/\lambda$ [16]. As a result of varying λ , we obtain the performance shown in Figures 11 and 12.

For different values of λ , Figure 11 depicts the throughput of the unslotted scheme as a function of the arrival rate at terminals. $\lambda = 0.1$ means that on average 0.1 packets per packet duration will become available for retransmission at each blocked terminal (the time is normalized in packet durations). This results in an average delay of 10 packet durations before a retransmission takes place. For low values of λ , the throughput is also low. This is because erroneous packets have to wait a long period before they may be retransmitted. The same explanation also applies to Figure 12 in which the delay is depicted versus the arrival rate.

C. Comparison between the slotted and unslotted scheme

The compasison between the slotted and unslotted hybrid CDMA/ISMA protocol is presented in Figures 13 and 14. The throughput comparison between the slotted and unslotted hybrid protocol is shown in Figure 13. The number of users is kept constant to 32. The parameter we vary in those curves is the number of codes used (this is also the number receivers or groups in the network). The range in which we vary this parameter is from 1 to 32. For low number of codes used, the performance is low for both slotted and unslotted scheme. Low number of codes used means a high number of users shares the same code because the number of users is kept constant. Here we see the effect of code sharing. For low arrival rates, code sharing has not much influence because the probability that the terminals in the same group start sending at the same time is very low.

For high arrival rates, the throughput performance of the unslotted scheme is better than the slotted one. The explanation can be found in the characteristics of the retransmission mechanism for both schemes. For the slotted scheme this is a random binary experiment and for the unslotted scheme this is the Poisson process with the the negative exponential distribution. In addition, for the unslotted scheme, data packets can be sent immediately if terminals are allowed to and do not have to wait to the start of a slots as in the slotted scheme. The delay characteristics of both the slotted and unslotted schemes are depicted in Figure 14.

V. CONCLUSIONS

The protocol performance is measured in terms of throughput S and delay D in relationship with the arrival rate at terminals P_{Arr} . The performance of the unslotted non-persistent hybrid CDMA/ISMA protocol has been investigated by computer simulation. Two decisive parameters which influences the protocol performance have been analyzed: the inhibit delay fraction and the average time a packet has to wait to be retransmitted. The increase of these parameters have a negative impact on both throughput and delay for high arrival rates.

The performance comparison between the slotted p-persistent and the unslotted non-persistent hybrid CDMA/ISMA protocol is also considered. For high traffic, the performance of the unslotted protocol is significantly better than that of the slotted protocol. This is caused by differences in the retransmission algorithms of both protocols. The unslotted protocol also shows superiority compared to the slotted protocol in terms of packets delay. This is because for the unslotted scheme, data packets can be sent immediately if terminals are allowed to and do not have to wait for the start of a slot as in the slotted scheme.

Computer simulations show that the unslotted scheme is the scheme of choice in the hybrid CDMA/ISMA protocol. The unslotted hybrid protocol will be investigated using a Markov model in future work.

REFERENCES

- R. Prasad,
 "Performance Analysis of Mobile Packet Radio Networks in Real Channels with Inhibit-Sense Multiple Access",
 IEE Proc. I, Vol. 138, pp. 458-464, October 1991.
- I. Widipangestu, A.J. 't Jong and R. Prasad,
 "Capture Probability and Throughput Analysis of Slotted ALOHA and Unslotted np-ISMA in a Rician/Rayleigh Environment",
 IEEE Transactions on Vehicular Technology, Vol. 43, No. 3, pp. 457-465, August 1994.
- K.J. Zdunek, D.R. Ucci and J.L. Locicero,
 "Throughput of Nonpersistent Inhibit Sense Multiple Access with Capture",
 Electronics Letters, Vol. 25, No. 1, pp. 30-31, 5th January 1989.
- [4] R. Prasad, and C.Y. Liu,
 "Throughput Analysis of Some Mobile Packet Radio Protocols in Rician Fading Channels", IEE Proceedings-I, Vol. 139, No. 3, pp. 297-302, June 1992.
- [5] M.G. Jansen, and R. Prasad,
 "Capacity, Throughput, and Delay Analysis of a Cellular DS CDMA System With Imperfect Power Control and Imperfect Sectorization",
 IEEE Transactions on Vehicular Technology, Vol. 44, No. 1, pp. 67-75, February 1995.
- [6] R.A. Scholtz,
 "The Origins of Spread-Spectrum Communications",
 IEEE Transactions on Communications, Vol. COM-30, No. 5, pp. 822-854, May 1982.
- [7] W.C.Y. Lee,
 "Overview of Cellular CDMA",
 IEEE Transactions on Vehicular Technology, Vol. 40, No. 2, pp. 291-302, May 1991.

"Performance Evaluation of Phase Coded Spread Spectrum Multiple Access Communication -Part I: System Analysis", IEEE Transactions on Communications, Vol. COM-25, No. 8, pp. 795-799, August 1977.

- [9] R.L. Pickholtz, D.L. Schilling and L.B. Milstein
 "Theory of Spread Spectrum Communications A Tutorial",
 IEEE Transactions on Communications, Vol. COM-30, No. 5, pp. 855-884, May 1982.
- K.S. Gilhousen, I.M. Jacobs, R. Padovani, A.J. Viterbi, L.A. Weaver, Jr. and C.E.Wheatley III,
 "On the Capacity of a Cellular CDMA System",
 IEEE Transactions on Vehicular Technology, Vol. 40, No. 2, pp. 303-312, May 1991.
- [11] H. van Roosmalen, J. Nijhof and R. Prasad,
 "Performance Analysis of a Hybrid CDMA/ISMA Protocol for Indoor Wireless Computer Communications",
 IEEE Journal on Selected Areas in Communications, Vol. 12, No. 5, pp. 909-916, June 1994.
- [12] C.A.F.J. Wijffels, H.S. Misser, and R. Prasad,
 "A Micro-Cellular CDMA System over Slow and Fast Rician Fading Channels with Forward Error Correcting Coding and Diversity",
 IEEE Transactions on Vehicular Technology, Vol. 42, No. 4, pp.570-580, November 1993.
- [13] A.A.M. Saleh and R.A. Valenzuela,
 "A Statistical Model for Indoor Multipath Propagation",
 IEEE Journal on Selected Areas on Communications, Vol. 5, pp. 138-146, February 1987.
- [14] A.B. Carlson,
 <u>Communication Systems</u>,
 McGraw-Hill International Editions, 1986
- [15] M. Kavehrad and B. Ramamurthi,
 "Direct -Sequence Spread Spectrum with DPSK Modulation and Diversity for Indoor Wireless Communications",

IEEE Transactions on Communications, Vol. COM-35, No. 2, pp. 224-236, February 1987.

^[8] M.B. Pursley,

[16] F.C. Schoute,

Prestatie-Analyse van Telecommunicatiesystemen,

Kluwer Technische Boeken B.V., 1989

List of Figure Captions

Figure 1:	Network structure
Figure 2:	Slotted p-persistent ISMA scheme
Figure 3:	Unslotted non-persistent ISMA scheme
Figure 4:	Inhibit delay fraction
Figure 5:	16-terminal network configuration
Figure 6:	(a) Interval-oriented simulation; (b) Event-oriented simulation
Figure 7:	PSD of main program
Figure 8:	Example of an event list
Figure 9:	Throughput S versus arrival rate P_{Arr} for different values of the inhibit delay fraction <i>d</i> parameter for the unslotted hybrid CDMA/ISMA protocol; Code length = 31; SNR = 20 dB; Number of groups = 8; Group size = 4.
Figure 10:	Delay D versus arrival rate P_{Arr} for different values of the inhibit delay fraction <i>d</i> parameter for the unslotted hybrid CDMA/ISMA protocol; Code length = 31; SNR = 20 dB; Number of groups = 8; Group size = 4.
Figure 11:	Throughput S versus arrival rate P_{Arr} for different values of λ for the unslotted hybrid CDMA/ISMA protocol; Code length = 31; SNR = 20 dB; af : 0.1, 0.5, 1, 2, 5, 10 for λ .
Figure 12:	Delay D against arrival rate P_{Arr} for different values of λ for the unslotted hybrid CDMA/ISMA protocol; Code length = 31; SNR = 20 dB; Number of groups = 8; Group size = 4; af : 0.1, 0.5, 1, 2, 5, 10 for λ .
Figure 13:	Throughput S versus arrival rate P_{Arr} for comparison between slotted and unslotted hybrid CDMA/ISMA protocol; Code length = 31; λ =1; d =0.01; Number of users = 32. The number of codes used vary from 1 to 32. Slotted(1) means the performance curve with 1 code for the slotted scheme, etc.
Figure 14:	Delay D versus arrival rate P_{Arr} for comparison between slotted and unslotted hybrid CDMA/ISMA protocol; Code length = 31, λ =1 and <i>d</i> =0.01; Number of users = 32; The number of codes used varies from 1 to 32. Slotted(1) means the performance curve with 1 code, etc.

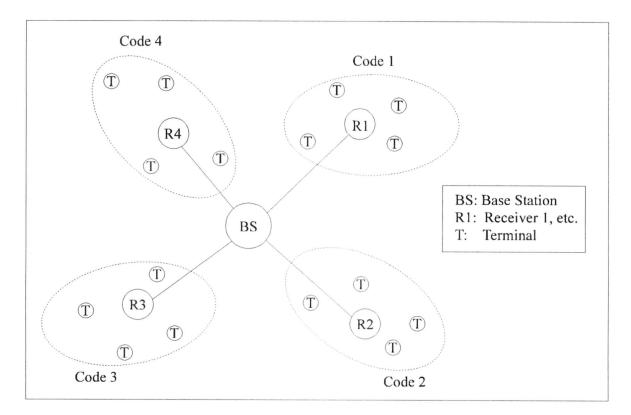


Figure 1: Network structure

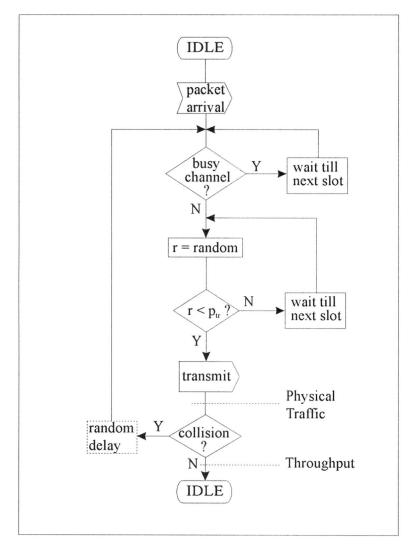


Figure 2: Slotted p-persistent ISMA scheme

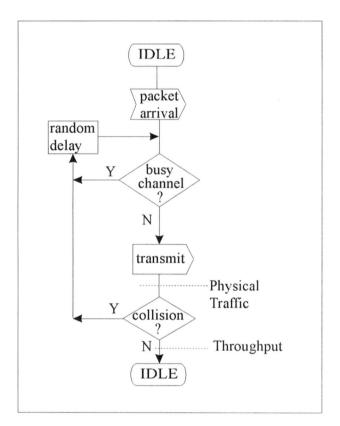


Figure 3: Unslotted non-persistent ISMA scheme

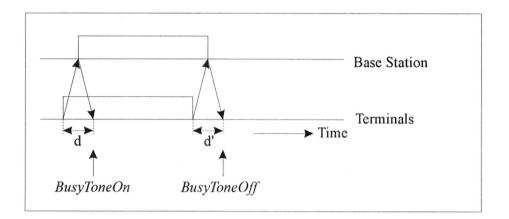


Figure 4: Inhibit delay fraction

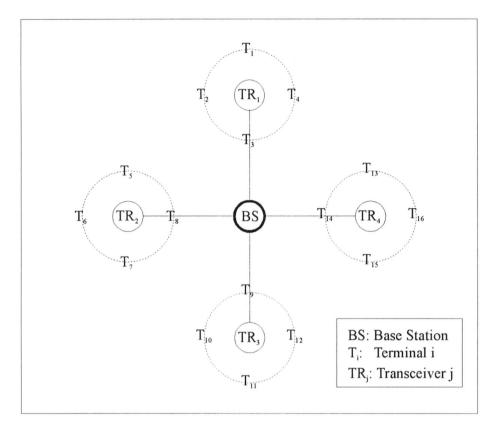


Figure 5: 16-terminal network configuration

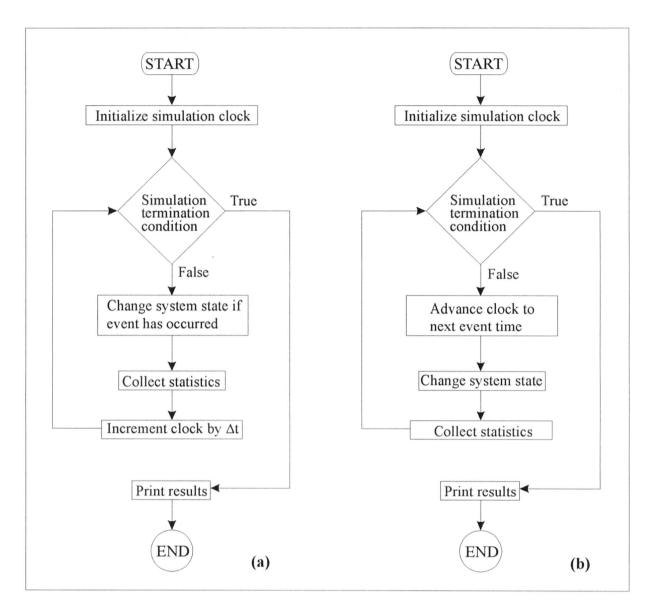


Figure 6: (a) Interval-oriented simulation; (b) Event-oriented simulation

Rese	Reset Counters						
While <i>Clock < EndTime</i> Do							
	Advance Clock to next event time						
	Process Event						
	Update Events List						
	Collect Statistics						

Figure 7: PSD of main program

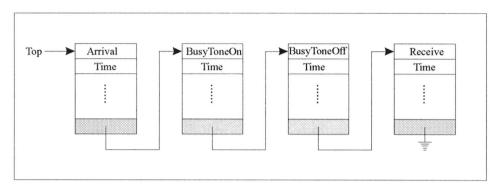


Figure 8:

Example of an event list

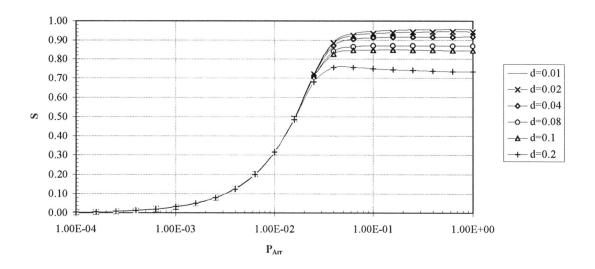


Figure 9: Throughput S versus arrival rate P_{Arr} for different values of the inhibit delay fraction *d* parameter for the unslotted hybrid CDMA/ISMA protocol; Code length = 31; SNR = 20 dB; Number of groups = 8; Group size = 4.

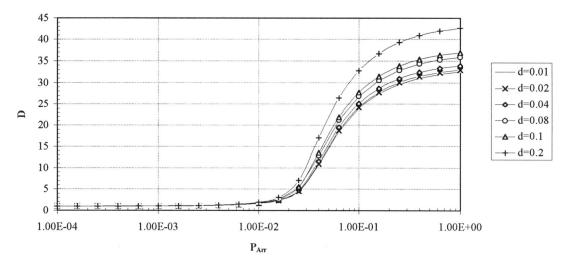


Figure 10: Delay D versus arrival rate P_{Arr} for different values of the inhibit delay fraction *d* parameter for the unslotted hybrid CDMA/ISMA protocol; Code length = 31; SNR = 20 dB; Number of groups = 8; Group size = 4.

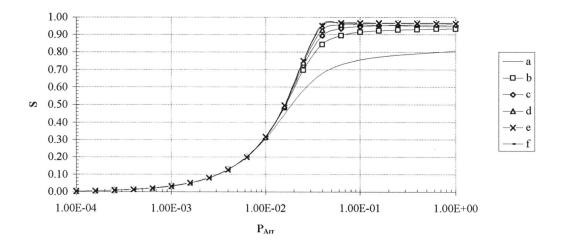


Figure 11: Throughput S versus arrival rate P_{Arr} for different values of λ for the unslotted hybrid CDMA/ISMA protocol; Code length = 31; SNR = 20 dB; a..f : 0.1, 0.5, 1, 2, 5, 10 for λ .

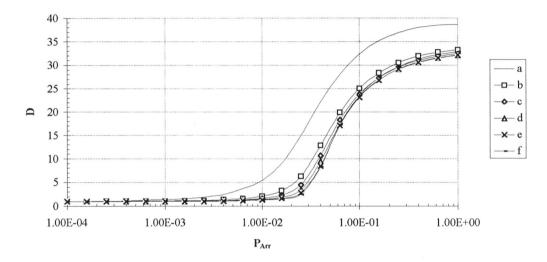


Figure 12: Delay D against arrival rate P_{Arr} for different values of λ for the unslotted hybrid CDMA/ISMA protocol; Code length = 31; SNR = 20 dB; Number of groups = 8; Group size = 4; a..f: 0.1, 0.5, 1, 2, 5, 10 for λ .

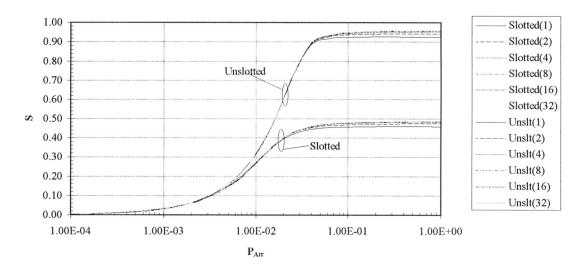


Figure 13: Throughput S versus arrival rate P_{Arr} for comparison between slotted and unslotted hybrid CDMA/ISMA protocol; Code length = 31; λ =1; d=0.01; Number of users = 32. The number of codes used vary from 1 to 32. Slotted(1) means the performance curve with 1 code for the slotted scheme, etc.

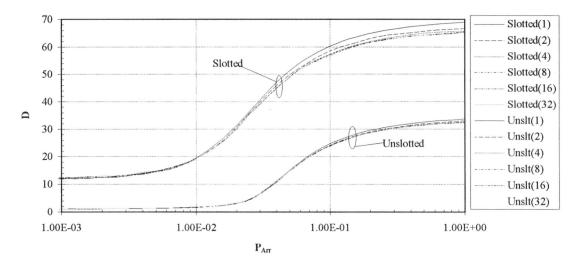


Figure 14: Delay D versus arrival rate P_{Arr} for comparison between slotted and unslotted hybrid CDMA/ISMA protocol; Code length = 31, λ =1 and *d*=0.01; Number of users = 32; The number of codes used varies from 1 to 32. Slotted(1) means the performance curve with 1 code, etc.

APPENDIX D:

Listings of Computer Programs

Three computer programs are included in this Appendix and are listed in the following order:

1. SimSlotted

This is a computer <u>simulation</u> program which simulates the behaviour of the <u>slotted</u> hybrid CDMA/ISMA protocol.

2. SimUnslotted

This is a computer <u>simulation</u> program which simulates the behaviour of the <u>unslotted</u> hybrid CDMA/ISMA protocol.

3. CalcUnslotted

This is a computer program which <u>calculates</u> the performance of the <u>unslotted</u> hybrid CDMA/ISMA protocol.

Page 1, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes.

1 {*****	*****	******	*****	******	****	***}	56 T	ype 1	TermState =	(IDLE, BLOCKED	, TRANSMIT, INTER	RFERE);	
2 {* Sim	Slotted is a p	rogram desig	ned wit	h the go	al to simulate the	*}	57	C	DnOff =	= (ON, OFF);			
3 (* 'SL	otted Hybrid C	DMA/ISMA pro	tocol'.	This pr	ogram calculates the	*}	58	M	Matrix =	Array[1MaxTe	rm, 1MaxTerm]	Of Extended;	
					ls by simulation.	*}	59	F	PosArr =	Array[1MaxTe	rm]	Of Extended;	
5 (*		,				*}	60	F	RArr =	Array[1MaxTe	rm, 1MaxL]	Of Extended;	
	t of the const	ants variab	les fi	Inctions	and procedures used here	*}	61			Array[1MaxTe		Of String;	
	ginates from H					*)	62		•	Array[1MaxTe		Of TermState;	
	ginates non n		materr.			*)	63			All ay Littlinaxie			
8 {*	terminals and	alustanad a	nound (lictnibut	ed receivers at a fixed	*}							3
						-							,
					tered around the base	*}	65	ŀ	p2CRec =	= ^CRec;			
					m the terminals are	*}	66						
					nt to the base station by	*}	67	(CRec =	Record			_
	e or lines. Th	ese lines ca	in be se	een as pr	imitive concentrators	*}	68				maxcodelengthma	axcodelength] Of ShortInt;	i
14 {*						*}	69			End;			
15 {* Huy	Linh Anh Le,	August 10 19	95			*}	70						
16 {*****	*******	*******	******	*******	*****	***}	71						
	SimSlotted(In						72	(CArr =	Array [1MaxT	erm, 1MaxTerm]	Of p2CRec;	
18	•••••••••••••••••••••••••••••••••••••••	,,					73						
19 Uses	Dos, Crt, Ult	rax.					74 {						}
20	000, 010, 010	i un					75	-	QueueEntry				
21 Const	MaxTerm	=	32. 1	Maximum	number of terminals allowe	۲ b	76			TerminalNo	:Integer;		
22	PacketLength				of bits per packet	ر `` }	77			Time	:Extended;		
	PacketLength	-	04, 1	Number	n bits per packet	,	78			END;	i Excended,		
23	5.4	_	1. (Trenewit	ten neven in Untto	}	79			LND,			
24	Pt	=			ter power in Watts	-	80						
25	LossExp	=			ion parameter for the				• • • • •	DECODD			
26			-		field model	}	81		QueueType =				
27	с		•		light in m/s	}	82			Count,			
28	f				ter center frequency	>	83			Front,			
29	lambda	=	c/f; {	Signal w	avelength	}	84			Rear: 0Max			
30							85				[1Maxterm] OF	QueueEntry;	
31	twoPi	= 2*	· Pi; {	Two time	s Pi	}	86			END;			
32	Pi2	= Pi*	Pi; {	Pi squar	ed	}	87						
33							88 (}
34	Ptr	= 0.99;			{ Transmission probabilit	y }	89	E	EventType =	= (Arrival, ReAr	rival, BusyToneOr	n, BusyToneOff, Receive);	
35	Alpha	= 0.1;			{ The number of Rearrivals	per	90						
36					PacketDuration	` >	91	1	ListPointer	<pre>r = `ListNode;</pre>			
37	bitrate	= 256*1024;			{ Bits transmitted per sec		92		ListNode	= RECORD			
38	PacketDuratio				{Time needed to send a Pac		93			Event	:EventType		
39	Packetbalatie						94			Time	:Extended;		
40	maxcodelength	- 127.			{ Maximum code size	}	95			TerminalNo	· · · · · · · · · · · · · · · · · · ·		
	-				{ Length of code	Ś	96			NextNode	:ListPointe	er	
41	codelength	= 31;				ź	97				a lot office		
42	Tb	= 1/bitrate			{ Bit duration	-	98			END;			
43	Tc		engtn*b		{ Chip duration	}				- 050000			
44	Tm	= 100E-9;			{ Maximum delay spread	}	99	1	LinkedList				
45	MaxL	= 8;			{ Maximum number of paths	}	100			Head	:ListPointe	er	
46	L	= Trunc(Tm/	(Tc)+1;		{ Number of paths	}	101			END;			
47							102						
48	WarmTime	= 2'	1000; {	Warm up	time in sec.	}							}
49	Interval	=	500; {	Time bet	ween screen updates	}	104 V	/ar (Count,				
50	EndTime	= 125	;000; {	Simulati	on time in sec.	>	105	1	Front,				
51	respondit 61 st						106	1	Rear	: Integer;	{ Counters for (QueueType	}
52	NumSim	=	20: {	Number o	of activity levels simulate	d }	107						
53	LRangeMin	=			activity from 10 ⁻ -LRangeM		108		CCat	: CArr;	{ Catalog of cod	de cross correlations	}
54	LRangeMax	=)^-LRangeMax)	109			•	50 - 1999 (1999) - 1993 - 1993 - 1993 - 1993 - 1993 - 1993 - 1993 - 1993 - 1993 - 1993 - 1993 - 1993 - 1993 - 1		
55			- / (2	110	1	PArrH,		{ Packet Arriva	l Probability Slotted	>
												en leveler ensembler and mental states of the second states of the	

Page 2, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes.

