DELFT UNIVERSITY OF TECHNOLOGY
Department of Electrical Engineering
Telecommunications and Traffic-Control Systems Group

Title: **Evaluation and performance analysis of the Circuit Reservation Multiple Access Protocol**

Author: M. Moretti

Type: Graduation thesis
Date: June 29, 1995
Pages: 110 + ix

Mentors: Prof. dr. R. Prasad and Ir. Nijhof

Period: January 1995 - July 1995

ABSTRACT

The Circuit Reservation Multiple Access (CRMA) protocol is a new protocol for wireless indoor communications. It integrates voice and data traffic on a Time Division Duplexed channel. As with Mthe well known Packet Reservation Multiple Access (PRMA) protocol, CRMA is a hybrid TDMA/Slotted ALOHA protocol. Speech packets follow a TDMA scheme and data packets are transmitted with Slotted ALOHA. For speech traffic, speech activity detection is used to enhance the system performances. To avoid collisions between speech and data packets, the stations transmitting data sense the channel before a transmission.

The protocol is designed to guarantee high quality speech transmission: the voice traffic does not suffer any delay and has a higher priority than data traffic. Nevertheless, data traffic has relatively high throughputs and low packet delays. An analytical model of the protocol is presented. The system performances are analysed and the results are compared with those of a PRMA system.

Indexing terms: Cellular packet communications, Circuit Reservation Multiple Access, Packet Reservation Multiple Access, Integrated voice/data protocol, Wireless office communications.

SUMMARY

The Circuit Reservation Multiple Access protocol is a protocol that uses a Time Divisioned Duplexed (TDD) radio link. A number of both Constant Bit Rate (CBR) and Variable Bit Rate (VBR) sources (respectively transmitting speech and data traffic), all located in a small geographical area or cell, share the same communication channel. The channel is Time Division Duplexd into frames, and each frame is divided into a number of slots. The channel for each cell is controlled by one Group Base Station (GBS).

The system is designed to have high quality speech transmission (i.e. no delay for voice packets). The data packets contend for the timeslots not occupied by speech traffic.

The system performances for data transmissions are analysed, considering different CBR loads. For each different channel occupation, the data throughput and the data time delay is calculated.

We consider a system with 20 CBR stations and 13 VBR stations. The computer simulations match perfectly the analytical results.

When the channel is almost all available to data transmissions, the performances are nearly equivalent to those of Slotted ALOHA. As the CBR occupation of the channel increases, the data throughputs decrease and the time delays increase. Even in the worse cases, the time delay is reasonably low (does not exceed the 100 timeslots for 65% of the channel occupied) and in every case the data throughputs are on the average around 40% of the available channel.

PREFACE

This graduation report was written as the final stage of my Master of Science (M.Sc.) degree. The research was carried out and completed at Delft University of Technology, where I studied on the ERASMUS exchange programme through the period from January, 1995 to July, 1995.

In this Preface I would firstly like to thank Professor Prasad and Ir. Nijhof for enabling me to accomplish my work in the Netherlands to the best of my ability. Also, I am grateful for the help of Professor Del Re and Professor Fantacci at my home university in Firenze.

Secondly, I must bring to your attention the vital role played by my "Bras Droit"; Gregor Hendrikse, without the help of whom, this report would not have attained the high standard it currently possesses.

Next I must show my appreciation to all the friends I made during this ERASMUS exchange, who all enabled me to enjoy the little time I spent outside of study. They provided me with the means to relax and entertain myself thoroughly, which helped me to concentrate with more enthusiasm to the working periods.

Finally, every young, male, Italian student can understand the utmost importance of having the truthful, encouraging and loving guidance of their family (and girlfriend) to carry them through the fog riden sea of uncertainty.

LIST OF ABBREVIATIONS

BACK	Backlogged state
CDMA	Code Division Multiple Access
CRMA	Circuit Reservation Multiple Access Protocol
FIFO	First In First Out
GBS	Group Base Station
ORIG	Originating state
PRMA	Packet Reservation Multiple Access Protocol
SIL	Silent state for the voice activity detector
TDD	Time Division Duplexing
TDMA	Time Division Multiple Access
TLK	Talking state for the voice activity detector
VBR	Variable Bite Rate
WAIT	Waiting state

LIST OF SYMBOLS

a	worst case propagation delay
α	probability of a packet arrival in one slot period
CBR_{thr}	continuous traffic throughput
$CTRL_{thr}$	number of successfully transmitted CONTROL packets per timeslot
D_{max}	maximum allowed time within a speech packet has to be transmitted, it will be dropped otherwise
γ	probability of a transition from the talking to the silent state
GBS_{thr}	throughput of the GBS
GBS_{weight}	weight factor to set the ratio between the VBR and GBS time delay
inb_outb	ratio between inbound and outbound traffic
K	number of circuits per frame available to continuous traffic
λ_n	call arrival rate at the state n
μ_n	call completion rate at the state n
N	number of CBR stations
nr_{CBR}	number of CBR users in the cell
nr_circ_{CBR}	average number of CBR circuits
nr_slots_{CBR}	average number of slots occupied by CBR traffic

nr_slots_{CBR}	number of slots in every frame assigned to CBR stations
nr_slots_{CTRL}	number of slots in every frame assigned to CONTROL data
$nr_{rtr\ GBS}$	average number of retransmissions accomplished by the GBS
$nr_{rtr\ VBR}$	average number of retransmissions accomplished by a VBR terminal
nr_{VBR}	number of VBR stations
p	number of stations in the WAITING state
P	transition probabilities matrix
P{BACK}	probability that a station is in the BACKLOGGED state
P{ORIG}	probability that a station is in the ORIGINATING state
P{SIL}	the probability for a station to be in the silent state
P{TLK}	the probability for a station to be in the talking state
P{TRANS}	probability that a station is in the TRANSMIT state
P{WAIT}	probability that a station is in the WAITING state
P_{arr}	Packet arrival rate at the system
P_B	blocking probability
P_d	retransmission probability for VBR terminals in the backlogged state
Pd	probability that a user terminal transmits a random packet from a data buffer in an unreserved slot
P_{d0}	probability a that new VBR packet arrives
P_{free}	the probability a slot is not occupied by CBR traffic

P_g	retransmission probability for the GBS in the backlogged state
P_{g0}	probability a that new GBS packet arrives
P_K	probability that CBR traffic occupies K slots
P_S	probability that a user terminal transmits a periodic packet from a data buffer in an unreserved slot
P_{succ}	probability that a packet is successfully transmitted
$P_{succGBS}$	probability of a successful GBS packet transmission
$P_{succVBR}$	probability of a successful packet transmission by a VBR terminal
P_{wd}	retransmission probability for VBR terminals in the waiting state
P_{wg}	retransmission probability for the GBS in the waiting state
q	number of stations in the BACK state
σ	probability of a transition from the silent to the talking state
slpfr	number of slots per frame
$Syst_{prfm}$	system performance function
$Syst_{thr}$	system throughput
Tack	duration of an acknowledgement
$t_{del\ collision\ GBS}$	average time delay in timeslots for the GBS due to collisions with other packets
$t_{del\ collision\ VBR}$	average time delay in timeslots for VBR terminals due to collisions with other packets

$t_{del\ WAIT\ STATE\ GBS}$	time delay for GBS packets in the WAIT state
$t_{del\ WAIT\ STATE\ VBR}$	time delay for VBR packets in the WAIT state
t_{delGBS}	GBS packet delay
t_{delVBR}	VBR packet delay
Tslot	duration of one timeslot
VBR_{thr}	throughput of all VBR terminals
VBR_{weight}	weight factor to set the ratio between the VBR and GBS time delay
$V_{steady\ state}$	Steady State vector
V_t	state probability vector at the time t
x	state of the GBS
X1	left border of the allowed interval for the penalty function
X2	right border of the allowed interval for the penalty function

CONTENTS

INTRODUCTION

In recent years there has been a steadily growing interest in wireless communication systems and in the integration of different types of traffic.

Wireless access provides flexibility in the placement of terminals and avoids pulling cables to any foreseeable location.

The electronic traffic can be divided in two groups: the traffic that requires a continuous bandwidth allocation and the traffic that has variable bandwidth demand. Continuous traffic does not tolerate time delays higher than a certain threshold and the quality of the transmission rapidly decreases when the delay increases. For the variable bandwidth traffic, the most important feature is the accuracy of the transmission. Continuous traffic sources are: telephone, fax, video, etc.; the variable traffic is mainly computer generated. Integration of traffic is required to use one communication system that provides all the different kinds of services.

Recently the concept of hybridisation has been accepted as a possible answer to many communications problems: by combining the best features of different protocols, new protocols have been created.

At the end of the 80's, PRMA, a hybrid protocol, was proposed by Goodman et al. as a "protocol for local wireless communications" [1] [2] [3]. In the early 90's at the Telecommunications and Traffic Control Systems group of the Delft University of Technology the CRMA protocol was developed [4] [5].

CRMA is a TDMA/Slotted ALOHA hybrid protocol and combine the channel efficiency and the high throughputs of TDMA with the Slotted ALOHA's traffic flexibility.

CRMA is a radio protocol that integrates the two types of traffic answering to the main problematic introduced by the different requirements: no time delay for continuous traffic, an efficient control error for the data traffic

All the terminals are connected with a base station in a star topology. The transmission time is divided into a fixed number of timeslots, like in the TDMA protocol. Each speech station is equipped with a speech activity detector.

The speech packets access the channel following a pure TDMA scheme, the data packets are transmitted using the Slotted ALOHA protocol

The conflict between the two types of traffic does not exist due to a mechanism of *channel sensing* performed by the data stations that gives a higher priority to the continuous traffic.

Differently from PRMA, there is no packet dropping and no speech packet is discarded.

In Chapter 1 we give a brief, introductory overview of the CRMA protocol.

The two different types of traffic considered, speech and data, are analysed in Chapter 2 and in Chapter 3. These two chapters present the timing and the analytical model for both traffics.

Chapter 4 analyses the system performances: throughput and time delay.

In Chapter 5 and Chapter 6 we present the results of the computer system analysis.

Chapter 7 is a qualitative (with some numerical results) comparison of CRMA and PRMA.

In Chapter 8 we propose a new hybrid CDMA/CRMA protocol that is an evolution of the studied system.

The conclusions and the recommendations are eventually discussed in Chapter 9.

1 CRMA PROTOCOL

1.1 Introduction to the CRMA protocol

CRMA is a protocol for wireless indoor communications. The main purpose is to efficiently integrate the speech and data traffic.

The stations, all located in a small area or cell, are connected with a Group Base Station (GBS) in a star topology. All the GBSs are connected with each other by a high speed back bone network as shown in figure 1.

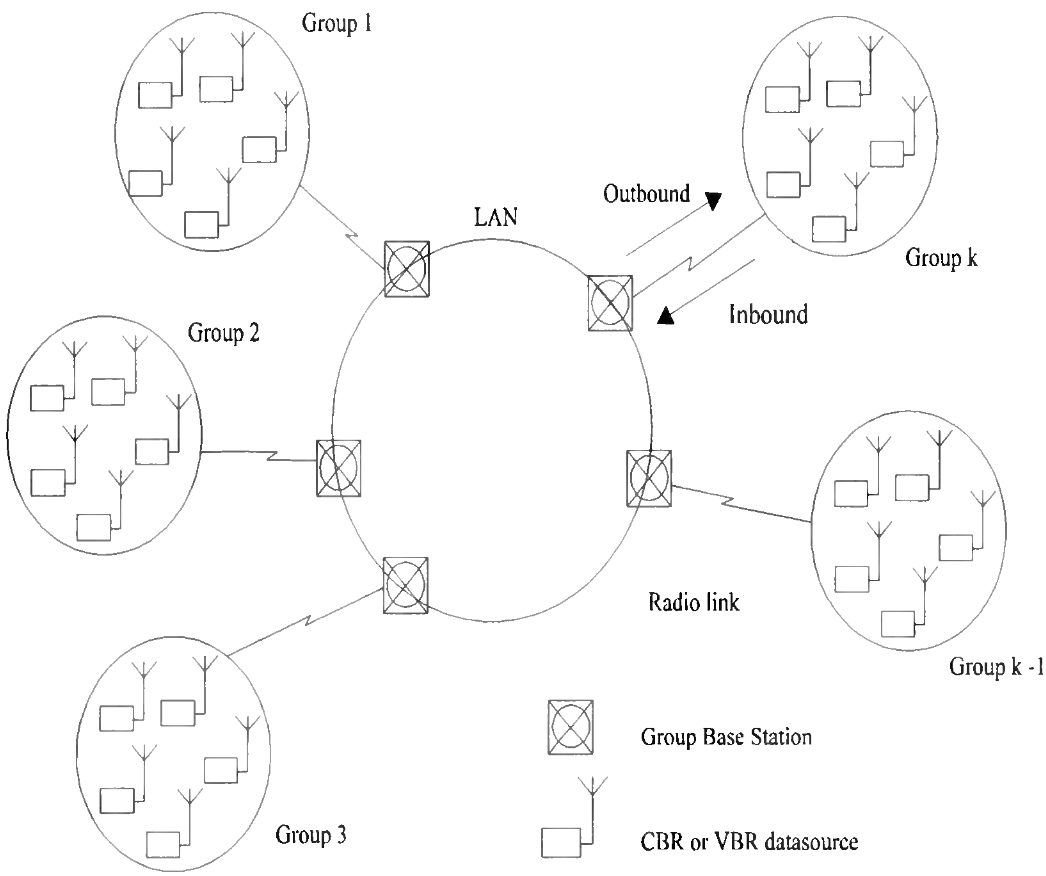

Figure 1: The CRMA system.

The stations transmitting over the channel can be divided in Continuous Bitstream Rate sources (CBR) and Variable Bitstream Rate (VBR) sources. Needless to say, speech traffic is handled by CBR stations and data traffic by VBR stations.

The GBS can be considered, depending on what it transmits, as a CBR or a VBR station.

The channel is Time Division Duplexed into frames and each frame is divided into a number of slots. The use of Time Division Duplexing (TDD) allows an optimal channel allocation between the inbound and outbound traffic.

The main features of CRMA are:

- It is a hybrid TDMA/Slotted ALOHA protocol. Speech traffic is handled following a pure TDMA scheme, data packets are transmitted with a Slotted ALOHA procedure.

- A certain amount of the channel (roughly 10%) is reserved for CONTROL packets. CONTROL packets are sent by the GBS in order to keep all the stations synchronised. Since the channel is slotted, the correct synchronisation of every station is vital for the good performances of the system. CONTROL packets are treated as CBR packets.

- VBR stations "listen" to the channel before transmitting a packet in order to avoid contention for the channel access with the speech traffic. If a VBR station finds the channel occupied, holds the transmission and tries to retransmit in the next timeslot with a retransmission probability chosen by the system engineer. Therefore, continuous traffic is independent and has a higher priority.

- VBR traffic can access only the slots not occupied by speech packets. If more than one data packet is transmitted within the same timeslot, all the collided packets are lost and they are retransmitted in the next timeslot with a probability value chosen by the system engineer.

- For each successfully received data packet, the receiving station sends a message, within the same timeslot, to acknowledge the good result of the transmission.

1.2 An example of CRMA accessing the TDD radio link

In figure 2 we give an example of three CBR and two VBR terminals accessing the radio link with nine timeslots per frame. For each slot a brief description is given.

Timeslot 1: The GBS sends a control packet. All VBR and CBR pick this packet from the radio link.

Timeslot 2: CBR 1 sends a packet, the GBS receives it and places it on the backbone network. VBR 2 senses the channel before sending the packet and finds it occupied.

Timeslot 3: VBR 1 and the GBS have sensed a free timeslot and both transmit a packet. A collision occurs, both packets are lost.

Timeslot 4: The GBS sends a CBR packet to CBR 2 terminal.

Timeslot 5: The GBS sends a VBR packet to VBR 2 terminal. The transmission is successful.

Timeslot 6: No station has transmitted in this timeslot. The channel is idle during this slot.

Timeslot 7: VBR 1 and VBR 2 have sensed a free timeslot and both transmit a packet. A collision occurs, both packets are lost.

Timeslot 8: CBR 3 sends a packet. The GBS receives it and places it on the backbone network.

Timeslot 9: VBR 1 successfully sends a packet.

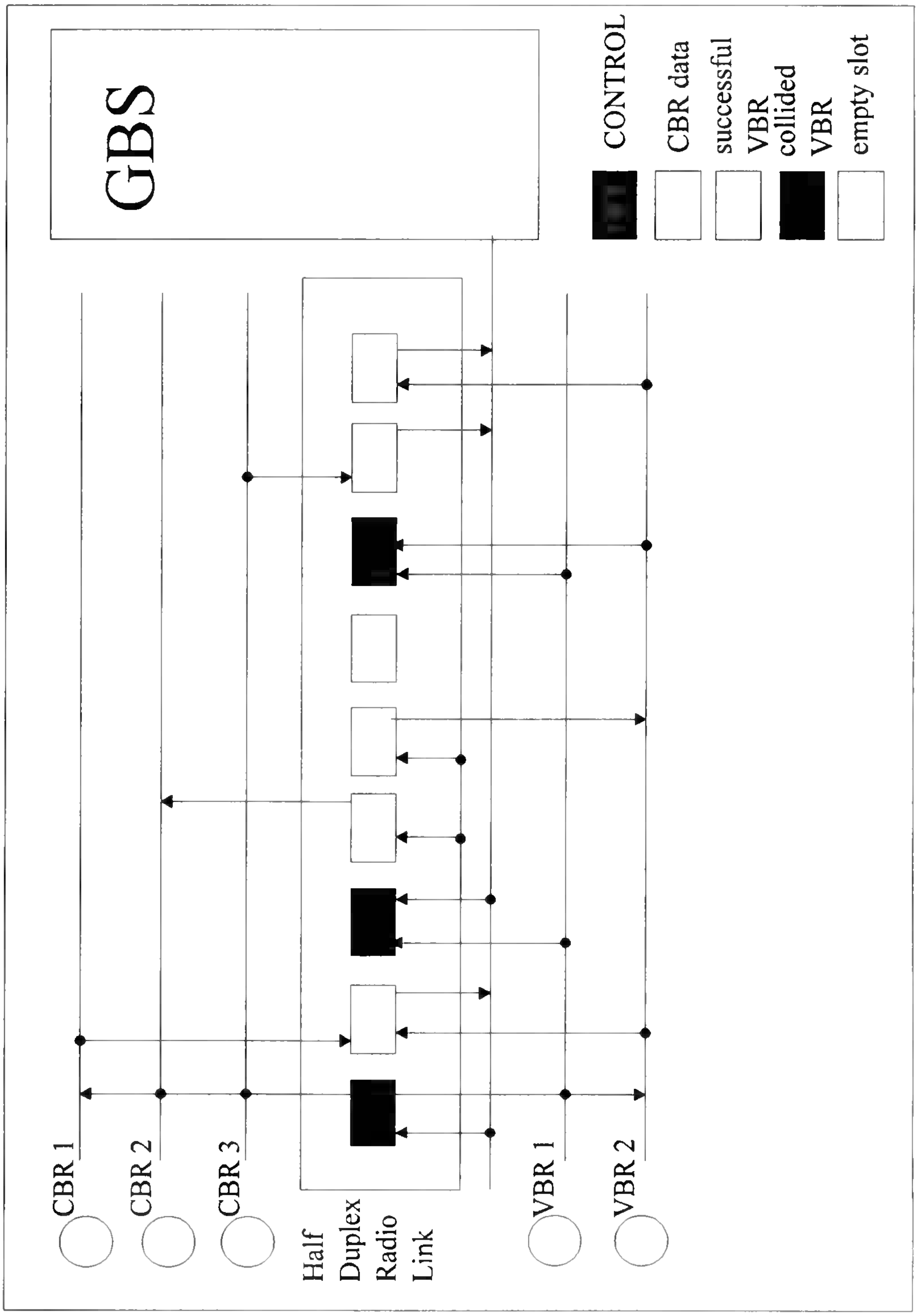

Figure 2: Accessing the half duplex radio link.

2 CONTINUOUS TRAFFIC

Continuous traffic is represented by CONTROL packets, sent by the GBS, and by the continuous packets generated by CBR sources and by the GBS.

2.1 CONTROL packets

Since CRMA is a TDD protocol, CONTROL packets are necessary to maintain the synchronisation between all the stations. They are sent by the GBS, have a high priority level and are treated like speech packets. The number of control packets is about 10% of the total number of slots per frame [4].

2.2 CBR sources

TDMA is the channel access scheme followed by the CBR sources. Whenever a CBR source requires a connection, it accesses the channel and there is always an available timeslot for the transmission. No two CBR sources will be allocated the same timeslot, so when the CBR source wishes to send data, no collisions will occur. Therefore, assuming the channel error free, there is no need to acknowledge the good receipt of the packets sent by CBR sources. The CBR stations use a speech activity detector. Hence, the channel is occupied only when there is something to transmit and released during silent periods.

2.3 Continuous traffic timing

In figure 3 the timing diagram of a continuous transmission is given. The CBR source starts transmitting immediately at the beginning of the timeslot and keeps transmitting almost until the end of the slot. At the end of each slot a short time of $2a$ is left not utilised, where a is the worst case of propagation delay. A lot of measurements of indoor characteristics have been done. From [6] and [7], it can be concluded that the worst case of propagation delay in an indoor environment (maximum cell diameter 300

[4]) does not exceed 1 μs. This includes a multipath time delay spread of about 400 ns. In the further calculations, for security *a* is assumed to be 5 μs.