			C. D. J. M. Andred Brokehilling Harland		1//
111	PArrL,		{ Packet Arrival Probability Unslotted	}	166 If b<>0
112	Rreceive,		{ Receiver radius around base station { Terminal radius around receivers	}	167 Then 168 power:= 0 { power(0,b) = 0 }
113 114	Rterm,	•Extended•	{ Signal to white Noise Ratio at receivers	-	168 power:= 0 { power(0,b) = 0 } 169 Else
114	SNR	:Extended;	(Signat to write Noise Ratio at receiver	,	170 power := 1; { power(0,0) = 1 }
116	Nt,		{ Number of terminals	}	171 End; {Power}
117	GroupSize,		{ Number of terminals per group	Ś	172
118	NumberOfLines	:Integer;	{ Number of groups in the system. This is	-	173 {************************************
119	Number of Effics	. Integer,	{ also the number of codes used.	}	174 { ShowState shows the terminal states on screen }
120				,	175 { }
121	DelayArray	:PosArr;	{ Delay administration for each terminal	3	176 { Depends on: config, numberOfLines, terminal }
122	DelayCount		{ Total delay time in Tpd	Ś	177 { Calls : none }
123	Detayoodite	i Externatory	c rocat actay child in the		178 { Modifies : none }
124	SuccessCount,		<pre>{ Number of successful packets</pre>	}	179 {************************************
125	ArrivalCount,		{ Number of accepted packets	5	180 Procedure ShowState;
126	GCount	:LongInt;	{ Number of collisions occured	5	181 Var
127			•	-	182 i : Integer;
128	S	:Matrix;	{ Power loss ratio matrix	}	183 c : Char;
129		•			184 Begin
130	OldClock,		{ Clock maintained for screen updates	}	185 GotoXY(12, 22);
131	Clock	:Extended;	{ Clock maintained by main program	}	186 Write(' 1 2 3 ');
132					187 GotoXY(12, 23);
133	EventList	:LinkedList	;{ Nodes of events linked together	}	188 Write('1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 3 2 4 5 6 7 8 9 0 1 2');
134	ReceiveQueue	:QueueType;	{ Queue for received packets	}	189 For i := 1 To Nt Do
135					190 Begin
136	BusyTone	:OnOff;	{ Indicates wheather the channel is free	}	191 GotoXY(10+2*i,24);
137			{ or not.	}	192 Case Terminal[i] Of
138	Terminal	:TermArr;	<pre>{ Terminal state information</pre>	}	193 IDLE : Write('i');
139					194 BLOCKED : Write('b');
140	i	:Integer;	{Loopvariable for terminal activity level	ls}	195 TRANSMIT : Write('t');
141		-			196 INTERFERE : Write('f');
142	ResultFile	:Text;	<pre>{ Result file with original values</pre>	}	197 End; {Case}
143	* L		f Theoryphonete forestion of the time		198 End; {For i := 1 TO Nt}
144	Throughput,		{ Throughput: fraction of the time	2	199 End;{ShowState} 200
145 146			<pre>carrying a successful transmission { Mean Delay normalized in slots</pre>	}	200 201 {************************************
140	D, G	.Extended.	{ Mean Delay: time between the arrival of	-	201 { R calculates a partial cross correlation }
147	a	.Extended,	the first bit till the receive of the		203 { calculates a partial closs correlation }
148			last bit in sec.	3	204 { Depends on: config, CCat, Tc }
150 7**	*****	*****	***************************************		205 { Calls : none }
	Power calculates a			Ś	206 { Modifies : none }
152 (TORCE CULCULULUS U			ŝ	207 {************************************
	Depends on: none			5	208 Function R(i,j:Integer; tau:Extended):Extended;
154 (Ś	209 Var
	Modifies : none			5	210 l, { Chip number belonging to tau }
156 (>	211 Cl, { Discrete correlation function for chip l }
	Pre : a >= 0 And b	>=0		5	212 Cl1 :Integer; { Discrete correlation function for chip l+1 }
158 (Post: Power = a^b			>	213 Begin
159 {**	*****	*****	***********	**}	214 l := Trunc(tau/Tc);
160 Fur	nction Power(a,b:Ext	ended):Exten	nded;		<pre>215 Cl1 := CCat[i,j]^.cc[l+1-codelength];</pre>
161 Beg	gin				216 If l >= 1
162	If a<>O				217 Then
163	Then				218 Begin
164	power := Exp(b*Ln(a))			<pre>219 Cl := CCat[i,j]^.cc[l-codelength];</pre>
165	Else				220 R := Cl*Tc+(Cl1-Cl)*(tau-l*Tc);

3, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes. Page

276

277

278 279

280

281

282

283

284

285

286

287

288

289

290

291

292

293 294

295

296

297

298

299 300

301

302

304

307 {

314

315

317

318

319

321

221	End		
222	Else		
223	R := tau*Cl1;		
	End; {R}		
225	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	*****	1
			}
227 228	And the second		5
229			5
230			5
231	{ Modifies : none		3
232	(**************************************	****	}
233	Function Rhat(i,j:Integer; tau:Ext	ended):Extended;	
234			
235			}
236			}
237		Discrete correlation function for chip l+1	}
	Begin		
239 240	<pre>l := Trunc(tau/Tc); cl := CCat[i i]^ cc[l];</pre>		
240	Cl := CCat[i,j] [*] .cc[l]; If l < codelength-1		
242	Then		
243	Begin		
244	Cl1 := CCat[i,j]^.cc[l-	+1];	
245	Rhat := Cl*Tc+(Cl1-Cl)*	(tau-l*Tc);	
246	End		
247	Else		
248	Rhat := ((l+1)*Tc-tau)*C	.;	
249	End;{Rhat}		
251	<i>{</i> ************************************	******	}
252			}
253	{ functions for a set of (Gold)	codes passed by gc. The catalog consists of	}
254	{ a two dimensional array of point	nters. These pointers refer to	}
255		ing the cross correlation information. This	
256			}
257			} }
258 259			, }
259			; ;
261			ŝ
262	{*************************************	***************	
	Procedure InitCCat(Var CCat:CArr;		
	Var		
265	i,	-	}
266	j,	• • • • • • • • • • • • • • • • • • • •	}
267	k,	-	}
268	ι,		}
269	t :Integer;	<pre>{ Correlation result for shift l</pre>	3
270	Begin Write('Calculating ', Nt, '	by '. Nt.' correlations'):	
272	For $i := 1$ To Nt Do		}
273	For j := 1 To Nt Do		
274	Begin		
275	GotoXY(70,1);	{ Show which codes are being done	3

Write(i:4,j:4); { Allocate space for new array > CCat[i,j] := New(p2CRec); { Calculate cross-correlation } For l := -codelength To codelength Do Begin { Reset correlation value t := 0: } If l < 0{ Different formulas for pos and } Then { neg phase shifts } For k := 0 To codelength-1+l Do If gc[i][1+((k-l) Mod codelength)] = gc[j][k+1] Then Inc(t) Else Dec(t) Else For k := 0 To codelength-l-1 Do If gc[i][1+(k Mod codelength)] = gc[j][1+((k+l) Mod codelength)] Then Inc(t) Else Dec(t); CCat[i,j]^.cc[l] := t; End; End; 303 End;{InitCCat} 306 { DisposeCCat frees memory space taken up by the correlation catalog 3 3 308 { Depends on: config 3 309 { Calls : none 3 310 { Modifies : none 3 312 Procedure DisposeCCat(CCat:CArr); 313 Var i, { Loop variable } { Loop variable } j :Integer; 316 Begin { Loop over all catalog entries } For i := 1 To Nt Do For i := 1 To Nt Do { Free memory of element i, j } Dispose(CCat[i,j]); 320 End; {DisposeCCat} 323 { InitTerm initializes all terminals as free 324 { 3 325 { Depends on: config 3 326 { Calls : none 3 327 { Modifies : pfn } 329 Procedure InitTerm(Var Terminal:TermArr); 330 Var

Page 4, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes.

331 i :Integer; { Loop variable } 332 Begin 387 { InitPathLoss sets up a table of transmitter to receiver path gains from 333 388 { all transmitters to all receivers. All gains are between 0 and 1. When 3 For i:= 1 To Nt Do 389 { groups are larger than one terminal duplicate entries will exist in the terminal[i] := IDLE; 334 390 { table because there are less receivers than terminals then. 335 End;{InitTerm} 391 { 336 392 { Depends on: config, lambda 3 393 { Calls : Power 3 338 PROCEDURE InitDelayArray(VAR DelayArray:PosArr); 394 { Modifies. : none 339 VAR 340 i : Integer; { Loop variable } 396 Procedure InitPathGain(Var S:Matrix; groupSize, numberOfLines :Integer; 341 BEGIN 342 FOR i := 1 TO MaxTerm DO 397 Rreceive, Rterm, lossexp:Extended); 398 Var 343 DelayArray[i] := 0; 399 { Angle of receiver relative to x-axis 344 END;{InitDelayArray} gamma. { Angle of transmitter relative to x-axis } 400 345 theta :Extended: { Transmitter coordinates 401 3 xt, yt, 347 PROCEDURE InitBusyTone(VAR BusyTone:OnOff); 402 xr, yr :PosArr; { Receiver coordinates 3 i, 403 { Loop variable > 348 BEGIN 349 BusyTone := Off; 404 j, { Loop variable 3 405 k { Index variable 350 END;{InitBusyTone} } :Integer: 406 Begin 351 407 For i := 1 To numberOfLines Do{ Loop over all receivers > 408 For j := 1 To groupSize Do{ Loop over all group terminals 353 { Initcode reads Gold codes from a file and has the cross correlation 3 409 354 { catalog setup Begin {Determine index current term. } 355 { 410 k := (i-1) * groupSize + j; gamma := 2*Pi * (i-1)/numberOfLines; {Calc. receiver angle 411 356 { Depends on: config 3 : initCCat 412 xr[k] := Rreceive * Cos(gamma); {Calc. receiver x-coord. } 357 { Calls 413 {Calc. receiver y-coord. } 358 { Modifies : pfn yr[k] := Rreceive * Sin(gamma); 414 theta := 2*Pi*(j-1)/groupSize+gamma+Pi;{Calc. transmitter angle } 415 xt[k] := xr[k] + Rterm * Cos(theta); {Calc. transmitter x-coord.} 360 Procedure InitCodes(Var CCat:CArr: groupSize, numCodes:Integer); yt[k] := yr[k] + Rterm * Sin(theta); {Calc. transmitter y-coord.} 416 361 Var 417 362 i, { Loop variable End; { Loop variable 418 For i := 1 To Nt Do {Double loop over all terminals} 363 i 3 :Integer; For i := 1 To Nt Do 364 :text; { File containing the Gold codes 419 f { Length of transmission code 420 Begin 365 codeCount, 3 { Calculate path gains according to far field model } 421 366 cfile :String; { Codefile name 3 422 S[i,j] := Power(lambda/(4*Pi*Sqrt(Sqr(xt[i]-xr[j])) { Gold codes as read from file 367 gc :StringArr; } 423 368 Begin +Sqr(yt[i]-yr[j]))),lossexp); 369 424 End ClrScr; 370 Str(codelength, codeCount); { Assign appropriate codefile name } 425 End;{InitPathGain} 371 cfile := 'codes\'+codeCount+'chip.dat'; 426 372 Assign(f, cfile); { Open code file } 373 Reset(f); 428 { Rayleigh randomly draws a value according to the rayleigh distribution 374 For i := 1 To Nt Do { Read the codes from file > 429 { 430 { Depends on: none 375 } ReadLn(f,gc[i]); 431 { Calls : duni 3 376 Close(f): { Reassign codes according to reuse scheme } 377 If numCodes <> Nt 432 { Modifies : pfn 378 Then { e.g. 12345678 -> 11223344 for 2 term/grp } 434 Function Rayleigh(ave:Extended):Extended; 379 For i := numCodes DownTo 1 Do 435 Var 380 For j := groupSize DownTo 1 Do 381 436 gc[(i-1)*groupSize+j] := gc[i]; r, 382 437 u, initCCat(CCat, gc); 438 383 { Initialize catalog of cross correlations } ٧, 384 End;{InitCodes} 439 x, 440 :Extended; 385 Y

Page 5, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes.

441	Begin	496	n := 0;
442	Repeat	497	Repeat
443	u := 2*duni-1;	498	n := n + 1;
444	v := 2*duni-1;	499	ac := d1+2*n*ab/(a*b)+aa;
445	r := sqr(u)+sqr(v);	500	bc := 1+2*n*bb/(a*b)+ba;
	Until (r<1.0) and (r>0.0);	501	d1 := dd*d1;
446		502	aa := ab;
447	r := Sqrt(-2*Ln(r)/r);	503	ab := ac;
448	$x := u^* r;$	503	
449	y := v*r;		ba := bb;
450	Rayleigh:=Sqrt(Sqr(x)+Sqr(y))*ave*Sqrt(2/Pi);	505	bb := bc;
451	End;	506	
452		507	If a < b
453	{*************************************	508	Then
454	{ MarcumQ calculates the Marcum Q-function }	509	MarcumQ := (ac/(2*bc))*Exp(-Sqr(a-b)/2)
455	{	510	Else
456	{ Depends on: none }	511	MarcumQ := 1-(ac/(2*bc))*Exp(-Sqr(a-b)/2);
457		512	End;
458	{ Modifies : pfn }	513	End;
459	(*************************************	514	
	Function MarcumQ(a,b:Extended):Extended;	515	{*****
461			{ NegExpIO approximates the modified Bessel function of the first kind }
			{ using the recurrence relation I (z) = I (z) + $2n/z$ I (z) }
462	aa,	518	
463	ab,	519	
464	ac,		{ and then multiplies the result with Exp(-(a ² +b ²)/2). }
465	bc,		
466	bb,	521	
467	ba,		t bepende ent here
468	dd,		
469	d1 :Extended;	524	{ Modifies : pfn }
470	n :Integer;		<pre>{************************************</pre>
471	Begin		Function NegExpI0(a,b:Extended):Extended;
472	If a = 0		Var
473	Then	528	
474	If b < 30	529	
475	Then	530	Ja, { I0,n-1 }
476	MarcumQ := Exp(-Sqr(b)/2)	531	Jb, { I0,n }
477	Else	532	Jc, { I0,n+1 }
478	MarcumQ := 0	533	tmp,
479	Else	534	
480	Begin		Begin
481	If a < b	536	
	Then	537	
482		538	If $x = 0$
483	Begin	539	
484	ab := 1;	540	
485	dd := a/b;		
486	End	541	
487	Else	542	
488	Begin	543	
489	ab := 0;	544	
490	dd := b/a;	545	
491	End;	546	
492	aa := 0 ;	547	
493	bb := 0.5;	548	
494	ba := 0 ;	549	
495	d1 := dd;	550	For n := 20 DownTo 1 Do

Page 6, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes.

```
551
           Begin
552
               Ja := (2*n/x)*Jb+Jc;
553
               Jc := Jb:
554
               Jb := Ja:
555
               t := t + Ja:
556
           End:
557
           NegExpI0 := Exp(x-(Sqr(a)+Sqr(b))/2)*Ja/(2*t-Ja);
558
       End:
559 End;
560
562 { CalcMu1 calculates the Mu1 parameter for the bit error formula in
563 { Rayleigh fading environments
                                                                 3
564 {
                                                                 3
565 { Depends on: L
                                                                 }
                                                                 3
566 { Calls
             : R, Rhat
567 { Modifies : none
                                                                 3
569 Function CalcMu1(u, h:Integer; tau:RArr; mu2:Extended):Extended;
570 Var
     tmp :Extended;
                                               { Intermediate result }
571
                                               { Loop variable
572
                                                                 3
     i :Integer;
573 Begin
574
       tmp := 0;
575
       For i := 1 To L Do
576
          If i \ll h
577
          Then
578
             tmp := tmp + R(u,u,tau[u,i])*Rhat(u,u,tau[u,i])*s[u,u];
579
580
       CalcMu1 := 4*2*Pt*tmp + mu2
581 End;
582
584 { CalcMu2 calculates the Mu2 parameter for the bit error formula in
585 { Rayleigh fading environments
                                                                 3
586 {
                                                                 3
587 { Depends on: config, L, Pt, terminal
                                                                 }
                                                                 3
588 { Calls
            : R, Rhat
589 { Modifies : none
591 Function CalcMu2(u, h:Integer; tau:RArr; sigma_n2:Extended):Extended;
592 Var
593
     tmp :Extended;
                                               { Intermediate result }
594
                                               { Loop variable
                                                                 }
     i,
595
                                               { Loop variable
                                                                 3
     i
        :Integer;
596 Begin
597
       tmp := 0;
598
       For i := 1 To Nt Do
599
         If (terminal[i] = INTERFERE) And (i <> u)
600
         Then
601
            For j := 1 To L Do
               tmp := tmp + (Sqr(R(u,i,tau[i,j]))
602
603
                        + Sqr(Rhat(u,i,tau[i,j])))*s[i,u];
604
605
       For j := 1 To L Do
```

```
606
         If i <> h
607
         Then
608
            tmp := tmp + (Sqr(R(u,u,tau[u,j]))
609
                     + Sqr(Rhat(u,u,tau[u,j])))*s[u,u];
610
611
       CalcMu2 := 2*(2*Pt*tmp+sigma n2):
612 End:
613
615 { CalcMu12 calculates the Mu12 parameter for the bit error formula in
616 { Rayleigh fading environments
617 {
618 { Depends on: config, L, Pt, terminal
619 { Calls
           : R, Rhat
620 { Modifies : none
                                                                3
622 Function CalcMu12(u,h:Integer;tau:RArr):Extended;
623 Var
624
     tmp :Extended;
                                               { Intermediate result }
                                               { Loop variable
625
     i,
                                                                3
626
                                               { Loop variable
                                                                }
     ī
        :Integer;
627 Begin
628
       tmp := 0;
629
       For i := 1 To Nt Do
630
         If (terminal[i] = INTERFERE) And (i<>u)
631
         Then
632
            For i := 1 To L Do
633
               tmp := tmp + R(u,i,tau[i,j])*Rhat(u,i,tau[i,j])*s[i,u];
634
635
       For i := 1 To L Do
636
         If i <> h
637
         Then
638
            tmp := tmp + (Sqr(Rhat(u,u,tau[u,j]))
639
                     + R(u,u,tau[u,j])*Rhat(u,u,tau[u,j]))*S[u,u];
640
641
       CalcMu12 := 2*2*Pt*tmp;
642 End:
643
645 { CalculatePbe computes the bit error probability for a terminal, given
646 { which other terminals transmit
647 {
648 { Depends on: config, L, Pt, S, Tb, terminal
649 { Calls
            : duni, IO, MarcumQ, Rayleigh
650 { Modifies : none
652 Function CalculatePbe(u:Integer):Extended;
653 Var
654
     mu1,
655
     mu2,
656
     mu12,
657
     a,
658
     b,
659
     bmax.
                              { Maximum path gain
                                                                }
660
     tmp,
```

7, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes. Page

>

}

}

}

}

```
661
       m,
662
       mq,
663
       mb,
       sigma n2:Extended;
664
                                     { Noise power
                                     { Rayleigh distributed path gains
665
       beta,
666
       tau
               :RArr;
                                     { Uniformly [0,Tb] distributed path delays }
667
                                     { Number of path with largest gain
      h,
668
       i,
                                     { Loop variable
669
                                     { Loop variable
       i
               :Integer;
670 Begin
671
         { Assign random delays and path gains to transmitting terminals }
672
         For i := 1 To Nt Do
673
            If terminal[i] = INTERFERE
674
            Then
675
                For j := 1 To L Do
676
                Begin
677
                     tau[i,j] := Tb*duni;
                     beta[i,j] := Rayleigh(Sqrt(S[i,u]));
678
679
                End
680
            Else
681
                For i := 1 To L Do
682
                Begin
683
                     tau[i,j] := 0;
684
                     beta[i,j] := 0;
685
                End;
686
687
         { Determine what the largest gain is }
688
         bmax := 0;
689
         h := 1;
690
         For i := 1 To L Do
691
            If beta[u,i]>bmax
692
            Then
693
            Begin
694
                 bmax := beta[u,i];
695
                 h := i;
696
            End;
697
698
         { Determine noise power relative to largest incoming signal }
         sigma n2 := power(10, -snr/10)*Sqr(bmax)*Pt*Tb*Tb;
699
700
701
         { Calculate bit error probability parameters }
702
         mu2 := CalcMu2(u, h, tau, sigma n2);
703
         mu1 := CalcMu1(u, h, tau, mu2);
704
         mu12 := CalcMu12(u, h ,tau);
705
           := Sqrt(2*Pt)*bmax*Tb;
         m
             := m*Abs(1/Sqrt(mu1)-1/Sqrt(mu2))/Sqrt(2);
706
         а
707
         b
             := m*(1/Sqrt(mu1)+1/Sqrt(mu2))/Sqrt(2);
708
             := MarcumQ(a,b);
         mq
709
         mb
            := NegExpIO(a,b);
710
711
         { Calculate the bit error probability }
712
         tmp := ( (1 + mu12/Sqrt(mu1*mu2))*mb )/2;
713
         CalculatePbe := mg-tmp
714 End;
715
```

716	{**************************************
717	
	{*************************************
719	•
720	
721	{**************************************
722	
723	
724	
	<pre>{************************************</pre>
	PROCEDURE CreateQueue(VAR Q:QueueType);
728	BEGIN
729	WITH Q DO BEGIN
730	Count := 0;
731	Front := 1;
732	Rear := 0
733	END
	END;{CreateQueue}
735	
736	{**************************************
737	PROCEDURE Append (TerminalNo:Integer; Time:Extended; VAR Q:QueueType);
738	
739	X : QueueEntry;
	BEGIN
741 742	WITH Q DO
742	IF Count = MaxTerm THEN WRITELN ('Error: Attempt to append an entry to a full queue')
744	ELSE
745	BEGIN
746	X.TerminalNo := TerminalNo;
747	X.Time := Time;
748	Count := Count + 1;
749	Rear := (Rear MOD MaxTerm) + 1;
750	Entry [Rear] := X;
751	END
	END; {Append}
753	(**************************************
	-
	PROCEDURE Remove (VAR X: QueueEntry; VAR Q:QueueType); BEGIN
757	WITH Q DO
758	IF Count = 0 THEN
759	WRITELN ('Error: Attempt to remove an entry from an empty queue')
760	ELSE
761	BEGIN
762	Count := Count - 1;
763	X := Entry [Front];
764	Front := (Front MOD MaxTerm) + 1
765	END
	END;{Remove}
767	{*****
	FUNCTION QueueSize (VAR Q: QueueType): INTEGER;
	BEGIN

Page 8, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes.

```
QueueSize := Q.Count
771
772 END; {QueueSize}
773
775 FUNCTION QueueEmpty (VAR Q: QueueType): BOOLEAN;
776 BEGIN
777
     QueueEmpty := (Q.Count = 0)
778 END; {QueueEmpty}
779
781 FUNCTION QueueFull (VAR Q: QueueType): BOOLEAN;
782 BEGIN
783
     QueueFull := (Q.Count = MaxTerm)
784 END; {QueueFull}
785
787 PROCEDURE ShowQueue (VAR Q : QueueType);
788 VAR
789
    x, y : INTEGER;
790
    X1 : QueueEntry;
791 BEGIN
792
     {WRITELN('De inhoud van de queue is:');}
793
     x := QueueSize (Q);
794
     WITH Q DO
795
        y := front:
796
        WHILE x > 0 DO
797
        BEGIN
798
           X1 := Q.Entry[y];
           WRITELN('TN: ',X1.TerminalNo,'Time: ',X1.Time);
799
800
           x := x-1;
801
           y := (y MOD MaxTerm) + 1
802
        END;
803
        WRITELN:
804 END; {ShowQueue}
805
806
808 {
                    LINKED-LIST
810
811
813 (* CreateList initializes linked list LL to be empty
                                                  *}
                                                  *3
814 {*
                                                  *}
815 {* Pre: None
                                                  *}
816 {* Post: The list LL has been created and is initialized to be empty
  817
818 PROCEDURE CreateList(VAR LL: LinkedList);
819 BEGIN
820 LL.head := NIL
821 END;{CreateList}
822
824 PROCEDURE InsertEvent(NewNode:ListPointer; T:Extended; VAR LL:LinkedList);
825 VAR
```

826 Current. 827 Trailing : ListPointer; 828 BEGIN 829 Current := LL.head; 830 Trailing := NIL; 831 WHILE (Current[^].Time <= T) AND (Current[^].NextNode <> NIL) DO 832 BEGIN 833 Trailing := Current; 834 Current := Current .NextNode; 835 END; 836 IF (Current[^].time <= T) AND (Current[^].NextNode = NIL) 837 THEN 838 BEGIN 839 Current .NextNode := NewNode; 840 NewNode .. NextNode := NIL; 841 END 842 ELSE IF (Current[^].Time > T) AND (Trailing = NIL) 843 THEN 844 BEGIN 845 NewNode .NextNode := LL.head; 846 := NewNode; LL.head 847 END 848 ELSE 849 BEGIN 850 NewNode .NextNode := Current; 851 Trailing .NextNode := NewNode; 852 END; 853 END;{InsertEvent} 854 856 PROCEDURE CreateEvent(E:EventType; T:Extended; TN:Integer; VAR LL:LinkedList); 857 858 VAR 859 NewNode : ListPointer; 860 BEGIN 861 New(NewNode); 862 NewNode^{*}.Event := E: 863 NewNode[^].Time := T; 864 NewNode^{*}.TerminalNo := TN; 865 IF LL.head = NIL 866 THEN 867 BEGIN 868 NewNode .NextNode := NIL; 869 := NewNode; LL.head 870 END 871 ELSE 872 BEGIN 873 InsertEvent(NewNode, T, LL); 874 END: 875 END;{CreateEvent} 876 878 PROCEDURE ShowList(VAR LL:LinkedList); 879 VAR 880 Current : ListPointer;

Page 9, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes.

881	i : Integer;	936	BEGIN
882	Event : String;	937	CreateEvent(ReArrival, Time + 1, TN, LL);
	BEGIN	938 939	DelayArray[TN] := DelayArray[TN] + 1; Inc(GCount);
884 885	i := 1; Current := LL.head;	940	END
886	ClrScr;	941	ELSE
887	WriteLn('The list consists of: (Press ENTER for the nodes to show up)');	942	BEGIN
888	WriteLn;	943	<pre>Terminal[TN] := TRANSMIT;</pre>
889	IF Current = NIL THEN	944	Append(TN, Time, Q);
890	BEGIN	945	CreateEvent(Receive, Time+ 0.95, TN, LL);
891	WRITELN('EventList is empty! (WriteList)');	946	<pre>DelayArray[TN] := DelayArray[TN] + 1;</pre>
892	END	947	END
893		948	END END (Process Appivel)
894	WHILE (Current <> NIL) DO	949 950	END;{ProcessArrival}
895 896	BEGIN CASE Current [^] .Event OF	951	<pre>{************************************</pre>
897	Arrival : Event := 'Arrival';		PROCEDURE ProcessReArrival(FirstNode:ListPointer; VAR BusyTone:OnOff;
898	ReArrival : Event := 'ReArrival';	953	VAR Terminal:TermArr; VAR LL:LinkedList;
899	BusyToneOn : Event := 'BusyToneOn';	954	VAR Q:QueueType);
900	BusyToneOFF : Event := 'BusyToneOff';	955	
901	Receive : Event := 'Receive';	956	RandomDelay,
902	ELSE WRITELN('ERROR in EventList (ShowList)!!!!');	957	Time,
903	END;	958 959	TInterArr : Extended; TN : Integer;
904 905	WRITELN('Node:',i,' Event:',Event:12,' Time:', Current^.Time:4,' Terminal:',Current^.TerminalNo);		BEGIN
905	i := i+1;	961	Clock := FirstNode^.Time;
907	Current := Current .NextNode;	962	<pre>TN := FirstNode^.TerminalNo;</pre>
908	ReadLn;	963	Time := FirstNode [*] .Time;
909	END;	964	IF DUNI > Ptr
910	WriteLn('End of list');	965	THEN
911	ReadLn;	966	BEGIN
	END;{ShowList}	967 968	CreateEvent (ReArrival, Time + 1, TN, LL); DelayArray[TN] := DelayArray[TN] + 1;
913	(**************************************	969	Inc(GCount);
	PROCEDURE ProcessArrival(FirstNode:ListPointer; VAR BusyTone:OnOff;	970	END
916	VAR Terminal:TermArr; VAR LL:LinkedList;	971	ELSE
917	VAR Q:QueueType);	972	BEGIN
918	VAR	973	Terminal[TN] := TRANSMIT;
919	RandomDelay,	974	Append(TN, Time, Q);
920	Time,	975	CreateEvent(Receive, Time+0.95, TN, LL);
921	TInterArr : Extended;	976 977	DelayArray[TN] := DelayArray[TN] + 1; END
922	TN : Integer;		END;{ProcessReArrival}
923	BEGIN Clock := FirstNode [*] .Time;	979	
925	TN := FirstNode [*] .TerminalNo;		<pre>{************************************</pre>
926	Time := FirstNode [*] .Time;	981	PROCEDURE ProcessReceive(FirstNode:ListPointer; VAR BusyTone:OnOff;
927	TInterArr := Trunc(-(ln(1-DUNI))/PArrH)+1;	982	VAR Terminal:TermArr; VAR LL:LinkedList;
928	CreateEvent(Arrival, Time + TInterArr, TN, LL);	983	VAR Q:QueueType);
929	IF Terminal[TN] = IDLE	984	
930	THEN	985	X2 : QueueEntry;
931	BEGIN	986 987	τΝ,
932 933	Terminal[TN] := BLOCKED; Inc(ArrivalCount);	988	TN2 : Integer;
934	IF (DUNI > Ptr)	989	
935	THEN	990	RandomDelay,

Page 10, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes.

991	Time,	1046	CASE FirstNode^.Event OF
992	Time2,	1047	Arrival : ProcessArrival(FirstNode, BusyTone, Terminal, LL, Q);
993	Pbe,	1048	
994		1049	
	Psp : Extended;	1050	
995	PEOL	1051	END;
	BEGIN	1052	
997	Clock := FirstNode .Time+0.05;		
998	Time := FirstNode^.Time+0.05;		END;{ProcessFirstNode}
999	IF QueueEmpty(Q) = TRUE	1054	· · · · · · · · · · · · · · · · · · ·
1000	THEN		{**************************************
1001	BEGIN	1056	<pre>PROCEDURE AdvanceNextEvent(VAR LL:LinkedList; VAR Q:QueueType);</pre>
1002	TN := FirstNode^.TerminalNo;	1057	VAR
1003	RandomDelay := Trunc(-(ln(1-DUNI))/Alpha)+1;	1058	FirstNode : ListPointer;
1004	Terminal[TN] := BLOCKED;	1059	BEGIN
1005	CreateEvent(ReArrival, Time+RandomDelay, TN, LL);	1060	IF LL.head = NIL
1006	<pre>DelayArray[TN] := DelayArray[TN] + RandomDelay;</pre>	1061	THEN
1007	Inc(GCount);	1062	BEGIN
1008	END	1063	
1000	ELSE	1064	END
	BEGIN	1065	ELSE IF LL.head^.NextNode = NIL
1010		1066	
1011	TN := FirstNode [*] .TerminalNo;	1067	
1012	Terminal[TN] := INTERFERE;		
1013	WHILE NOT QueueEmpty(Q) DO	1068	
1014	BEGIN	1069	
1015	Remove(X2, Q);	1070	
1016	TN2 := X2.TerminalNo;	1071	
1017	Terminal[TN2] := INTERFERE;	1072	
1018	END;	1073	
1019	Pbe := CalculatePbe(TN);	1074	
1020	<pre>Psp := Power(1-Pbe, PacketLength);</pre>	1075	LL.head := LL.head^.NextNode;
1021		1076	<pre>ProcessFirstNode(FirstNode, LL, Q);</pre>
1022	IF DUNI < Psp	1077	END;
1023	THEN	1078	Dispose(FirstNode);
1024	BEGIN	1079	END; {AdvanceNextEvent}
1025	Inc(SuccessCount);	1080	
1026	Terminal[TN] := IDLE;	1081	{**************************************
1020	DelayCount := DelayCount + DelayArray[TN] + 0.5;		PROCEDURE InitEventList(VAR LL:LinkedList);
		1083	
1028	DelayArray[TN] := 0;	1084	
1029	END		
1030	ELSE	1085	
1031	BEGIN		BEGIN
1032	Terminal[TN] := BLOCKED;	1087	
1033	RandomDelay := Trunc(-(ln(1-DUNI))/Alpha)+1;	1088	
1034	CreateEvent(ReArrival, Time+RandomDelay, TN, LL);	1089	
1035	DelayArray[TN] := DelayArray[TN] + RandomDelay;	1090	CreateEvent(Arrival, Time, i, LL);
1036	Inc(GCount);	1091	END
1037	END	1092	END;{InitEventList}
1038	END	1093	
	END; {ProcessReceive}	1094	{*************************************
1040			FUNCTION CountList(VAR LL:LinkedList):Integer;
1041	(**************************************	1096	
	PROCEDURE ProcessFirstNode(FirstNode:ListPointer; VAR LL:LinkedList;	1097	
1042	VAR Q:QueueType);	1098	
	the deduction in the second se		BEGIN
1044	PECIN	1100	
1045	BEGIN	1100	

Page 11, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes.

1101	Current := LL.head;	1156 ({ WarnBeep emits a half second warning tone and then waits 4.5 seconds	}
1102	WHILE Current <> NIL DO	1157 (C	}
1103	BEGIN	1158 ({ Depends on: none	3
1104	<pre>Current := Current^.NextNode;</pre>			3
1105	i := i + 1;	1160 (}
1106	END:	1161 ({*************************************	•}
1107	CountList := i;		Procedure WarnBeep;	
	END; {CountList}	1163 \		
1109		1164	i : Integer;	
1110	{**************************************	1165 E		
	PROCEDURE RemoveList(VAR LL:LinkedList);	1166	FOR i := 1 TO 5 DO	
1112		1167	BEGIN	
1113		1168	Sound(500+i*20);	
	BEGIN	1169	Delay(300);	
1115	Current := LL.head;	1170	END;	
1116	WHILE Current <> NIL DO	1171	NoSound;	
1117	BEGIN	1172	Delay(1000);	
1118	Current := Current [^] .NextNode;		End;{WarnBeep}	
1119	Dispose(LL.head);	1174		
1120	LL.head := Current;		{**************************************	3
1121	END;	1176 (ŝ
			{*************************************	n's
1123	END;{RemoveList}	1178		,
1120	{**************************************	1179 E	Regin	
	PROCEDURE OpenResultFile(VAR OutFile:Text);	1180	ClrScr;	
1126		1181	WriteLn('Output Testprogramma:');	
		1182		
1127 1128	GS,	1183	{Initialize variables}	
1120	NL, FileName : String;	1184	Rreceive := 30;	
	BEGIN	1185	Rterm := 5;	
1131	Str(GroupSize, GS);	1186	SNR := 20;	
1132	Str(NumberOfLines, NL);	1187		
1133	FileName := 'V1SLT2P7.DAT';	1188	Nt := 32;	
1134	Assign(OutFile, FileName);	1189	NumberOfLines := 8;	
1135	Assignoutrite, ritename,	1190	GroupSize := 4;	
1136	ReWrite(OutFile);	1191		
1137		1192	<pre>OpenResultFile(ResultFile);</pre>	
	WriteLn(OutFile);	1193	InitCodes(CCat, GroupSize, NumberOfLines);	
1138 1139	WriteLn(OutFile,'Size of group : ', GroupSize);	1194	InitPathGain(S, GroupSize, NumberOfLines, Rreceive, Rterm, LossExp);	
	WriteLn(OutFile, Number of groups: ', NumberOfLines);	1195	intractidating, droupoize, numberorentes, krecerve, kterin, Eossexp),	
1140		1196	FOR i := 0 TO NumSim DO { Loop over terminal activity levels	ъ
1141 1142	WriteLn(OutFile); WriteLn(OutFile,'Ptr : ', Ptr);	1197	BEGIN	,
		1198	{Set terminal activity}	
1143	WriteLn(OutFile, 'Alpha : ', Alpha);	1199	PArrH := Power(10, -(LRangeMin + i*(LRangeMax-LRangeMin)/NumSim));	
1144	WriteLn(OutFile,'SNR : ', SNR);	1200	rInit(3647523, 65321); (Initialize FSU Ultra random generator	~
1145	WriteLn(OutFile);	1200		ś
1146	WriteLn(OutFile,'Warm up time : ', WarmTime);	1202		ś
1147	<pre>WriteLn(OutFile,'Simulation time: ', EndTime-WarmTime);</pre>	1202		3
1148	WriteLn(OutFile);	1203	CreateList(EventList); CreateQueue(ReceiveQueue);	
1149	WriteLn(OutFile); } WriteLn(OutFile, 'G ':17,	1204		}
1150	<pre>} WriteLn(OutFile, 'G ':17, 'S ':16,</pre>	1205		
1151		1208	InitDelayArray(DelayArray); {Reset all terminal delays to zero	3
1152		1207	{Show configuration to user}	
	END;{OpenResultFile}	1208	ClrScr;	
1154	{*************************************	1210	WriteLn('Receiver Radius: ', Rreceive:7:2);	
1122		1210		

Page 12, listing of V1SLOT1.PAS, date is 30-10-95, file date is 28-09-95, size is 52584 bytes.

1211	WriteLn('Terminal Radius: ', Rterm:7:2);	1266	WriteLn;
		1267	WriteLn('Delay :', DelayCount);
1212	WriteLn('Number of users: ', Nt:4);		
1213	<pre>WriteLn('Size of group : ', GroupSize:4);</pre>	1268	WriteLn('Clock :', Clock);
1214	WriteLn('Number of codes: ', NumberOfLines:4);	1269	ShowState;
1215	<pre>WriteLn('Code length : ', codelength:4);</pre>	1270	OldClock := Clock;
1216	WriteLn('Packet length : ', Packetlength:4);	1271	END;
1217	WriteLn;	1272	GoToXY(40,4);
1218	<pre>WriteLn('Terminal activity (PArrH) : ',PArrH:8:5);</pre>	1273	WriteLn('SIMULATING !!!!');
1219		1274	GoTOXY(40,6);
	(Decet event countered)	1275	
1220	{Reset event counters}		Write('STATUS: ',i,'/',NumSim);
1221	ArrivalCount := 0;	1276	END;{Simulation Loop}
1222	SuccessCount := 0;	1277	
1223	DelayCount := 0;	1278	{Calculate values for Physical Traffic, Throughput and Delay}
1224	GCount := 0;	1279	<pre>Throughput := SuccessCount/(EndTime-WarmTime);</pre>
1225		1280	D := DelayCount/SuccessCount;
1226	{Warm up the network}	1281	<pre>G := (ArrivalCount+GCount)/(EndTime-WarmTime);</pre>
1227	Clock := 0;	1282	
		1283	(Unite simulation results in files for pressoning purposes later on)
1228	OldClock := 0;		{Write simulation results in files for processing-purposes later on}
1229	While Clock < WarmTime DO	1284	WriteLn(ResultFile , G :16,
1230	BEGIN	1285	Throughput :16,
1231	AdvanceNextEvent(EventList, ReceiveQueue);	1286	D :16);
1232	IF (Clock - OldClock) > Interval	1287	
1233	THEN	1288	RemoveList(EventList);
1234	BEGIN	1289	
1235	GOTOXY(1,12);	1290	END;{Terminal activity level loop}
1236	WriteLn('Arrivals :', ArrivalCount:8);	1291	
1237	<pre>WriteLn('Success :', SuccessCount:8);</pre>	1292	Close(ResultFile);
1238	WriteLn;	1293	
1239	WriteLn('Delay :', DelayCount);	1294	{WarnBeep;}
1240	WriteLn('Clock :', Clock);	1295	GoToXY(30,16);
1241	ShowState;	1296	Write('I AM READY MISTER LE !!!!!');
1242	OldClock := Clock;	1297	GoToXY(30,17);
1243	END;	1298	Write('Press ENTER to return to program.');
1244	GOTOXY(40,4);	1299	{ReadLn;}
		1300 END.	
1245	WriteLn('SIMULATING !!!!');	1300 END.	
1246	GoTOXY(40,6);		
1247	<pre>Write('STATUS: ',i,'/',NumSim);</pre>		
1248	END;{Warm up loop}		
1249			
1250	{Reset event counters}		
1251	ArrivalCount := 0;		
1252	SuccessCount := 0;		
1253	GCount := 0;		
1254			
	DelayCount := 0;		
1255			
1256	{Main simulation Loop}		
1257	While Clock < EndTime DO		
1258	BEGIN		
1259	AdvanceNextEvent(EventList, ReceiveQueue);		
1260	IF (Clock - OldClock) > Interval		
1261	THEN		
1262	BEGIN		
1263	GOTOXY(1,12);		
1265			
	WriteLn('Arrivals :', ArrivalCount:8);		
1265	<pre>WriteLn('Success :', SuccessCount:8);</pre>		

Page 1, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes.

2 (* Sim 3 (* 'Un 4 (* thr 5 (* 6 (* Mos 7 (* ori 8 (* 9 (* The	Winslotted is a slotted Hybric oughput and de t of the const ginates from H terminals are tance. The dis	a prog CDMA lay o ants, uub v clus	ram designed /ISMA protoc of a network variables, van Roosmale stered around	d with the col'. This of termin functions n. d distribu	e goal to sin program cal mals by simul and procedu ted receiver	culates the lation. ures used here rs at a fixed	**> **> *> *> *> *> *> *> *> *> *> *> *>	57 58 59 60 61 62 63	Туре	OnOff = Matrix = PosArr = RArr = StringArr = TermArr =	= (ON, OFF); = Array[1MaxTe = Array[1MaxTe = Array[1MaxTe = Array[1MaxTe = Array[1MaxTe	rm, 1MaxL] rm] rm]	
12 {* rec 13 {* wir 14 {*	12 (* received by the distributed receivers and sent to the base station by 13 (* wire or lines. These lines can be seen as primitive concentrators 14 (*							66 67 68 69 70		CRec =	= Record cc :Array[- End;	maxcodelengthn	maxcodelength] Of ShortInt;
17 Program 18	SimUnslotted(Input		*******	**********	******	**}	71 72 73			0 but is bound. This is a strationary strate	erm, 1MaxTerm]] Of p2CRec;
19 Uses 20 21 Const 22 23 24	Dos, Crt, Ult MaxTerm PacketLength Pt	=	64;	{ Number	n number of t of bits per tter power i		} }	74 75 76 77 78 79	{	QueueEntry		:Integer; :Extended;	
25 26 27 28 29 30 31	LossExp c f lambda twoPi	=	2; 299792458; 1700000000; c/f; 2 * Pi;	<pre>{ Attenua {fai { Speed o { Transm { Signal { Two tim</pre>	ation paramet field model of light in m tter center wavelength mes Pi	ter for the l n/s	> > > >	80 81 82 83 84 85 86		QueueType =	Count, Front, Rear: OMax	term; [1Maxterm] OF	F QueueEntry;
32 33 34 35	Pi2 Alpha	= = 0.		{ Pi squa	{ The number	r of Rearrivals p tDuration	} er }	87 88 89 90	{				On, BusyToneOff, Receive);
36 37 38 39 40 41	bitrate PacketDuratic Fraction maxcodelength codelength	on= 64 = 0. n = 12 = 3	2; 27; 31;		{Time neede { Inhibit de { Maximum d { Length of	f code	et} }	91 92 93 94 95 96		ListPointer ListNode	r = ^ListNode; = RECORD Event Time TerminalNo NextNode	:EventType :Extended; :Integer; :ListPoint	;
42 43 44 45 46 47	Tb Tc Tm MaxL L	= 1/ = 10 = 8; = Tr	00E-9; runc(Tm/Tc)+	1;	{ Maximum r { Number of	ation delay spread number of paths f paths	> > > >	97 98 99 100 101 102		LinkedList	Head END;	:ListPoint	
48 49 50 51 52	WarmTime Interval EndTime NumSim	= = =	500; 125000; 20;	{ Time be { Simular { Number		n updates Tpd levels simulated		103 104 105 106 107	{ Var	Count, Front, Rear		{ Counters for	QueueType }
53 54 55	LRangeMin LRangeMax	=	4;	{ Termina		from 10 ⁻ -LRangeMi		108 109 110		CCat PArrH,	: CArr;	•	ode cross correlations } al Probability Slotted }

Page 2, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes.