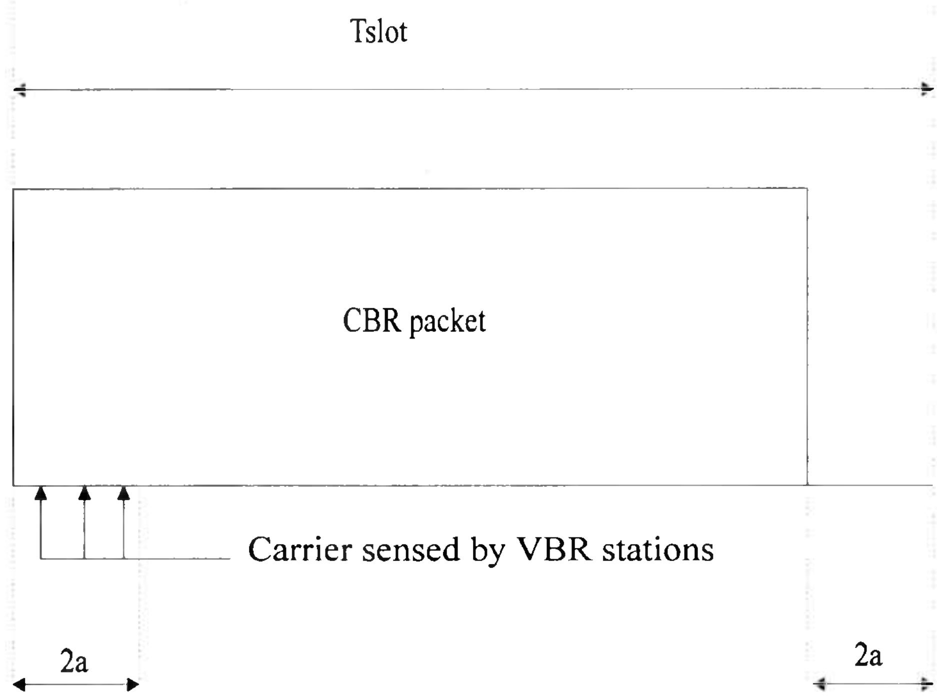

Figure 3: CBR packet timing.

The channel is sensed for a period of *2a* by the VBR stations in order to avoid collisions with the CBR packets.

Since we are using a slotted protocol, we have to take into account the problems of the stations synchronisation and propagation delay.

The stations synchronisation problem is solved by the use of CONTROL packets, which are sent regularly in fixed timeslots: their only function is to update the internal clock of the stations with the GBS's master clock.

As we can see in figure 3, at the end of each slot a portion of *2a* of the channel is not used for transmission because of the propagation delay.

The first a is accumulated in the transmission of the CONTROL packets: due to the physical distance between the GBS and the stations, perfect synchronisation is not feasible and there is always a maximum a error in the synchronisation. This error can not be neglected. In the actual transmission of a packet, again some propagation delay will be added. An extra silent gap must be considered to avoid collision of bits from packets sent in adjacent slots

Both above mentioned aspects result in the necessary gap of $2a$. For this reason, the CBR stations can not transmit for the complete timeslot.

2.4 Analytical model

For simplicity speech traffic is taken as representative of the continuous traffic. Speech traffic has a higher priority than data traffic and is independent from data traffic.

2.4.1 Model for the voice activity detector

CRMA uses a voice activity detector to improve the system performances by multiplexing speech and data. The major advantage of a voice activity detector is that the pauses in a conversation can be used to send data without degrading the quality of speech transmission. The model we describe is studied in [1] and [8]. In the simplest form of this model, a conversation is divided into periods of speech and periods of silence where the silence periods are due to pauses between words and sentences and pauses caused by listening to the other party. The duration of talkspurts and gaps can be approximated by an exponential distribution. For slotted time, the model can be described by the 2-state discrete Markov model shown in figure 4:

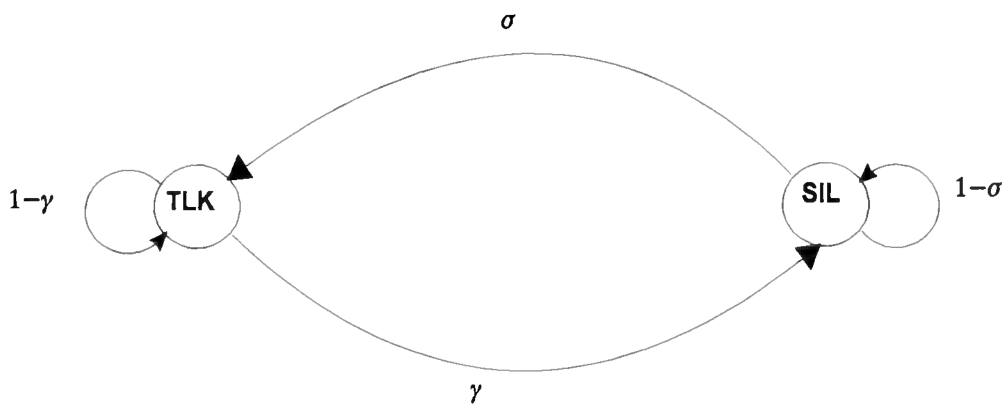

Figure 4: Model of the speech activity detector.

The probability σ of a transition from the silent to the talking state is equal to the probability of a silent period ending within a timeslot of duration T_{slot}:

$$\sigma = 1 - e^{-\frac{T_{slot}}{t_1}}. \tag{1}$$

where t_1 is the average duration of a silent period.

The probability γ of a transition from the talking to the silent state is equal to the probability of a spurt ending within a timeslot of duration T_{slot}:

$$\gamma = 1 - e^{-\frac{T_{slot}}{t_2}}. \tag{2}$$

where t_2 is the average duration of a talkspurt.

Given the model and the values of σ and γ, the values of $P\{TLK\}$ and $P\{SIL\}$, respectively the probability of finding a station talking or silent, are:

$$P\{TLK\} = \frac{\gamma}{\gamma + \sigma}. \tag{3}$$

$$P\{SIL\} = \frac{\sigma}{\gamma + \sigma}. \tag{4}$$

2.4.2 Analytical model for the continuous traffic

To analyse the CBR traffic we make the following assumptions:

- The destination terminals are assumed to belong to an infinite population. Therefore, the probability of blocking is neglected.
- For every CBR circuit two slots are assigned: one for the inbound and one for the outbound channel of the full duplex end to end connection.
- The arrival process for the calls to CBR stations is a Poisson process.
- The CBR stations service-time is exponentially distributed.

Under these hypothesises, we can model our system as a M/M/K/K/N queuing system (K: number of circuits per frame available to continuous traffic; N: number of CBR stations). The system is in state n when n circuits are active. Therefore, for λ_n (call arrival rate at the state n) and μ_n (call completion rate at the state n) are valid these formulas:

$$\lambda_n = (m-n)\lambda \quad \begin{cases} 0 \leq n \leq K \\ K \leq N \end{cases} \tag{5}$$

$$\mu_n = n\mu \quad 1 \leq n \leq N. \tag{6}$$

In figure 5 we show the transition diagram of this Markov model.

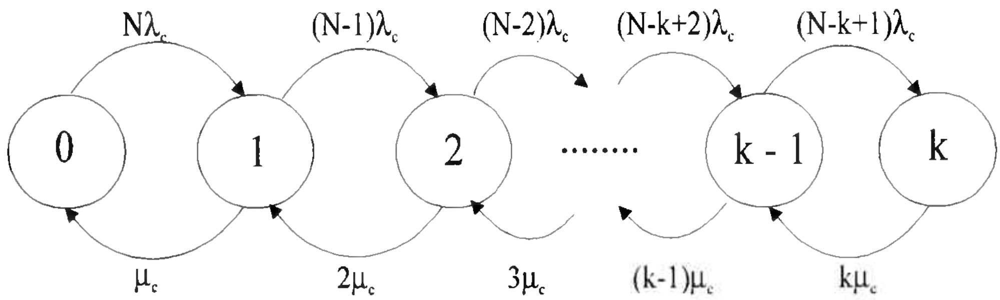

Figure 5: State diagram of the CBR model

The probability of being in the state *j* is [9]:

$$P\{x=j\} = \frac{\binom{N}{j}\left(\frac{\lambda}{\mu}\right)^j}{\sum_{i=0}^{k}\binom{N}{i}\left(\frac{\lambda}{\mu}\right)^i}. \quad (7)$$

The average number of CBR circuits is:

$$nr\_circ_{CBR} = \sum_{j=0}^{K} j \times P\{x=j\}. \quad (8)$$

Since every CBR circuit requires two slots and those two slots are occupied with a probability P*{TLK}*, the average number of slots occupied by CBR traffic is :

$$nr\_slots_{CBR} = 2 \times nr\_circ_{CBR} \times P\{TLK\}. \quad (9)$$

Our model has no waiting room, hence it is called a blocking system: if *K* calls are already in progress, any additional calls arriving are blocked [9].

The system engineer can decide the maximum number *K* of slots simultaneously occupied by the continuous traffic, to reserve a minimum level of the link capacity to the data traffic.

To determine the value of the blocking probability, we assume that any conversation, as long as it is active, occupies continuously only one timeslot. This corresponds to an even distribution of the channel between the two *talking* stations. For our purposes, considering that the value of *P{TLK}* is 43%, this assumption is reasonable. In reality, the number of timeslots occupied by a CBR circuit can be *0* (i.e. no one is speaking in that moment), *1* (i.e.only one station is transmitting) or *2* (i.e. the two callers are speaking simoultaneously). Under this hypothesis, the number of slots occupied by CBR traffic is equal to the number of active circuits.

Therefore, the blocking probability P_B , (i.e.the probability of being in state K) is the probability that CBR traffic occupies K slots.

$$P_B = P_K = \frac{\binom{N}{K}\left(\frac{\lambda}{\mu}\right)^K}{\sum_{i=0}^{k}\binom{N}{i}\left(\frac{\lambda}{\mu}\right)^i}. \quad (10)$$

The maximum number of slots for speech traffic is determined by the allowed blocking probability according the following formula:

$$P_B \triangleq \frac{1}{J_i(N, \lambda)}. \quad (11)$$

Where J_i is function of the maximum number of circuits i and can be calculated with the recursive procedure:

$$J_0 = 1. \quad (12)$$

$$J_i(N, \lambda) = J_{i-1}(N, \lambda) \times \frac{i \times \mu}{(N - i + 1) \times \lambda} + 1. \quad (13)$$

Where:

λ call arrival rate per speech station [min^{-1}].

μ call completion rate [min^{-1}].

3 VBR TRAFFIC

3.1 VBR sources

A VBR station can be one of the VBR terminals or the GBS. The probability of arrival of a packet at such a VBR station in a slot period is assumed constant and independent of previous arrival events.

Let x be the stochastic variable defined by the number of slots between the current slot and the slot in which the next packet arrives. If α is the probability of a packet arrival in one slot period, then the probability the first packet arrives in the i^{th} timeslot from the current timeslot can be calculated with:

$$P\{x = i\} = (1-\alpha)^{i-1} \times \alpha . \qquad (14)$$

This results in a geometrically distributed time between two data deliveries. This inter arrival time has an average value of $1/\alpha$ slot periods. For any of the VBR terminals, new packets arrive in a timeslot with probability P_{d0}. This arrival probability is P_{g0} for the GBS.

3.2 VBR traffic timing

Figure 6 shows the timing diagram for a VBR packet transmission. At the beginning of each timeslot in which there is a transmission, the VBR stations listen to the channel for a $2a$ period. As we have already seen for the description of CBR traffic timing, the station has to wait a time a because of the synchronisation lag and due to the propagation delay another a interval is on top of it to avoid any collision.

Contrary to the CBR transmission, there is a chance that the packet is lost, therefore it is necessary to acknowledge the good receipt of the transmission. Because of the TDD channel, the acknowledgement must be issued within the same timeslot of the transmission. This results in a minor portion of the channel available to data transmission. The timing for the acknowledgement is also shown in figure 6. In the worst case, an amount a of time is necessary for the propagation delay of the transmission and to it must be added the time spent for the signal processing.

As for the CBR packets the data transmission must end a period of *2a* before the end of the timeslot to avoid possible collisions.

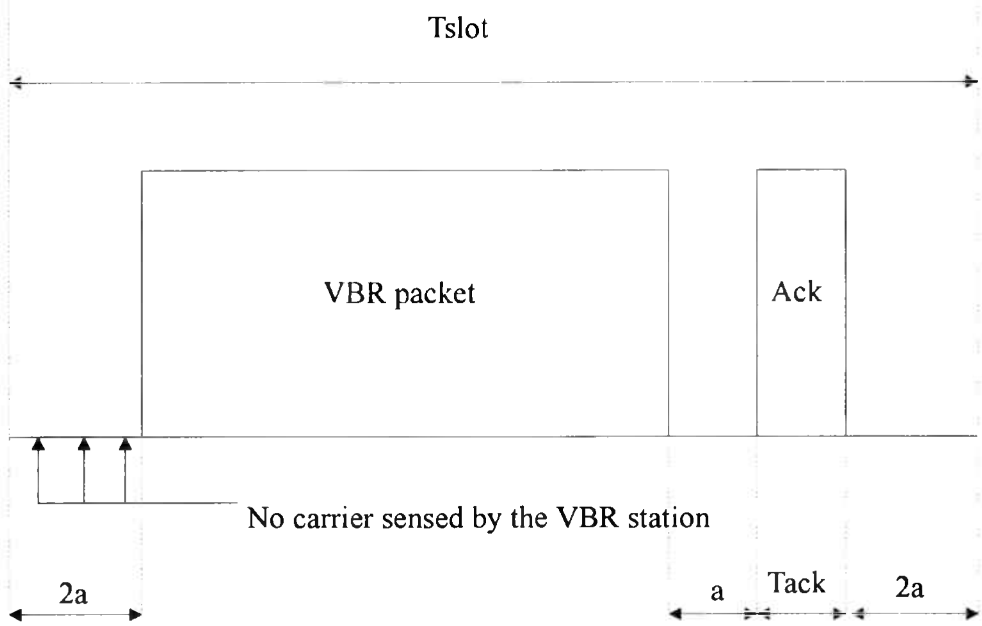

Figure 6: VBR packet timing

3.3 Contentions for the channel

If more than one station issues a request for the channel in the same timeslot there is a collision and all the packets will be lost.

When a VBR packet is lost, it has to be retransmitted because this kind of traffic is not allowed to have any data loss. The VBR terminals retransmit a lost packet with probability P_d. The GBS retransmits a lost packet with probability P_g.

If a VBR station tries to transmit a packet and the channel is already occupied by CBR traffic, there is no transmission.

The VBR station retransmits the packet in the first free slot with probability P_{wd} if it is a VBR terminal and with probability P_{wg} if it is the GBS.

The behaviour of a VBR station in the contention for the channel with CBR stations, is the main difference with the protocol in [4] and [5].

3.4 The VBR states

A VBR station has four different states: originating state (*ORIG*), transmitting state (*TRANS*), backlogged state (*BACK*) and the waiting state (*WAIT*).

If a VBR station attempts a transmission, it will only enter the transmission state until the end of the current slot period. At the end of a transmission slot, the station is in the *ORIG* state if the transmission has been successful, in the *BACK* state if there was a collision and in the *WAIT* state if a CBR station occupied the timeslot.

If the station is in backlogged state or in waiting state new packets are discarded. A model of the four-state Markov Chain of a VBR station is shown in figure 7.

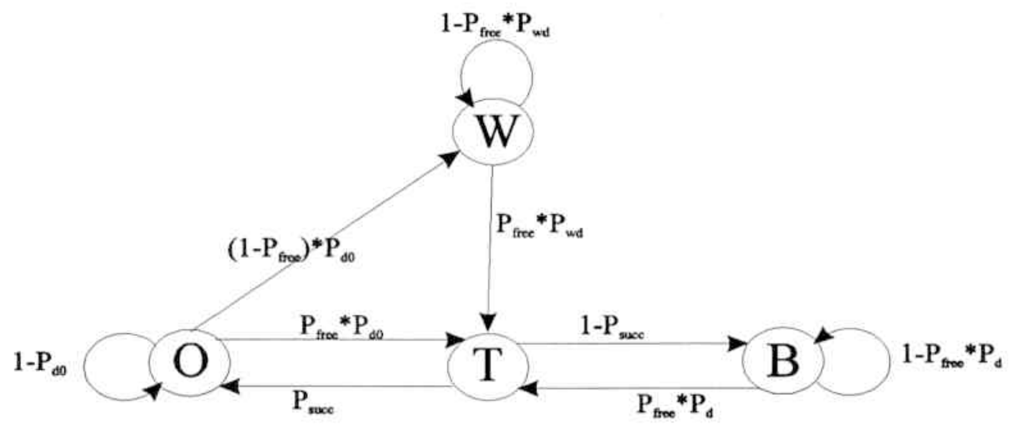

Figure 7: Single VBR terminal states and transitions

As can be seen from the figure, certain transitions are not present (for example: from *BACK* state to *WAIT* state), meaning their transition probabilities are zero.

When a VBR terminal is in *ORIG* state, it transmits with probability P_{d0}, that depends on the arrival rate of the external (inbound) traffic.

$$P_{d0} = \frac{inb\_outb \times P_{arr}}{nr_{VBR} \times (1 + inb\_outb)}. \qquad (15)$$

P_{arr}:	Packet arrival rate at the system. This rate is split into inbound and outbound traffic.
nr_{VBR}:	Number of VBR stations.
Inb_outb:	Ratio between inbound and outbound traffic.

The values of the retransmission probabilities P_d and P_{wd} are chosen by the system engineer to optimise the behaviour of the model.

The probability P_{succ} that a packet is successfully transmitted or not $(1\text{-}P_{succ})$, is dependent on the state of the Markov model.

P_{free} is the probability a slot is not occupied by CBR traffic:

$$P_{free} = \frac{slpfr - nr\_slots_{CTRL} - nr\_slots_{CBR}}{slpfr}. \qquad (16)$$

with:

slpfr:	Number of slots per frame.
nr_slots_{CTRL}:	Number of slots in every frame assigned to CONTROL data.
nr_slots_{CBR}:	Number of slots in every frame assigned to CBR stations.

3.5 The VBR station channel access mechanism

This section describes the access mechanism to the channel for a VBR terminal:

a) The station senses the channel for a CBR carrier during a period *2a* at the beginning of the slot.

b) If the channel is sensed *busy* there are two possibilities:

  i) If the station is in the *ORIG* state AND a new packet has arrived, the station enters the *WAIT* state.

  ii) If the station is in the *ORIG* state AND no new packet has arrived or the station is in the *BACK* or in the *WAIT* state, then the station does not change state and there is no transmission.

c) If the channel is sensed *idle* there are four possibilities:

  i) If the station is in the *ORIG* state AND a new packet has arrived, the packet is transmitted.

  ii) If the station is in the *ORIG* state AND no new packet has arrived, there is no transmission.

  iii) The station is in the *BACK* state: the backlogged packet is transmitted with probability P_d.

  iv) The station is in the *WAIT* state: the waiting packet is transmitted with probability P_{wd}.

When the GBS is transmitting data, the access mechanism is the same with different values of probability.

3.6 The Markov chain model of a CRMA system

At the end of every timeslot each VBR station will be in one of the three states (*ORIG*, *WAIT*, *BACK*). We can describe the system state by denoting the number of terminals in each of the terminal states.

We indicate with *o* the number of stations in *ORIG* state, with *p* the number of stations in *WAIT* state and with *q* the number of the station in *BACK* state.

With a fixed number nr_{VBR} of terminals in the system, the sum of $o+p+q$ is equal to nr_{VBR}, so to describe the system state only two of the three parameters are sufficient.

Since the GBS has different transmission probabilities, it can not be considered as the other stations. Therefore, with a *0*, *1*, *2* we indicate respectively that the GBS is in *ORIG*, *WAIT* or *BACK* state.

The system state is given by the vector (*p*, *q*, *x*), where:

p Number of the stations in *WAIT* state.

q Number of the stations in *BACK* state.

x The state of the GBS ($x = 0, 1, 2$).

3.7 System transition probabilities

Assuming that the stations are all independent from each other, the number *n* of stations in a certain state that transmits every new timeslot, follows the binomial distribution.

Therefore, if *m* is the total number of stations in a certain state, and *p* is the probability for each station to transmit:

P {*n stations are transmitting in the current time slot*} = bin (n, m, p).

It is important to mark out the statistical independence between the event "*A backlogged station decides to retransmit*" and the event "*A CBR station is occupying the channel*".

Every new timeslot, a backlogged station takes the decision whether to retransmit or not. As we have seen in the previous section, if the station 'decides' to retransmit and the channel is sensed idle the station completes the retransmission. If the channel is sensed busy, there is no retransmission and in the next timeslot the backlogged station decides again with the same probability P_d. The statistical independence of these two events allows us to consider:

P {*A backlogged station decides to retransmit / A CBR station is not occupying the channel*} = P {*A backlogged station decides to retransmit*} × P {*A CBR station is not occupying the channel*}.

The purpose of this section is to calculate the transition probabilities from $(p_{old}, q_{old}, x_{old})$ to $(p_{new}, q_{new}, x_{new})$.

Every state transition which is not covered by one of the following formulae has a transition probability equal to zero.

The possible values of p, q are:

$p \in [0, nr_{VBR}]$

$q \in [0, nr_{VBR}-p]$

The probability is presented in "words".

All the transitions can be divided in three main groups:

1) The state transitions with $x_{old} = 0$ (GBS in *ORIG* state).

2) The state transitions with $x_{old} = 1$ (GBS in *WAIT* state).