113Rterm, (1 minist radius around receivers)168power := 0(power(0,b) = 0114SNR:Extended;(Signal to white Noise Ratio at receiver)169Else(power(0,b) = 1115Nt, (Composite, (Number of terminals per group)171Iff End(fOwer)(power(0,b) = 1116Nt, (Number of terminals per group)172(power(0,c))(power(0,c))1117GroupSite, (Number of terminals per group)173(momber of codes used.)174(ShouState shows the terminal states on screen 176 (Depends on: config, numberOfLines, terminal 177 (Calls : none120DelayArray:PosArr; (Lotal delay time in Tpd)177 (Calls : none178(Modifies : none123SuccessCount, (Number of succesful packets)170(Modifies : none178(Modifies : none124SuccessCount, (Clock maintained for screen updates)180Procedure ShocKate; 180180178(Modifies : none125ArrivalCourt, (Clock maintained for screen updates)185GotXV(12, 22); 184182123133EventList:Linedjit; (Nodes of events linked together)186Write('1 2 3 4 5 6 7 8 9 0 1 3 2	111	PArrL,		{ Packet Arrival Probability Unslotted	}	166 If b<>0 167 Then	
iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii	112 113	Rreceive, Rterm,		{ Receiver radius around base station { Terminal radius around receivers	} }		
Nt. (Number of terminals per group) 171 End;(Cover) 111 Consplize, Conspliz		SNR	:Extended;	{ Signal to white Noise Ratio at receiver	• }		
118 Number Of Lines : Integer; C. also the number of codes used. 173 (************************************		Nt,		{ Number of terminals	}	ine pener e il	
110 1111 111 111					-		
121 DelayArray :PosArr; C Delay administration for each terminal 175 (Depends on: config, numberOfLines, terminal) 122 DelayCount :Extended; (Total delay time in Tpd 177 (C Delay administration for each terminal) 123 SuccessCount; (Number of successful packets) 177 (C Calls : none 124 SuccessCount; (Number of accepted packets) 178 (C Modifies : none 125 ArrivalCount; (Number of accepted packets) 178 (C Modifies : none 125 Gount :Longin; (Number of accepted packets) 178 (C Modifies : none 126 Glock (Clock maintained for screen updates) 178 (Modifies : none 125 OldClock, (Clock maintained by main program) 186 (Write('' I 2 3 4 5 6 7 8 9 0 1 3 2 4 5 6 7 8 9 0 1 3 2 4 5 6 7 8 9 0 1 3 2 4 5 6 7 8 9 0 1 3 2 4 5 6 7 8 9 0 1 1 2 2 4 5 6 7 8 9 0 1 3 2 4 5 6 7 8 9 0		NumberOfLines	:Integer;				
121 DelayArray :PosArry (Delay duministration for each terminal.) 176 (Depende on: config. numberOfLines, terminal.) 122 DelayCount (Number of successful packets.) 177 (Call delay time in Tpd. 123 SuccessCount, (Number of accepted packets.) 177 (Call delay time in Tpd. 124 SuccessCount, (Number of accepted packets.) 177 (Call delay time in Tpd. 125 ArrivalCount, (Number of accepted packets.) 179 (Percedure ShocKate; 126 Count LongInt; (Number of accepted packets.) 180 (Call delay, time d				{ also the number of codes used.	}		
122 DelayCount :Extended; C Total delay time in Tpd) 177 C Calls : none 123 SuccessCount, (Number of successful packets) 170 C Modifies : none 124 SuccessCount, (Number of successful packets) 180 Procedure Shoultate; 125 ArrivalCount, (Number of collisions occured) 181 Var 126 GCount :LongInt; (Number of collisions occured) 181 Var 126 GCount :LongInt; (Number of collisions occured) 181 Var 127 GCount :LongInt; (Number of collisions occured) 181 Var 126 GCount :LongInt; (Number of collisions occured) 181 Var 128 S :Matrix; (Cock maintained for screen updates) 185 GotoXr(12, 22); 1 2 3 4 5 6 7 8 9 0 1 3 2 4 5 6 7 8 9 0 1 1 2 4 5 6 7 8		DelavArray	·PosArr.	{ Delay administration for each terminal	3		
124 SuccessCount, (Number of successful packets) 179 (************************************							
125 Arrivalization, i C Number of accepted pakkets) 180 Procedure ShouState; 126 GCount Linght, i Number of accepted pakkets) 181 Var 127 S :Matrix; i (Power loss ratio matrix)) 181 Var 182 i: Integer; 128 S :Matrix; i (Power loss ratio matrix)) 183 c: Char; 130 OldClock, :Extended; i (Clock maintained for screen updates)) 186 detrift 1 2 3 ') 131 Clock :Extended; i (Clock maintained by main program) 187 detrift 1 2 3 ') 132 Exernitist: Indedistri; (Nodes of events linked torgether) 188 detrift 1 2 3 ') 133 Terminal : TermAr; (Convariable for terminal catrivity levels) 197 dotX/(10-2*i,24); 1 2 3 ') 134 i : Integer; (Loopvariable for terminal catrivity levels) 197 dotX/(10-2*i,24); 1						178 { Modifies : none }	
126 GCount :LongInt; (Number of collisions occured) 181 Var 127 S :Matrix; (Power loss ratio matrix) 182 c: Char; 129 OldLock, :Clock maintained by main program) 185 GotXVI[2, 22]; 1 2 3 ') 131 Clock ::Extended; :LinedList;(Nodes of events linked together) 186 Write(12, 23); 1 2 3 ') 133 Eventiat ::IntedList;(Nodes of events linked together) 188 Write(12, 23); 1 2 3 ') 134 ReceiveQueue ::QueueType; (Queue for received packets) 189 For i:=1 To Nt Do Begin 5 6 5 100 Begin 6 5 100 E ::Write(1'); 1 2 3 ') 135 Terminal ::TermAr; (Convariable for terminal activity levels) 197 GotXX(10-2*1,24); 6 101 E ::Write(1*1); 102 E ::Writ	124	SuccessCount,			-		
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$				{ Number of accepted packets	-		
128 129 129133 128 129135 128 129135 129 120 120 121 121 121 121 121 121 		GCount	:LongInt;	{ Number of collisions occured	}		
120 01dClock, (Clock maintained for screen updates) 184 Begin 130 01dClock, (Extended; (Clock maintained by main program) 185 GotoXY(12, 23); GotoXY(12, 23); 1 2 3 1) 132 EventList (LinkedList;(Nodes of events linked together) 188 Write('1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 3 2 4 5 6 7 8 9 0 1 1 2		<u>c</u>	Motnix	C Pouce Loca patio matrix	1		
130 OldClock, (Clock maintained for screen updates) 185 GotoXY(12, 22); 131 Clock : Extended; (Clock maintained by main program) 186 GotoXY(12, 23); 132 EventList : LinkedList; (Nodes of events linked together) 187 GotoXY(12, 23); 134 ReceiveQueue : QueueType; (Queue for received packets) 189 For i := 1 To NE Do 135 GotoXY(12, 23); (Indicates wheather the channel is free) 191 GotoXY(12, 24); 136 BusyTone :OnOff; (Indicates wheather the channel is free) 192 Case Terminal 10 of 137 GotoXY(12, 22); 189 For i := 1 To NE Do Begin 136 BusyTone :OnOff; (Indicates wheather the channel is free) 191 GotoXY(12, 24); 137 GotoXY(12, 22) Case Terminal 10 of 193 IDLE : Write('1'); 138 Terminal : ItemArr; (Coopyariable for terminal activity levels) 195 IRMKPARE 144 Throughput, (Incoughuzt: fraction of the time intervision) 195 End;(Chaso) 105 145 carrying a successful transmission) 220 Cacluates a partial cross correlation 102		5	:Matrix;		,		
131Clock:Extended;(Clock maintained by main program)186Write(11231'132EventList:LinkedList; (Nodes of events Linked together)188Write(11 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 3 2 4 5 6 7 8 9 0 1 1 2 1 2 1 5 6 7 8 0 0 1 1 2 1 0 1 1 1 0 f135Chorewere and the approximal state information196IDEE : Write('1'0;141Throughput;fraction of the time142ResultFileThroughput;143Throughput;fraction of the time144Throughput;Throughput;150(Mean Delay romalized in slots)161(M		OldClock.		{ Clock maintained for screen updates	}		
133 EventList :LinkedList; Nodes of events linked together) 188 Write(1' 2 3 4' 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 3 2 4 5 6 7 8 9 0 1 2 2 4 5 6 7 8 9			:Extended;		}		
15ReceiveQueue:QueueType; (Queue for received packets189For i:= 1 To Nt Do155BusyTone <td::onoff; (="")<="" channel="" free="" indicates="" is="" td="" the="" wheather="">190Begin157Cor not.192Case Terminal(1) Of158Terminal<td:termarr; (="")<="" information="" state="" td="" terminal="">193DLE: Write('i');160i:Integer; (Loopvariable for terminal activity levels)195TRANSMIT : Write('t');161i:Integer; (Coopvariable for terminal activity levels)196INTERFERE : Write('t');162ResultFile:Text; (Result file with original values)197End;CFor i:= 1 To Nt)164Throughput, (Throughput: fraction of the timecarrying a guecessful transmission)200165D, (Mean Delay normalized in slots)201 (************************************</td:termarr;></td::onoff;>	132						
155Inclusion (inclusion (incl				•	-		
BusyTone ::0nOff; (Indicates wheather the channel is free) For the system of the sy		ReceiveQueue	:QueueType;	{ Queue for received packets	}		
137Low function192Case Terminal (1) Of138Terminal :TermArr;(Terminal state information)193IDLE : Write('i');139i:Integer;(Loopvariable for terminal activity levels)194BLOCKDWrite('b');140i:Integer;(Loopvariable for terminal activity levels)195TRANSNIT : Write('t');141196INTERFERE : Write('t');142Resultfile :Text;(Result file with original values))197End;(Case)143198End;(For i := 1 to Nt)144Throughput,(Introughput: transmission)200145cerrying a successful transmission)200146D,(Mean Delay normalized in slots)201147G:Extended;(Mean Delay: time between the arrival of202 (R calculates a partial cross correlation)148lest bit in sec.)205 (Calls : none205 (Calls : none)151(Depends on: none)206 Function R(i,j:Integer; tau:Extended):Extended;152 (Depends on: none)209 Var208 Function R(i,j:Integer; tau:Extended):Extended;155 (Modifies : none)210 L,(Chip number belonging to tau156 (Grunt : none)211 Cl,(Discrete correlation function for chip L157 (Pre: a >= 0 And b >=0)213 Begin158 (Post: Power = a'b)213 Begin164power := Exp(b#Ln(a))218 Begin165If a <>0218 Begin166 <td< td=""><td></td><td>PuevTopo</td><td>•OpOff•</td><td>[Indicates wheather the channel is free</td><td>3</td><td></td><td></td></td<>		PuevTopo	•OpOff•	[Indicates wheather the channel is free	3		
139 139 139Terminal:Terminal state information103 104 104 104 105 		Busyrone	.01011,				
100i:Integer;(Loopvariable for terminal activity levels)195TRANSNIT:Write('t');141141196INTERFERE:Write('t');142ResultFile:Text;(Result file with original values)196INTERFERE:Write('t');143144Throughput,(Throughput: fraction of the timecarrying a successful transmission)198End;(For i := 1 TO Nt)1440,(Mean Delay: time between the arrival ofclast bit in sec.)2002011460,(Mean Delay: time between the arrival ofclast bit in sec.)202 (R calculates a partial cross correlationclast bit in sec.)205 (Calls : noneclast bit in sec.)150(************************************		Terminal	:TermArr;		3	193 IDLE : Write('i');	
141196INTERFER: $Write('f');$ 142Result File : Text;(Result file with original values)196INTERFER: : Write('f');143Throughput,(Throughput: fraction of the timecarrying a successful transmission)198End;(Gos)144Throughput,(Throughput: fraction of the timecarrying a successful transmission)199End;(Gos)145carrying a successful transmission)200200146D,(Mean Delay normalized in slots)201(************************************	139						
142ResultFile :Text; (Result file with original values)197End;(Case)143Throughput, (Throughput: fraction of the timecarrying a successful transmission)198End;(Core)144Throughput, (Mean Delay normalized in slots)200145carrying a successful transmission)200146D, (Mean Delay normalized in slots)201147G :Extended; (Mean Delay: time between the arrival oflast bit in sec.)202 (R calculates a partial cross correlationlast bit in sec.)148last bit in sec.)204 (Depends on: config, CCat, Tc150(************************************		i	:Integer;	{Loopvariable for terminal activity level	ls}		
143144145144145198End;(For i := 1 T0 Nt)144144145carrying a successful transmission)200146D,(Mean Delay normalized in slots)201147G:Extended; (Mean Delay: time between the arrival of202 (R calculates a partial cross correlation148the first bit till the receive of the203 (149tast bit in sec.)206 (Calls : none151Power calculates a raised to the power of b)206 (Modifies : none152 (last bit in sec.)200 (***********************************				C Develo (its with an initial values			
144Throughput, carrying a successful transmission)199 End;(ShowState)145carrying a successful transmission)200146D,(Mean Delay normalized in slots)201147G:Extended;(Mean Delay: time between the arrival of the first bit till the receive of the the first bit till the receive of the time receive of the receive of the receive of the receive		ResultFile	:lext;	{ Result file with original values	3		
<pre>145carrying a successful transmission) 146 D, (Mean Delay normalized in slots) 147 G :Extended; (Mean Delay: time between the arrival of 148the first bit till the receive of the 149last bit in sec.) 150 (************************************</pre>		Throughput		{ Throughput: fraction of the time			
146D,(Mean Delay normalized in slots)201 (************************************		in oughput,			}	200	
148the first bit till the receive of the203 (149last bit in sec.)150 (************************************	146	D,			2		
140last bit in sec.204 { Depends on: config, CCat, Tc150 (************************************		G	:Extended;				
150 (************************************							
151 (Power calculates a raised to the power of b 206 (Modifies : none 152 (207 (************************************	149	*****	*****	last Dit in sec.		Eer (bepende en een ij veder ie	
152 (153 (Depends on: none154 (Calls : none155 (Modifies : none155 (Modifies : none156 (157 (Pre : a >= 0 And b >=0158 (Post: Power = a^b159 (************************************	-				}	206 { Modifies : none }	
153 (Depends on: none } 208 Function R(i,j:Integer; tau:Extended):Extended; 154 (Calls : none } 209 Var 155 (Modifies : none } 210 l, (Discrete correlation function for chip l 156 (} 211 Cl, (Discrete correlation function for chip l 157 (Pre: a >= 0 And b >=0 212 Cl1 :Integer; (Discrete correlation function for chip l+1) 158 (Post: Power = a^b) 213 Begin 159 (************************************		Toker catedatates a			3	207 {************************************	
155 (Modifies : none210 l,(Chip number belonging to tau156 (3211 Cl,(Discrete correlation function for chip l157 (Pre : a >= 0 And b >=03212 Cl1 :Integer;(Discrete correlation function for chip l+1158 (Post: Power = a b3213 Begin159 (************************************		Depends on: none			}	208 Function R(i,j:Integer; tau:Extended):Extended;	
156 (211 Cl,Discrete correlation function for chip l156 (}211 Cl,(Discrete correlation function for chip l157 (Pre : a >= 0 And b >=0}212 Cl1 ::Integer;(Discrete correlation function for chip l+1158 (Post: Power = a^b}213 Begin159 (************************************	154 (Calls : none			}		
<pre>157 { Pre : a >= 0 And b >=0 } 158 { Post: Power = a^b } 159 {************************************</pre>	and the second	Modifies : none			}		
158 { Post: Power = a^b } 159 { ***********************************	-		N-0		-		
159 (************************************	-		2-0		3		
160 Function Power(a,b:Extended):Extended; 215 Cl1 := CCat[i,j]^.cc[l+1-codelength]; 161 Begin 216 If l >= 1 162 If a<>0 217 Then 163 Then 218 Begin 164 power := Exp(b*Ln(a)) 219 Cl := CCat[i,j]^.cc[l-codelength];	159 (*	**************************************	******	*****	**}		
161 Begin 216 If l >= 1 162 If a<>0 217 Then 163 Then 218 Begin 164 power := Exp(b*Ln(a)) 219 Cl := CCat[i,j]^.cc[l-codelength];	_				-		
163 Then 218 Begin 164 power := Exp(b*Ln(a)) 219 Cl := CCat[i,j]^.cc[l-codelength];		-		-			
164 power := Exp(b*Ln(a)) 219 Cl := CCat[i,j]^.cc[l-codelength];							
			(D. LU(a))				

3, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes. Page

276

277

278 279

280

281

282

283

284

285

286 287

288

289

290

291

292

293

294

295

296

297

298

299 300

301

302

304

307 {

314

315

317

318

319

321

324 {

221	End		
222	Else		
223	R := tau*Cl1;		
224	End; {R}		
225			-
226	<*************************************	*****	
227	{ Rhat calculates a partial cross correla		}
228	{		}
229	{ Depends on: config, CCat, Tc		}
230	{ Calls : none		}
231	{ Modifies : none		}
232	{*************************************	*****	·}
233	<pre>Function Rhat(i,j:Integer; tau:Extended):E</pre>	xtended;	
234			
235		ber belonging to tau	>
236		correlation function for chip l	}
237	• •	correlation function for chip l+1	}
	Begin		
239	<pre>l := Trunc(tau/Tc);</pre>		
240	Cl := CCat[i,j]^.cc[l];		
241	If l < codelength-1		
242	Then		
243	Begin		
244	Cl1 := CCat[i,j]^.cc[l+1];	\ .	
245 246	Rhat := Cl*Tc+(Cl1-Cl)*(tau-l*Tc	/,	
240	End Else		
247	Rhat := ((l+1)*Tc-tau)*Cl;		
	End; (Rhat)		
250			
251	{****	*****	3
252			3
253			3
254			3
255		ross correlation information. This	3
256			}
257			}
258			}
259	{ Depends on: config, p2CRec		}
260			}
261	{ Modifies : none		}
262	{****	*****	۲
263	Procedure InitCCat(Var CCat:CArr; gc:Strin	gArr);	
264	Var		
265	i,	{ Loop variable	}
266	j,	{ Loop variable	}
267	k,	{ Loop variable	}
268	ι,	{ Discrete phase shift	}
269	t :Integer;	<pre>{ Correlation result for shift l</pre>	}
	Begin		
271	Write('Calculating ', Nt, ' by ', Nt,		
272		{ Double loop over all codes	}
273	For j := 1 To Nt Do		
274	Begin CotoVV(70 1):	{ Show which codes are being done	3
275	GotoXY(70,1);	t show whitch codes are being done	,

Write(i:4,j:4); { Allocate space for new array } CCat[i,j] := New(p2CRec); { Calculate cross-correlation } For l := -codelength To codelength Do Begin { Reset correlation value 3 t := 0; If l < 0{ Different formulas for pos and } { neg phase shifts } Then For k := 0 To codelength-1+1 Do If gc[i][1+((k-l) Mod codelength)] = gc[j][k+1] Then Inc(t) Else Dec(t) Else For k := 0 To codelength-l-1 Do If gc[i][1+(k Mod codelength)] = gc[j][1+((k+l) Mod codelength)] Then Inc(t) Else Dec(t); CCat[i,j]^.cc[l] := t; End; End; 303 End; {InitCCat} 306 { DisposeCCat frees memory space taken up by the correlation catalog } 3 308 { Depends on: config } 3 309 { Calls : none 310 { Modifies : none } 312 Procedure DisposeCCat(CCat:CArr); 313 Var { Loop variable } i, { Loop variable } j :Integer; 316 Begin For i := 1 To Nt Do { Loop over all catalog entries } For i := 1 To Nt Do { Free memory of element i, j } Dispose(CCat[i,j]); 320 End;{DisposeCCat} 323 { InitTerm initializes all terminals as free > 3 } 325 { Depends on: config 326 { Calls : none } 327 { Modifies : pfn > 329 Procedure InitTerm(Var Terminal:TermArr); 330 Var

Page 4, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes.

331	i :Integer; { Loop variable }	386	{*************************************
	Begin	387	{ InitPathLoss sets up a table of transmitter to receiver path gains from }
333	For i:= 1 To Nt Do		{ all transmitters to all receivers. All gains are between 0 and 1. When }
334	<pre>terminal[i] := IDLE;</pre>		· Our de la company and
	End;{InitTerm}		{ table because there are less receivers than terminals then. }
336		391	{
337	{**************************************	392	{ Depends on: config, lambda }
	PROCEDURE InitDelayArray(VAR DelayArray:PosArr);	393	{ Calls : Power }
339		394	{ Modifies : none }
		305	{*************************************
340			
	BEGIN		Procedure InitPathGain(Var S:Matrix; groupSize, numberOfLines :Integer;
342	FOR i := 1 TO MaxTerm DO	397	
343	DelayArray[i] := 0;	398	Var
344	END;{InitDelayArray}	399	gamma, { Angle of receiver relative to x-axis }
345		400	
7/4	{*************************************	401	,
	Concentration of a solution was prevented from the set of the solution of the	402	
	PROCEDURE InitBusyTone(VAR BusyTone:OnOff);		
	BEGIN	403	
349	BusyTone := Off;	404	
350	END;{InitBusyTone}	405	k :Integer; { Index variable }
351		406	Begin
352	{**************************************	407	
	{ Initcode reads Gold codes from a file and has the cross correlation }	408	
354		409	
355	{	410	
356	{ Depends on: config }	411	<pre>gamma := 2*Pi * (i-1)/numberOfLines; {Calc. receiver angle }</pre>
357	{ Calls : initCCat }	412	<pre>xr[k] := Rreceive * Cos(gamma); {Calc. receiver x-coord. }</pre>
358	(Modifies : pfn)	413	<pre>yr[k] := Rreceive * Sin(gamma); {Calc. receiver y-coord. }</pre>
350	{*************************************	414	
		415	
	<pre>Procedure InitCodes(Var CCat:CArr; groupSize, numCodes:Integer);</pre>		
361		416	
362	i, { Loop variable }	417	
363	j :Integer; { Loop variable }	418	
364	f :text; { File containing the Gold codes }	419	For j := 1 To Nt Do
365	codeCount, { Length of transmission code }	420	Begin
366	cfile :String; { Codefile name }	421	{ Calculate path gains according to far field model }
367	gc :StringArr; { Gold codes as read from file }	422	• • • • • • • • • • • • • • • • • • • •
		423	
	Begin		
369	ClrScr;	424	
370	<pre>Str(codelength, codeCount); { Assign appropriate codefile name }</pre>	425	End;{InitPathGain}
371	cfile := 'codes\'+codeCount+'chip.dat';	426	
372	Assign(f, cfile); { Open code file }	427	<pre>{************************************</pre>
373	Reset(f);		{ Rayleigh randomly draws a value according to the rayleigh distribution }
		429	
374			•
375	ReadLn(f,gc[i]);		{ Depends on: none }
376	Close(f);		{ Calls : duni }
377	If numCodes <> Nt { Reassign codes according to reuse scheme }	432	{ Modifies : pfn }
378	Then { e.g. 12345678 -> 11223344 for 2 term/grp }	433	{*************************************
379	For i := numCodes DownTo 1 Do	434	Function Rayleigh(ave:Extended):Extended;
380	For j := groupSize DownTo 1 Do		Var
381		436	
	<pre>gc[(i-1)*groupSize+j] := gc[i]; initCot(Cot, co);</pre>	430	
382	initCCat(CCat, gc);		-1
383	{ Initialize catalog of cross correlations }	438	•
	End;{InitCodes}	439	
385		440	y :Extended;

Page 5, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes.