3) The state transitions with $x_{old} = 2$ (GBS in *BACK* state).

1) <u>The state transitions with $x_{old} = 0$ (GBS in *ORIG* state)</u>.

$(p, q, 0) \rightarrow (p, q, 0)$ (17)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap \text{all } BACK_{no\ trm}\}+$

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap 2 \text{ or more } BACK_{trm} \cap GBS_{no\ trm}\}+$

$P\{CBR_{no\ trm} \cap 1\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{no\ trm}\}+$

$P\{CBR_{trm} \cap \text{all } ORIG_{no\ trm} \cap GBS_{no\ trm}\}$

$(p, q, 0) \rightarrow (p-1, q+1, 0)$ (18)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm.} \cap 1\ WAIT_{trm} \cap 1 \text{ or more } BACK_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 0) \rightarrow (p, q+1, 0)$ (19)

$P\{CBR_{no\ trm} \cap 1\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm} \cap 1 \text{ or more } BACK_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 0) \rightarrow (p-n, q+n, 0)$ $n>1$; (20)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap n\ WAIT_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 0) \rightarrow (p, q+m, 0)$ $m>1$; (21)

$P\{CBR_{no\ trm} \cap m\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm} \cap GBS_{no\ trm}\}$

$(p, q, 0) \rightarrow (p-n, q+n+m, 0)$ $n>0, m>0$; (22)

$P\{CBR_{no\ trm} \cap m\ ORIG_{trm} \cap n\ WAIT_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 0) \rightarrow (p, q, 2)$ (23)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap 1 \text{ or more } BACK_{trm} \cap GBS_{trm}\}$

$(p, q, 0) \rightarrow (p-n, q+n, 2)$ $n>0;$ (24)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap n\ WAIT_{trm} \cap GBS_{trm}\}$

$(p, q, 0) \rightarrow (p, q+m, 2)$ $m>0;$ (25)

$P\{CBR_{no\ trm} \cap m\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm} \cap GBS_{trm}\}$

$(p, q, 0) \rightarrow (p-n, q+n+m, 2)$ $n>0, m>0;$ (26)

$P\{CBR_{no\ trm} \cap m\ ORIG_{trm} \cap n\ WAIT_{trm} \cap GBS_{trm}\}$

$(p, q, 0) \rightarrow (p-1, q, 0)$ (27)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap 1\ WAIT_{trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{no\ trm}\}$

$(p, q, 0) \rightarrow (p, q-1, 0)$ (28)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap 1 BACK_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 0) \rightarrow (p+m, q, 0)$ $m>0;$ (29)

$P\{CBR_{trm} \cap m\ ORIG_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 0) \rightarrow (p+m, q, 1)$ (30)

$P\{CBR_{trm} \cap m\ ORIG_{trm} \cap GBS_{trm}\}$

2) The state transitions with $x_{old} = 1$ (GBS in *WAIT* state)

$(p, q, 1) \rightarrow (p, q, 1)$ (31)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap \text{all } BACK_{no\ trm} + GBS_{no\ trm}\} +$

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap 2 \text{ or more } BACK_{trm} \cap GBS_{no\ trm}\} +$

$P\{CBR_{no\ trm} \cap 1\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{no\ trm}\} +$

$P\{CBR_{trm} \cap \text{all } ORIG_{no\ trm}\}$

$(p, q, 1) \rightarrow (p, q, 0)$ (32)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{trm}\}$

$(p, q, 1) \rightarrow (p-1, q+1, 1)$ (33)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm.} \cap 1\ WAIT_{trm} \cap 1 \text{ or more } BACK_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 1) \rightarrow (p, q+1, 1)$ (34)

$P\{CBR_{no\ trm} \cap 1\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm} \cap 1 \text{ or more } BACK_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 1) \rightarrow (p-n, q+n, 1)$ $n>1$; (35)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap n\ WAIT_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 1) \rightarrow (p, q+m, 1)$ $m>1$; (36)

$P\{CBR_{no\ trm} \cap m\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm} \cap GBS_{no\ trm}\}$

$(p, q, 1) \rightarrow (p\text{-}n, q\text{+}n\text{+}m, 1)$ $n>0, m>0;$ (37)

$P\{CBR_{no\ trm} \cap m\ ORIG_{trm} \cap n\ WAIT_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 1) \rightarrow (p\text{-}1, q, 1)$ (38)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap 1\ WAIT_{trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{no\ trm}\}$

$(p, q, 1) \rightarrow (p, q\text{-}1, 1)$ (39)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap 1\ BACK_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 1) \rightarrow (p, q, 2)$ (40)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap 1 \text{ or more } BACK_{trm} \cap GBS_{\ trm}\}$

$(p, q, 1) \rightarrow (p\text{-}n, q\text{+}n, 2)$ $n>0;$ (41)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap n\ WAIT_{trm} \cap GBS_{trm}\}$

$(p, q, 1) \rightarrow (p, q\text{+}m, 2)$ $m>0;$ (42)

$P\{CBR_{no\ trm} \cap m\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm} \cap GBS_{trm}\}$

$(p, q, 1) \rightarrow (p\text{-}n, q\text{+}n\text{+}m, 2)$ $n>0, m>0;$ (43)

$P\{CBR_{no\ trm} \cap m\ ORIG_{trm} \cap n\ WAIT_{trm} \cap GBS_{trm}\}$

$(p, q, 1) \rightarrow (p\text{+}m, q, 1)$ $m>0;$ (44)

$P\{CBR_{trm} \cap m\ ORIG_{trm}\}$

3) The state transitions with $x_{old}= 2$ (GBS in *BACK* state)

$(p, q, 2) \rightarrow (p, q, 2)$ (45)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm}\}$ -

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap 1\ BACK_{trm} \cap GBS_{no\ trm}\}$ -

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{trm}\}$ +

$P\{CBR_{no\ trm} \cap 1\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{no\ trm}\}$ +

$P\{CBR_{trm} \cap \text{all } ORIG_{no\ trm}\}$

$(p, q, 2) \rightarrow (p, q, 0)$ (46)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{trm}\}$

$(p, q, 2) \rightarrow (p-1, q+1, 2)$ (47)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap 1\ WAIT_{trm}\}$ -

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap 1\ WAIT_{trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{no\ trm}\}$

$(p, q, 2) \rightarrow (p, q+1, 2)$ (48)

$P\{CBR_{no\ trm} \cap 1\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm}\}$ -

$P\{CBR_{no\ trm} \cap 1\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{no\ trm}\}$

$(p, q, 2) \rightarrow (p-n, q+n, 2)$ $n>1$; (49)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap n\ WAIT_{trm}\}$

$(p, q, 2) \rightarrow (p, q+m, 2)$ $m>1$; (50)

$P\{CBR_{no\ trm} \cap m\ ORIG_{trm} \cap \text{all } WAIT_{no\ trm}\}$

$(p, q, 2) \rightarrow (p-n, q+n+m, 2)$ $n>0, m>0$; (51)

$P\{CBR_{no\ trm} \cap m\ ORIG_{trm} \cap n\ WAIT_{trm}\}$

$(p, q, 2) \rightarrow (p-1, q, 2)$ (52)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap 1\ WAIT_{trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{no\ trm}\}$

$(p, q, 2) \rightarrow (p, q-1, 2)$ (53)

$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap 1\ BACK_{trm} \cap GBS_{no\ trm}\}$

$(p, q, 2) \rightarrow (p+m, q, 2)$ $m>0$; (54)

$P\{CBR_{trm} \cap m\ ORIG_{trm}\}$

4 SYSTEM PERFORMANCES

4.1 The steady state

All transition probabilities (calculated with the formulas 17-54), are grouped together in the square matrix P. In this matrix the element in row x and column y denotes the transition probability from state x to state y, where x is the old state and y is the new state.

At any moment, the system is described by the state probability vector. Each elements of the state probability vector represents the probability the system is in a certain state.

As we have seen in the previous chapter, the system state is described by the three element vector (p, q, x). Therefore, the dimension of the state probability vector is $3 \times \frac{(nr_{VBR}+1)\times(nr_{VBR}+2)}{2}$. The GBS can be in three different states (*ORIG*, *WAIT* and *BACK*). For each value of x (GBS state), the number of states for VBR terminals is $\frac{(nr_{VBR}+1)\times(nr_{VBR}+2)}{2}$, since this is the number of the possible combinations of p (number of waiting stations) and q (number of backlogged stations), with p ranging between 0 and nr_{VBR} and q ranging between 0 and $nr_{VBR} - p$.

The state probability vector v_t at the time t, can be found by multiplying the previous state probability vector v_{t-1} at the time $t-1$ with the transition matrix P as in formula 55:

$$V_t = V_{t-1} \times P. \tag{55}$$

Assumed that the model is ergodic [10], the Markov model will end up in the steady state vector ($V_{steady\ state}$) after an infinite number of steps.

We can define $V_{steady\ state}$ in this way:

$$V_{steady\ state} = \lim_{t\to\infty} V_t. \tag{56}$$

Once the system is in the steady state, the state probability vector does not change anymore.

The formula (55) becomes:

$$V_{steady\ state} = V_{steady\ state} \times P. \tag{57}$$

This yields the following formula:

$$V_{steady\ state} \times (P - I) = 0_V. \tag{58}$$

This describes a set of $3 \times \frac{(nr_{VBR}+1)\times(nr_{VBR}+2)}{2}$ equations, which are not independent. To find the steady state vector it is therefore necessary to consider the following condition:

$$\sum_{x=0}^{2} \sum_{p=0}^{nr_{VBR}} \sum_{q=0}^{nr_{VBR}-p} v_{steadystate}(p,q,x) = 1. \tag{59}$$

Eventually, it is possible to calculate the value of the $V_{steady\ state}$ [11]:

1) Construct the matrix: $M = P\text{-}I$.

2) Replace the last column of M by ones.

3) Construct a new vector $E = [0, 0, \ldots\ldots, 0, 1]$.

4) Calculate $V_{steady\ state}$ by:

$$V_{steady\ state} = E \times M^{-1}. \tag{60}$$

4.2 System performances

This section studies the system performances by analysing two parameters:

a) *The system throughput.*

b) The VBR packet delay.

The system throughput

The throughput is defined as the mean number of successfully transmitted packets per timeslot.

The system throughput is the sum of the throughputs of the various types of traffic that the system integrates.

$$Syst_{thr} = VBR_{thr} + CBR_{thr.} + CTRL_{thr} \qquad (61)$$

The VBR traffic is the traffic produced by all the VBR terminals and by the GBS when transmitting VBR packets. To calculate the throughput for the VBR terminals and the GBS transmitting data, first we need to find the probability of a successful packet transmission.

The probability of a successful packet transmission by a VBR terminal $P_{succVBR}$ is given in 'words' by:

$$P_{succVBR}(p, q, x) = \qquad (62)$$

$$P\{CBR_{no\ trm} \cap 1\ ORIG_{trm} \cap all\ WAIT_{no\ trm} \cap all\ BACK_{no\ trm} \cap GBS_{no\ trm}\} +$$
$$P\{CBR_{no\ trm} \cap all\ ORIG_{no\ trm} \cap 1\ WAIT_{trm} \cap all\ BACK_{no\ trm} \cap GBS_{no\ trm}\} +$$
$$P\{CBR_{no\ trm} \cap all\ ORIG_{no\ trm} \cap all\ WAIT_{no\ trm} \cap 1\ BACK_{trm} \cap GBS_{no\ trm}\}.$$

To calculate the VBR throughput, it is necessary to find the mean value of $P_{succVBR}$, which can be obtained from the following formula:

$$P_{succVBR} = \frac{1}{nr_{VBR}} \times \sum_{x=0}^{2} \sum_{p=0}^{nr_{VBR}} \sum_{q=0}^{nr_{VBR}-p} PsuccVBR(p, q, x) \times P\{st = p,q,x\}. (63)$$

where:

$P_{succVBR}(p, q, x)$	is the value of $P_{succVBR}$ when the system state is (p, q, x).
$P\{st = p, q, x\}$	is the probability that the system state is (p, q, x).

Since the throughput of a single VBR terminal is equal to $P_{succVBR}$, the throughput of all the terminals is:

$$VBRterminals_{thr} = nr_{VBR} \times P_{succVBR}. \tag{64}$$

For the GBS the probability of a successful packet transmission $P_{succGBS}$, is given in 'words' by:

$$P_{succGBS} = \tag{65}$$

$$P\{CBR_{no\ trm} \cap \text{all } ORIG_{no\ trm} \cap \text{all } WAIT_{no\ trm} \cap \text{all } BACK_{no\ trm} \cap GBS_{trm}\}.$$

This leads to the formula:

$$GBS_{thr} = P_{succGBS} = \sum_{x=0}^{2} \sum_{p=0}^{nr_{VBR}} \sum_{q=0}^{nr_{VBR}-p} PsuccGBS\ (p,\ q,\ x) \times P\{st = p, q, x\}. \tag{66}$$

The continuous traffic is produced by the CBR terminals and the CONTROL packets. In this case, the continuous throughput is the number of slots occupied by the CBR and CONTROL packets divided by the number of slots per frame.

We are now able to express the system throughput as:

$$Syst_{thr} = \frac{nr\_slots_{CBR}}{slpfr} + \frac{nr\_slots_{CTRL}}{slpfr} + VBRterminals_{thr} + GBS_{thr} \tag{67}$$

Where:

nr_slots_{CBR}	number of slots occupied by CBR packets in a frame.
nr_slots_{CTRL}	number of slots occupied by CONTROL packets.
slpfr	number of timeslot per frame.

The VBR packet delay

The VBR packet delay has to be calculated for the packets transmitted by both the VBR terminals ($t_{del\ VBR}$) and the GBS ($t_{del\ GBS}$). In the calculations for the VBR packet delay, the influence of continuous traffic can not be neglected.

The continuous traffic affects the VBR traffic in two ways:

- If a VBR station in the *ORIG* state wants to transmit a packet and the channel is already occupied by continuous traffic, the station enters the *WAIT* state and the packet has to wait before being transmitted.
- If a VBR station is in the *WAIT* or *BACK* state, the station cannot always access the channel to retransmit a packet. This increases the average number of slot periods required for each retransmission.

The VBR packet delay is the result of the sum of four different factors:

a) Time necessary for synchronisation.

b) Time necessary for a successful transmission.

c) Time lost due to the channel being occupied by continuous type traffic.

d) Time lost due to collisions with other packets.

a) On average, a new packet has to wait half a slot period before it is synchronised with a timeslot.

b) Any successful transmission requires one timeslot to be accomplished.

c) The average time a packet is delayed due to collisions with continuous traffic can be calculated as the probability a VBR packet has to enter the *WAIT* state (i.e. to find the channel occupied when it tries to access the channel for the first time) multiplied by the average time spent in that state:

$$t_{del\ WAIT\ STATE\ VBR} = (1 - P_{free}) \times \frac{1}{P_{wd} \times P_{free}}. \quad (68)$$

$$t_{del\ WAIT\ STATE\ GBS} = (1 - P_{free}) \times \frac{1}{P_{wg} \times P_{free}}. \quad (69)$$

Where:

$(1 - P_{free})$	probability for a VBR station to enter the WAIT state.
$\frac{1}{P_{wd} \times P_{free}}$	average number of timeslots required to leave the WAIT state.

Since a station can access the *WAIT* state only from the *ORIG* state, it may seem strange that the probability of this transition is independent of the value of the probability of being in the *ORIG* state ($P\{ORIG\}$). By definition of our model, the probability that a packet is in the *ORIG* state when it enters the channel for the first time is equal to 1. Therefore, assuming the two types of traffic independent, the probability for a VBR packet to find the channel occupied when it is first transmitted, is equal to $1 - P_{free}$.

d) When a packet collides after a transmission, it takes on the average $\frac{1}{P_d \times P_{free}}$ (or $\frac{1}{P_g \times P_{free}}$ for the GBS) timeslots before a new transmission attempt is made. The average number of retransmissions accomplished by a VBR terminal or by the GBS is defined as $nr_{rtr\ VBR}$ and $nr_{rtr\ GBS}$ respectively.

Therefore, in timeslots the average packet delay due to collisions with other packets, is:

$$t_{del\ collision\ VBR} = nr_{rtr\ VBR} \times \frac{1}{P_d \times P_{free}}. \quad (70)$$

$$t_{del\ collision\ GBS} = nr_{rtr\ GBS} \times \frac{1}{P_g \times P_{free}}. \quad (71)$$

The number of retransmissions can be calculated as:

$$nr_{rtr} = \frac{\text{normalized carried load}}{\text{throughput}} - 1. \quad (72)$$

In the previous section we have seen that the throughput for a VBR terminal is $P_{succVBR}$ and $P_{succGBS}$ for the GBS.

The normalised carried load is the load generated by the newly arrived packets and the collided packets and it is equal to the probability that a station is in the TRANSMIT state (*TRANS*), defined as $P\{TRANS\}$. Hence, we can write formula 72 as:

$$nr_{rtr} = \frac{P\{TRANS\}}{P_{succ}} - 1. \quad (73)$$

From the formulas (62-66, 72-75) is interesting to note, that as a result of the model chosen, the average number of retransmissions for a packet is independent of P_{free} and the continuous traffic influences only the duration of each retransmission. Both $P\{TRANS\}$ and P_{succ} are the result of the multiplication of P_{free} by other factors. Therefore, we cross out P_{free}.

$$P\{TRANS\}_{VBR} = \quad (74)$$

$$P\{ORIG\}_{VBR} \times P_{d0} \times P_{free} + P\{WAIT\}_{VBR} \times P_{wd} \times P_{free} + P\{BACK\}_{VBR} \times P_d \times P_{free}.$$

$$P\{TRANS\}_{GBS} = \tag{75}$$

$$P\{ORIG\}_{GBS} \times P_{g0} \times P_{free} + P\{WAIT\}_{GBS} \times P_{wg} \times P_{free} + P\{BACK\}_{GBS} \times P_{g} \times P_{free}.$$

Where:

$P\{ORIG\} \times P_{d0} \times P_{free}$	Probability for a station in the ORIG state to go in the TRANS state.
$P\{WAIT\} \times P_{wd} \times P_{free}$	Probability for a station in the WAIT state to go in the TRANS state.
$P\{BACK\} \times P_{d} \times P_{free}$	Probability for a station in the BACK state to go in the TRANS state.

P{ORIG}, *P{WAIT}*, *P{BACK}* are all calculated as mean values:

$$P\{WAIT\} = \frac{1}{nr_{VBR}} \times \sum_{x=0}^{2} \sum_{q=0}^{nr_{VBR}} \sum_{p=o}^{nr_{VBR}-q} p \times P\{st = p,q,x\}. \tag{76}$$

$$P\{BACK\} = \frac{1}{nr_{VBR}} \times \sum_{x=0}^{2} \sum_{p=0}^{nr_{VBR}} \sum_{q=o}^{nr_{VBR}-p} q \times P\{st = p,q,x\}. \tag{77}$$

$$P\{ORIG\} = 1-[P\{WAIT\}+P\{BACK\}] \tag{78}$$

We are now able to find the average VBR packet delay:

$$t_{delVBR} = 1.5 + \frac{1 - P_{free}}{P_{wd} \times P_{free}} + \frac{nr_{rtr\,VBR}}{P_{d} \times P_{free}}. \tag{79}$$

$$t_{delGBS} = 1.5 + \frac{1 - P_{free}}{P_{wg} \times P_{free}} + \frac{nr_{rtr\,GBS}}{P_{g} \times P_{free}}. \tag{80}$$

5 OPTIMISING THE CRMA PROTOCOL

5.1 System Performance

The behaviour of the CRMA protocol depends on the values of many parameters. Due to its independence, the continuous traffic throughput is assumed constant in the calculation of the system performances. The CBR traffic throughput and the number of CONTROL packets, is used to find the value of P_{free}, as can be seen in formula (81):

$$P_{free} = 1 - CBR_{thr} - CTRL_{thr} \qquad (81)$$

To optimise the CRMA protocol, we consider three variables.

The first variable we use is the system throughput ($Syst_{thr}$). The second variable is the time delay for VBR packets (t_{delVBR}). The third variable is the time delay for the GBS (t_{delGBS}).

In the previous chapter we presented expressions for VBR_{thr}, GBS_{thr}, t_{delVBR} and t_{delGBS}.

The higher the throughput, the better the performance of the system. On the other hand, when the delays increase, the system performance decreases.

The system performance function ($Syst_{prfm}$), combines the three described variables into one formula:

$$Syst_{prfm} = Syst_{thr} - \left(GBS_{weight} \times t_{delGBS}\right) - \left(VBR_{weight} \times t_{delVBR}\right) \qquad (82)$$

In this formula:

GBS_{weight}: A weight factor factor for the GBS time delay. You can use this factor to set the ratio between the t_{delVBR} and t_{delGBS}.

When you increase this factor, the result of the optimisation will give a lower value for t_{delGBS}. The value of t_{delVBR} will be higher.

VBR_{weight}: A weight factor for the VBR time delay. You can use this factor to set the ratio between the t_{delVBR} and t_{delGBS}. An increased VBRweight factor results in a lower VBR delay and a higher GBS delay.

To evaluate the system performances of the CRMA protocol, we define two classes of parameters:

Class A: The parameters in this class, describe the environment in which the system is operating.

P_{arr}: P_{arr} is the packet arrival rate per slot at the system.

P_{free}: P_{free} is the probability that a slot is available for the next datapacket transmission.

Inb_outb: We define the inbound traffic as the data traffic from a terminal towards the GBS and the outbound traffic as the traffic from the GBS towards a terminal. The *inb_outb* parameter expresses a ratio between the two types of traffic. P_{d0} and P_{g0} are set by values assigned to P_{arr} and *inb_outb*.

nr_{VBR}: Number of VBR terminals in the cell.

nr_{CBR}: Number of CBR users in the cell.

P_B: Maximum allowed blocking probability for CBR sources.

λ_C: CBR call arrival rate for every free CBR source [hour $^{-1}$].

μ_C: Call completion rate [min $^{-1}$].

τ_{tlk}: Average length of a talkspurt.

τ_{sil}: Average length of a silent period.

Class B: The parameters in this class are the parameters on which the system engineer can operate to improve the system performances.

P_d, P_g:

P_d and P_g are respectively the probability of retransmission for a backlogged terminal and a backlogged GBS.

P_{wd}, P_{wg}:

P_{wd} and P_{wg} are the probabilities of retransmission for a terminal and the GBS when they are in the waiting state.

5.2 <u>The penalty function</u>

The best way to optimise the protocol performances is choosing proper values for P_d, P_g ,P_{wd} and P_{wg}. Maximisation of the $Syst_{prfm}$ gives the best solution.

To maximise the $Syst_{prfm}$ we can minimise:

$$f(x) = 1 - \text{Syst}_{\text{prfm}} \tag{83}$$

This is a constrained problem. Since we can perform unconstrained minimisations more easily than constrained minimisations, we try to convert the constrained problem into an unconstrained problem. There are several techniques for converting constrained optimisation problems. A method which has a wide applicability is the penalty function method [11].

The basic idea of the penalty function method, is to modify the objective function *f(x)* in such a way, that the values of the modified objective function are equal to the objective function within the range of interest. Outside this region, the values of the modified objective function are very large compared with those of the objective function.

We can modify *f(x)* by adding a function of the constraint equations.

In our case, an example of the ideal penalty function would be two delta functions $\delta(x)$ and $\delta(x-1)$.

A good approximation of this ideal function, in case the constraint is $X1 \le x \ge X2$, would be a function of the form :

$$p(x) = c \times \left[\frac{2x - (X1 + X2)}{X2 - X1}\right]^p \tag{84}$$

In this formula:

X1: the left border of the allowed interval.

X2: the right border of the allowed interval.

c: a constant to set the function to the right value between *a* and *b*.

p: the power of the function.

The parameters to be optimised are probabilities, and they must vary in the interval of interest. Therefore in our case, *X1* is equal to 0 and *X2* is equal 1. In order to get the steepest curve possible, for *c* and *p* we choose ½ and 100 respectively. This way the following penalty function is obtained:

$$p(x) = \frac{1}{2} \times (2x - 1)^{100} \tag{85}$$

This penalty function is shown in figure 8.

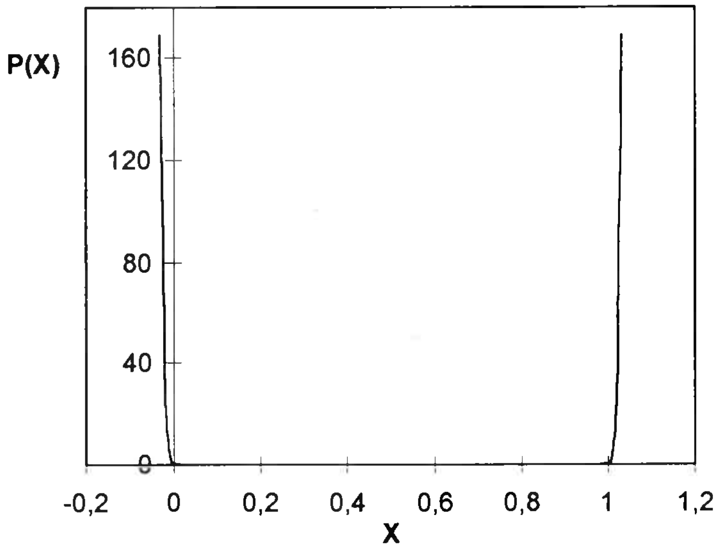

Figure 8: Penalty function $p(x) = \frac{1}{2} \times (2x - 1)^{100}$.

When we apply the penalty function for the four different retransmission probabilities in the $Syst_{prfm}$ function, we obtain the following formula which has to be optimised:

$$\mathrm{Syst}_{\mathrm{prfm}} = \mathrm{Syst}_{\mathrm{thr}} - \left(\mathrm{GBS}_{\mathrm{weight}} \times \mathrm{t}_{\mathrm{delGBS}}\right) - \left(\mathrm{VBR}_{\mathrm{weight}} \times \mathrm{t}_{\mathrm{delVBR}}\right) - \left\{\frac{1}{2}\left(2\mathrm{P}_{\mathrm{d}} - 1\right)^{100} + \frac{1}{2}\left(2\mathrm{P}_{\mathrm{wd}} - 1\right)^{100} + \frac{1}{2}\left(2\mathrm{P}_{\mathrm{g}} - 1\right)^{100} + \frac{1}{2}\left(2\mathrm{P}_{\mathrm{wg}} - 1\right)^{100}\right\} \quad (86)$$

In case x violates a constraint, a certain amount of penalty is incurred by $Syst_{prfm}$ to increase its value.

We can use the MATLAB built-in minimisation function 'FMINS', to optimise the protocol performances. This function uses a Simplex search method and needs two input vectors. The first vector contains the starting values for P_d, P_{wd}, P_g and P_{wg}. The control parameters, for instance the maximum number of steps and the termination tolerance, can be set in the second vector.

5.3 The computer program

The first intention was to write the program in PASCAL. To handle matrices with a dimension greater than one hundred by one hundred appeared not to be possible in PASCAL. Therefore, we changed the language to C. In C there are no constraints for the size of a matrix.

The program calculates the protocol performances for thirteen stations and a GBS. It is not possible to increase the number of stations. The maximum size of a variable in DOS is 1 Mb (20 bits addresses), and when the number of stations is greater than thirteen, the memory size of the matrix becomes too large for a variable.

The computer program can be divided in three different parts:

1. The first part is written in C. It starts with the initialisation of the variables and parameters. After the initialisation, the state transition probabilities are calculated and filled in a matrix (P).

   The use of pointers made it possible to locate the variables and parameters as P_{arr}, P_{free} and *inb_outb*, in a data file 'DATA.DAT'.

   This way they can be easily changed for another optimisation.

2. The second part is a MATLAB program, that inverts the state transition probability matrix, in order to get the steady state vector.

   The MATLAB built-in matrix inversion routine takes advantage of the fact that the state transition probability matrix contains a lot of zeros.

   A procedure called 'SPARSE' converts a full matrix to a 'SPARSE' form, by squeezing out the zero elements.

   The benefit of this storage method is the memory reduction that can be accomplished. Since only 15 % of our matrix consists of non-zero elements, a lot of memory space can be saved. Once the inverse of the matrix is known, the steady state vector can be found from formula (60).

3) The final part of the program is written in C and calculates the throughput and the average VBR packet delay of the CRMA protocol. In order to find the values for the delay and throughput the following steps are taken :

- The probability of being in the three different states *ORIG*, *WAIT* and *BACK* are calculated according to the formulas (76) - (78).
- We can now obtain the number of retransmissions using formulas (73)-(75).
- Finallay we can fill the results in the expressions for the t_{delVBR} and t_{delGBS} (formulas (79) and (80)).

The results are saved in a file 'RESULTS.DAT'.

6 RESULTS

In this chapter, we will comment on the graphs with the optimised results for the CRMA protocol performances.

Pfree

For *Pfree*, as significant values we have chosen 0.35, 0.60, 0.85 to represent the three cases:

- the channel is mainly occupied by continuous traffic (*Pfree* = 0.35),
- the amount of CBR traffic is almost equal to the data traffic (*Pfree* = 0.60),
- the channel is almost fully available for data traffic (*Pfree* = 0.85).

Inb_outb

To simulate the different cases whether the stream of data is going towards the GBS or the terminals, 0.5, 1.0, 2.0 are assumed as representative values of *inb_outb*.

The throughput and the packet delay are calculated for values of P_{arr} ranging from 0.1 to 0.9. The results are plotted in graphical form, where P_{arr} is the abscissa and the throughput or the time delay (in time slots) is the ordinate.

6.1 Graphs

On each page two figures are shown. In the top figure the system throughput is shown, the bottom figure shows the time delay. In the throughput diagram, the fraction of the total link capacity used for VBR traffic, GBS traffic and the total link utilisation (system throughput) are marked out.

In the delay plots, the average delay for a VBR packet and a GBS packet are marked out. In both cases, the solid lines are the results of the analysis and the blocks and triangles mark the results of the simulations.

If we look at the plots with the results of the analysis and the simulations, it points out that both models match very well. Therefore it is assumed that the results of the analytical model can be used for evaluating the CRMA protocol performances.

The plots will be discussed hereafter, to find out the behaviour of the CRMA protocol for different values of P_{free} and the *inb_outb* ratio.

For all the simulations, frame sizes of 41 slots per frame (*slpfr* = 41) are considered, and the number of CONTROL slots is embedded in the continuous traffic throughput.

6.2 Comments on the graphs

- *Pfree = 0.85*

In this case, the channel is almost completely available for data traffic. When GBS and VBR packets are treated equally, the GBS packets will suffer a lower delay than the VBR packets, because all packets transmitted by the GBS are placed in different timeslots and therefore will not collide with eachother. In order to compensate the expected higher VBR delays, we must increase the weight factor for the VBR timedelay VBR_{weight} and decrease GBS_{weight}.

Since the curve of the average VBR time delay appeared to be very steep, we should run the optimisation for a relatively high packet arrival rate in order to reduce the delay as much as possible.

The shown graphs (9-14), are obtained for the following parameter settings:

VBR_{weight}:	0.002
GBS_{weight}:	0.001
P_{arr}:	0.75

As expected, when the *inb_outb* ratio is equal to 2.0, the VBR throughput is greater than the GBS throughput, because in this case the VBR terminals are less likely to find a free transmission slot. Therefore they are less frequently backlogged and the throughput is high.

The maximum packet delay for the VBR terminals is 27 timeslots and 20 timeslots for *inb_outb* ratio's of 2.0 and 0.5 respectively.The GBS suffers a maximum packet delay of 21 timeslots. The maximum $Syst_{thr}$ is achieved for an inb_outb of 0.5, and is equal to 0.498.

Pfree:	0.85	Parr:	0.70
Inb outb:	2.0	VBR_{thr}:	0.269
Pd:	0.116	GBS_{thr}:	0.054
Pwd:	0.041	$Syst_{thr}$:	0.473
Pg:	0.115	T_{delVBR}:	21.9
Pwg:	0.156	T_{delGBS}:	15.8

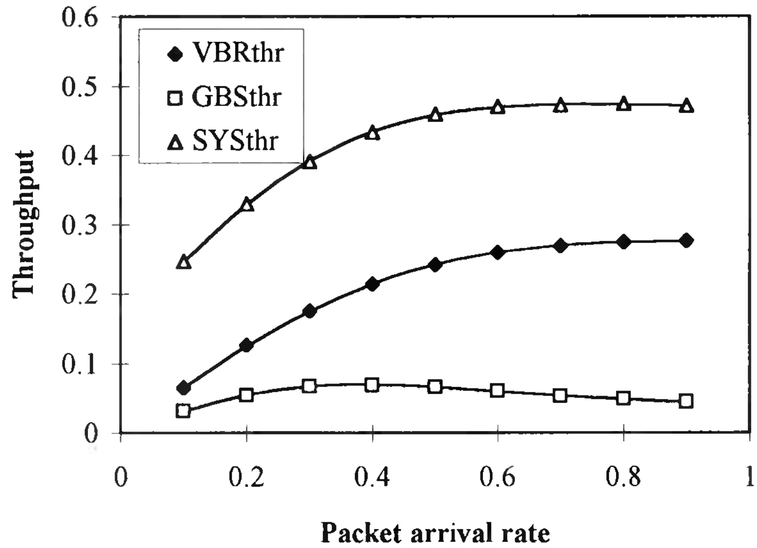

Figure 9: Throughput for a CRMA system (P_{free} = 0.85 and inb_outb = 2.0).

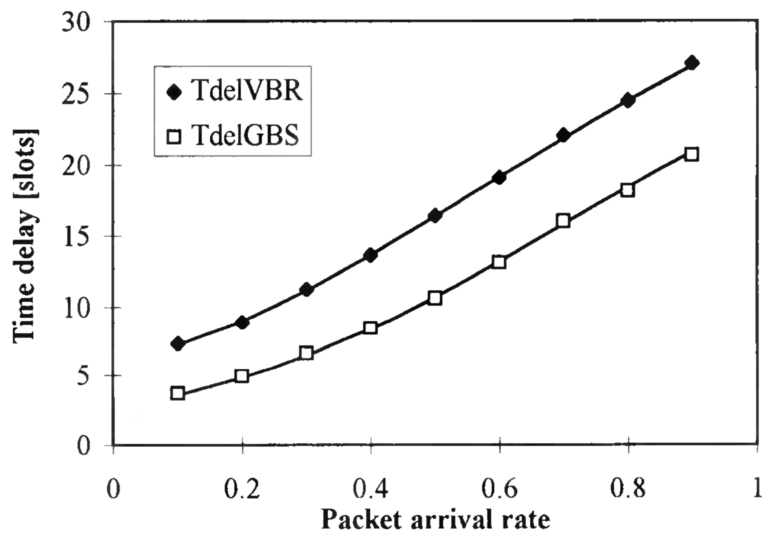

Figure 10: Time delay for a CRMA system (P_{free} = 0.85 and inb_outb = 2.0).

Pfree:	0.85	Parr:	0.70
Inb outb:	1.0	VBR_{thr}:	0.242
Pd:	0.123	GBS_{thr}:	0.081
Pwd:	0.056	$Syst_{thr}$:	0.473
Pg:	0.135	T_{delVBR}:	18.0
Pwg:	0.176	T_{delGBS}:	11.0

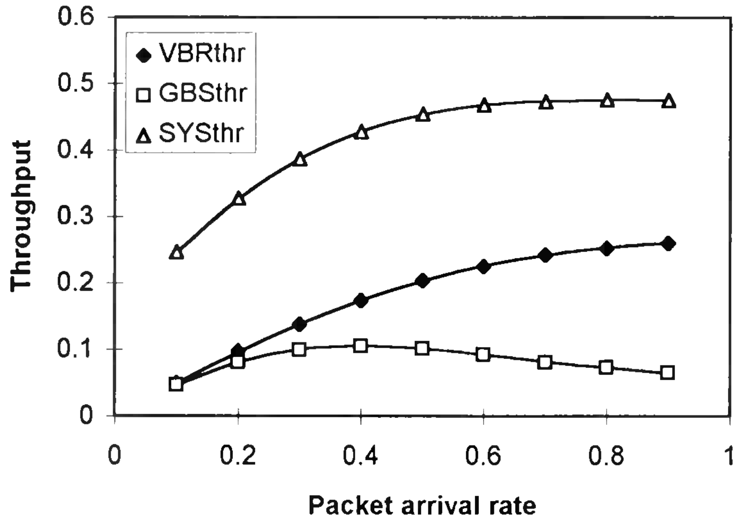

Figure 11: Throughput for a CRMA system (P_{free} = 0.85 and inb_outb = 1.0).

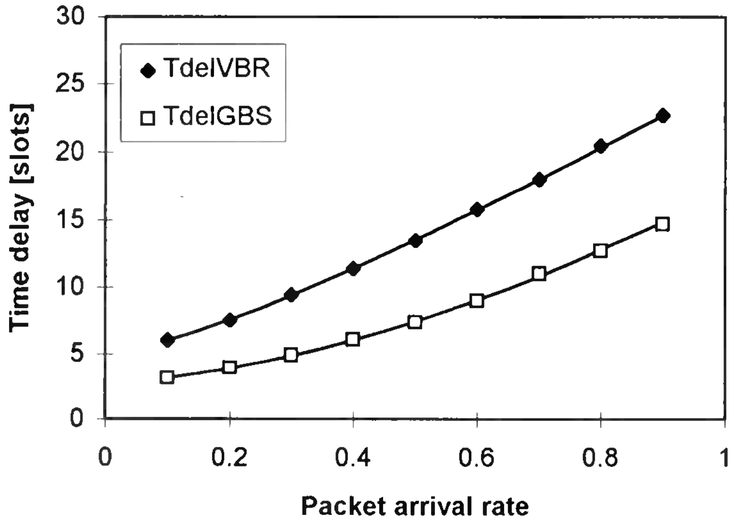

Figure 12: Time delay for a CRMA system (P_{free} = 0.85 and inb_outb = 1.0).

Pfree:	0.85	Parr:	0.70
Inb outb:	0.5	VBR_{thr}:	0.188
Pd:	0.154	GBS_{thr}:	0.159
Pwd:	0.144	$Syst_{thr}$:	0.497
Pg:	0.222	T_{delVBR}:	15.1
Pwg:	0.940	T_{delGBS}:	5.6

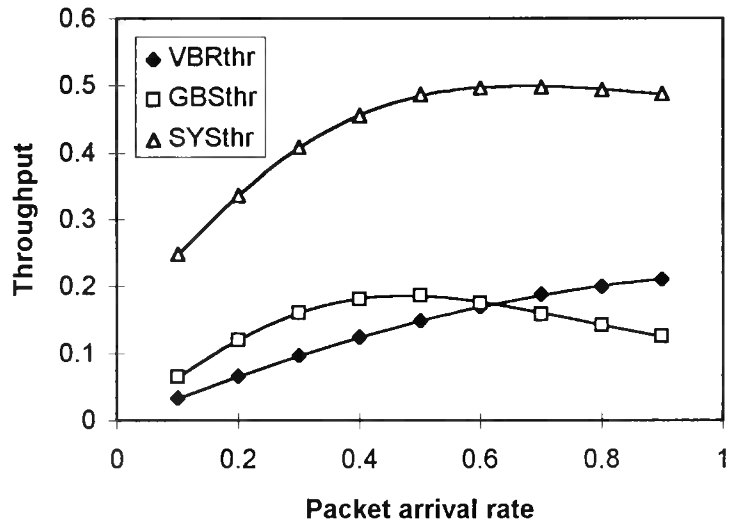

Figure 13: Throughput a CRMA system (P_{free} = 0.85 and inb_outb = 0.5).

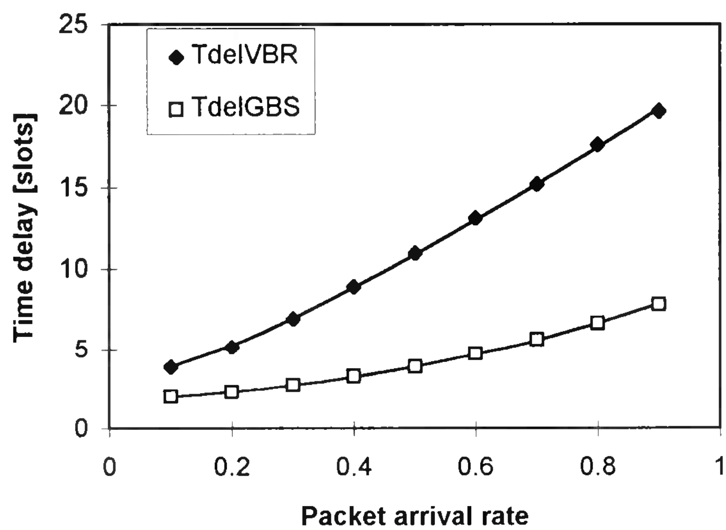

Figure 14: Time delay for a CRMA system (P_{free} = 0.85 and inb_outb = 0.5).

- *Pfree = 0.60*

We optimized the system for the following parameter settings:

VBR_{weight}:	0.0012
GBS_{weight}:	0.0008
P_{arr}:	0.60

The *SYSthr* is equal to 0.63, and is nearly constant for the three different inb_out ratio's. When the inb_outb ratio decreases from 2.0 to 0.5, the VBR_{thr} decreases and the GBS_{thr} increases, as expected.

The average VBR packet delay has a maximum of 53 slots, the GBS suffers a maximum delay of 17 slot periods.

Pfree:	0.60	Parr:	0.60
Inb outb:	2.0	VBR_{thr}:	0.176
Pd:	0.081	GBS_{thr}:	0.056
Pwd:	0.093	$Syst_{thr}$:	0.633
Pg:	0.179	T_{delVBR}:	42.7
Pwg:	0.511	T_{delGBS}:	14.2

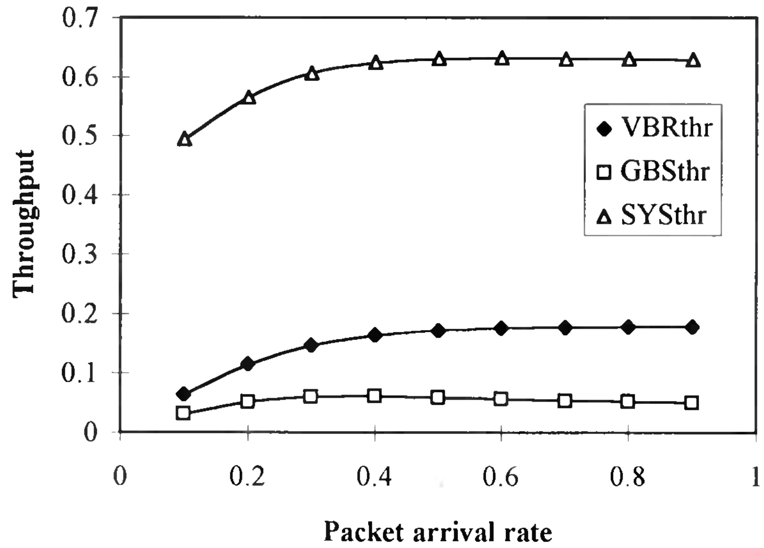

Figure 15: Throughput for a CRMA system (P_{free} = 0.6 and inb_outb = 2.0).