441	Begin	496	n := 0;
442	Repeat	497	Repeat
443	u := 2*duni-1;	498	n := n + 1;
444	v := 2*duni-1;	499	ac := d1+2*n*ab/(a*b)+aa;
445	r := sqr(u)+sqr(v);	500	bc := 1+2*n*bb/(a*b)+ba;
446	Until (r<1.0) and (r>0.0);	501	d1 := dd*d1;
447	r := Sqrt(-2*Ln(r)/r);	502	aa := ab;
448	$x := u^*r;$	503	ab := ac;
440		504	
	$y := v^* \Gamma;$		ba := bb;
450	<pre>Rayleigh:=Sqrt(Sqr(x)+Sqr(y))*ave*Sqrt(2/Pi);</pre>	505	bb := bc;
	End;	506	Until bc > 1e12;
452		507	If a < b
	{**************************************	508	Then
454	{ MarcumQ calculates the Marcum Q-function }	509	MarcumQ := (ac/(2*bc))*Exp(-Sqr(a-b)/2)
455	(510	Else
456	{ Depends on: none }	511	MarcumQ := 1-(ac/(2*bc))*Exp(-Sqr(a-b)/2);
457	{ Calls : none }	512	End;
458	{ Modifies : pfn }	513	End;
	{*************************************	514	
460	Function MarcumQ(a,b:Extended):Extended;		<*************************************
461			{ NegExpIO approximates the modified Bessel function of the first kind }
462	aa,		{ using the recurrence relation I (z) = I (z) + $2n/z$ I (z) }
463	ab,	518	
464		519	
465	ac,		
	bc,		{ and then multiplies the result with Exp(-(a ² +b ²)/2). }
466	bb,	521	
467	ba,		{ Depends on: none }
468	dd,		{ Calls : none }
469	d1 :Extended;		{ Modifies : pfn }
470	n :Integer;	525	{**************************************
	Begin		Function NegExpI0(a,b:Extended):Extended;
472	If a = 0	527	Var
473	Then	528	n :Integer; { Loop variable }
474	If b < 30	529	х,
475	Then	530	Ja, { I0,n-1 }
476	MarcumQ := Exp(-Sqr(b)/2)	531	Jb, { 10,n }
477	Else	532	Jc, { 10,n+1 }
478	MarcumQ := 0	533	tmp,
479	Else	534	t :Extended; { Intermediate result }
480	Begin		Begin
481	If a < b	536	
482		537	<pre>tmp := (Sqr(a)+Sqr(b))/2;</pre>
	Then		x := a*b;
483	Begin	538	If $x = 0$
484	ab := 1;	539	Then
485	dd := a/b;	540	If tmp < 150
486	End	541	Then
487	Else	542	NegExpIO := Exp(-tmp)/2
488	Begin	543	Else
489	ab := 0;	544	negExpIO := 0
490	dd := b/a;	545	Else
491	End;	546	Begin
492	aa := 0 ;	547	Jc := 0;
493	bb := 0.5;	548	Jb := 1;
494	ba := 0 ;	549	t := 0;
495	d1 := dd;	550	For n := 20 DownTo 1 Do

Page 6, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes.

```
551
           Begin
552
               Ja := (2*n/x)*Jb+Jc;
553
               Jc := Jb;
554
               Jb := Ja;
555
               t := t + Ja:
556
           End:
557
           NegExpI0 := Exp(x-(Sqr(a)+Sqr(b))/2)*Ja/(2*t-Ja);
558
       End:
559 End;
560
562 { CalcMu1 calculates the Mu1 parameter for the bit error formula in
                                                                3
563 { Rayleigh fading environments
                                                                }
564 {
                                                                }
565 { Depends on: L
                                                                3
                                                                3
566 { Calls
             : R, Rhat
567 { Modifies : none
569 Function CalcMu1(u, h:Integer; tau:RArr; mu2:Extended):Extended;
570 Var
571
     tmp :Extended;
                                               { Intermediate result }
572
    i :Integer;
                                               { Loop variable
                                                                }
573 Begin
574
       tmp := 0:
575
       For i := 1 To L Do
576
          If i <> h
577
          Then
578
             tmp := tmp + R(u,u,tau[u,i])*Rhat(u,u,tau[u,i])*s[u,u];
579
580
       CalcMu1 := 4*2*Pt*tmp + mu2
581 End;
582
584 { CalcMu2 calculates the Mu2 parameter for the bit error formula in
585 { Rayleigh fading environments
                                                                3
586 {
                                                                3
587 { Depends on: config, L, Pt, terminal
                                                                }
                                                                3
588 { Calls
             : R, Rhat
589 { Modifies : none
591 Function CalcMu2(u, h:Integer; tau:RArr; sigma_n2:Extended):Extended;
592 Var
593
     tmp :Extended;
                                               { Intermediate result }
594
                                               { Loop variable
     i,
                                                                }
595
                                               { Loop variable
                                                                }
    j
        :Integer;
596 Begin
597
       tmp := 0;
598
       For i := 1 To Nt Do
         If (terminal[i] = INTERFERE) And (i <> u)
599
600
         Then
601
            For j := 1 To L Do
602
               tmp := tmp + (Sqr(R(u,i,tau[i,j]))
603
                        + Sqr(Rhat(u,i,tau[i,j])))*s[i,u];
604
605
       For j := 1 To L Do
```

If i <> h 606 607 Then 608 tmp := tmp + (Sqr(R(u,u,tau[u,j]))609 + Sqr(Rhat(u,u,tau[u,j])))*s[u,u]; 610 611 CalcMu2 := 2*(2*Pt*tmp+sigma_n2); 612 End; 613 615 { CalcMu12 calculates the Mu12 parameter for the bit error formula in 3 616 { Rayleigh fading environments 3 617 { } } 618 { Depends on: config, L, Pt, terminal 3 619 { Calls : R, Rhat 620 { Modifies : none } 622 Function CalcMu12(u,h:Integer;tau:RArr):Extended; 623 Var 624 tmp :Extended; { Intermediate result } { Loop variable 625 i, } 626 { Loop variable 3 j :Integer; 627 Begin 628 tmp := 0; 629 For i := 1 To Nt Do 630 If (terminal[i] = INTERFERE) And (i<>u) 631 Then 632 For i := 1 To L Do 633 tmp := tmp + R(u,i,tau[i,j])*Rhat(u,i,tau[i,j])*s[i,u]; 634 635 For i := 1 To L Do 636 If i <> h 637 Then 638 tmp := tmp + (Sqr(Rhat(u,u,tau[u,j])) 639 + R(u,u,tau[u,j])*Rhat(u,u,tau[u,j]))*S[u,u]; 640 641 CalcMu12 := 2*2*Pt*tmp; 642 End: 643 645 { CalculatePbe computes the bit error probability for a terminal, given 646 { which other terminals transmit 647 { 648 { Depends on: config, L, Pt, S, Tb, terminal 649 { Calls : duni, IO, MarcumQ, Rayleigh 650 { Modifies : none 3 652 Function CalculatePbe(u:Integer):Extended; 653 Var 654 mu1, 655 mu2, 656 mu12, 657 a, 658 b, 659 bmax, } { Maximum path gain 660 tmp,

7, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes. Page

```
661
       m,
662
       mq,
663
       mb,
       sigma_n2:Extended;
                                      { Noise power
                                                                                }
664
665
       beta,
                                      { Rayleigh distributed path gains
                                                                                }
                                      { Uniformly [0,Tb] distributed path delays }
666
       tau
               :RArr;
                                      { Number of path with largest gain
                                                                                >
667
       h,
                                      { Loop variable
                                                                                }
668
       i,
                                      { Loop variable
                                                                                }
669
      i
               :Integer;
670 Begin
         { Assign random delays and path gains to transmitting terminals }
671
672
         For i := 1 To Nt Do
            If terminal[i] = INTERFERE
673
674
            Then
675
                For j := 1 To L Do
676
                Begin
                     tau[i,j] := Tb*duni;
677
                     beta[i,j] := Rayleigh(Sqrt(S[i,u]));
678
679
                End
680
            Else
                For j := 1 To L Do
681
682
                Begin
683
                     tau[i,j] := 0;
684
                     beta[i,j] := 0;
685
                End;
686
687
         { Determine what the largest gain is }
         bmax := 0;
688
689
         h := 1;
690
         For i := 1 To L Do
691
            If beta[u,i]>bmax
692
            Then
693
            Begin
694
                 bmax := beta[u,i];
695
                 h
                     := i:
696
            End;
697
698
         { Determine noise power relative to largest incoming signal }
699
         sigma n2 := power(10, -snr/10)*Sqr(bmax)*Pt*Tb*Tb;
700
701
         { Calculate bit error probability parameters }
702
         mu2 := CalcMu2(u, h, tau, sigma n2);
703
         mu1 := CalcMu1(u, h, tau, mu2);
704
         mu12 := CalcMu12(u, h ,tau);
705
         m
            := Sqrt(2*Pt)*bmax*Tb;
             := m*Abs(1/Sqrt(mu1)-1/Sqrt(mu2))/Sqrt(2);
706
         а
707
         b
              := m*(1/Sqrt(mu1)+1/Sqrt(mu2))/Sqrt(2);
708
         mq := MarcumQ(a,b);
709
         mb := NegExpIO(a,b);
710
         { Calculate the bit error probability }
711
712
         tmp := ( (1 + mu12/Sqrt(mu1*mu2))*mb )/2;
713
         CalculatePbe := mq-tmp
714 End;
715
```

716	<pre>{************************************</pre>
717	C QUEUES >
718	<pre>{************************************</pre>
719	
720	
721	{*************************************
722	
	{ Pre: None }
724	{ Post: The queue Q has been created and is initialized to be empty. }
	(**************************************
	PROCEDURE CreateQueue(VAR Q:QueueType);
	BEGIN
728 729	WITH Q DO
729	BEGIN Count of Ot
731	Count := 0; Front := 1;
732	Rear := 0
733	END
	END;{CreateQueue}
735	
736	<pre>{************************************</pre>
737	PROCEDURE Append (TerminalNo:Integer; Time:Extended; VAR Q:QueueType);
738	VAR
739	X : QueueEntry;
	BEGIN
741	WITH Q DO
742	IF Count = MaxTerm THEN
743	WRITELN ('Error: Attempt to append an entry to a full queue')
744 745	ELSE BEGIN
745	X.TerminalNo := TerminalNo;
740	X.Time := Time;
748	Count := Count + 1;
749	Rear := (Rear MOD MaxTerm) + 1;
750	Entry [Rear] := X;
751	END
752	END;{Append}
753	
	<pre>{************************************</pre>
	PROCEDURE Remove (VAR X: QueueEntry; VAR Q:QueueType);
	BEGIN
757	WITH Q DO
758 759	IF Count = 0 THEN
760	WRITELN ('Error: Attempt to remove an entry from an empty queue') ELSE
761	BEGIN
762	Count := Count - 1;
763	X := Entry [Front];
764	Front := (Front MOD MaxTerm) + 1
765	END
766	END;{Remove}
767	· · · · · · · · · · · · · · · · · · ·
	<pre>(************************************</pre>
	FUNCTION QueueSize (VAR Q: QueueType): INTEGER;
110	BEGIN

Page 8, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes.

```
771
     QueueSize := Q.Count
772 END; {QueueSize}
773
775 FUNCTION QueueEmpty (VAR Q: QueueType): BOOLEAN:
776 BEGIN
777
     QueueEmpty := (Q.Count = 0)
778 END; {QueueEmpty}
779
781 FUNCTION QueueFull (VAR Q: QueueType): BOOLEAN;
782 BEGIN
783
     QueueFull := (Q.Count = MaxTerm)
784 END; {QueueFull}
785
787 PROCEDURE ShowQueue (VAR Q : QueueType);
788 VAR
789
    x, y : INTEGER;
790
    X1 : QueueEntry;
791 BEGIN
792
     {WRITELN('De inhoud van de queue is:');}
793
     x := QueueSize (Q);
794
     WITH Q DO
795
        y := front;
796
        WHILE x > 0 DO
797
        BEGIN
798
           X1 := Q.Entry[y]:
799
           WRITELN('TN: ',X1.TerminalNo,'Time: ',X1.Time);
800
           x := x - 1:
801
           y := (y MOD MaxTerm) + 1
802
        END:
803
        WRITELN;
804 END; {ShowQueue}
805
806
808 (
                    LINKED-LIST
810
811
813 {* CreateList initializes linked list LL to be empty
                                                  *}
                                                  *}
814 (*
815 (* Pre: None
                                                  *3
                                                  *}
816 {* Post: The list LL has been created and is initialized to be empty
818 PROCEDURE CreateList(VAR LL: LinkedList);
819 BEGIN
820
  LL.head := NIL
821 END;{CreateList}
822
824 PROCEDURE InsertEvent(NewNode:ListPointer; T:Extended; VAR LL:LinkedList);
825 VAR
```

826 Current, 827 Trailing : ListPointer; 828 BEGIN 829 Current := LL.head; 830 Trailing := NIL: 831 WHILE (Current[^].Time <= T) AND (Current[^].NextNode <> NIL) DO 832 BEGIN 833 Trailing := Current; 834 Current := Current .NextNode; 835 END: 836 IF (Current[^].time <= T) AND (Current[^].NextNode = NIL) 837 THEN 838 BEGIN 839 Current .NextNode := NewNode; 840 NewNode .. NextNode := NIL: 841 END 842 ELSE IF (Current[^].Time > T) AND (Trailing = NIL) 843 THEN 844 BEGIN 845 NewNode .NextNode := LL.head; 846 LL.head := NewNode; 847 END 848 ELSE 849 BEGIN 850 NewNode .. NextNode := Current; 851 Trailing .NextNode := NewNode; 852 END: 853 END;{InsertEvent} 854 856 PROCEDURE CreateEvent(E:EventType; T:Extended; TN:Integer; VAR LL:LinkedList); 857 858 VAR 859 NewNode : ListPointer; 860 BEGIN 861 New(NewNode); 862 NewNode . Event := E: 863 NewNode^{*}.Time := T; 864 NewNode^{*}.TerminalNo := TN; 865 IF LL.head = NIL 866 THEN 867 BEGIN 868 NewNode .. NextNode := NIL; 869 := NewNode; LL.head 870 END 871 ELSE 872 BEGIN 873 InsertEvent(NewNode, T, LL); 874 END: 875 END;{CreateEvent} 876 878 PROCEDURE ShowList(VAR LL:LinkedList); 879 VAR 880 Current : ListPointer;

Page 9, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes.

B82 Event : string; 937 RandomDelay := <(ln(1-DUNIJ)/Alpha; B83 SEGN 937 CreateVent(Revirus) [ime / RandomDelay, TN, L1); B84 1 1 1 1 1 B85 1 1 1 1 1 1 1 B85 1 1 1 1 1 1 1 1 B85 1 1 1 1 1 1 1 1 B85 1 1 1 1 1 1 1 1 B85 1 1 1 1 1 1 1 1 1 B87 WriteLn('The ist consists of: (Press ENTER for the nodes to show up)'); 92 2	881	i : Integer;	936	BEGIN
B35 EELN 938 CreateVent(ReArrival, Time + RandomDelay, TN, LL); Durrent: # LL.head; B36 EELN 939 DelayArray(TN) = DelayArray(TN) = RandomDelay, TN, LL); DelayArray(TN) = DelayArray(TN) = RandomDelay; B37 EELN 940 Inc(GCGUNT); B38 EELN 940 Inc(GCGUNT); B38 EELN 940 Inc(GCGUNT); B39 IF Current: # LL THEN 943 BEEIN B40 ECIN 943 BEEIN B41 ECIN 943 BEEIN B41 ECIN 943 BEEIN B41 ECIN 943 BEEIN B42 ECIN 943 BEEIN B42 ECIN 943 DelayArray(TN) = TRANSHIT; B42 ELSE 945 DelayArray(TN) = DelayArray(TN) + 1 + Fraction; B42 ELSE 945 DelayArray(TN) = DelayArray(TN) + 1 + Fraction; B43 ELSE 945 DelayArray(TN) = DelayArray(TN) + 1 + Fraction; B43 ELSE 945 DelayArray(TN) = DelayArray(TN) + 1 + Fraction; B44 Hitel (Yournet - VNIL) DO 945 END B450 Readori = P			937	RandomDelay := -(ln(1-DUNI))/Alpha;
B8 i i = 1; 930 DelsyArray(TN] := DelsyArray(TN] := DelsyArray(TN] := Randombelay; B8 Current := LL.hed; 941 B80 Current := LL TEM 941 B80 B8 F Current := LL TEM 944 B80 B8 F Current := LL TEM 944 B80 B8 F Current := LL TEM 944 Terminal (TN) := TANSHIT; B9 B8 F Current := LL TEM 944 Terminal (TN) := TANSHIT; B9 B8 F Current := LL TEM 944 Terminal (TN) := TANSHIT; B9 B9 F Current := TANSHIT; 946 Terminal (TN) := TANSHIT; B9 B8 F Current := TANSHIT; 946 Terminal (TN) := TANSHIT; B9 B9 F Current := TANSHIT; 946 Terminal (TN) := TANSHIT; B9 B9 F Current := Tanshit; 945 ELSE F Current := Tanshit; F Current := Tanshit; F Current := Tanshit; F F F Current := Tanshit; F F F F Current := Tanshit; F F F F Current := Tanshit; F F F F F F F F F F F F F F F F F F F			938	
Base Current := Lichand; 040 Inc(dicount); Base ClifScr; 941 EDD Base Virteln('The list consists of: (Press ENTER for the nodes to show up)'); 942 ELSE Base Virteln('The list consists of: (Press ENTER for the nodes to show up)'); 942 ELSE Base Virteln('The list consists of: (Press ENTER for the nodes to show up)'); 942 ELSE Base Virteln('The list consists of: (Press ENTER for the nodes to show up)'); 942 ELSE Base Virteln('The list consists of: (Press ENTER for the nodes to show up)'); 942 ELSE Base Virteln('The list consists of: (Press ENTER for the nodes to show up)'); 942 ELSE Base Virteln('The list consists) (Press ENTER for the nodes to show up)'); 942 ELSE Base Virteln('The list consists) (Press ENTER for the nodes to show up)'); 942 ELSE Base Virteln('The list consists) (Press ENTER for the nodes to show up)'); 943 ELSE Base Virteln('The list consists) (Press ENTER for the nodes to show up)'); 944 ELSE Base Virteln('The list consists) (Press ENTER for the nodes to show up)'); 945 Press				
041 END 042 ELSE 041 END 042 ELSE 043 ELSE 044 ELSE 045 If Current = NLL 046 ELSE 047 CreateSvert(Reury(nost), Time+Fraction, TM, LL); 048 ELSE 049 ELSE 041 ELSE 041 ELSE 043 ELSE 044 ELSE 045 ELSE 046 DelayArray(TN) := DelayA				
B87 writein("The List consists of: (Press ENTER for the nodes to show up)"); 942 ELSE B89 Writein; 943 BEGIN B89 If Current = NL THEN 944 Terminal(TM) := TRANSHIT; B89 WRIELEN("EventList is emptyl (WriteList)"); 945 Constant (TM) := TRANSHIT; B80 WRIELEN("EventList is emptyl (WriteList)"); 945 Constant (TM) := TRANSHIT; B80 WRIELE (Current = NL) D0 945 DelayArray(TN) := DelayArray(TN) := DelayArray(TN) := Transton; B90 EASE Current'-Event OF 951 END B90 EASE Qurrent'-Event := ManyToneOh'; 955 UXR EndotDelay, B90 ELSE event := WanyToneOh'; 955 UXR EndotDelay, B91 Event := WanyToneOh'; 955 VXR EndotDelay, B92 Event := WanyToneOh'; 956 VXR EndotDelay,				
B88 Writein; 943 BEGIN B90 IF Current = NLL THEN 944 Terminal(TN) := TRANSMIT; B90 BEGIN 945 Append(TN, Time, 0); B90 BEGIN 945 CreatEdvent(Receive, Time+Fraction, 0, LL); B91 FG Current = NLL THEN 945 Append(TN, Time, 0); B92 END 945 CreatEdvent(Receive, Time+Fraction, 0, LL); B93 EGIN 945 CreatEdvent(Receive, Time+Fraction, 0, LL); B94 KHITELK(FewntList is emptyl (WriteList)'); 945 CreatEdvent(Receive, Time+Fraction, 0, LL); B95 CASE Current - F.Vent 10 951 END B96 CASE Current'-F.Vent 11 States (States (
289 if Ourrent = NIL THEN 944 Terminal(TM) := TARABNIT; 290 BEGIN 945 Append(TM, Time, 0); 291 WRITELK("EventList is emptyl (WriteList)"); 946 CreateEvent(Basylonedon, Time+Fraction, 0, LL); 293 ELSE 946 DelsyArray(TN) := DelsyArray(TN) := TelsyNT; 893 BEGIN 946 DelsyArray(TN) := DelsyArray(TN) := TelsyNT; 894 Maile (Current - Kvent Of 956 END 895 Rearival : Event := 'Rearival'; 952 END 906 BusyToneOF : Event := 'BusyToneOff'; 955 VAR Terminal:Term#r; VAR BusyToneOff'; 907 Exes (WRITELK("ERROR In EventList (Shoulst):TITI'); 957 VAR Torminal:Term#r; VAR BusyToneOff'; 907 Exes (WRITELK("ERROR In EventList (Shoulst): Current'.TerminalNo); 957 InterArr : Extended; 907 Current := Current'.NextNode; 958 Tore = FirstNode'.Time; 907 EVEN Current'.Terminal:', Current'.TerminalNo); 950 Time; 908 Read(n; Terminal:', Current'.TerminalNo); 951 Time; 907 Current': NextNode; 956 Time; 111terArr : Extended; 907 Current :- Vertick(FirstNode:ListPointer; VAR BusyTone:OnOff; 957 Time; 907				
BEGIN965Append(TN, Time, 0);801WRITELW('EventList is empty! (WriteList)');966CreatEvent(BusyToneOn, Time+Traction, D, LL);802END967CreatEvent(BusyToneOn, Time+Traction, TN, LL);803WHILE (Current ~ NIL) DO969END804WHILE (Current ~ Nil) DO960END805CASE Current'.Event Of950END806CASE Current'.Event Of950END807BusyToneOn : Event := 'BusyToneOn';955951808CASE Current'.Event::= 'BusyToneOn';955VAR Terminal:TermArr; VAR BusyToneOn';809BusyToneOf : Event := 'BusyToneOn';955VAR Terminal:TermArr; VAR BusyToneOrf;809BusyToneOf : Event := 'BusyToneOff';955VAR BusyToneOnceDoff;800EUK MITELW('Hode:',i,' Event::Pacetive'; 'Terminal:',Current'.Terminalko;957VAR BusyToneOnceDoff;801EVENCEEvent::Pacetive'; 'Terminal:',Current'.Terminalko;957VAR BusyToneOnceDoff;803END;957VARBusyToneOnceDoff;804EVENCEEvent::Pacetive:Faceti				
b) WiTELW('EventList is empty! (WriteList)'); 64 CreateEvent(Receive, Time:H=ration, 0, LL); 873 ELSE 64 DelayArray[TN] := DelayArray[TN] + 1 + Fraction; 874 WHILE (Current ↔ NL) D0 64 END 875 ELSE 64 DelayArray[TN] := DelayArray[TN] + 1 + Fraction; 876 CASE Current '-Event OF 640 END 877 Rearing : Sean Sean 878 Rearing : Sean Sean 879 BusyToneOF : Event := 'Arrival'; Sean Sean 870 Reacive : Event := 'Arrival'; Sean Sean Sean 870 BusyToneOF : Event := 'BusyToneOff'; Sean Sean <td></td> <td></td> <td></td> <td></td>				
B02END947CreateSventReceive, Time+1+Fraction, TW, LL);894WHILE (Current > NIL) D0948DelayArray(TN) := DelayArray(TN) = DelayArray(TN) + 1 + Fraction;895BESIN950END896CASE Current : Event OF950END897Arrival : Event := 'Arrival';952898Bestfondoff : Event := 'BayGroupOff';954800Bestfondoff : Event := 'BayGroupOff';955901Receive : Event := 'Neventive,'955902ELSE WRITELW('KoRo in EventList (ShowList)!!!!');957903END;957904WIRTELM('Kode:',i,' Event:', EventLi', Event				
Big ELSE 948 DelayArray[TN] := DelayArray[TN] + 1' + Fraction; Big Construct ← WIL) DD 96 DelayArray[TN] := DelayArray[TN] + 1' + Fraction; Big Construct ← WIL) DD 96 END Big Construct ← WIL] DD 96 END Big Construct ← WIL; Event := 'Arrival'; 95 END Big Construct ← WIL; Event := 'Neckrival'; 95 END Big Construct ← WIL; Event := 'Neckrival'; 95 END Big Construct ← WIL; Event := 'Neckrival'; 95 END Big Construct ← WIL; Event := 'Neckrival'; 95 WR TENUE (Mode:ListPointer; WR BusyTone:OnOff; Big Construct ← Time:A, 'Second + Event := 'Neckrival'; 95 NR Arrival WILL:NeckList; Big Construct := Current '.NextNode; Time:', 'Second + Event := 'Neckrival'; 95 Big Construct := Current '.NextNode; Gendomoleay; 96 Time := FirstNode'.Time; Big EastNot 96 Time := FirstNode'.Time; 96 Big Construct := Current '.NextNode; 96 Time := FirstNode'.Time; 96 Big Construct := Current '.NextNode; 96 Time := FirstNode'.Time; 96 Big Construct := Current '.NextNode; 96 Time := F				
00/2 WillE (Current *> NiL) D0 949 END 055 BEGIN 950 END 056 CASE Current *: Event GF 951 END 057 Arrival : Event := 'Nerrival'; 952 058 Readon := Event := 'Newrival'; 952 050 BusyToneOff :: Event := 'Newrival'; 952 051 Receive := Event := 'Newrival'; 954 052 ELSE WRITELM('FERGE in Eventis' (Shoutist)!!!!'); 955 053 ELSE WRITELM('Node:', i, ' Event:', Event': Event': Terminal', Current'. Terminalk0; 054 KuritELM('Node:', i, ' Terminal', Current'. Terminalk0; 055 Current : Time:', ' 956 END 056 Current : Time:', ' 957 VAR 057 Current : Time:', ' 957 VAR 058 Readury 11ne; ' 11ne; ' 059 END; 963 ECOL := FirstWode'.Time; 051 Current'. NextNode; 964 Time := FirstWode'.Time; 951 PROCEDUKE ProcessArrival (firstWode:ListPointer; VAR BusyTone:Onff; 964 Time := FirstWode'.Time; 951 Rocadur; 964 Time := FirstWode'.Time; 951 Rocadur; 967 EEON 951<				
BEGIN 590 END 896 CAE Current'.Event 0F 591 897 Arrival : Event := "Arrival'; 592 898 ReArrival : Event := "ReArrival'; 593 899 BusyToneOf : Event := "BusyToneOf'; 594 800 Editation 595 801 Readrive := BusyToneOf'; 595 802 Editation 594 803 BusyToneOf : Event := "BusyToneOf'; 595 804 Readrive := SusyToneOf'; 595 805 Editation 596 806 Current'.HanceAction 595 807 Editation 596 808 Formaliation 596 809 Editation 596 800 Editation 596 801 Formaliation 597 802 Editation 596 803 Clock := FirstNode: Time; 597 804 Formaliation 596 805 Formaliation 596 806 Formaliation 597 807 Current': NextNode; 597 908 Formaliation 598 909 END; 596 901 Foreatefine 596				
CASE Current'.Event OFPSI807Arrival : Event := 'Arrival';PSI808ReArrival : Event := 'Arrival';PSI809BusyToneOFF : Event := 'BusyToneOff';PSI801Receive : Event := 'BusyToneOff';PSI802ELSE WRITELN('ERROR in Event := 'Receive';PSI803Redrive : Event := 'Receive';PSI804Freminal:TermArr; VAR LL:LinkedList;805Current'.Imer;PSI806Freminal:Y, Event::Event::E,'Time;807Functi::Eurent'.NextNode;PSI808Current'.Imer;PSI809EUS: WRITELN('ERROR in Event::E,'Time;',804Functi::Eurent'.NextNode;PSI805Current'.Imer;',PSI806Functi::Eurent'.NextNode;PSI807Functi::Eurent'.NextNode;PSI808EEGNPSI809Functi::Eurent'.NextNode;PSI809Functi::Eurent'.NextNode;PSI809Functi::Eurent'.NextNode;PSI800EEGNPSI801Functi::Eurent'.NextNode;PSI802Functi::Eurent'.NextNode;PSI803EEGNPSI804Functi::Eurent'.NextNode;PSI804Functi::Eurent'.NextNode;PSI805Functi::Eurent'.NextNode;PSI806Functi::Eurent'.NextNode;PSI807Functi::Eurent'.NextNode;PSI808Featurent'.Functi::Eurent'.NextNode;PSI<	894	WHILE (Current <> NIL) DO		
Boy Boy Boy Boy Boy Boy Boy 	895	BEGIN		
Box Box BusyToneOFF : Event := 'ReArrival';953 (************************************	896	CASE Current [°] .Event OF	951	END;{ProcessArrival}
300BusyToneOn : Event := FlusyToneOn';954 FROCEDURE ProcessReArrival(FirstNode:ListPointer; VAR BusyTone:OnOff;900BusyToneOff : Event := PlusyToneOff;955VAR Terminal:TermArr; VAR LL:LinkedList;901Receive : Event := PlusyToneOff;955VAR Q:QueueType);902ELSE WATELM('ERROR in EventList (ShowList)!!!!');957VAR Q:QueueType);903END;Current :Ine:4,' Terminal:',Current '.TerminalNo);958ReadomDelay,905Current :Ine:4,' Terminal:',Current '.TerminalNo);960TInterArr : Extended;906i:=i1;961Clock := FirstNode'.Time;962907Current :- NextNode;962EEGIN908ReadIn;963Clock := FirstNode'.Time;909END;964TN := FirstNode'.Time;909END;964TN := FirstNode'.Time;909END;965Clock := FirstNode'.Time;909END;966IE BusyTone = ON911ReadIn;967TIme912END;CNOURE ProcessArrival(FirstNode:ListPointer; VAR BusyTone:OnOff;967913FROCEDURE ProcessArrival(FirstNode:ListPointer; VAR BusyTone:OnOff;976914CreateFvent (Rehrival, Time + RandomDelay, TN, LL);976915FROCEDURE ProcessArrival(FirstNode:ListPointer; VAR BusyTone:OnOff;977916VAR G:QueuFype);973EEGIN917VAR G:QueuFype);974CreateFvent(Reatrival, Time + RandomDelay, TN, LL);918FirstNode'.Time;975BEGIN<	897	Arrival : Event := 'Arrival';		
900BusyTomeOFF : Event := FlexelyTomeOff;955VAR Deminal:TermArr; VAR LL:LinkedList;901Receive : Event := Receive;955VAR Q:QueueType);903ELSE WAITELM('HERRG in EventList (ShowList)!!!!');957VAR904WRITELM('Hode:',i,' Event:',Event:12,' Time:',957VAR905Current'.Time:4,' Terminal:',Current'.TerminalN0;950Time,906i:= i+1;951Current'.Time:4,' Terminal:',Current'.TerminalN0;953907Current := Current'.NextNode;963Clock := FirstNode'.Time;908ReadIn;963Clock := FirstNode'.Time;909END;964TN := FirstNode'.Time;901WriteLn('Hod of List');964TN := FirstNode'.Time;912KeadIn;965Time := FirstNode'.Time;913914(************************************	898	ReArrival : Event := 'ReArrival';	953	{*************************************
OnlReceive: Event := 'Receive';956VAR G:QueueType);002ELSE WRITELK('ERROR in EventList (ShowList)!!!');957 VAR003END;Current'.Time:4,' Terminal:*,Current',TerminalNo);958 RandomDelay,005Current'.Time:4,' Terminal:*,Current',TerminalNo);960 TinterArr : Extended;006i := 1:1;958 Cacchar007Current: -NextNode;962 EEGIN08Readin;963 Clock := FirstNode'.Time;090END;964 TN := FirstNode'.Time;091WriteLn('End of list');965 Time := FirstNode'.Time;092END;966 Time := FirstNode'.Time;093Keadin;966 Time := FirstNode'.Time;094END;966 Time := FirstNode'.Time;095FROCEDURE ProcessArrival(FirstNode:ListPrinter; VAR BusyTone:DnOff;970 CreateEvent (ReArrival, Time + RandomDelay, TN, LL);194VAR erminal:TermArr; VAR LL:LinkedList;971 DelayArray(TN] := DelayArray(TN] + RandomDelay;195PROCEDURE ProcessArrival(FirstNode'.Time;973 END198VAR e:QueueType);974 ELSE209Time, := FirstNode'.Time;975 EGIN209Time;976 CreateEvent(Rearrival, Time + Farction, 0, LL);219Clock := FirstNode'.Time;971 CreateEvent(Receive, Time+I+Fraction, 0, LL);220Time := FirstNode'.Time;972 CreateEvent(Kaceive, Time+I+Fraction, TN, LL);231Clock := FirstNode'.Time;981 END232Time := FirstNode'.Time;981 END233CreateEvent(Karceive, Time+I+Fracti	899	BusyToneOn : Event := 'BusyToneOn';	954	PROCEDURE ProcessReArrival(FirstNode:ListPointer; VAR BusyTone:OnOff;
901 Receive : Event := 'Receive'; 956 VAR 0:GueueType); 902 ELSE WITELKUYTERROR in EventList (Stoulist)!!!!!'); 957 VAR 903 END; 958 RandomDelay, 905 Current '.Time:4,' Terminal:/,Current '.TerminalNo); 950 Time, 906 i:=11; 951 Time, 907 Current '.NextNode; 952 EEGIN 908 ReadIn; 952 FirstNode'.TerminalNo; 909 END; 954 Time := FirstNode'.Time; 909 END; 954 Time := FirstNode'.Time; 911 ReadIn; 956 Time := FirstNode'.Time; 912 END;(ShowList) 967 THEN 913 ProceEDURE ProcessArrival(FirstNode:ListPointer; VAR BusyTone:DnOff; 970 CreateEvent (ReArrival, Time + RandomDelay, TN, LL); 914 VAR Q:QueueType); 972 Inc(Gcount); ELSE 915 PROCEDURE ProcessArrival(FirstNode:Time; 974 ELSE 913 RandomDelay, 974 ELSE	900	<pre>BusyToneOFF : Event := 'BusyToneOff';</pre>	955	VAR Terminal:TermArr; VAR LL:LinkedList;
902 ELSE WRITELN('ERROR in EventList (ShowList)!!!!'); 957 VAR 903 END; 958 RandomDelay, 904 WRITELN('Node:',i,' Event:',Event:12,' Time:', 958 RandomDelay, 905 Current.Time:4,' Terminal:',Current'.TerminalNo); 950 Time, 906 I:= i=i1; Current'.NextNode; 950 Time, 907 Current'.NextNode; 963 Clock := FirstNode'.Time; 908 ReadLn; 963 Time, := FirstNode'.Time; 909 FIG. 964 TN := FirstNode'.Time; 909 END; 964 TN := FirstNode'.Time; 909 FIG. 965 Time, := FirstNode'.Time; 965 Time, := FirstNode'.Time; 908 VAR Terminal:T			956	VAR Q:QueueType);
903 END; 958 RandomDelay, 904 KRITELK/Node:',i,' Event:', Event:12,' Time,' 905 Current.Time:4,' Terminal:', Current'.TerminalNo); 960 Time, 905 Current.Time:4,' Terminal:', Current'.TerminalNo); 960 Time, 907 Current := Current'.NextNode; 962 EEGIN 908 ReadIn; 962 EEGIN 909 END; 964 Time := FirstNode'.Time; 909 END; 964 Time := FirstNode'.Time; 901 WriteLn('End of list'); 965 Time := FirstNode'.Time; 913 ReadIn; 966 IF BusyTone = ON 914 Kernerminal:Terninal:			957	VAR
904WRITELN('Wode:',i,' Event:',Event:',Event:',Event:',Event:',Event:', Second Sec			958	RandomDelay,
005Current'.Time:4,'Terminal:',Current'.TerminalNo);960TinterArr : Extended;9061 Integer;961TN: Integer;907Current := Current'.NextNode;963Clock := FirstNode'.Time;908ReadLn;964TN:= FirstNode'.Time;909END;965Time := FirstNode'.Time;910WriteLn('End of list');965Time := FirstNode'.Time;911ReadLn;965Time := FirstNode'.Time;912EDD;(ShowList)967THEN913967THEN914(************************************			959	
905i = i=i;961TN: Integer;907Current := Current '.NextNode;962BEGIN908ReadLn;963Clock := FirstNode'.Time;909END;964TN:= FirstNode'.Time;909END;964Time := FirstNode'.Time;901WriteLn('End of list');964Time := FirstNode'.Time;911ReadLn;965Time := FirstNode'.Time;912END; (ShouList)966IF BusyTone = ON913968EEGIN914(************************************				
907Current'.NextNode;962BEGIN908ReadLn;963Clock := FirstNode'.Time;909WriteLn('End of List');964TN := FirstNode'.Time;910WriteLn('End of List');965Time := FirstNode'.Time;911ReadLn;965Time use FirstNode'.Time;912END; (ShowList)966The WayTone = ON913914(************************************				
963Clock := FirstNode'.Time;909END;963Clock := FirstNode'.Time;909END;964TN := FirstNode'.Time;909END;965Time := FirstNode'.Time;901ReadLn;965Time := FirstNode'.Time;911ReadLn;966IF BusyTone = ON912END; ShouList)967THEN913968BEGIN968914(************************************				• /
900END;"964TN:= FirstNode*.TerminalNo;910WriteLn('End of list');965Time := FirstNode*.Time;911ReadIn;965Time := FirstNode*.Time;912END;(ShowList)967THEN914(************************************				
910WriteLn('End of list'); ReadLn;965Time := FirstNode'.Time; 966911ReadLn;966IF BusyTone = ON912END;(ShowList)967THEN913968BEGIN914(************************************				
911ReadLn;966IF BusyTone = ON912END;(ShowList)967THEN913914(************************************				
912END;(ShowList)967THEN913968BEGIN914(************************************				
913968BEGIN914 (************************************				
914 (************************************		END;{SHOWLIST		
915 PROCEDURE ProcessArrival(FirstNode:ListPointer; VAR BusyTone:DnOff;970CreateEvent (ReArrival, Time + RandomDelay, TN, LL);916VAR Terminal:TermArr; VAR LL:LinkedList;971DelayArray[TN] := DelayArray[TN] + RandomDelay;917VAR Q:QueueType);972Inc(GCount);918 VAR973END919 RandomDelay,974ELSE920 Time,975BEGIN921 TInterArr : Extended;976Terminal[TN] := TRANSMIT;922 TN: Integer;977Append(TN, Time, Q);923 BEGIN978CreateEvent(Receive, Time+Fraction, 0, LL);924 Clock := FirstNode^.Time;979CreateEvent(Receive, Time+1+Fraction, TN, LL);925 TN:= FirstNode^.Time;980DelayArray[TN] := DelayArray[TN] + 1 + Fraction;926 Time := firstNode^.Time;981END927 TInterArr := -(ln(1-DUNI))/PArrH;982END;(ProcessReArrival)928 CreateEvent(Arrival, Time + TInterArr, TN, LL);983929 IF Terminal[TN] = IDLE984 (************************************	915	·····		
916VAR Terminal:TermArr; VAR LL:LinkedList;971DelayArray[TN] := DelayArray[TN] + RandomDelay;917VAR Q:QueueType);972Inc(GCount);918 VAR973END919RandomDelay,974ELSE920Time,975BEGIN921TInterArr : Extended;976Terminal[TN] := TRANSMIT;922TN: Integer;977Append(TN, Time, Q);923BEGIN978CreateEvent(BusyToneOn, Time+Fraction, O, LL);924Clock := FirstNode^.TerminalNo;979CreateEvent(Receive, Time+1+Fraction, TN, LL);925TN:= FirstNode^.Time;981END927TInterArr := -(In(1-DUNI))/PArrH;982END;{ProcessReArrival}929THEN984(************************************				
917VAR Q:QueueType);972Inc(GCount);918 VAR973END919 RandomDelay,974ELSE920 Time,975BEGIN921 TInterArr : Extended;976Terminal[TN] := TRANSMIT;922 TN: Integer;977Append(TN, Time, Q);923 EEGIN978CreateEvent(BusyToneOn, Time+Fraction, O, LL);924 Clock:= FirstNode*.Time;979CreateEvent(Receive, Time+1+Fraction, TN, LL);925 TN:= FirstNode*.Time;981END927 TInterArr := -(ln(1-DUNI))/PArrH;982END;(ProcessReArrival)928 CreateEvent(Arrival, Time + TInterArr, TN, LL);983984929 TH Terminal[TN] := BLOCKED;985PROCEDURE ProcessBusyToneOn(FirstNode:ListPointer; VAR BusyTone:OnOff;933 Inc(ArrivalCount);988Clock := FirstNode*.Time;				
918 VAR973END919RandomDelay,974ELSE920Time,975BEGIN921TInterArr: Extended;977Append(TN, Time, Q);922TN: Integer;977Append(TN, Time, Q);923BEGIN978CreateEvent(BusyToneOn, Time+Fraction, O, LL);924Clock:= FirstNode*.Time;977CreateEvent(Receive, Time+Fraction, TN, LL);925TN:= FirstNode*.Time;980DelayArray[TN] := DelayArray[TN] + 1 + Fraction;926Time:= FirstNode*.Time;981END927TInterArr: := -(In(1-DUNI))/PArrH;982END;(ProcessReArrival)928CreateEvent(Arrival, Time + TInterArr, TN, LL);983929IF Terminal[TN] = IDLE984(************************************				
Normalize974ELSE920Time,975BEGIN921TInterArr : Extended;976Terminal(TN] := TRANSMIT;922TN: Integer;977Append(TN, Time, Q);923BEGIN978CreateEvent(BusyToneOn, Time+Fraction, O, LL);924Clock:= FirstNode^.TerminalNo;979CreateEvent(Receive, Time+1+Fraction, TN, LL);925TN:= FirstNode^.TerminalNo;980DelayArray[TN] := DelayArray[TN] + 1 + Fraction;926Time:= FirstNode^.Time;981END927TInterArr := -(In(1-DUNI))/PArrH;982END; (ProcessReArrival)929IF Terminal[TN] = IDLE983984(************************************				
920Time,975BEGIN921TInterArr : Extended;976Terminal[TN] := TRANSMIT;922TN: Integer;977Append(TN, Time, Q);923BEGIN977Append(TN, Time, Q);924Clock:= FirstNode^.Time;979CreateEvent(BusyToneOn, Time+Fraction, O, LL);925TN:= FirstNode^.TerminalNo;979CreateEvent(Receive, Time+1+Fraction, TN, LL);925TN:= FirstNode^.Time;980DelayArray[TN] := DelayArray[TN] + 1 + Fraction;926Time:= FirstNode^.Time;981END927TInterArr := -(ln(1-DUNI))/PArrH;982END;{ProcessReArrival}929IF Terminal[TN] = IDLE984(************************************				
921TinterArr : Extended;976Terminal[TN] := TRANSMIT;922TN: Integer;977Append(TN, Time, Q);923BEGIN978CreateEvent(BusyToneOn, Time+Fraction, O, LL);924Clock:= FirstNode^.TerminalNo;979CreateEvent(Receive, Time+1+Fraction, TN, LL);925TN:= FirstNode^.Time;980DelayArray[TN] := DelayArray[TN] + 1 + Fraction;926Time:= FirstNode^.Time;981END927TInterArr := -(ln(1-DUNI))/PArrH;982END;{ProcessReArrival}928CreateEvent(Arrival, Time + TInterArr, TN, LL);983929IF Terminal[TN] = IDLE984 (************************************				
922TN: Integer;977Append(TN, Time, Q);923BEGIN978CreateEvent(BusyToneOn, Time+Fraction, 0, LL);924Clock:= FirstNode^.TerminalNo;979CreateEvent(Receive, Time+1+Fraction, TN, LL);925TN:= FirstNode^.TerminalNo;980DelayArray[TN] := DelayArray[TN] + 1 + Fraction;926Time:= FirstNode^.Time;981END927TInterArr := -(ln(1-DUNI))/PArrH;982END;{ProcessReArrival}928CreateEvent(Arrival, Time + TInterArr, TN, LL);983929IF Terminal[TN] = IDLE984 {************************************		Time,		
923BEGIN978CreateEvent(BusyToneOn, Time+Fraction, 0, LL);924Clock := FirstNode^.Time;979CreateEvent(Receive, Time+1+Fraction, TN, LL);925TN := FirstNode^.TerminalNo;980DelayArray[TN] := DelayArray[TN] + 1 + Fraction;926Time := FirstNode^.Time;981END927TInterArr := -(ln(1-DUNI))/PArrH;982END;{ProcessReArrival}928CreateEvent(Arrival, Time + TInterArr, TN, LL);983929IF Terminal[TN] = IDLE984{************************************		Reservation relationships and a special less and		
924Clock := FirstNode^.Time;979CreateEvent(Receive, Time+1+Fraction, TN, LL);925TN := FirstNode^.TerminalNo;980DelayArray[TN] := DelayArray[TN] + 1 + Fraction;926Time := FirstNode^.Time;981END927TInterArr := -(ln(1-DUNI))/PArrH;982END;{ProcessReArrival}928CreateEvent(Arrival, Time + TInterArr, TN, LL);983929IF Terminal[TN] = IDLE984 (************************************		· ·		
925TN:= FirstNode^.TerminalNo;980DelayArray[TN] := DelayArray[TN] + 1 + Fraction;926Time:= FirstNode^.Time;981END927TInterArr := -(ln(1-DUNI))/PArrH;982END;{ProcessReArrival}928CreateEvent(Arrival, Time + TInterArr, TN, LL);983929IF Terminal[TN] = IDLE984 {************************************	923	BEGIN		
926Time:= FirstNode^.Time;981END927TInterArr := -(ln(1-DUNI))/PArrH;982END;{ProcessReArrival}928CreateEvent(Arrival, Time + TInterArr, TN, LL);983929IF Terminal[TN] = IDLE984 {************************************	924	Clock := FirstNode^.Time;		
927TInterArr := -(ln(1-DUNI))/PArrH;982 END;{ProcessReArrival}928CreateEvent(Arrival, Time + TInterArr, TN, LL);983929IF Terminal[TN] = IDLE984 {************************************	925	TN := FirstNode^.TerminalNo;	980	DelayArray[TN] := DelayArray[TN] + 1 + Fraction;
928CreateEvent(Arrival, Time + TInterArr, TN, LL);983929IF Terminal[TN] = IDLE984 (************************************	926	Time := FirstNode^.Time;	981	END
929 IF Terminal[TN] = IDLE 984 {************************************	927	TInterArr := -(ln(1-DUNI))/PArrH;	982	END;{ProcessReArrival}
929 IF Terminal[TN] = IDLE 984 (************************************	928	CreateEvent(Arrival, Time + TInterArr, TN, LL);		
930 THEN 985 PROCEDURE ProcessBusyToneOn(FirstNode:ListPointer; VAR BusyTone:OnOff; 931 BEGIN 986 VAR LL:LinkedList); 932 Terminal[TN] := BLOCKED; 987 BEGIN 933 Inc(ArrivalCount); 988 Clock := FirstNode^.Time;			984	{*************************************
931 BEGIN 986 VAR LL:LinkedList); 932 Terminal[TN] := BLOCKED; 987 BEGIN 933 Inc(ArrivalCount); 988 Clock := FirstNode^.Time;			985	PROCEDURE ProcessBusyToneOn(FirstNode:ListPointer; VAR BusyTone:OnOff;
932Terminal[TN] := BLOCKED;987 BEGIN933Inc(ArrivalCount);988Clock := FirstNode^.Time;				
933 Inc(ArrivalCount); 988 Clock := FirstNode [*] .Time;			987	
	934	IF BusyTone = ON	989	BusyTone := ON;
935 THEN 990 END; {ProcessBusyToneOn}			990	

Page 10, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes.