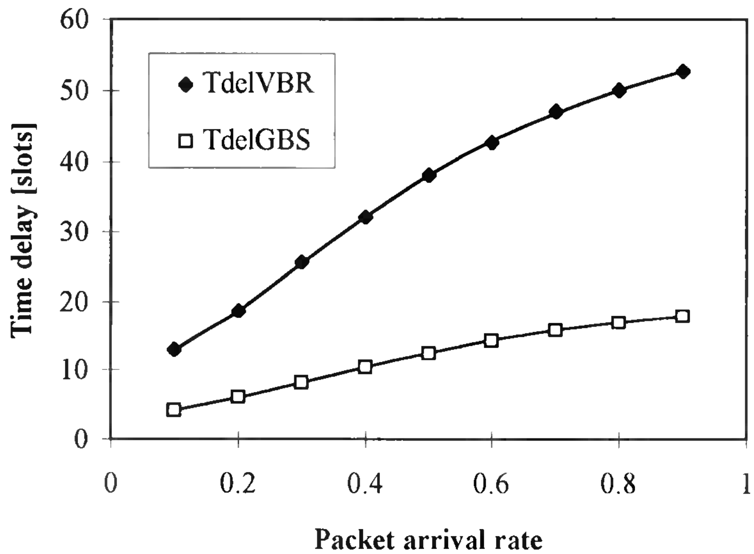

Figure 16: Time delay for a CRMA system (P_{free} = 0.6 and inb_outb = 2.0).

Pfree:	0.60	Parr:	0.60
Inb outb:	1.0	VBR_{thr}:	0.170
Pd:	0.102	GBS_{thr}:	0.062
Pwd:	0.127	$Syst_{thr}$:	0.633
Pg:	0.164	T_{delVBR}:	34.4
Pwg:	0.684	T_{delGBS}:	14.3

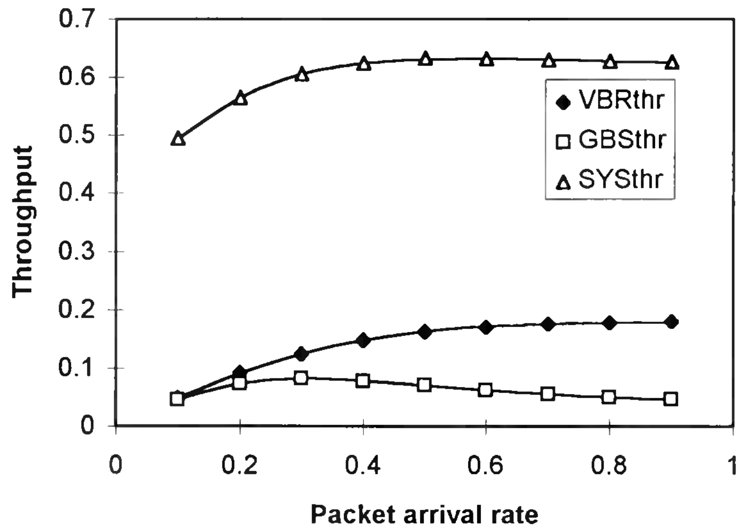

Figure 17: Throughput for a CRMA system (P_{free} = 0.6 and inb_outb = 1.0).

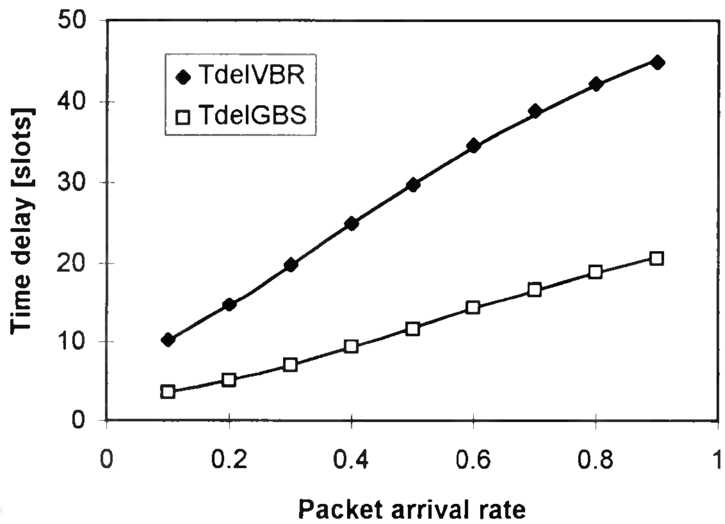

Figure 18: Time delay for a CRMA system (P_{free} = 0.6 and inb_outb = 1.0).

Pfree:	0.60	Parr:	0.60
Inb outb:	0.5	VBR_{thr}:	0.146
Pd:	0.147	GBS_{thr}:	0.094
Pwd:	0.160	$Syst_{thr}$:	0.639
Pg:	0.206	T_{delVBR}:	25.7
Pwg:	0.77	T_{delGBS}:	9.7

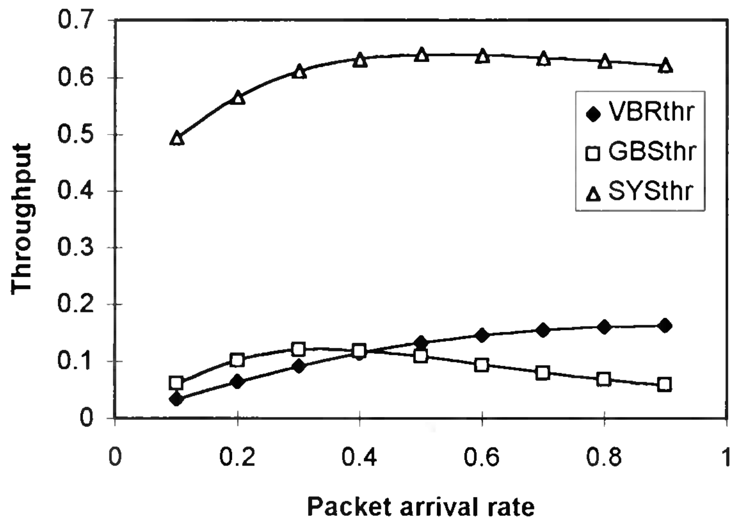

Figure 19: Throughput for a CRMA system (P_{free} = 0.6 and inb_outb = 0.5).

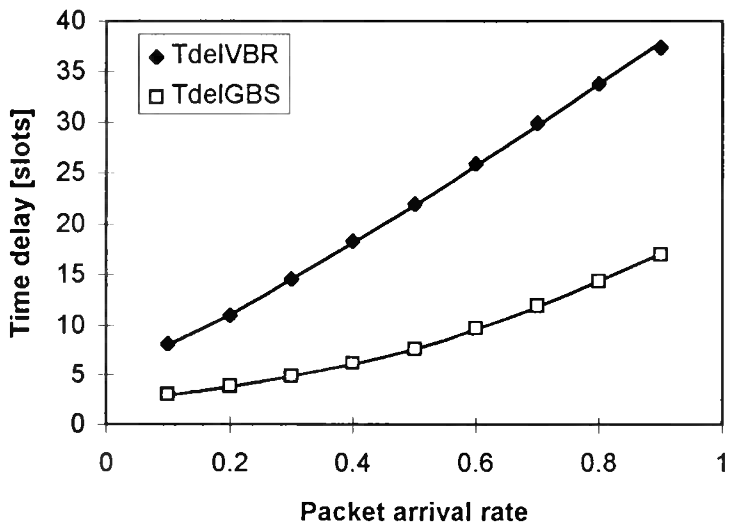

Figure 20: Time delay for a CRMA system (P_{free} = 0.6 and inb_outb = 0.5).

- *Pfree = 0.35*

The shown graphs are obtained for the following parameter settings:

VBR_{weight}:	0.0012
GBS_{weight}:	0.0008
P_{arr}:	0.60

In case the value for P_{free} is 0.35, the channel is mainly occupied by CBR traffic. Figures (21-26) show that the throughput only varies for a low packet arrival rate. For higher packet arrival rates ($P_{arr} > 0.7$) the channel will become saturated, because of the heavy CBR traffic occupation.

For an arrival rate of 0.6, (the value for which we optimised) the protocol behaves as expected. The VBR throughput decreases and the GBS throughput increases when the *inb_outb* ratio changes from 2.0 to 0.5.

The maximum VBR and GBS delay is greater in this case, because the channel is almost fully available for speech traffic.

Pfree:	0.35	Parr:	0.60
Inb outb:	2.0	VBR_{thr}:	0.108
Pd:	0.073	GBS_{thr}:	0.028
Pwd:	0.079	$Syst_{thr}$:	0.785
Pg:	0.156	T_{delVBR}:	89.8
Pwg:	0.261	T_{delGBS}:	32.6

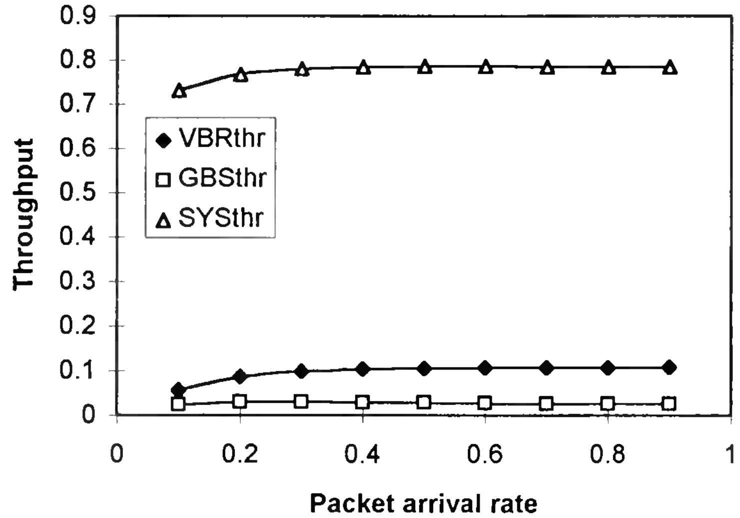

Figure 21: Throughput for a CRMA system (P_{free} = 0.35 and inb_outb = 2.0).

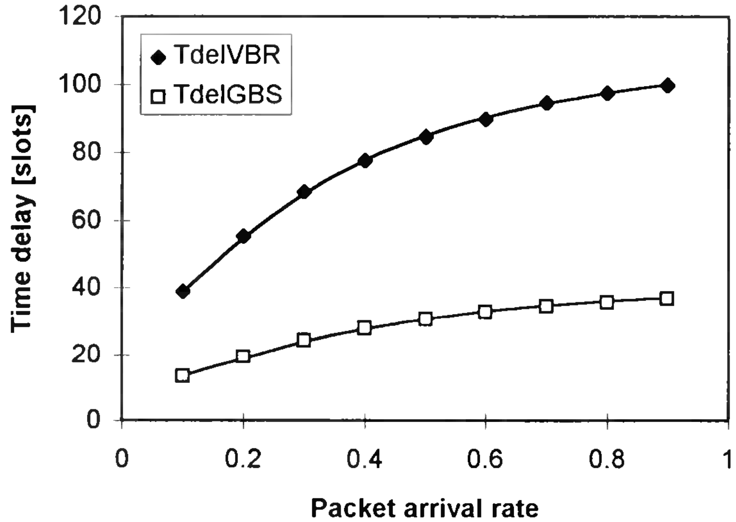

Figure 22: Time delay for a CRMA system (P_{free} = 0.35 and inb_outb = 2.0).

Pfree:	0.35	Parr:	0.60
Inb outb:	1.0	VBR_{thr}:	0.107
Pd:	0.084	GBS_{thr}:	0.028
Pwd:	0.089	$Syst_{thr}$:	0.785
Pg:	0.149	T_{delVBR}:	80.3
Pwg:	0.276	T_{delGBS}:	33.5

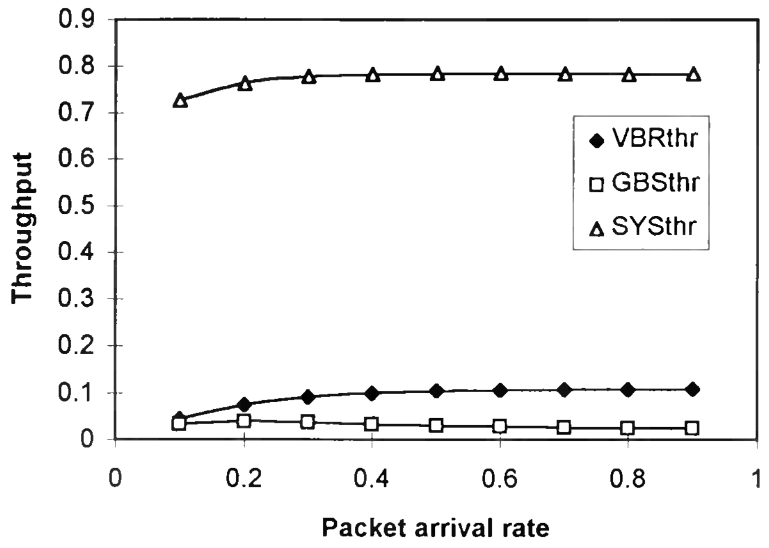

Figure 23: Throughput for a CRMA system (P_{free} = 0.35 and inb_outb = 1.0).

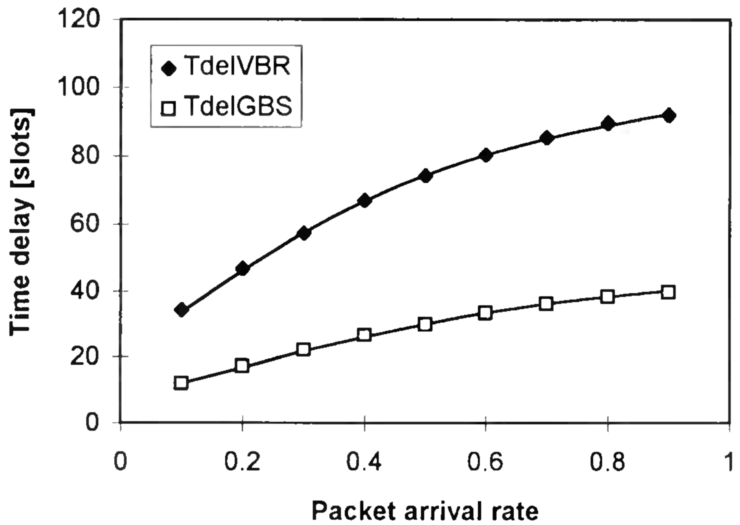

Figure 24: Time delay for a CRMA system (P_{free} = 0.35 and inb_outb = 1.0).

Pfree:	0.35	Parr:	0.60
Inb outb:	0.5	VBR_{thr}:	0.105
Pd:	0.106	GBS_{thr}:	0.029
Pwd:	0.115	$Syst_{thr}$:	0.784
Pg:	0.123	T_{delVBR}:	60.5
Pwg:	0.335	T_{delGBS}:	32.9

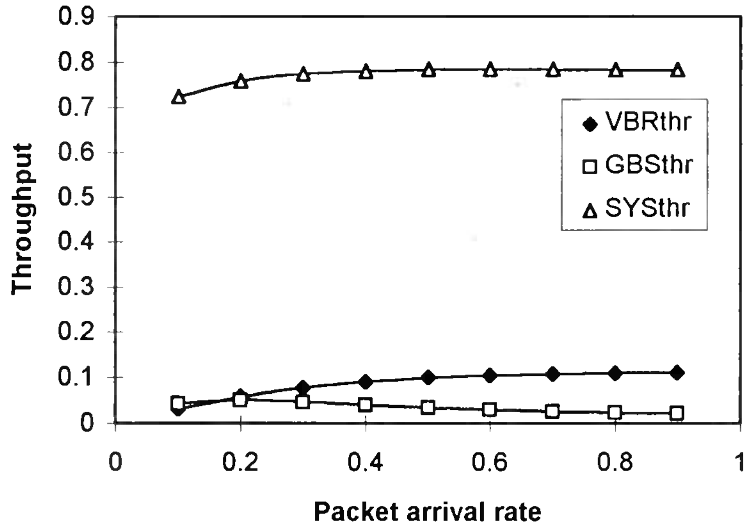

Figure 25: Throughput for a CRMA system (P_{free} = 0.35 and inb_outb = 0.5).

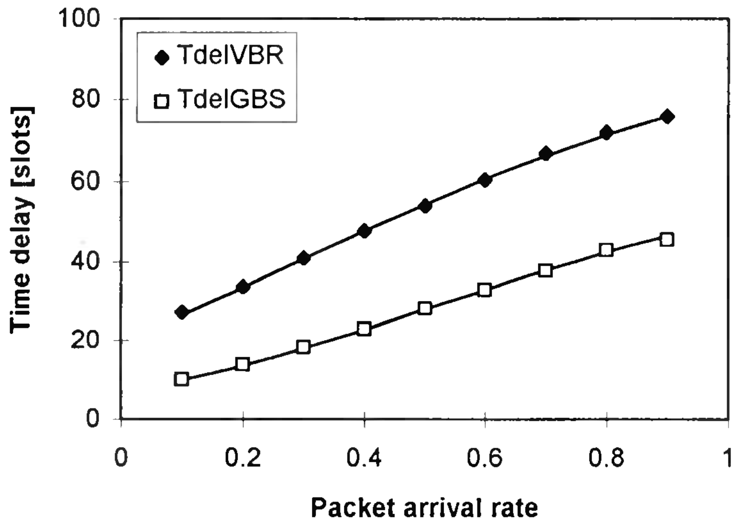

Figure 26: Time delay for a CRMA system (P_{free} = 0.35 and inb_outb = 0.5).

6.3 System throughput

It appears that the more of the total traffic stream is produced by the GBS, the greater the total throughput can be. For an *inb_outb* ratio of 2.0, 1.0 and 0.5, the maximum link utilisation can be 32%, 33% and 35% respectively (for $P_{free} = 0.85$). The difference between these percentages can be greater when for example 10, 5, 1 and 0.1 are taken as values for the *inb_outb* ratio.

These percentages of link utilisation show what we expected, because packets transmitted by the GBS are placed in different timeslots, and therefore will not collide with each other.

The control packets sent by the GBS are taken into account in the calculation of the system throughput, as can bee seen in formula (61).

Therefore, we should subtract 10% from the system throughput in order to get a throughput that represents the number of information packets that has been successfully transmitted.

As can be seen from the graphs, with higher CBR traffic the total channel utilisation increases, but the VBR and GBS throughput decreases.

Figure 27 shows the system throughput for different inb_outb ratio's and packet arrival rates. It is obvious that the higher the portion of the channel that is available for continuous traffic, the higher the system throughput will be. The inb_outb ratio has an impact on the performances only when the channel is mainly available for data traffic.

The VBR and GBS throughput appears to be the same fraction of the remaining link capacity, after the CBR portion has been subtracted from the link capacity.

CBR utilisation of the channel	Remaining capacity	Percentage of the remaining capacity used by VBR/GBS
15 %	85 %	38.0 %
40 %	60 %	38.7 %
65 %	35 %	38.5 %

Table 1: Percentage of the channel used by VBR and GBS traffic.

The sum of the VBR and GBS traffic throughput, has a maximum equal to the ideal throughput of a Slotted ALOHA system. This is 0.368 [13] in case the number of stations is infinite. In our simulations, we use 13 stations and therefore the throughput can be greater than this theoretical upperlimit.

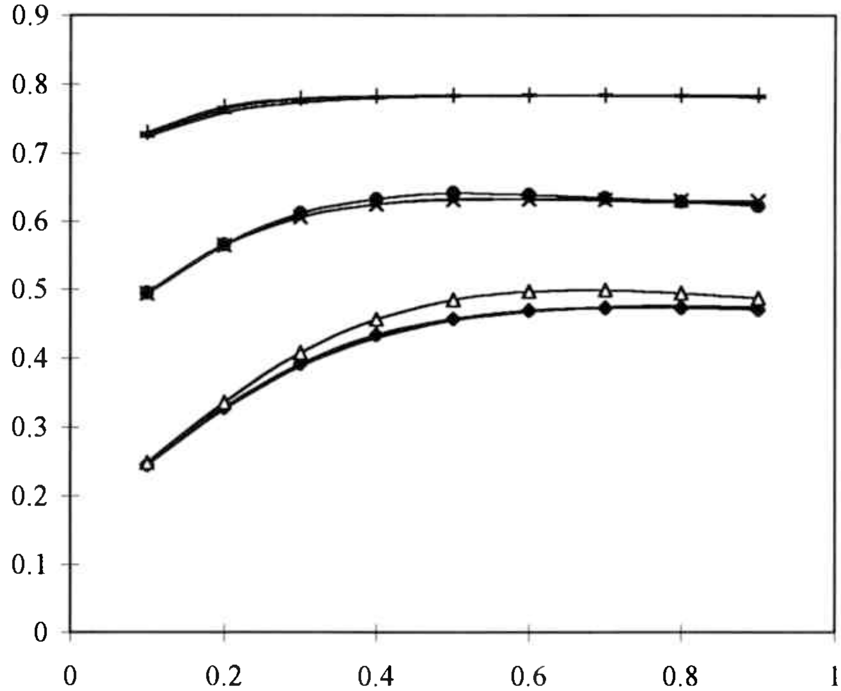

Figure 27: Throughput for a CRMA system (P_{free} = x, inb_outb = y).

Marker	X	Y
◆	0.85	2.0
▲	0.85	1.0
■	0.85	0.5
●	0.60	2.0
✖	0.60	1.0
	0.60	0.5
+	0.35	2.0
I	0.35	1.0
–	0.35	0.5

7 EVALUATION AND COMPARISON WITH PRMA

After the performance analysis, it is possible to present a final evaluation of the CRMA protocol and to make a qualitative comparison with PRMA.

7.1 Evaluation of the CRMA protocol

In this section, we present the main characteristics of CRMA:

a) About continuous traffic:

- A CRMA system always allocates the necessary bandwidth for the speech traffic.
- Continuous traffic does not experience any delay.

As a consequence of this two points, the overall quality of the speech transmissions is very high.

- The channel occupation of the CONTROL packets is small, only 10%.

The use of CONTROL packets allows the implementation of the TDD channel (without the CONTROL packets the stations synchronisation would not be possible).

b) About the data traffic:

- The channel utilisation of the data traffic, always rather high, is sensible to the inbound outbound traffic ratio. The more the traffic is handled by the GBS (i.e. the smaller is the ratio), the higher is the data throughput.
- System performances are very sensible to the variations of the retransmission probabilities. Therefore, it is possible to keep the data delay time fairly low, even when the channel is heavily occupied by continuous traffic, just choosing the right values for P_d, P_g, P_{wd} and P_{wg}.

- Due to implementation of the error control, acknowledged data transmissions are virtually error free.

Hence, even if the CRMA protocol is mainly designed to have good quality speech transmissions, the issues of the data traffic are not neglected and the global system performances are good enough to allow comparison with another, more famous, hybrid protocol for indoor, wireless communications: the Packet Reservation Multiple Access protocol.

7.2 Brief description of PRMA protocol

The Packet Reservation Multiple Access (PRMA) [14] protocol allows mobile wireless terminals to transmit packetised information over a full duplex channel to the Group Base Station (GBS). The terminals communicate each other via one base station.

7.2.1 Channel description

The PRMA channel is divided into frames, which are again divided into a fixed number of timeslots where one timeslot can hold one packet. A timeslot can be either reserved or unreserved.

- In a reserved timeslot only the terminal holding reservation for that timeslot is allowed to transmit a packet.
- An unreserved timeslot can be accessed by any terminal.

There are two kinds of frames both with the same duration:

a) Frames in an *upstream* channel, used by the terminals to transmit packets to the base station.

b) Frames in a *downstream* channel, used by the base station to transmit packets to the terminals.

Of the two communication channels, only the uplink is a multiple access channel. Since only the base station controls the downlink channel, there is no contention on it. The downlink channel is used by the GBS to transmit speech, data and feedback packets to the terminals. The feedback packets acknowledge the successful transmissions, synchronise the stations with the GBS and indicate to the stations which timeslot in the frame is reserved.

7.2.2 Channel access and transmission probabilities

In the PRMA protocol a difference is made between two types of packets:

1. Periodic packets.

Periodic packets are part of a long stream of information. We assume that all speech packets are periodic. In the PRMA protocol, the speech terminals use speech activity detection during a conversation to determine whether the user is speaking or silent. Upon generating a stream of packets, the user terminal stores it in a (periodic data) buffer and transmits the first packet of this buffer in an unreserved slot with probability *Ps*. If the transmission is unsuccessful, the station will retransmit in the next unreserved timeslot with probability *Ps*, until it succeeds in its transmission. After a successful transmission, the terminal receives the reservation for that timeslot in every frame. In a reserved timeslot only the terminal holding the reservation for that timeslot can transmit. If the buffer is empty, the terminal will transmit an empty packet in its reserved slot, indicating that the reservation on that slot is released.

2. Random packets.

Random packets consist of isolated data. We assume that all data packets are random. Upon generating a random packet, the user terminal stores it in a buffer and transmits the first packet of this buffer in an unreserved slot with probability *Pd*. If the transmission is unsuccessful, the station will retransmit in the next unreserved timeslot with probability *Pd* until it succeeds in its transmission.

The speech and data are transmitted in an unreserved slot with transmission probability respectively of *Ps* and *Pd*. Usually the probability *Ps* is much greater than *Pd*, thus realising a higher priority for the transmission of the speech packets.

7.2.3 Buffer and packet dropping probability

Both data and speech packets have their own buffer and both buffers are of the same type: First In First Out (FIFO). Because a high delay in speech packet communication is unacceptable, a speech packet will be dropped if it is not transmitted after a time D_{max} (usually 40 ms [14]). Therefore, the speech packets buffer has a finite length. The data packets buffer is assumed to be infinitely long.

Packet dropping leads to a speech degradation. The upper limit for the packet dropping probability is 1% of the overall packets transmitted [1]. An increase in the number of users leads to an increase in the number of collisions, which results in an increase of the packet dropping probability. Therefore, the maximum number of users is limited by the packet dropping probability.

7.3 Theoretical comparison of the two protocols

As we have seen in the previous paragraphs, there are some common features shared by the two protocols.

- *Hybrid protocols (TDMA/Slotted ALOHA)*

The two protocols are both hybrid: they combine the good qualities of TDMA and Slotted ALOHA in order to obtain an enhanced hybrid protocol. Since the continuous traffic can not tolerate high delays, TDMA is used and its scarce flexibility to traffic variations is balanced by the use of speech activity detector and by the hybridisation with Slotted ALOHA. On the other hand Slotted ALOHA alone does not guarantee high throughputs.

- *Higher priority for continuous traffic*

In both systems, continuous traffic has a higher priority due to the constraints on the speech packets delay.

More interesting are the differences between the two protocols:

- *Bandwidth allocation*

One disadvantage of the PRMA protocol is that the bandwidth allocation for the uplink and downlink channel is constant and sometimes it may not be optimally used. When the difference between the stream of the inbound and the outbound traffic is large, at least one of the two links has a poor utilisation.

For CRMA, small streams have a minor impact on the system throughput and therefore hardly affects the link utilisation. This results in a higher flexibility to the traffic load variations for CRMA.

- *Traffic contentions*

The contentions between continuous traffic and data traffic are resolved in different ways for CRMA and PRMA.

For PRMA, before obtaining the reservation from GBS, the speech packets have to win a contention for the channel with the packets from other stations. If a speech packet stays for too long in a buffer, waiting for accessing the channel, it is lost and the quality of the overall transmission decreases.

For CRMA, the speech stations gain access to the channel whenever they require it due to the implementation of pure TDMA protocol. The implementation of a pure TDMA scheme for speech packets is possible because of the *channel sensing* performed by all the data stations before accessing the channel. As we have seen in Chapter 3, the data stations *listen* to the channel in order to avoid collisions with CBR packets. If the channel is occupied by continuous traffic, the station does not transmit. As a consequence of the channel sensing, not the whole timeslot is available to data transmission. This worsens the global performances of the protocol. On the other hand the data stations do not need to be informed by the GBS which timeslot is reserved and which is not. Eventually, the channel loss is partially compensated by the fact that with PRMA, whenever a periodic transmission is finished, the transmitting station has to send a blank slot to the GBS carrying no information.

- *Packet dropping probability and blocking probability*

With PRMA, sometimes the speech packets can not access the channel for a relatively long time, due to the contentions with the other packets. This results in a delay that exceeds the time constraints (D_{max}) for having intelligible speech. Packets delayed for too long are discarded.

It is therefore necessary to introduce an upper limit (usually 0.01) for the packet dropping probability, so that the quality of the transmission is not too poor.

With CRMA, there is no delay for continuous packets but is necessary to introduce a blocking probability (i.e. a maximum number of simultaneous continuous transmissions allowed) to reserve a minimum level of the link capacity to the data traffic.

7.4 System variables

We are now going to analyse and confront the most important system variables to have a comparable view of PRMA [14] and CRMA.

- *Channel rate*

Usually PRMA is implemented with a channel rate of 720 kb/s for the uplink channel and the downlink channel. Since CRMA is implemented on a TDD channel, we assume a channel rate of 720 kb/s $\times 2$ = 1.44 Mb/s.

- *Frame duration*

The frame duration for PRMA is 16 ms , there is no reason to consider a different value for CRMA.

- *Speech encoder rate and voice activity*

The speech encoder rate is the rate at which the speech encoder generates digitised speech. For PRMA and CRMA is taken the same value of 32 kb/s. For the two protocols is assumed true the same value of 43% voice activity.

- *Speech packet header*

Since continuous packet handling is more complex with PRMA than with CRMA, PRMA speech packets headers require a larger number of bits than CRMA speech packets headers.

PRMA header = 64 bits.

CRMA header = 32 bits (standard value for pure TDMA).

- *Speech packet size*

The speech encoder rate and the frame duration are the same for the two protocols. Hence, the number of information bits per packet is the same. Number of information bits per packet: 16 ms × 32 kb/s = 512 bits. Due to the different packet header size, the total number of bits for packet is different:

512+64 = 576 for PRMA.

512 +32 = 544 for CRMA.

- *Slot duration*

The slot duration is different for the two protocols, due to the different packet size and channel rate.

PRMA slot duration: 576 bit / 720 kb/s = 800 μs.

At the end of every CRMA transmission, there is a blank period of $2a$, due to the propagation and synchronisation delay, where a is the worst case of propagation delay. In Chapter 2 we assumed as a reasonable value for a 5 μs.

CRMA slot duration: 544 bits / 1.44 Mb/s + 10 μs = 388 μs.

- *Data packet header*

The data packet header size is 64 bits for PRMA and CRMA.

- *Propagation and processing time*

Propagation and processing time have the same values (respectively 5 μs and 7 μs [4]) for PRMA and CRMA. Propagation and processing time do not influence the PRMA system performances, since acknowledgement

packets are sent by the GBS on the downstream channel. On the contrary, they are very important for CRMA performance evaluation because the acknowledgement packets are sent by the receiving station within the same timeslot, thus reducing the time available for data transmission.

- *Data packet size*

PRMA data packet size is the same of the speech packet size.

PRMA packet size = 576 bits.

For CRMA data transmission not the whole slot is available: a portion of it is spent in the packet acknowledgement, that is sent by the GBS within the same timeslot.

CRMA packet size: (388-32) $\mu s \times 1.44$ Mb/s = 512 bits.

- *Number of slots per frame*

For PRMA, the number of slots per frame is 16 ms / 800 μs = 20 slots.

For CRMA the number of slots per frame is $\lfloor 16 \text{ ms} / 388\ \mu\text{s} \rfloor = 41$. Where $\lfloor x \rfloor$ denotes the biggest integer $\leq x$.

The difference is because of the double channel rate for CRMA and the smaller size of speech packets.

- *Packet dropping probability and blocking probability*

With the PRMA protocol, if a speech packet stays for too long in the buffer, it is discarded. A value of 1% for the packet dropping probability is considered acceptable. The 1% constraint limits the maximum number of active users that can transmit over the channel (this number ranges between 30 and 35 users). This is not a great disadvantage since this protocol is mainly designed for indoor applications with a relative small number of users.

For the CRMA does not exist any packet dropping probability, since every speech station gains the access to the channel whenever it needs it. Therefore, it is necessary to introduce a blocking probability to reserve a certain percentage of the link to the data traffic.

We have modelled our system with 20 CBR stations, 41 slots and 4 CONTROL packets for frame. To calculate the blocking probability, we assume that to each CBR circuit is assigned only one slot. Hence the maximum number of slots occupied by continuous traffic is 20 + 4 = 24 slots.

In our case there is no need to introduce a blocking probability since continuous traffic occupies at the most 59% of the channel.

As an example, we consider a system with a higher number of CBR stations, for instance 30. If we prevent the continuous traffic to use more than 65% of the channel (i.e. 23 slots for CBR traffic and 4 for CONTROL packets), then the blocking probability is equal to 1.5e-7.

In table 2 we present a the results of this paragraph:

	PRMA	**CRMA**
Channel rate	720 Kb/s	1.44 Mb/s
Frame duration	16 ms	16 ms
Speech encoder rate	32 kb/s	32 kb/s
Voice activity	43 %	43 %
Speech packet header	64 bits	32 bits
Speech packet size	576 bits	544 bits
Slot duration	800 μs	388 μs
Data packet header	64 bits	64 bits
Propagation delay	5 μs	5 μs
Processing delay	7 μs	7 μs
Data packet size	576 bits	512 bits
Number of slots per frame	20	41
Packet dropping probability	0.01	0
Blocking probability*	0	1.75e-7

Table 2: Confront of PRMA and CRMA system variables

* For a system with 30 CBR stations.

7.5 Numerical results of the comparison

Finally, we are able to compare the performances of the two protocols. The following parameters are considered:

- throughput;

- average delay of a data packet.

The performance analysis of the two protocols are obtained following different approaches, hence only a qualitative comparison is possible.

The results for PRMA we are considering are presented in [15].

In figure 28 and figure 29 we represent the throughput and the average data delay for PRMA as a function of the total number of terminals (P_s is the retransmission probability for a continuous transmission, P_d for a data transmission).

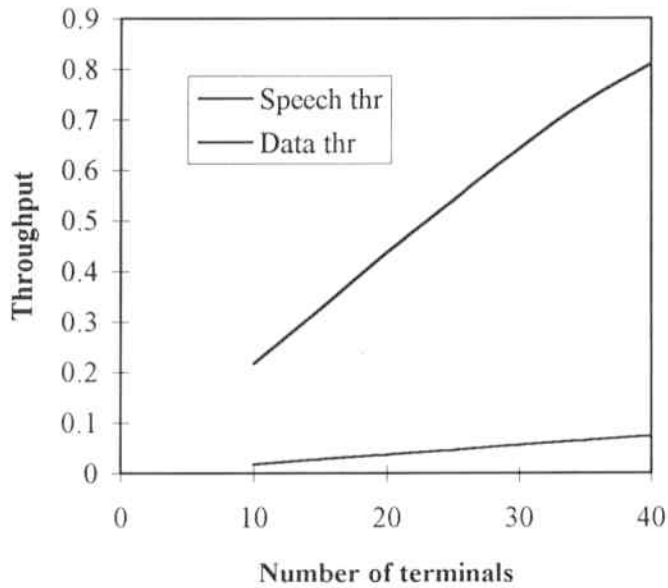

Figure 28: Throughput for a PRMA system ($P_s = 0.1$, $P_d = 0.044$)

The performance evaluation is performed considering capture in an error free channel with only near-far effect. The results for a protocol using capture are expected to be reasonably better than the results for the same protocol without capture.

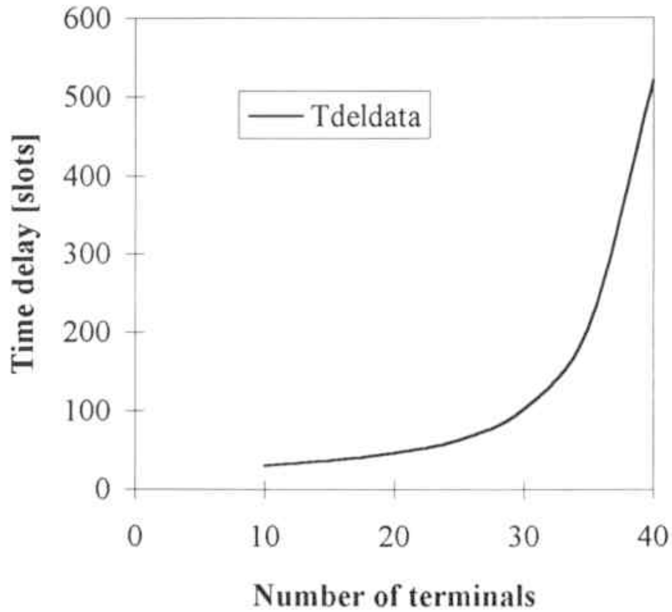

Figure 29: Time delay for a PRMA system (P_s = 0.1, P_d = 0.044)

About these results, they are obtained considering any of the PRMA terminals able to transmit either speech or data.

For PRMA performances analysis, the overall data packets delay is calculated as the sum of the time spent in the buffer and the time necessary for a successful transmission. CRMA data performances do not consider the time spent in the buffers.

The packet dropping limits the number of users between 30 and 35 for PRMA (the results presented are obtained for 33 users).

The results for CRMA protocol, presented in Chapter 6, are obtained for a overall number of 33 users (20 speech stations and 13 data stations).

For a fair comparison, it is necessary to remember that for PRMA the packets on the downlink channel suffer very low delays and have high throughput.

Table 3 presents some numerical results:

	PRMA	CRMA
Speech throughput	0.70	0.65
Data throughput	0.06	0.11 ($GBS_{througput}$ = 0.03)
Data delay VBR	192	61
Data delay GBS	0	33

Table 3: Numerical results of the comparison

7.6 Conclusions

One of the major problems with PRMA is that there is a balance between the quality of speech transmissions and the data packets delay:

- good speech quality implies a high data delay,
- low data delay implies a high packet dropping probability.

The general solution is to limit the total number of users.

CRMA, also because of the implementation of the channel sensing, is expected to guarantee comparable performances with a higher number of users, specially if the data traffic is not too high.

8 PROPOSAL FOR A HYBRID CDMA/CRMA SYSTEM

8.1 Hybrid CDMA/CRMA protocol

The performances of CRMA protocol are quite satisfactory but there are two drawbacks.

- The data throughput has an upper ideal limit of 0.36, that is the boundary for Slotted Aloha [9].
- T_{del} for data packets is somewhat high under heavy continuous traffic loads.

To overcome these two limits, in case of heavy data traffic, a possible solution would be to implement a hybrid CDMA/CRMA protocol.

Due to CDMA protocol, the number of successfully transmitted packets in the same timeslot can be greater than one. Therefore, the number of destroying collisions will be reduced, the throughput will increase and the T_{del} will be smaller.

The possible combinations for the hybrid protocol are two:

a) Hybrid CDMA/TDMA and hybrid CDMA/Slotted Aloha.

b) Pure TDMA and hybrid CDMA/Slotted Aloha.

The requirements for continuous traffic are different from the requirements for data traffic. Because of its nature, speech traffic does not tolerate high delay, therefore if a continuous packet is lost it can not be retransmitted. Hence, since CDMA protocol allows packet collisions, we analyse pure TDMA and hybrid CDMA/Slotted Aloha.

8.2 The model for the hybrid protocol

The CDMA protocol, assigning different codes to different stations allows up to c stations to transmit simultaneously without any packet loss. If the number of transmitting stations is greater than c, all the transmitted packets will be lost.

The proposed Markov model for pure CRMA is valid also for the hybrid CDMA/CRMA protocol.

Once a packet arrives, either to a VBR terminal or to a GBS, immediately the station tries to transmit it. If the channel is occupied by a continuous transmission, the station enters the *WAIT* state and tries to retransmit every new timeslot with probability *Pwd* (*Pwg* for a GBS).

When a packet accesses the channel:

a) More than *c* stations are transmitting simultaneously: the packet is lost and all the stations attempting to access the channel enter the *BACK* state. Once a station is in the *BACK* state , it tries to retransmit the packet every new time slot with probability *Pd* (*Pg* for a GBS).

b) A number of stations less or equal to *c* transmits simultaneously: the packet is transmitted successfully and the stations go back to the *ORIG* state and are ready for a new transmission.

Therefore, the nature of the hybrid CDMA/CRMA system is not very different from that of the pure CRMA protocol, the only difference is that the hybrid model can tolerate much higher data traffic loads

- *Advantages*

  The main advantage of a CRMA/CDMA is in the increase of the system performances. It is possible to transmit more than one data packet per slot. Therefore, better *throughputs* and *time delays* are obtained. The results depend on the efficiency of the code and on the maximum number of users allowed to transmit at one time.

  Because of the independence of the continuous traffic from the data traffic, if a hybrid CDMA/TDMA, CDMA/Slotted ALOHA model is implemented than is possible to *reuse* the same codes for data and speech traffic.

- *Disadvantages*

  Some of the features that make pure CRMA protocol very interesting are the small bandwidth required and the simplicity to implement it. A hybrid CDMA/CRMA protocol requires a *larger bandwidth* and a *greater complexity* due to implementation of coded transmissions.

Table 4 presents the results of a qualitative comparison of the three protocols:

	PRMA	CRMA	CDMA/CRMA
1 **Speech traffic**			
Packet loss (is a function of the number of users).	Depends on the packet dropping probability.	Depends on the blocking probability (less than PRMA).	Depends on the blocking probability (less than PRMA).
Throughput.	High	High	Very high.
Packet delay.	Depends on the time spent in buffer (maximum 40 ms).	Virtually 0.	Virtually 0.
2 **Data traffic**			
Throughput*.	Worse than CRMA (there are collisions with speech packets).	Roughly 40% of the available channel (depends on continuous traffic)	Limited only by the number of packets transmitted without collision.
Packet delay*.	Worse than CRMA (there are more collisions).	Comparable to ALOHA (depends on the continuous traffic).	Relatively small.

Table 4: Comparisons of the three protocols: PRMA, CRMA and CDMA/CRMA

* The results for PRMA are obtained for the upstream channel

9 CONCLUSIONS AND RECOMMENDATIONS

9.1 Conclusions

This report presents the performance evaluation of the Circuit Reservation Multiple Access protocol. Our investigations led to the following results:

- The development of an analytical model for both data and speech traffic.
- The design and implementation of a program to calculate the performance of the data traffic model (the speech traffic model has already been widely researched in literature [17]).
- The design and the implementation of a program that simulates the behaviour of CRMA protocol.
- A qualitative comparison with PRMA, a hybrid indoor wireless protocol that has been researched for a longer period.
- A proposal for a hybrid CDMA/CRMA protocol. The hybrid CDMA/CRMA protocol is designed to combine the advantages of CDMA and CRMA and allows higher throughputs and lower delays.