991		1046 Inc(SuccessCount):
	{**************************************	1046 Inc(SuccessCount); 1047 Terminal[TN] := IDLE;
	PROCEDURE ProcessBusyToneOff (FirstNode:ListPointer; VAR BusyTone:OnOff);	1048 DelayCount := DelayCount + DelayArray[TN];
	BEGIN	1049 DelayArray[TN] := 0;
995	Clock := FirstNode^.Time;	1050 END
996	BusyTone := OFF;	1051 ELSE
	END;{ProcessBusyToneOff}	1052 BEGIN
998		1053 Terminal[TN] := BLOCKED;
999	{**************************************	1054 RandomDelay := -(ln(1-DUNI))/Alpha;
1000	PROCEDURE ProcessReceive(FirstNode:ListPointer; VAR BusyTone:OnOff;	1055 CreateEvent(ReArrival, Time+RandomDelay, TN, LL);
1001	VAR Terminal:TermArr; VAR LL:LinkedList;	1056 DelayArray[TN] := DelayArray[TN] + RandomDelay;
1002	VAR Q:QueueType);	1057 Inc(GCount);
1003	VAR	1058 END
1004	X2 : QueueEntry;	1059 END
1005		1060 END;{ProcessReceive}
1006	ΤΝ,	1061
1007	TN2 : Integer;	1062 {************************************
1008		1063 PROCEDURE ProcessFirstNode(FirstNode:ListPointer; VAR LL:LinkedList;
1009	RandomDelay,	1064 VAR Q:QueueType);
1010	Time,	1065
1011	Time2,	1066 BEGIN
1012		1067 CASE FirstNode .Event OF
1013	Psp : Extended;	1068 Arrival : ProcessArrival(FirstNode, BusyTone, Terminal, LL, Q);
1014	PEOL	1069 ReArrival : ProcessReArrival(FirstNode, BusyTone, Terminal, LL, Q);
	BEGIN	1070 BusyToneOn : ProcessBusyToneOn(FirstNode, BusyTone, LL);
1016	Clock := FirstNode ² .Time;	1071 BusyToneOff : ProcessBusyToneOff(FirstNode, BusyTone); 1072 Receive : ProcessReceive(FirstNode, BusyTone, Terminal, LL, Q);
1017 1018	Time := FirstNode [*] .Time;	
1018	IF QueueEmpty(Q) = TRUE THEN	1073 ELSE WRITELN('ERROR in EventList (ProcessFirstNode)!!!!!'); 1074 END;
1020		1075
1020	TN := FirstNode^.TerminalNo;	1075 END;{ProcessFirstNode}
1022	RandomDelay := -(ln(1-DUNI))/Alpha;	1077
1023	Terminal[TN] := BLOCKED;	1078 {************************************
1024	CreateEvent(ReArrival, Time+RandomDelay, TN, LL);	1079 PROCEDURE AdvanceNextEvent(VAR LL:LinkedList; VAR Q:QueueType);
1025	<pre>DelayArray[TN] := DelayArray[TN] + RandomDelay;</pre>	1080 VAR
1026	Inc(GCount);	1081 FirstNode : ListPointer;
1027	END	1082 BEGIN
1028	ELSE	1083 IF LL.head = NIL
1029	BEGIN	1084 THEN
1030	TN := FirstNode [^] .TerminalNo;	1085 BEGIN
1031	Terminal[TN] := INTERFERE;	1086 WRITELN('Error: EventList is empty (AdvanceNextEvent)');
1032	WHILE NOT QueueEmpty(Q) DO	1087 END
1033	BEGIN	1088 ELSE IF LL.head^.NextNode = NIL
1034	Remove(X2, Q);	1089 THEN
1035	TN2 := X2.TerminalNo;	1090 BEGIN
1036	Time2 := X2.Time;	1091 FirstNode := LL.head;
1037	Terminal[TN2] := INTERFERE;	1092 LL.head := NIL;
1038	END;	1093 ProcessFirstNode(FirstNode, LL, Q);
1039	Pbe := CalculatePbe(TN);	1094 END
1040	<pre>Psp := Power(1-Pbe, PacketLength);</pre>	1095 ELSE
1041	CreateEvent(BusyToneOff, Time2+1+Fraction, 0, LL);	1096 BEGIN
1042		1097 FirstNode := LL.head; 1098 LL.head := LL.head^.NextNode:
1043 1044	IF DUNI < PSp THEN	
1044	BEGIN	1099 ProcessFirstNode(FirstNode, LL, Q); 1100 END;
1045		

Page 11, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes.

1101 Dispose(FirstNode);	1156 FileName := 'V2H8W3.DAT';
1102 END;{AdvanceNextEvent}	1157 Assign(OutFile, FileName);
1103	1158
1104 (************************************	1159 ReWrite(OutFile);
1105 PROCEDURE InitEventList(VAR LL:LinkedList);	1160 { WriteLn(OutFile,'Size of group : ', GroupSize);
1106 VAR	1161 WriteLn(OutFile, Number of groups: ', NumberOfLines);
1107 i : Integer;	1162 WriteLn(OutFile);
1108 Time : Extended;	1163 WriteLn(OutFile,'Inhibit delay fraction: ', Fraction);
1109 BEGIN	1164 WriteLn(OutFile,'Alpha : ',Alpha);
	1166 WriteLn(OutFile);
1112 Time := -(ln(1-DUNI))/PArrH;	1167 }
1113 CreateEvent(Arrival, Time, i, LL);	1168 WriteLn(OutFile, 'G ':17,
1114 END	1169 's ':16,
1115 END;{InitEventList}	1170 'D ':16);
1116	1171 END;{OpenResultFile}
1117 (**********************************	1172
1118 FUNCTION CountList(VAR LL:LinkedList):Integer;	1173 {************************************
1119 VAR	1174 (WarnBeep emits a half second warning tone and then waits 4.5 seconds }
1120 i : Integer;	1175 { }
1121 Current : ListPointer;	1176 { Depends on: none }
1122 BEGIN	
1123 i := 0;	1178 { Modifies : pfn } 1179 {************************************
1124 Current := LL.head;	
1125 WHILE Current <> NIL DO	1180 Procedure WarnBeep;
1126 BEGIN	1181 VAR
1127 Current := Current ^.NextNode;	1182 i : Integer;
1128 i := i + 1;	1183 BEGIN
1129 END;	1184 FOR i := 1 TO 5 DO
1130 CountList := i;	1185 BEGIN
1131 END;{CountList}	1186 Sound(500+i*20);
1132	1187 Delay(300);
1133 (**********************************	1188 END;
1134 PROCEDURE RemoveList(VAR LL:LinkedList);	1189 NoSound;
1135 VAR	1190 Delay(1000);
1136 Current : ListPointer;	1191 End; {WarnBeep}
1137 BEGIN	1192
1138 Current := LL.head;	1193 (************************************
1139 WHILE Current <> NIL DO	1194 { MAINLINE }
1140 BEGIN	1195 {************************************
1141 Current := Current [^] .NextNode;	1196
1142 Dispose(LL.head);	1197 Begin
1143 LL.head := Current;	1198 ClrScr;
1144 END;	1199 WriteLn('Output Testprogramma:');
1145 END; (RemoveList)	1200
1146	1201 {Initialize variables}
1147 {************************************	1202 Rreceive := 30;
Contraction and the second s	1203 Rterm := 5;
1148 PROCEDURE OpenResultFile(VAR OutFile:Text);	
1149 VAR	1204 SNR := 6;
1150 GS,	1205
1151 NL,	1206 Nt := 32;
1152 FileName : String;	1207 NumberOfLines := 4;
1153 BEGIN	1208 GroupSize := 8;
1154 Str(GroupSize, GS);	1209
1155 Str(NumberOfLines, NL);	1210 OpenResultFile(ResultFile);

Page 12, listing of V2SIMUN1.PAS, date is 30-10-95, file date is 01-10-95, size is 53363 bytes.

		10//	
1211	InitCodes(CCat, GroupSize, NumberOfLines);	1266	END;{Warm up loop}
1212	InitPathGain(S, GroupSize, NumberOfLines, Rreceive, Rterm, LossExp);	1267	
1213		1268	{Reset event counters}
1214	FOR i := 0 TO NumSim DO { Loop over terminal activity levels }	1269	ArrivalCount := 0;
1215	BEGIN	1270	SuccessCount := 0;
1216	{Set terminal activity}	1271	GCount := 0;
1217	<pre>PArrH := Power(10, -(LRangeMin + i*(LRangeMax-LRangeMin)/NumSim));</pre>	1272	DelayCount := 0;
1218	rInit(3647523, 65321); {Initialize FSU Ultra random generator}	1273	
1219	InitTerm(Terminal); {Reset terminal states }	1274	{Main simulation Loop}
1220	InitBusyTone(BusyTone); {Set BusyTone to OFF }	1275	While Clock < EndTime DO
1221	CreateList(EventList);	1276	BEGIN
1222	CreateQueue(ReceiveQueue);	1277	AdvanceNextEvent(EventList, ReceiveQueue);
1223	<pre>InitEventList(EventList); {Initialize EventList }</pre>	1278	IF (Clock - OldClock) > Interval
1224	<pre>InitDelayArray(DelayArray); {Reset all terminal delays to zero }</pre>	1279	THEN
1225		1280	BEGIN
1226	{Show configuration to user}	1281	GoToXY(1,12);
1227	ClrScr;	1282	WriteLn('Arrivals :', ArrivalCount:8);
	WriteLn('Receiver Radius: ', Rreceive:7:2);	1283	WriteLn('Success :', SuccessCount:8);
1228		1284	WriteLn;
1229	WriteLn('Terminal Radius: ', Rterm:7:2);	1285	WriteLn('Delay :', DelayCount);
1230	WriteLn('Number of users: ', Nt:4);	1286	
1231	<pre>WriteLn('Size of group : ', GroupSize:4);</pre>		WriteLn('Clock :', Clock);
1232	WriteLn('Number of codes: ', NumberOfLines:4);	1287	ShowState;
1233	<pre>WriteLn('Code length : ', codelength:4);</pre>	1288	OldClock := Clock;
1234	<pre>WriteLn('Packet length : ', Packetlength:4);</pre>	1289	END;
1235	WriteLn;	1290	GoToXY(40,4);
1236	WriteLn('Terminal activity (PArrH) : ',PArrH:8:5);	1291	<pre>WriteLn('SIMULATING !!!!');</pre>
1237		1292	GoTOXY(40,6);
1238	{Reset event counters}	1293	Write('STATUS: ',i,'/',NumSim);
1239	ArrivalCount := 0;	1294	END;{Simulation Loop}
1240	SuccessCount := 0;	1295	
1241	DelayCount := 0;	1296	{Calculate values for Physical Traffic, Throughput and Delay}
1242	GCount := 0;	1297	Throughput := SuccessCount/(EndTime-WarmTime);
1243		1298	D := DelayCount/SuccessCount;
1244	{Warm up the network}	1299	<pre>G := (ArrivalCount+GCount)/(EndTime-WarmTime);</pre>
1245	Clock := 0;	1300	
1246	OldClock := 0;	1301	{Write simulation results in files for processing-purposes later on}
1247	While Clock < WarmTime DO	1302	WriteLn(ResultFile , G :16,
1248	BEGIN	1303	Throughput :16,
1249	AdvanceNextEvent(EventList, ReceiveQueue);	1304	D :16);
1250	IF (Clock - OldClock) > Interval	1305	
1251	THEN	1306	RemoveList(EventList);
1252	BEGIN	1307	Kemovel belleventer berg
		1308	END;{Terminal activity level loop}
1253	GOTOXY(1,12);	1309	END, (Terminiat activity level toop)
1254	<pre>WriteLn('Arrivals :', ArrivalCount:8);</pre>		
1255	<pre>WriteLn('Success :', SuccessCount:8);</pre>	1310	Close(ResultFile);
1256	WriteLn;	1311	
1257	WriteLn('Delay :', DelayCount);	1312	{WarnBeep;}
1258	WriteLn('Clock :', Clock);	1313	GoToXY(30,16);
1259	ShowState;	1314	Write('I AM READY MISTER LE !!!!!');
1260	OldClock := Clock;	1315	GoToXY(30,17);
1261	END;	1316	Write('Press ENTER to return to program.');
1262	GoToXY(40,4);	1317	{ReadLn;}
1263	WriteLn('SIMULATING !!!!');	1318 END.	
1264	GoTOXY(40,6);		
1265	Write('STATUS: ',i,'/',NumSim);		

Page 1, listing of V2_CALC1.PAS, date is 30-10-95, file date is 25-09-95, size is 39354 bytes.

6 5444	a sta alla alla alla alla alla alla alla	****	****							
2 {*	1 {************************************									
2 (*										
4 {*										
5 {*	a mathematical f		*}							
6 (*	a mathematicat i		*}							
7 {*	Most of the cons	nts, variables, functions and proc								
8 {*	originates from		*}							
9 (*	originates from	b van Koosmateri.	*}							
10 (*	The terminals ar	lustered around distributed recei	5							
11 (*		ibuted receivers are clustered ar								
12 (*		distance. The packets from the te								
13 (*		tributed receivers and sent to th								
14 {*		e lines can be seen as primitive								
15 {*		(a) the standard of the strength of the state of the s	*}							
16 {*	Most time parame	s are normalized to the PacketDur	ation. To have a *}							
17 (*	time parameter i	econds one needs to multiply by t	he PacketDuration *}							
18 {*			*}							
19 (*	Huy Linh Anh Le,	ıly 16 1994	*}							
20 {***	******	****	**************************							
21										
22 Prog	gram CalcUnslotte	nput, Output);								
23										
24 Uses	B Dos, Crt, Ul	x;								
25										
26 Cons		1000;								
27	MaxTerm	32; { Maximum number o								
28	XMax		of interference pckts }							
29	PacketLength	64; { Number of bits p	per packet }							
30 31	Pt	: 1; { Transmitter powe	er in Watts }							
32	LossExp	2; { Attenuation para								
33	LUSSEXP	{ far field mo								
34	с	299792458; { Speed of light i	-							
35	f	1700000000; { Transmitter cent								
36	lambda	c/f; { Signal wavelengt								
37	Camboda									
38	twoPi	2 * Pi; { Two times Pi	}							
39	Pi2	Pi * Pi; { Pi squared	5							
40										
41	Alpha	0.1; { x retransmission	per PacketDuration }							
42	Fraction	0.01; { Inhibi	t delay fraction (d) }							
43										
44	bitrate	256*1024; { Bits t	ransmitted per sec. }							
45	PacketDurati	: 64/bitrate; {Time ne	eded to send a Packet}							
46	maxcodelengt		m code size }							
47	CodeLength		of code }							
48	Tb	: 1/bitrate; { Bit du								
49	Tc	<pre>1/(codelength*bitrate); { Chip c</pre>								
50	Tm		m delay spread }							
51	MaxL	-1	m number of paths }							
52	L	Trunc(Tm/Tc)+1; { Number	of paths }							
53		20								
54	NumSim	= 20;								
55	LRangeMin	: 3;								

56	LRangeMax	= -2;			
57 58 Type 59 60 61 62 63 64 65 66 66 67 68 {	Matrix= ArPosArr= ArRArr= ArStringArr= ArTermArr= ArConfigArr= ArPpsArr= ArExtArr= Ar	ray[1MaxTe ray[1MaxTe ray[1MaxTe ray[1MaxTe ray[1MaxTe	rm, 1MaxL] rm] rm]	Of Extended; Of Extended; Of Extended; Of String; Of TermState; Of Integer; Of Extended; Of Extended; Of Extended;	
69	p2CRec = ^C	Rec;			3
70 71 72 73	CRec = Re En	cc :Array[-	maxcodelengthmaxcodele	ngth] Of ShortInt;	
74 75 76 77			erm, 1MaxTerm] Of p2CR	ec;	
78 79 Var	CCat	: CArr;	{ Catalog of code cross	correlations	}
80 81 82 83 84	Rreceive, Rterm, SNR	:Extended;	{ Receiver radius aroun { Terminal radius aroun { Signal to white Noise	d receivers	} } }
85 86 87 88	Nt, GroupSize, NumberOfLines	:Integer;	<pre>{ Number of terminals { Number of terminals p { Number of groups in t { also the number of</pre>	he system. This is	} } }
89 90 91	S	:Matrix;	{ Power loss ratio matr	ix	}
92 93	Terminal	:TermArr;	{ Terminal state inform	ation	}
94 95 96 97	X, i, j	:Integer;	{Loopvariable for termi	nal activity levels	s}
98 99	CountConfig StartConfig,	:LongInt;			
100 101 102	MaxConfig PpsAveXArr SArray	:ConfigArr; :PpsArr; :SArr;			
103 104 105	ResultFile	:Text;			
106 107 108 109 110	PpsAveX, PpsXZero, PConfl, PConflTemp, PNoConfl,				

Page 2, listing of V2_CALC1.PAS, date is 30-10-95, file date is 25-09-95, size is 39354 bytes.

123 124 125 126 127	<pre>{ Power calculates a { { Compared to be a compared to be compared to be a compared</pre>	: Extended; : ExtArr; ***********************************	***************************************	169 170 171 172 173 174 175 176 177 178 179 180 181 182	7 Be Be 99 11 22 33 44 55 66 77 88 99 00 11 22	INTERFERE End;{Case} End;{For i := 1 TO N	4);] Of : Write(' : Write(' : Write(' : Write('	i'); b'); t');	2 8901	32456	3 57890	.,
128		>=0	}			nd;{ShowState}						
	<pre>{ Pre : a >= 0 And b { Post: Power = a^b</pre>	>=0	}	184 185		*****	******	********	******	*******	*******	*****3
131	{*************************************	******	***************************************			R calculates a partial						}
	Function Power(a,b:Ex	tended):Extended;		187								}
133 134	Begin If a<>0					Depends on: config, CCa Calls : none	at, Tc					}
134	Then					Modifies : none						}
136	power := Exp	(b*Ln(a))				*****	******	*****	*******	******	******	*****3
137	Else			192	2 Fu	unction R(i,j:Integer; ta	au:Extende	d):Extended	;			
138	If b<>0				3 Va							
139	Then	0		194		ι,		hip number		-		}
140	power :=	0	{ power(0,b) = 0 }	195 196		Cl, Cl1 :Integer;		iscrete cor				
141 142	Else power :=	1.	{ power(0,0) = 1 }		5 7 Be		τυ	iscrete cor	retation	Tunction	TOP CIT	p (+1)
	End; {Power}	.,		198		l := Trunc(tau/Tc);	:					
144				199	9	Cl1 := CCat[i,j]^.cc		ength];				
145	{*****	*****	*********************	200	-	If l >= 1						
146	{ InitFact		}	201	-	Then						
		***************************************	***************************************	202 203		Begin	^	al angth] .				
140	Procedure InitFact (Va Var	ar fact:ExtArr);		203		Cl := CCat[i,j] R := Cl*Tc+(Cl						
150				205		End		(10)				
	Begin			206		Else						
152	fact[0] := 1;			207	7	R := tau*Cl1;						
153	For i := 1 TO XMa					nd;{R}						
154	fact[i] := i	*Fact[i-1];		209		a ale ale ale ale ale ale ale ale ale al		***	والمروان والمروان والمروان والمروان	والمرام والمروان والمروان والمروان والمروان	alle alle alle alle alle alle alle alle	a dhadhadhadhadha 3
155	End;{InitFact}				-	Phot coloulatoo a post				******	******	******}
	{*****	*****	*****	212		Rhat calculates a part	Tat cross	correlation				2
		e terminal states on screen	\$		_	Depends on: config, CCa	at, Tc					Ś
159			3			Calls : none						>
		, numberOfLines, terminal	>			Modifies : none						}
	{ Calls : none		>		-	******				*******	******	*****}
162	{ Modifies : none	******	{ /************		7 Fu 8 Va	<pre>unction Rhat(i,j:Integer;</pre>	; tau:Exte	naea):Exten	aea;			
	Procedure ShowState;		,	219		ur L	{ C	hip number	belongin	a to tau		3
165	the second se			220		ci,		iscrete cor	-	-	for chi	pl 3

Page 3, listing of V2_CALC1.PAS, date is 30-10-95, file date is 25-09-95, size is 39354 bytes.

221 CL1 { Discrete correlation function for chip l+1 } :Integer; 222 Begin 223 := Trunc(tau/Tc); 224 Cl := CCat[i,i]^.cc[l]: 225 If l < codelength-1226 Then 227 Begin 228 Cl1 := CCat[i,j]^.cc[l+1]; 229 Rhat := Cl*Tc+(Cl1-Cl)*(tau-l*Tc); 230 Fnd 231 Else 232 Rhat := ((l+1)*Tc-tau)*Cl; 233 End; {Rhat} 234 236 { InitCCat sets up a catalog of discrete aperiodic cross correlation 237 { functions for a set of (Gold) codes passed by gc. The catalog consists of } 238 { a two dimensional array of pointers. These pointers refer to 239 { one-dimensional arrays containing the cross correlation information. This } 240 { information is in essence the number of simularities minus the numbers } 241 { of dissimilarities between two codes for a given discrete phase shift 3 242 { 3 243 { Depends on: config, p2CRec } 244 { Calls 3 : none 245 { Modifies : none } 247 Procedure InitCCat(Var CCat:CArr; gc:StringArr); 248 Var i, 249 { Loop variable } 250 { Loop variable 3 j, 251 { Loop variable } k, 252 ι. { Discrete phase shift 3 253 { Correlation result for shift l } t :Integer; 254 Begin 255 Write('Calculating ', Nt, ' by ', Nt,' correlations'); 256 { Double loop over all codes For i := 1 To Nt Do } 257 For i := 1 To Nt Do 258 Begin 259 { Show which codes are being done } GotoXY(70,1); 260 Write(i:4, j:4); 261 CCat[i,j] := New(p2CRec); { Allocate space for new array } 262 263 { Calculate cross-correlation } 264 For l := -codelength To codelength Do 265 Begin { Reset correlation value 266 t := 0; } 267 If l < 0{ Different formulas for pos and } 268 Then { neg phase shifts } 269 For k := 0 To codelength-1+1 Do 270 If gc[i][1+((k-l) Mod codelength)] = gc[j][k+1] 271 Then 272 Inc(t) 273 Else 274 Dec(t) 275 Else

276 For k := 0 To codelength-l-1 Do 277 If $qc[i][1+(k \mod codelength)] =$ 278 gc[j][1+((k+l) Mod codelength)] 279 Then 280 Inc(t) 281 Flse 282 Dec(t); 283 284 CCat[i,j]^.cc[l] := t; 285 End: 286 End: 287 End; (InitCCat) 288 290 { DisposeCCat frees memory space taken up by the correlation catalog 291 { 292 { Depends on: config 3 293 { Calls : none 3 294 { Modifies : none 296 Procedure DisposeCCat(CCat:CArr); 297 Var 298 i, { Loop variable } 299 i :Integer: { Loop variable } 300 Begin 301 For i := 1 To Nt Do { Loop over all catalog entries } 302 For i := 1 To Nt Do 303 Dispose(CCat[i,i]); { Free memory of element i, j } 304 End; {DisposeCCat} 305 307 { InitTerm initializes all terminals as free 308 { 3 309 { Depends on: config 3 310 { Calls : none 3 311 { Modifies : pfn 3 313 Procedure InitTerm(Var Terminal:TermArr); 314 Var 315 i :Integer; { Loop variable } 316 Begin For i:= 1 To Nt Do 317 318 terminal[i] := IDLE; 319 End: {InitTerm} 320 322 { Initcode reads Gold codes from a file and has the cross correlation 323 { catalog setup 324 (3 325 { Depends on: config 326 { Calls : initCCat 327 { Modifies : pfn 329 Procedure InitCodes(Var CCat:CArr; groupSize, numCodes:Integer); 330 Var

Page 4, listing of V2_CALC1.PAS, date is 30-10-95, file date is 25-09-95, size is 39354 bytes.