The results of analytical model calculations and of the simulations match each other perfectly. Therefore, it is assumed that the CRMA protocol can be very well described by the proposed model.

At this point of research, CRMA gave results good enough to justify further investigations. The comparison with PRMA can not be completely fair due to the fact that capture has not been studied yet with CRMA and the packets that found the terminal occupied are discarded and not stored in buffers.

9.2 Recommendations

Further studies on CRMA can develop in many directions, this is what we recommend for the future:

- Develop a model with at least *20* CBR and *20* VBR stations.

- Implement a model with buffers for data packets, so that the packets that arrive to a VBR station and find it in the *WAIT* or *BACK* state are not discarded but stored in a buffer.

- Add capture for collided packets to the model for data traffic.

- Study a more appropriated distribution for data traffic. An assumption that is used in the analysis of CRMA protocol is that the data traffic arriving to the system follows a *geometrical distribution*. We made this assumption because it simplifies the protocol analysis and is generally accepted in literature [9]. In reality, the data traffic is not always exactly described by this distribution. The latest researches seem to model the data traffic as statistically *self similar distributed* [18]. Unfortunately studies on self similarity are not enough developed to be used for our applications.

- Investigations on stability of the CRMA protocol. An underlying assumption in the analysis of CRMA protocol is a *stability* assumption, namely, that the number of backlogged users with packets awaiting to be retransmitted is not steadily growing. In other words, it is assumed that packets are entering and

REFERENCES

[1] Goodman, D. J.; Valenzuela, R. A.; Galyliard, K. T.; Ramamurthi, B.; 'Packet Reservation Multiple Access for local wireless communications'. IEEE Transaction on Communications, vol. 37, pp. 885-890, August 1989.

[2] Goodman, D. J.; Wei, S.X.; 'Factors affecting the bandwidth of packet reservation multiple access'. Proceedings of the 39th IEEE Vehicular Technology conference, San Francisco, pp. 292-299, May 1989.

[3] Goodman, D.J.; Wei, S.X.; 'Efficiency of packet reservation multiple access'. IEEE Transactions on Vehicular Technology, Vol. 40, pp. 170-176, February 1991.

[4] Vliet, E. J. M. van; 'A wireless office communication system for constant and variable bandwidth demand traffic'. Delft University of Technology, Telecommunications and Traffic-Control System Group, Task report, October 1992.

[5] Vliet, E. J. M. van; 'Performances analysis of the Circuit Reservation Multiple Access protocol for wireless office communications'. Delft University of Technology, Telecommunications and Traffic-Control System Group, Thesis report, March 1993.

[6] Devasirvatham, D. M. J.; 'Multipath time delay spread in the digital portable radio environment'. IEEE Communication Magazine, Vol. 25, N. 6, pp. 13-21, June 1987.

[7] Saleh, A. A. M.; Valenzuela, R. A.; 'A statistical model for indoor multipath propagation'. IEEE Journal on Selected Areas in Communication, Vol. SAC-5, N. 2, pp. 128-137, February 1987.

[8] Brady, P. T.; 'A model for generating on-off speech patterns in two way conversation'. Bell System Technical Journal, Vol. 48, pp. 2445-2471, September 1969.

[9] Schwartz, M.; 'Telecommunication Networks, protocols, modelling and analysis'. Addison-Wesley, Reading, 1987.

[10] Leon-Garcia, A.; 'Probability and Random Processes for Electrical Engineering', Addison-Wesley, Reading, 1989.

[11] Schoute, F.C.; 'Prestatie-analyse van telecommunicatiesystemen'. Kluwer Technische Boeken B.V.; 1989.

[12] Walsh, G.R.;'Methods of Optimization'. Department of Mathematics, University of York, 1975.

[13] Tanenbaum, A.S.; 'Computer Networks', second edition. Prentice-Hall International, Inc, 1989.

[14] Broek, C. van den; 'Modelling and simulation of the Packet Reservation Multiple Access Protocol for Wireless speech/data communications'. Delft University of Technology, Telecommunications and Traffic-Control System Group, TWAIO thesis, January 1994.

[15] Lee, T. H.; 'Performance analysis of the Packet Reservation Multiple Access protocol in a slow and fast fading channel with near-far effect, using simulation'. Delft University of Technology, Telecommunications and Traffic-Control System Group, Task report, February 1994.

[16] Proakis, J.G.; 'Digital Communications'. McGraw-Hill Book Company, New York, 1983.

[17] Kleinrock L.; 'Queuing Systems, Volume 1: Theory'. John Wiley & Sons, New York, 1975.

[18] Leland, W., E.; Taqqu, M., S.; Willinger, W.; Wilson, D., V.; 'On the Self-Similar Nature of Ethernet Traffic'. IEEE/ACM Transaction on Networking, vol.2, pp. 1-15, February 1994.

[19] Rom R., Sidi M., 'Multiple Access protocol: performance and analysis'. Springer, New York, 1990.

APPENDIX A: THE ANALYTICAL PROGRAM

This section presents the listing of the analytical programs.

There are two programs: MARGRE.M and MARGRE1.M. They both have the vector of the retransmission probabilties (P_d, P_{wd}, P_g and P_{wg}) as input.

MARGRE.M gives as a result the system performence value. Therefore, it is optimised with the MATLAB built in function FMINS.

MARGRE1.M prints on the screen (and in the file RESULTS.DAT) the following parameters P_{arr}, P_{d0}, P_{g0}, VBR throughput, GBS throughput, VBR time delay, GBS time delay and System throughput.

Both MARGRE.M and MARGRE1.M are linked with two executable programs written in the C language: AJAX.EXE and VIOLA.EXE.

AJAX.EXE fills in the transition probabilities matrix.

VIOLA.EXE calculates the system performance from the steady state.

The sources of AJAX and VIOLA are splitted in several files which are grouped in projects:

The C sources for the AJAX project are :

CRMA2.CPP and REKEN1.CPP which include the files: MARCO.H GREGOR.H and DATA.H.

The C sources for the VIOLA project are :

CRMA3.CPP and REKEN2.CPP which include the files: MARCO2.H GREGOR2.H and DATA.H.

Here follows the listing of the programs.

The order is:

MARGRE.M;

MARGRE1.M;

CRMA2.CPP;

REKEN1.CPP;

MARCO.H;

GREGOR.H;

DATA.H;

CRMA3.CPP;

REKEN2.CPP;

MARCO2.H;

GREGOR2.H.

Page 1, listing of MARGRE.M, date is 04-07-95, file date is 04-07-95, size is 820 bytes.

```
function margre = margre(Vettore)

% Att.: The order of the values is :
%
% Pd    Pwd   Pg    Pwg

save Pvalues.dat Vettore -ascii

% In the vector data are inputted the system parameters:
% Pfree,
% Parr,
% inb_outb ratio.

data =[.85 .5  2]

save data.dat data -ascii

dos('ajax.exe');

u=13;
j=3*(u+1)*(u+2)/2;

load output.dat
p=output;
clear output;
i=eye(j);
m=p-i;
clear i;
clear p;
c=ones(j);
m(:,j)=c(:,j);
clear c;
a=sparse (m);
clear m
nono=inv(a);
clear a;
s=full(nono);
clear nono;
B=s(j,:);
clear s;
save temp.dat B -ascii

clear B;
dos('viola.exe');

% In the file Result1.dat it is stored the value of the system performance function

load Result1.dat

% It is necessary to subtract because the function FMINS finds the minimum

margre=1-Result1;

```

Page 1, listing of MARGRE1.M, date is 04-07-95, file date is 04-07-95, size is 1092 bytes.

```
function margre1 = margre1(Vettore)
% Att.: The order of the values is :
%
% Pd    Pwd   Pg    Pwg
% this version of margre is for Hungry people!!!
Store=zeros(9,8);
data = zeros(1,3);
t=.1:.1:.9
for i1=1:9,
% The first column of the matrix data is the value of Pfree,
% the second column of the matrix is the value of Parr (it changes at every row),
% the third column of the matrix is the value of the inbound_outbound ratio.
data(i1,:) =[ .85 t(i1) 2]
mentu = data(i1,:)
save data.dat mentu -ascii
save Pvalues.dat Vettore -ascii
dos('ajax.exe');
u=13;
j=3*(u+1)*(u+2)/2;
load output.dat
p=output;
clear output;
i=eye(j);
m=p-i;
clear i;
clear p;
c=ones(j);
m(:,j)=c(:,j);
clear c;
a=sparse (m);
clear m
a=inv(a);
s=full(a);
B=s(j,:);
clear s;
save temp.dat B -ascii
clear B;
dos('viola.exe');
load results.dat
results=results';
Store (i1,:) =results
load Result1.dat
allora = Result1
margre1 = 1 - allora;
end
% In the file Mammamia.dat are stired all the system values
save Mammamia.dat Store -ascii
```

Page 1, listing of CRMA2.CPP, date is 04-07-95, file date is 04-07-95, size is 13162 bytes.

```
# include <stdio.h>
# include <math.h>
# include "f:\users\gregor\marco.h"

extern float Pfree;
int dim = 3*((N+1)*(N+2))/2;
int Err;
FILE *stream;
FILE *stream1;
FILE *stream2;

float transitionprob (statetype State1, statetype State2)
{
    float Result;
    int   P,Q;

        P = State1.P;
        Q = State1.Q;

          if ((State2.P-State1.P == 0) && (State2.Q-State1.Q == 0) && (State2.X-State1.X == 0) &&
              (State1.X == 0))
              /*Form1*/
              {
                Result = Pfree * pow (1-Pd0,N-P-Q) * pow (1-Pwd,P) * pow (1-Pd,Q) +
                         Pfree * pow (1-Pd0,N-P-Q)* pow (1-Pwd,P) * (1 - pow (1-Pd,Q)-bin (1,Q,Pd))*(1-Pg0)+
                         Pfree * bin (1,N-P-Q,Pd0) * pow (1-Pwd,P) * pow (1-Pd,Q) * (1-Pg0) +
                          (1-Pfree) * pow (1-Pd0,N-P-Q) * (1-Pg0);
              } else
          if ((State2.P-State1.P == -1) && (State2.Q-State1.Q == 1) && (State2.X-State1.X == 0) &&
              (State1.X == 0))
              /*Form2*/
              {
                Result = Pfree * pow (1-Pd0,N-P-Q) * bin (1,P,Pwd) * (1 - pow (1-Pd,Q)) * (1-Pg0);
                } else
          if ((State2.P-State1.P == 0) && (State2.Q-State1.Q == 1) && (State2.X-State1.X == 0)
              && (State1.X == 0))
              /*Form3*/
              {
                Result = Pfree * bin (1,N-P-Q,Pd0) * pow (1-Pwd,P) * (1 - pow (1-Pd,Q)) * (1-Pg0);
              } else
          if ((State2.P-State1.P < -1) && (State2.P-State1.P + State2.Q-State1.Q==0)
              &&(State2.X-State1.X==0)&&(State1.X==0))
              /*Form4*/
              {
                Result = Pfree * pow (1-Pd0,N-P-Q) * bin (State1.P-State2.P,P,Pwd) * (1-Pg0);
              } else
          if ((State2.P-State1.P == 0)&&(State2.Q-State1.Q > 1)&&(State2.X-State1.X == 0)&&(State1.X == 0))
              /*Form5*/
              {
                Result = Pfree * bin (State2.Q-State1.Q,N-P-Q,Pd0) * pow (1-Pwd,P) * (1-Pg0);
              } else
          if ((State2.P-State1.P<0)&&(State2.Q-State1.Q>1)
             &&(State1.P+State1.Q<State2.P+State2.Q) && (State2.X-State1.X==0)&&(State1.X==0))
              /*Form6*/
              {
                Result = Pfree * bin (State2.Q-(State1.Q-State2.P)-State1.P,N-P-Q,Pd0) *
                bin(State1.P-State2.P,P,Pwd)*(1-Pg0);
              } else
          if ((State2.P-State1.P == 0) && (State2.Q-State1.Q == 0) && (State2.X-State1.X == 2))
             /*Form7*/
              {
                Result = Pfree * pow (1-Pd0,N-P-Q) * pow (1-Pwd,P) * (1 - pow (1-Pd,Q)) * Pg0;
              } else
          if ((State2.P-State1.P < 0)&&(State2.P-State1.P + State2.Q-State1.Q==0)&&(State2.X-State1.X==2))
             /*Form8*/
              {
                Result = Pfree * pow (1-Pd0,N-P-Q) * bin (State1.P-State2.P,P,Pwd) * Pg0;
```

Page 2, listing of CRMA2.CPP, date is 04-07-95, file date is 04-07-95, size is 13162 bytes.

```
              } else
          if ((State2.P-State1.P == 0) && (State2.Q-State1.Q > 0) && (State2.X-State1.X == 2))
             /*Form 9*/
              {
                Result = Pfree * bin (State2.Q-State1.Q,N-P-Q,Pd0) * pow (1-Pwd,P) * Pg0;
              } else
          if ((State2.P-State1.P < 0) && (State2.Q-State1.Q > 1)
             && (State1.P + State1.Q < State2.P+State2.Q) && (State2.X-State1.X == 2))
             /*Form10*/
              {
                Result = Pfree * bin (State2.Q-State1.Q+State2.P-State1.P,N-P-Q,Pd0)
                * bin (State1.P-State2.P,P,Pwd) * Pg0;
              } else
          if ((State2.P-State1.P == -1) && (State2.Q-State1.Q == 0)
              && (State2.X-State1.X  == 0) && (State1.X == 0))
             /*Form11*/
              {
                Result = Pfree * pow (1-Pd0,N-P-Q) * bin (1,P,Pwd) * pow (1-Pd,Q) * (1-Pg0);
              } else
          if ((State2.P-State1.P == 0) && (State2.Q-State1.Q == -1) && (State2.X-State1.X == 0)
              && (State1.X == 0))
             /*Form12*/
              {
                Result = Pfree * pow (1 - Pwd,P) * pow (1-Pd0,N-P-Q) * bin (1,Q,Pd) * (1-Pg0);
              } else
          if ((State2.P-State1.P > 0)&&(State2.Q-State1.Q == 0)&&(State2.X-State1.X == 0)&&(State1.X == 0))
             /*Form13*/
              {
                Result = (1 - Pfree) * bin (State2.P-State1.P,N-P-Q,Pd0) * (1-Pg0);
              } else
          if ((State2.P-State1.P >= 0)&&(State2.Q-State1.Q == 0)&&(State2.X-State1.X == 1)&&(State1.X == 0))
             /*Form14*/
              {
                Result = (1 - Pfree) * bin (State2.P-State1.P,N-P-Q,Pd0) * Pg0;
              } else
          if ((State2.P-State1.P == 0)&&(State2.Q-State1.Q == 0)&&(State2.X-State1.X == 0)&&(State1.X == 1))
             /*Form15*/
              {
                Result = Pfree * pow (1-Pd0,N-P-Q) * pow (1-Pwd,P) * (1-Pwg) * (1 - bin (1,Q,Pd)) +
                         Pfree * bin (1,N-P-Q,Pd0) * pow (1-Pwd,P) * pow (1-Pd,Q) * (1-Pwg) +
                         (1 - Pfree) * pow (1-Pd0,N-P-Q);
              } else
          if ((State2.P-State1.P == 0) && (State2.Q-State1.Q == 0) && (State2.X-State1.X == -1)
              && (State1.X == 1))
             /*Form16*/
              {
                Result = Pfree * pow (1-Pd0,N-P-Q) * pow (1-Pwd,P) * pow (1-Pd,Q) * Pwg;
              } else
          if ((State2.P-State1.P == -1)&&(State2.Q-State1.Q == 1)&&(State2.X-State1.X == 0)&&(State1.X==1))
             /*FORM17*/
              {
                Result = Pfree * pow (1-Pd0,N-P-Q) * bin (1,P,Pwd) * (1 - pow(1-Pd,Q)) * (1-Pwg);
              } else
          if ((State2.P-State1.P == 0)&&(State2.Q-State1.Q == 1)&&(State2.X-State1.X == 0)&&(State1.X == 1))
             /*Form18*/
              {
                Result = Pfree * bin (1,N-P-Q,Pd0) * pow (1-Pwd,P) * (1 - pow (1-Pd,Q)) * (1-Pwg);
              } else
          if ((State2.P-State1.P < -1) && (State2.P-State1.P + State2.Q-State1.Q==0)
              && (State2.X-State1.X==0) && (State1.X==1))
             /*Form19*/
              {
                Result = Pfree * pow (1-Pd0,N-P-Q) * bin (State1.P-State2.P,P,Pwd) * (1-Pwg);
              } else
          if ((State2.P-State1.P == 0)&&(State2.Q-State1.Q > 1)&&(State2.X-State1.X == 0)&&(State1.X == 1))
             /*Form20*/
              {
```

Page 3, listing of CRMA2.CPP, date is 04-07-95, file date is 04-07-95, size is 13162 bytes.

```
              Result = Pfree * bin (State2.Q-State1.Q,N-P-Q,Pd0) * pow (1-Pwd,P) * (1-Pwg);
            } else
        if ((State2.P-State1.P < 0) && (State2.Q-State1.Q > 1) && (State2.P+State2.Q > State1.P+State1.Q)
          && (State2.X-State1.X == 0) && (State1.X ==1))
           /*Form21*/
            {
              Result = Pfree * bin (State2.Q-State1.Q+State2.P-State1.P,N-P-Q,Pd0)
              * bin(State1.P-State2.P,P,Pwd) * (1-Pwg);
            } else
        if ((State2.P-State1.P == -1)&&(State2.Q-State1.Q == 0)&&(State2.X-State1.X == 0)&&(State1.X==1))
           /*Form22*/
            {
              Result = Pfree * pow (1-Pd0,N-P-Q) * bin (1,P,Pwd) * pow (1-Pd,Q) * (1-Pwg);
            } else
        if ((State2.P-State1.P == 0) && (State2.Q-State1.Q == -1) && (State2.X-State1.X == 0)
            && (State1.X == 1))
           /*Form23*/
            {
              Result = Pfree * pow (1-Pd0,N-P-Q) * pow (1-Pwd,P) * bin (1,Q,Pd) * (1-Pwg);
            } else
        if ((State2.P-State1.P == 0)&&(State2.Q-State1.Q == 0)&&(State2.X-State1.X == 1)&&(State1.X == 1))
           /*Form24*/
            {
              Result = Pfree * pow (1-Pd0,N-P-Q) * pow (1-Pwd,P) * (1 - pow (1-Pd,Q) ) * Pwg;
            } else
        if ((State2.P-State1.P < 0) && (State2.P-State1.P + State2.Q-State1.Q == 0)
            && (State2.X-State1.X==1) &&(State1.X==1))
           /*Form25*/
            {
              Result = Pfree * pow (1-Pd0,N-P-Q) * bin (State2.Q-State1.Q,P,Pwd) * Pwg;
            } else
        if ((State2.P-State1.P == 0)&&(State2.Q-State1.Q > 0)&&(State2.X-State1.X == 1)&&(State1.X == 1))
           /*Form 26*/
            {
              Result = Pfree * bin (State2.Q-State1.Q,N-P-Q,Pd0) * pow (1-Pwd,P) * Pwg;
            } else
        if ((State2.P-State1.P < 0) && (State2.Q-State1.Q > 1) &&
            (State2.Q + State2.P > State1.P + State1.Q) && (State2.X-State1.X == 1) && (State1.X == 1))
           /*Form27*/
            {
              Result = Pfree * bin (State2.Q-State1.Q+State2.P-State1.P,N-P-Q,Pd0)
              * bin (State1.P-State2.P,P,Pwd) * Pwg;
            } else
        if ((State2.P-State1.P > 0)&&(State2.Q-State1.Q == 0)&&(State2.X-State1.X == 0)&&(State1.X == 1))
           /*Form28*/
            {
              Result = (1 - Pfree) * bin (State2.P-State1.P,N-P-Q,Pd0);
            } else
        if ((State2.P-State1.P == 0) && (State2.Q-State1.Q == 0)
            && (State2.X-State1.X == 0) && (State1.X == 2))
           /*Form 29*/
            {
              Result = Pfree * pow (1-Pd0,N-P-Q) * pow (1-Pwd,P) *(1-((1-Pg)*bin(1,Q,Pd)+Pg*pow(1-Pd,Q)))+
                          Pfree * bin (1,N-P-Q,Pd0) * pow (1-Pwd,P) * pow (1-Pd,Q) * (1-Pg) +
                          (1 - Pfree) * pow (1-Pd0,N-P-Q);
            } else
        if ((State2.P-State1.P == 0) && (State2.Q-State1.Q == 0) && (State1.X-State2.X == 2))
           /*Form30*/
            {
              Result = Pfree * pow (1-Pd0,N-P-Q) * pow (1-Pwd,P) * pow (1-Pd,Q) * Pg;
            } else
        if ((State2.P-State1.P == -1) && (State2.Q-State1.Q == 1) && (State2.X-State1.X == 0)
            && (State1.X == 2))
           /*Form 31*/
            {
              Result = Pfree * pow (1-Pd0,N-P-Q) * bin (1,P,Pwd) * (1 - pow(1-Pd,Q) * (1 - Pg));
            } else
```