331	i,	{ Loop variable	} 3	886	End;	
332	j :Integer;	{ Loop variable	} 3	887	For i := 1 To Nt Do {	[Double loop over all terminals}
333	f :text;	<pre>{ File containing the Gold codes</pre>	} 3	888	For j := 1 To Nt Do	
334	codeCount,	<pre>{ Length of transmission code</pre>	} 3	38 9	Begin	
335	cfile :String;	{ Codefile name	} 3	590	{ Calculate path gains accord	<pre>ing to far field model }</pre>
336	gc :StringArr;	{ Gold codes as read from file	3 3	591	S[i,j] := Power(lambda/(4*Pi*	<pre>Sqrt(Sqr(xt[i]-xr[j])</pre>
337	Begin		3	392	+Sqr(yt[i]-yr[j])))	,lossexp);
338	ClrScr;		3	393	End	
339	<pre>Str(codelength, codeCount);</pre>	{ Assign appropriate codefile name	} 3	\$94	End;{InitPathGain}	
340	cfile := 'codes\'+codeCount+'	chip.dat';		395		
341	Assign(f, cfile);	{ Open code file	} 3	396	{**************************************	***************************************
342	Reset(f);		3	397	<pre>{* InitStartConfig</pre>	
343	For i := 1 To Nt Do	{ Read the codes from file	} 3	398	{*********	***************************************
344	ReadLn(f,gc[i]);		3	399	PROCEDURE InitStartConfig (VAR Start:ConfigArr;	X:Integer);
345	Close(f);		4	00	VAR	
346	If numCodes <> Nt	{ Reassign codes according to reuse scheme	3 4	01	i : Integer;	
347	Then	{ e.g. 12345678 -> 11223344 for 2 term/grp	3 4	02	BEGIN	
348	For i := numCodes Dow	InTo 1 Do	4	03	FOR i := 1 TO X DO	
349	For j := groupSiz	e DownTo 1 Do	4	04	<pre>Start[i] := i+1;</pre>	
350	gc[(i-1)*grou	ıpSize+j] := gc[i];	4	05	END;{InitStartConfig}	
351	initCCat(CCat	, gc);	4	06		
352		{ Initialize catalog of cross correlations	} 4	07	{*************************************	***************************************
353	End;{InitCodes}		4	804	<pre>{* InitMaxConfig</pre>	
354				09	{*****	***************************************
355	{*****	*****	*} 4	10	PROCEDURE InitMaxConfig (VAR Max:ConfigArr; X:I	nteger);
356	{ InitPathLoss sets up a table of	transmitter to receiver path gains from			VAR	
357	{ all transmitters to all receive	ers. All gains are between 0 and 1. When	} 4	12	i : Integer;	
358	{ groups are larger than one term	inal duplicate entries will exist in the	} 4	13	BEGIN	
359	{ table because there are less re	ceivers than terminals then.	3 4	14	FOR i := X DOWNTO 1 DO	
360	(3 4	15	Max[i] := Nt-(X-i);	
	{ Depends on: config, lambda		> 4	16	END;{InitMaxConfig}	
	{ Calls : Power			17		
363	{ Modifies : none		} 4	18	{*****	**********************************
364	{****	*****	*} 4	19	<pre>{* WriteConfig</pre>	*}
365	Procedure InitPathGain(Var S:Matri	x; groupSize, numberOfLines :Integer;			{****	***************************************
366		Rreceive, Rterm, lossexp:Extended);			Procedure WriteConfig (Var Config:ConfigArr; X:	-
367	Var	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1			Var	
368	gamma,	{ Angle of receiver relative to x-axis		23		
369	theta :Extended;	{ Angle of transmitter relative to x-axis	-		Begin	
370	xt, yt,	{ Transmitter coordinates		25	<pre>Write('Configuration = ');</pre>	
371	xr, yr :PosArr;	{ Receiver coordinates		26		
372	i,	{ Loop variable		27		
373	j,	{ Loop variable	-	28		
374	k :Integer;	{ Index variable	-		End;{WriteConfig}	
	Begin		-	30		
376	For i := 1 To numberOfLines D	of Loop over all receivers			{*****	**********************************
377					{ Rayleigh randomly draws a value according to	-
378	Begin	er meh eter att 3. och terminate	-	33		s the rayterigh aroth batton 3
379	k := (i-1) * groupSize	+ j; {Determine index current term.			{ Depends on: none	3
380	gamma := 2*Pi * (i-1)/n				{ Calls : duni	3
381	xr[k] := Rreceive * Cos		-		{ Modifies : pfn	3
382	yr[k] := Rreceive * Sin				{*************************************	· {********************************
383					Function Rayleigh(ave:Extended):Extended;	,
384		* Cos(theta); {Calc. transmitter x-coord.			Var	
385	yt[k] := yr[k] + Rterm			40		
			-			

Page 5, listing of V2_CALC1.PAS, date is 30-10-95, file date is 25-09-95, size is 39354 bytes.

441	u,	496	aa := 0 ;
442	v,	497	bb := 0.5;
443	×,	498	ba := 0 ;
444	y :Extended;	499	d1 := dd;
	Begin	500	n := 0;
446	Repeat	501	Repeat
447	u := 2*duni-1;	502	n := n + 1;
448	v := 2*duni-1;	503	ac := d1+2*n*ab/(a*b)+aa;
449	r := sqr(u)+sqr(v); Until (r<1.0) and (r>0.0);	504 50 5	bc := 1+2*n*bb/(a*b)+ba;
450 451	r := Sqrt(-2*Ln(r)/r);	506	d1 := dd*d1; aa := ab;
452	$x := u^{r};$	507	ad :- ab;
453	y := v*r;	508	ba := bb;
454	Rayleigh:=Sqrt(Sqr(x)+Sqr(y))*ave*Sqrt(2/Pi);	509	bb := bc;
455 1		510	Until bc > 1e12;
456	,	511	If a < b
	(**************************************	512	Then
458		513	MarcumQ := (ac/(2*bc))*Exp(-Sqr(a-b)/2)
459		514	Else
460 .	[Depends on: none }	515	MarcumQ := 1-(ac/(2*bc))*Exp(-Sqr(a-b)/2);
461 +	[Calls : none }	516	End;
462 -	[Modifies : pfn }		End;
	***************************************	518	
	Function MarcumQ(a,b:Extended):Extended;		(**************************************
465 \			{ NegExpIO approximates the modified Bessel function of the first kind }
466	aa,	521	•
467	ab,	522 523	
468 469	ac,	524	
409	bc, bb,	525	
471	ba,		{ Depends on: none }
472	dd,	527	
473	d1 :Extended;		{ Modifies : pfn }
474	n :Integer;	529	<pre>(************************************</pre>
475 E	Begin	530	Function NegExpIO(a,b:Extended):Extended;
476	If a = 0		Var
477	Then	532	n :Integer; { Loop variable }
478	If b < 30	533	х,
479	Then	534	Ja, { I0,n-1 }
480	MarcumQ := Exp(-Sqr(b)/2)	535	Jb, { 10,n }
481	Else	536	Jc, { I0,n+1 }
482	MarcumQ := 0	537	tmp,
483	Else	538	t :Extended; { Intermediate result }
484	Begin		Begin
485	If a < b	540	tmp := (Sqr(a)+Sqr(b))/2;
486 487	Then Begin	541 542	x := a*b; If x = 0
488	ab := 1;	543	Then
489	dd := a/b ;	544	If tmp < 150
490	End	545	Then
491	Else	546	NegExpIO := Exp(-tmp)/2
492	Begin	547	Else
493	ab := 0;	548	negExpIO := 0
494	dd := b/a;	549	Else
495	End;	550	Begin

Page 6, listing of V2_CALC1.PAS, date is 30-10-95, file date is 25-09-95, size is 39354 bytes.

```
551
           Jc := 0;
552
           Jb := 1;
553
           t := 0;
           For n := 20 DownTo 1 Do
554
555
           Begin
556
               Ja := (2*n/x)*Jb+Jc;
557
               Jc := Jb;
558
               Jb := Ja;
559
               t := t + Ja;
560
           End:
561
           NeqExpIO := Exp(x-(Sar(a)+Sar(b))/2)*Ja/(2*t-Ja):
562
       End;
563 End;
564
566 { CalcMu1 calculates the Mu1 parameter for the bit error formula in
567 { Rayleigh fading environments
568 {
569 { Depends on: L
                                                                3
570 { Calls
             : R, Rhat
571 { Modifies : none
573 Function CalcMu1(u, h:Integer; tau:RArr; mu2:Extended):Extended;
574 Var
575
                                               { Intermediate result }
     tmp :Extended;
                                               { Loop variable
576
    i
        :Integer;
                                                                3
577 Begin
578
       tmp := 0;
579
       For i := 1 To L Do
580
          If i <> h
581
          Then
582
             tmp := tmp + R(u,u,tau[u,i])*Rhat(u,u,tau[u,i])*s[u,u];
583
584
       CalcMu1 := 4*2*Pt*tmp + mu2
585 End:
586
588 { CalcMu2 calculates the Mu2 parameter for the bit error formula in
589 { Rayleigh fading environments
                                                                3
590 {
                                                                3
591 { Depends on: config, L, Pt, terminal
                                                                3
592 { Calls
            : R, Rhat
593 { Modifies : none
595 Function CalcMu2(u, h:Integer; tau:RArr; sigma_n2:Extended):Extended;
596 Var
597
     tmp :Extended;
                                              { Intermediate result }
                                               { Loop variable
598
     i,
                                                                }
599
                                               { Loop variable
                                                                }
        :Integer;
600 Begin
601
       tmp := 0;
602
       For i := 1 To Nt Do
603
         If (terminal[i] = INTERFERE) And (i <> u)
604
         Then
605
            For j := 1 To L Do
```

```
606
               tmp := tmp + (Sqr(R(u,i,tau[i,j]))
607
                        + Sqr(Rhat(u,i,tau[i,j])))*s[i,u];
608
609
       For i := 1 To L Do
         If i <> h
610
611
         Then
612
            tmp := tmp + (Sqr(R(u,u,tau[u,j]))
613
                     + Sqr(Rhat(u,u,tau[u,j])))*s[u,u];
614
615
       CalcMu2 := 2*(2*Pt*tmp+sigma n2);
616 End:
617
619 { CalcMu12 calculates the Mu12 parameter for the bit error formula in
                                                                 3
620 { Rayleigh fading environments
                                                                 3
621 {
                                                                 3
622 { Depends on: config, L, Pt, terminal
623 { Calls
             : R. Rhat
624 { Modifies : none
                                                                 3
626 Function CalcMu12(u,h:Integer;tau:RArr):Extended;
627 Var
628
                                               { Intermediate result }
     tmp :Extended;
629
                                               { Loop variable
     i,
                                                                 }
630
                                               { Loop variable
                                                                 }
     j
        :Integer;
631 Begin
632
       tmp := 0:
633
       For i := 1 To Nt Do
634
         If (terminal[i] = INTERFERE) And (i<>u)
635
         Then
636
            For i := 1 To L Do
637
               tmp := tmp + R(u,i,tau[i,j])*Rhat(u,i,tau[i,j])*s[i,u];
638
639
       For j := 1 To L Do
640
         If j <> h
641
         Then
642
             tmp := tmp + (Sqr(Rhat(u,u,tau[u,j]))
643
                     + R(u,u,tau[u,j])*Rhat(u,u,tau[u,j]))*S[u,u];
644
645
       CalcMu12 := 2*2*Pt*tmp;
646 End;
647
649 { CalculatePbe computes the bit error probability for a terminal, given
650 { which other terminals transmit
651 {
652 { Depends on: config, L, Pt, S, Tb, terminal
                                                                 >
653 { Calls
             : duni, IO, MarcumQ, Rayleigh
                                                                 3
654 { Modifies : none
656 Function CalculatePbe(u:Integer):Extended;
657 Var
658
     mu1,
659
     mu2,
660
     mu12,
```

Page 7, listing of V2_CALC1.PAS, date is 30-10-95, file date is 25-09-95, size is 39354 bytes.

```
716
                                                                                         tmp := ( (1 + mu12/Sqrt(mu1*mu2))*mb )/2;
661
      a,
                                                                                717
                                                                                         CalculatePbe := mg-tmp
662
      b,
                                                                                718 End;
663
                                  { Maximum path gain
                                                                        }
      bmax.
                                                                                 719
664
      tmp,
                                                                                 635
      m,
                                                                                 721 {* CalcPpsXZero
                                                                                                                                                         *}
1.5
      mq.
                                                                                 667
      mb,
663
      sigma n2:Extended;
                                  { Noise power
                                                                         3
                                                                                 723 FUNCTION CalcPpsXZero : Extended;
                                  { Rayleigh distributed path gains
                                                                        }
                                                                                 724 VAR
      bata,
                                                                                 725 i
                                  { Uniformly [0, Tb] distributed path delays }
                                                                                                 : Integer;
      tau
             :RAFF:
                                  { Number of path with largest gain
                                                                        }
                                                                                 726
                                                                                      PpsTot
                                                                                                 : Extended;
671
      h,
                                  { Loop variable
                                                                        }
                                                                                 727 BEGIN
672
      i,
                                  { Loop variable
                                                                        }
                                                                                 728
                                                                                         InitTerm(Terminal);
673
      ī
             :Integer;
                                                                                 729
                                                                                         Terminal[1] := INTERFERE;
674 Begin
        { Assign random delays and path gains to transmitting terminals }
                                                                                 730
                                                                                         PpsTot
                                                                                                   := 0;
675
                                                                                 731
                                                                                         If L=1
676
        For i := 1 To Nt Do
                                                                                 732
677
          If terminal[i] = INTERFERE
                                                                                         Then
                                                                                 733
678
                                                                                            CalcPpsXZero := Power(1-CalculatePbe(1), PacketLength)
          Then
                                                                                 734
679
              For i := 1 To L Do
                                                                                         Else
                                                                                 735
                                                                                         Begin
680
              Begin
                                                                                 736
                                                                                            For i := 1 To MonteCarlo Do
681
                   tau[i,j] := Tb*duni;
                                                                                 737
682
                   beta[i,j] := Rayleigh(Sqrt(S[i,u]));
                                                                                            BEGIN
                                                                                 738
                                                                                               PpsTot := PpsTot + Power(1-CalculatePbe(1), PacketLength);
683
              End
                                                                                 739
                                                                                            END:
684
          Else
                                                                                 740
                                                                                            CalcPpsXZero := PpsTot/MonteCarlo;
685
              For i := 1 To L Do
                                                                                 741
                                                                                         End
686
              Begin
687
                   tau[i,i] := 0;
                                                                                 742 End; {CalcPpsXZero}
688
                   beta[i,j] := 0;
                                                                                 743
                                                                                 689
              End:
                                                                                 745 {* CalcPpsAveX
690
                                                                                 691
        { Determine what the largest gain is }
692
        bmax := 0:
                                                                                 747 FUNCTION CalcPpsAveX (X:Integer):Extended;
                                                                                 748 VAR
693
        h := 1:
                                                                                 749
                                                                                       i,ii,j
694
        For i := 1 To L Do
                                                                                                 : Integer;
                                                                                 750
695
          If beta[u,i]>bmax
                                                                                       Pbe,
                                                                                 751
696
                                                                                       PpsTot
                                                                                                 : Extended;
          Then
697
                                                                                 752
                                                                                       TempConfig : ConfigArr;
          Begin
                                                                                 753 BEGIN
               bmax := beta[u,i];
698
                                                                                 754
                                                                                         InitTerm(Terminal);
699
               h
                   := i;
                                                                                 755
                                                                                         Terminal[1] := INTERFERE;
700
          End;
                                                                                 756
                                                                                         TempConfig := StartConfig;
701
                                                                                 757
702
        { Determine noise power relative to largest incoming signal }
                                                                                         PpsTot := 0;
                                                                                 758
        sigma n2 := power(10, -snr/10)*Sqr(bmax)*Pt*Tb*Tb;
                                                                                         CountConfig := 0;
703
                                                                                 759
                                                                                         FOR ii := 1 TO X DO
704
                                                                                 760
                                                                                            Terminal[TempConfig[ii]] := INTERFERE;
705
        { Calculate bit error probability parameters }
706
        mu2 := CalcMu2(u, h, tau, sigma_n2);
                                                                                 761
                                                                                         FOR ii := 1 TO MonteCarlo DO
                                                                                 762
707
        mu1 := CalcMu1(u, h, tau, mu2);
                                                                                         Begin
                                                                                 763
708
        mu12 := CalcMu12(u, h ,tau);
                                                                                             Pbe := CalculatePbe(1);
                                                                                 764
                                                                                             PpsTot := PpsTot + Power(1-Pbe, PacketLength);
709
            := Sqrt(2*Pt)*bmax*Tb;
        m
                                                                                 765
                                                                                         End:
710
            := m*Abs(1/Sqrt(mu1)-1/Sqrt(mu2))/Sqrt(2);
        а
            := m*(1/Sqrt(mu1)+1/Sqrt(mu2))/Sqrt(2);
                                                                                 766
                                                                                         CountConfig := CountConfig + 1;
711
        b
                                                                                 767
                                                                                         {WriteConfig(TempConfig, X);}
712
            := MarcumQ(a,b);
        mq
                                                                                 768
                                                                                         i := X;
713
            := NegExpIO(a,b);
        mb
                                                                                 769
                                                                                         WHILE (i<>1) OR (TempConfig[1]<>MaxConfig[1]) DO
714
                                                                                 770
                                                                                         BEGIN
715
        { Calculate the bit error probability }
```

Page 8, listing of V2_CALC1.PAS, date is 30-10-95, file date is 25-09-95, size is 39354 bytes.

826 WriteLn(OutFile, 'SNR : ', SNR); 771 i := X; 827 772 IF TempConfig[i] = Nt WriteLn(OutFile); 828 773 THEN WriteLn(OutFile);} 829 END: {OpenResultFile} 774 REGIN 75 WHILE TempConfig[i] = MaxConfig[i] DO 830 776 i := i-1: 777 TompConfig[i] := TempConfig[i] + 1; 832 { WarnBeep emits a half second warning tone and then waits 4.5 seconds 773 FOR 1 := (1+1) TO X DO 833 (7.7 834 { Depends on: none TempConfig[j] := TempConfig[j-1] + 1; 835 { Calls : none 780 END 836 { Modifies : pfn 781 ELSE 782 TempConfig[i] := TempConfig[i] + 1; 783 838 Procedure WarnBeep: InitTerm(Terminal); 784 839 VAR Terminal[1] := INTERFERE; 785 FOR ii := 1 TO X DO 840 i : Integer; 786 Terminal[TempConfig[ii]] := INTERFERE; 841 BEGIN FOR i := 1 TO 5 DO 787 ShowState; 842 843 788 FOR ii := 1 TO MonteCarlo DO BEGIN Sound(500+i*20); 789 844 Begin 790 845 Delay(300); Pbe := CalculatePbe(1); 791 PpsTot := PpsTot + Power(1-Pbe, PacketLength); 846 END: 847 792 End: NoSound: 848 Delay(1000); 793 CountConfig := CountConfig + 1; 849 End; {WarnBeep} 794 {WriteConfig(TempConfig, X);} 795 850 END: 796 CalcPpsAveX := PpsTot/(MonteCarlo*CountConfig); {WriteLn(X, ' ', CountConfig, ' ', MonteCarlo*CountConfig);} 852 { 797 MAIN LINE 798 END; {CalcPpsAveX} 854 Begin 799 855 ClrScr: 856 WriteLn('Output Testprogramma:');WriteLn; 801 PROCEDURE OpenResultFile(VAR OutFile:Text); 802 VAR 857 858 GS, {Initialize Variables} 803 859 Rreceive := 30: 804 NL. 860 Rterm := 5; 805 FileName : String; 861 SNR := 6; 806 BEGIN 807 862 Str(GroupSize, GS); 863 := 32; 808 Str(NumberOfLines, NL); Nt 864 NumberOfLines := 2; 809 FileName := '2SDK7L2.dat'; 865 GroupSize := 16; 810 Assign(OutFile, FileName); 866 811 867 812 { Initialize Functions and Procedures } ReWrite(OutFile); 868 InitCodes(CCat, GroupSize, NumberOfLines); 813 { WriteLn(OutFile, 'H.L.A. Le'); WriteLn(OutFile, 'RESULTS FROM CALCULATIONS (v.2):'); 869 InitPathGain(S, GroupSize, NumberOfLines, Rreceive, Rterm, LossExp); 814 870 815 WriteLn(OutFile); InitFact(Faculteit); 816 WriteLn(OutFile,'Receiver Radius: ', Rreceive); 871 WriteLn('NumberOfLines: ',NumberOfLines); 872 817 WriteLn(OutFile, 'Terminal Radius: ', Rterm); WriteLn: 873 818 WriteLn(OutFile): 874 819 { Calculate PpsAveX } WriteLn(OutFile,'Number of users : ', Nt); 875 OpenResultFile(ResultFile); 820 WriteLn(OutFile,'Size of group : ', GroupSize); WriteLn(OutFile, 'Number of groups: ', NumberOfLines); 876 FOR X := 1 TO XMax Do 821 877 BEGIN 822 WriteLn(OutFile); 878 823 WriteLn(OutFile,'Code length : ', codelength); InitStartConfig(StartConfig, X); 879 824 WriteLn(OutFile, 'Packet length : ', Packetlength); InitMaxConfig(MaxConfig, X); 880 PpsAveX := CalcPpsAveX(X); 825 WriteLn(OutFile, 'Inhibit delay fraction: ', Fraction);

Page 9, listing of V2_CALC1.PAS, date is 30-10-95, file date is 25-09-95, size is 39354 bytes.

881 PpsAveXArr[X] := PpsAveX; WriteLn(ResultFile, PpsAveX); 882 883 Flush(ResultFile); 884 END; 885 WriteLn(ResultFile); 886 WriteLn(ResultFile, 'G: 1:24, 'S(4): 1:23, 887 888 'D(4): 1:23); 839 \$90 { Throughput and Delay Calculation } 891 PpsXZero := CalcPpsXZero; For i := 0 To NumSim Do 892 893 Begin := Power(10, -(LRangeMin+i*(LRangeMax-LRangeMin)/NumSim)); 894 G 895 ExpGd := Exp(-G*fraction); TCycle := (1+(2*fraction))+(ExpGd/G); 896 897 PNoConfl := ExpGd*PpsXZero; := 1/(G*(1+(2*fraction))+ExpGd); 898 Pidle PConflTemp := 0; 899 900 Write(ResultFile, G:23); For j := 1 To XMax Do 901 902 Begin Poisson := Power(G*fraction,j) * ExpGd/Faculteit[j]; 903 PConflTemp := PConflTemp + Poisson*PpsAveXArr[j]; 904 905 End; := PConflTemp; 906 PConfl Psuccess := PNoConfl+PConfl; 907 908 Throughput := Psuccess/TCycle; := (((G*Pidle)/Throughput)-1)*(1+fraction+(1/Alpha)) 909 D 910 +(G*(1-Pidle)/Throughput)*(1/Alpha) +(1+fraction); 911 912 Write(ResultFile, Throughput:23); 913 WriteLn(ResultFile, D:23); 914 End; 915 Close(ResultFile); 916 917 918 {WarnBeep;} 919 WriteLn; 920 WriteLn; Write('I AM READY MISTER LE !!!!!!'); 921 922 GoToXY(35,17); Write('Press ENTER to return to program.'); 923 924 {ReadLn;}

925 END.