Page 4, listing of CRMA2.CPP, date is 04-07-95, file date is 04-07-95, size is 13162 bytes.

```
        if ((State2.P-State1.P == 0)&&(State2.Q-State1.Q == 1)&&(State2.X-State1.X == 0)&&(State1.X == 2))
           /*Form32*/
            {
              Result = Pfree * bin (1,N-P-Q,Pd0) * pow (1-Pwd,P) * (1 - pow (1-Pd,Q) * (1 - Pg));
            } else
        if ((State2.P-State1.P < 1) && (State2.P-State1.P + State2.Q-State1.Q == 0)
            && (State2.X-State1.X==0) && (State1.X==2))
           /*Form33*/
            {
              Result = Pfree * pow (1-Pd0,N-P-Q) * bin (State2.Q-State1.Q,P,Pwd);
            } else
        if ((State2.P-State1.P == 0)&&(State2.Q-State1.Q > 1)&&(State2.X-State1.X == 0)&&(State1.X == 2))
           /*Form34*/
            {
              Result = Pfree * bin (State2.Q-State1.Q,N-P-Q,Pd0) * pow (1-Pwd,P);
            } else
        if ((State2.P-State1.P < 0) && (State2.Q-State1.Q > 1)
            && (State2.Q + State2.P > State1.P + State1.Q) && (State2.X-State1.X == 0) && (State1.X == 2))
           /*Form35*/
            {
              Result = Pfree * bin (State2.P-State1.P+State2.Q-State1.Q,N-P-Q,Pd0)
              * bin (State1.P-State2.P,P,Pwd);
            } else
        if ((State2.P-State1.P == -1)&&(State2.Q-State1.Q == 0)&&(State2.X-State1.X==0)&&(State1.X == 2))
           /*Form36*/
            {
              Result = Pfree * pow (1-Pd0,N-P-Q) * bin (1,P,Pwd) * pow (1-Pd,Q) * (1-Pg);
            } else
        if ((State2.P-State1.P == 0) && (State2.Q-State1.Q == -1) && (State2.X-State1.X == 0)
            && (State1.X == 2))
           /*Form37*/
            {
              Result = Pfree * pow (1-Pd0,N-P-Q) * pow (1-Pwd,P) * bin (1,Q,Pd) * (1 - Pg);
            } else
        if ((State2.P-State1.P > 0) && (State2.Q-State1.Q == 0) && (State2.X-State1.X == 0) && (State1.X == 2
           /*Form38*/
            {
              Result = (1 - Pfree) * bin (State2.P-State1.P,N-P-Q,Pd0);
            } else
        Result = 0;

        return Result;

      };

void clearmatrix (void)
{

 int i1,j1;
 for  (i1 = 0; i1 < matmax; i1++){
      for  (j1 = 0; j1 < matmax; j1++)
           M[i1][j1] = 0;
 };
}


void vulmatrix (void)
{

 statetype State1,State2;
 int i1,j1;

 State1.P = 0;
 State1.Q = 0;
 State1.X = 0;
 i1 = 0;

```

Page 5, listing of CRMA2.CPP, date is 04-07-95, file date is 04-07-95, size is 13162 bytes.

```
cento:

j1 = 0;
State2.P = 0;
State2.Q = 0;
State2.X = 0;

M[i1][j1] = transitionprob(State1,State2);

duecento:

if (State2.P == N)
        goto trecento;
else{

       while (State2.Q < (N - State2.P)){
              State2.Q += 1;
              j1 += 1;
              M[i1][j1] = transitionprob(State1,State2);
       };
};
while (State2.P < N){
              State2.Q = 0;
              State2.P += 1;
              j1 += 1;
              M[i1][j1] = transitionprob(State1,State2);
              goto duecento;
};

trecento:

if (State2.X == 2)
       goto Quattrocento;
else{

       while (State2.X < 2){
                     State2.X += 1;
                     State2.P = 0;
                     State2.Q = 0;
                     j1 += 1;
                     M[i1][j1] = transitionprob(State1,State2);
                     goto duecento;
       };
};


Quattrocento:

if (State1.P == N)
        goto cinQuecento;
else{

       if (State1.Q == (N - State1.P))
              goto QuattrocentocinQuanta;
       else{

              State1.Q += 1;
              i1 += 1;
              goto cento;
       };
};
QuattrocentocinQuanta:


State1.P += 1;
State1.Q = 0;
i1 += 1;
```

Page 6, listing of CRMA2.CPP, date is 04-07-95, file date is 04-07-95, size is 13162 bytes.

```
 goto cento;


 cinQuecento:

 if (State1.X == 2)
        goto seicento;
 else{

        while (State1.X < 2){
                State1.X += 1;
                State1.P = 0;
                State1.Q = 0;
                i1 += 1;
                goto cento;
        };
 };
 seicento:

 M[i1][j1] = transitionprob(State1,State2);

 }



void rowsum (void)
{

float rowsum;
int i1,j1;

rowsum = 0;
i1 = 0;
j1 = 0;

cento:

for (j1 = 0; j1 <  matmax; j1++)
   rowsum +=  M[i1][j1];

fprintf (stream,"\nThe sum of row %2u is %6f", i1+1, rowsum);
fprintf (stream,"%c%2u",' ',j1);
if (i1 == matmax-1)
  goto duecento;
else
 {i1 += 1;
  rowsum = 0;
  goto cento;}

duecento:

}

void dumpmatrix (void)

{

int     i1,j1;

for (i1 = 0; i1 < matmax; i1++){
    fprintf (stream,"%c%c",'\n',' ');
        for (j1 =0; j1 < matmax; j1++){
                fprintf (stream,"%g%c", M[i1][j1],' ');

        }
     }
}
```

Page 7, listing of CRMA2.CPP, date is 04-07-95, file date is 04-07-95, size is 13162 bytes.

```

void valueassign (void)
{

 (fscanf(stream1, "%f", &Pd));
 (fscanf(stream1, "%f", &Pwd));
 (fscanf(stream1, "%f", &Pg));
 (fscanf(stream1, "%f", &Pwg));
 printf ("\n%f %f %f %f",Pd,Pwd,Pg,Pwg);
}

void data (void)
{
 (fscanf(stream2, "%f", &Pfree));
 (fscanf(stream2, "%f", &Parr));
 (fscanf(stream2, "%f", &Inb_outb));
}

void main(void)

{
   /* open a file for update */
   stream = fopen("OUTPUT.DAT", "w+");
   if (stream == NULL)
      printf("\ncan not open OUTPUT.DAT ");

   stream1 = fopen("Pvalues.DAT", "r+");
   if (stream1 == NULL)
      printf("\ncan not open Pvalues.DAT ");

   stream2 = fopen("DATA.DAT", "r+");
   if (stream2 == NULL)
      printf("\ncan not open DATA.DAT ");

   data();

   valueassign();

   values();
//   printf ("\n Questi sono i valori per Pd0 Pg0 %f %f",Pd0 ,Pg0);
   clearmatrix();
   vulmatrix();
   dumpmatrix();
//   printf ("\n Questi sono i valori scelti per Pfree Parr e Inb_outb %f %f %f",Pfree,Parr,Inb_outb);
   /* close the file */
   fclose(stream);
   fclose(stream1);
   fclose(stream2);
}
```

Page 1, listing of REKEN1.CPP, date is 04-07-95, file date is 11-04-95, size is 705 bytes.

```

# include <stdio.h>
# include <math.h>
# include "f:\users\gregor\gregor.h"

void values(void)
{
 Pd0 = (Inb_outb * Parr) / (N * (1 + Inb_outb));
 Pg0 = Parr / (1 + Inb_outb);
};



float fact (int n)     /*Returns n! for n >== 0*/

{
  float F1;
  F1 = 1;
  while (n> 0){
        F1 = F1 * n;
        n = n-1;
  };
  return F1;
};


float over (int a, int b)

{
  float over1;
  if (b > a)
        over1 = 0;
  else{
        if ((a == 0) && (b == 0))
                over1 = 1;
        else
                over1 = (fact(a)) / ((fact(a-b))*(fact(b)));
  };
  return over1;
};


float bin ( int a, int b, float PPP)

  {
    float bin1;
    bin1 = over (b,a) * pow (PPP,a) * pow (1-PPP,b-a);
    return bin1;
  };


```

Page 1, listing of MARCO.H, date is 04-07-95, file date is 07-04-95, size is 1047 bytes.

```
# include <stdio.h>
# include <math.h>
# include "f:\users\gregor\data.h"



typedef struct {
        int P ;  /*Number of stations in the waiting state*/
        int Q ;  /*Number of stations in the backlogged state*/
        int X ;  /*State of the GBS*/
} statetype;

 float huge M[matmax][matmax];
 float V[matmax];

 float      Pd=0.10;
 float      Pg=0.5;
 float      Pwd=0.1;
 float      Pwg=0.5;
 float      Pd0=0.3;
 float      Pg0=0.6;
 float      PoVBR,PbVBR,PwVBR,PtVBR,NrtrVBR,Pfree,Parr,Inb_outb;
 float      PoGBS,PbGBS,PwGBS,PtGBS,NrtrGBS;
 float      VBRthr,GBSthr,CTRLthr,CBRthr,SYSTthr;


float fact (int n);
float over (int a, int b);
float bin (int a, int b, float PPP);
float PsuccVBR(void);
void GETVBRthr(void);
float PsuccGBS(void);
void GETGBSthr(void);
float TdelVBR(void);
float TdelGBS(void);
void GETSYSTthr(void);
void values(void);
float transitionprob (statetype State1, statetype State2);
void clearmatrix (void);
void vulmatrix (void);
void dumpmatrix(void);
void valueassign (void);
```

Page 1, listing of GREGOR.H, date is 04-07-95, file date is 07-04-95, size is 1160 bytes.

```
# include <stdio.h>
# include <math.h>
# include "f:\users\gregor\data.h"


typedef struct {
        int P ;  /*Number of stations in the waiting state*/
        int Q ;  /*Number of stations in the backlogged state*/
        int X ;  /*State of the GBS*/
} statetype;

extern float huge M[matmax][matmax];
extern float huge V[matmax];

extern float      Parr,Inb_outb;
extern float      Pd,Pg;

extern float      Pwd,Pwg;
extern float      Pd0,Pg0,Pfree;
extern double     PoVBR,PbVBR,PwVBR,PtVBR,NrtrVBR;
extern double     PoGBS,PbGBS,PwGBS,PtGBS,NrtrGBS;
extern float      VBRthr,GBSthr,CTRLthr,CBRthr,SYSTthr;

float fact (int n);
float over (int a, int b);
float bin (int a, int b, float PPP);
float PsuccVBR(void);
void GETVBRthr(void);
float PsuccGBS(void);
void GETGBSthr(void);
float TdelVBR(void);
float TdelGBS(void);
void GETSYSTthr(void);
float transitionprob (statetype State1, statetype State2);
void clearmatrix (void);
void dumpmatrix(void);
void vulmatrix (void);
void values(void);
void rowsum (void);
void rowsumv (void);
void dumpvector(void);
void readvec (void);
void valueassign (void);
void data (void);
```

Page 1, listing of DATA.H, date is 04-07-95, file date is 05-06-95, size is 93 bytes.

```

# define  GBSw  0.0008
# define  VBRw  0.0012

# define  N 13
# define  matmax  315

```

Page 1, listing of CRMA3.CPP, date is 04-07-95, file date is 29-05-95, size is 2333 bytes.

```
# include <stdio.h>
# include <math.h>
# include <stdlib.h>
# include "f:\users\moretti\perfeval\calculat\marco2.h"

FILE *stream;
FILE *stream1;
FILE *stream2;
FILE *stream3;
FILE *stream4;

void readvec (void)
{

  int i1;

 /* read the data  */
  for (i1 = 0; i1 < matmax; i1++){
     (fscanf(stream, "%f", &V[i1]));
     }
 }


void dumpvector (void)
{

 int i1;
 fprintf(stream1,"%c",'\n');

 for (i1 = 0; i1 < matmax; i1 ++)
     fprintf (stream1,"%g%c", V[i1],' ');
}



void rowsumv (void)
{

int i1;
float rowsumv1 ;
rowsumv1 = 0;

for (i1 = 0; i1 != matmax; i1++)
  rowsumv1 += V[i1];
   fprintf (stream1,"\nThe sum of the vector is : %g", rowsumv1);
}

void data (void)
{
 (fscanf(stream4, "%f", &Pfree));
 (fscanf(stream4, "%f", &Parr));
 (fscanf(stream4, "%f", &Inb_outb));
}


void dumpresults(void)
{
  fprintf(stream1,"\n%g", Parr);
  fprintf(stream1,"\n%g",Pd0);
  fprintf(stream1,"\n%g",Pg0);
  fprintf(stream1,"\n%g",TdelVBR());
  fprintf(stream1,"\n%g",TdelGBS());
  fprintf(stream1,"\n%g", VBRthr);
  fprintf(stream1,"\n%g", GBSthr);
  fprintf(stream1,"\n%g",SYSTthr);

}
```

Page 2, listing of CRMA3.CPP, date is 04-07-95, file date is 29-05-95, size is 2333 bytes.

```
void main(void)

{
   /* open file with steady state */

   stream = fopen("temp.dat","r+");
      if (stream == NULL)
         printf("can not open TEMP.DAT");

   stream1 = fopen("RESULTS.DAT","w+");
      if (stream1 == NULL)
        printf("can not open RESULTS.DAT");


    stream2 = fopen("Pvalues.dat","r+");
      if (stream2 == NULL)
        printf("can not open Pvalues.DAT");

    stream3 = fopen("Result1.dat","w+");
      if (stream3 == NULL)
        printf("can not open Result1.DAT");

    stream4 = fopen("DATA.dat","r+");
      if (stream4 == NULL)
        printf("can not open DATA.DAT");

   data();
   printf("\n%f %f %f",Pfree,Parr,Inb_outb);
//   fprintf(stream1,"\n%g",Pd0);
//   fprintf(stream1,"\n%g",Pg0);

   values();
   valueassign();
   readvec();
   GETGBSthr();
   GETVBRthr();
   TdelVBR();
   TdelGBS();
   GETSYSTthr();
   dumpresults();
   Performance();
   fprintf (stream3,"\n%f",SP);
   printf ("\n Questi sono i valori scelti per Pfree Parr e Inb_outb %f %f %f",Pfree,Parr,Inb_outb);
   /* close the files */

   fclose(stream);
   fclose(stream1);
   fclose(stream2);
   fclose(stream3);
   fclose(stream4);
}
```

Page 1, listing of REKEN2.CPP, date is 04-07-95, file date is 26-05-95, size is 5604 bytes.

```
# include <stdio.h>
# include <math.h>
# include "f:\users\moretti\perfeval\calculat\gregor2.h"

extern float VBRthr,GBSthr,CTRLthr,CBRthr,SYSTthr,TdelVBRwait,TdelVBRback;
extern float V[matmax];
extern FILE *stream2;

void values(void)
{
 Pd0 = (Inb_outb * Parr) / (N * (1 + Inb_outb));
 Pg0 = Parr / (1 + Inb_outb);
};

void valueassign (void)
{

 (fscanf(stream2, "%f", &Pd));
 (fscanf(stream2, "%f", &Pwd));
 (fscanf(stream2, "%f", &Pg));
 (fscanf(stream2, "%f", &Pwg));
//  printf ("\n%f %f %f %f",Pd,Pwd,Pg,Pwg);
}



float fact (int n)     /*Returns n! for n >== 0*/
{
  float F1;
  F1 = 1;
  while (n> 0){
        F1 = F1 * n;
        n = n-1;
  };
  return F1;
};



float over (int a, int b)
{
  float over1;
  if (b > a)
        over1 = 0;
  else{
        if ((a == 0) && (b == 0))
                over1 = 1;
        else
                over1 = (fact(a)) / ((fact(a-b))*(fact(b)));
  };
  return over1;
};



float bin ( int a, int b, float PPP)
  {
    float bin1;
    bin1 = over (b,a) * pow (PPP,a) * pow (1-PPP,b-a);
    return bin1;
  };


float PsuccVBR(void)

{
 float Delta, Result, Ambra;
```

Page 2, listing of REKEN2.CPP, date is 04-07-95, file date is 26-05-95, size is 5604 bytes.

```
int    P, Q, i1;

 Delta = 0;
 Result = 0;
 i1 = 0;
   for (P = 0; P <= N; P++){
      for (Q = 0; Q <= (N - P); Q++){
          Result = V[i1] * Pfree * (1-Pg0) *
                   (((bin(1,N-P-Q,Pd0) * pow(1-Pwd,P) * pow(1-Pd,Q)) +
                   (pow(1-Pd0,N-P-Q) * bin(1,P,Pwd) * pow(1-Pd,Q)) +
                   (pow(1-Pd0,N-P-Q) * pow(1-Pwd,P) * bin(1,Q,Pd))));
                   Delta += Result;
                  i1 += 1;
      }
 };



  for (P = 0; P <= N; P++){
     for (Q = 0; Q <= (N - P); Q++){
         Result = V [i1] * Pfree * (1-Pwg) *
                 (((bin(1,N-P-Q,Pd0) * pow(1-Pwd,P) * pow(1-Pd,Q)) +
                 (pow(1-Pd0,N-P-Q) * bin(1,P,Pwd) *  pow(1-Pd,Q))  +
                 (pow(1-Pd0,N-P-Q) * pow(1-Pwd,P) *  bin(1,Q,Pd))));
                 Delta += Result;
                 i1 += 1;
      }
 };

int for (P = 0; P <= N; P++){
     for (Q = 0; Q <= (N - P); Q++){
         Result = V[i1] * Pfree * (1-Pg)  *
                 (((bin(1,N-P-Q,Pd0) * pow(1-Pwd,P) * pow(1-Pd,Q)) +
                 (pow(1-Pd0,N-P-Q) *  bin(1,P,Pwd) * pow(1-Pd,Q))  +
                 (pow(1-Pd0,N-P-Q) *  pow(1-Pwd,P) * bin(1,Q,Pd))));
                  Delta += Result;
                  i1 += 1;
     }
 };
 Ambra = Delta/N;
 return Ambra;
};


void GETVBRthr(void)
{
 VBRthr = PsuccVBR() * N;
};


float PsuccGBS(void)

{
float  Delta, Result, Ambra;
static int    P,Q,i1;

 Delta = 0;
 Result = 0;
 i1 = 0;

 for (P = 0; P <= N; P++){
     for (Q = 0; Q <= (N - P); Q++){
            Result = V[i1] * Pg0 * Pfree *
                     pow(1-Pd0,N-P-Q) * pow(1-Pwd,P) * pow(1-Pd,Q);
            Delta += Result;
            i1 += 1;
     }
```

Page 3, listing of REKEN2.CPP, date is 04-07-95, file date is 26-05-95, size is 5604 bytes.

```
  };

  for (P = 0; P <= N; P++){
      for (Q = 0; Q <= (N - P); Q++){
         Result =  V[i1] * Pwg * Pfree *
                   pow(1-Pd0,N-P-Q) * pow(1-Pwd,P) * pow(1-Pd,Q);
         Delta += Result;
         i1 += 1;
     }
  };

  for (P = 0; P <= N; P++){
      for (Q = 0; Q <= (N - P); Q++){
            Result = V[i1] * Pg * Pfree *
                     pow(1-Pd0,N-P-Q) * pow(1-Pwd,P) * pow(1-Pd,Q);
            Delta += Result;
            i1 += 1;
      }
  };

  return Delta;
};


void GETGBSthr(void)
{
   GBSthr = PsuccGBS();
 };


float TdelVBR(void)
{

  float Delta1, Delta2, Delta3, Result1, Result2, Result3, TdelVBR1;
  float PoVBR,PbVBR,PwVBR,PtVBR,NrtrVBR;
  int   i1, P, Q, X;

  Delta1 = 0;
  Delta2 = 0;
  Delta3 = 0;
  Result1 = 0;
  Result2 = 0;
  Result3 = 0;
  i1 = 0;

  for (X = 0; X <= 2; X++){
     for (P = 0; P <= N; P++){
        for (Q = 0; Q <= (N - P); Q++){
           Result1 = P * V[i1];
           Result2 = Q * V[i1];
           Result3 = (N-(P+Q)) * V[i1];
           Delta1 += Result1;
           Delta2 += Result2;
           Delta3 += Result3;
           i1 += 1;
        }
     }
  };

  PwVBR = Delta1 / N;
  PbVBR = Delta2 / N;
  PoVBR = Delta3 / N;
  PtVBR = Pfree * (PoVBR*Pd0 + PwVBR*Pwd + PbVBR*Pd);

  NrtrVBR = PtVBR / PsuccVBR()  - 1;
  TdelVBR1 = 1.5 + (NrtrVBR * (1 / (Pd * Pfree))) + ((1 - Pfree) / (Pwd*Pfree));
  TdelVBRwait = ((1 - Pfree) / (Pwd*Pfree));
```

Page 4, listing of REKEN2.CPP, date is 04-07-95, file date is 26-05-95, size is 5604 bytes.

```
  TdelVBRback = (NrtrVBR * (1 / (Pd * Pfree)));
  return TdelVBR1;
};

float TdelGBS(void)

{

 float  Delta1, Delta2, Delta3, Result1, Result2, Result3, TdelGBS1;
 float  PtGBS, PoGBS, PwGBS, PbGBS, NrtrGBS;
 int    i1, P, Q;


  Delta1 = 0;
  Delta2 = 0;
  Delta3 = 0;
  Result1 = 0;
  Result2 = 0;
  Result3 = 0;
  i1 = 0;

/* Calculation for PoGBS */

 for (P = 0; P <= N; P++){
      for (Q = 0; Q <= (N - P); Q++){
          Result1 = V[i1];
          Delta1 += Result1;
          i1 += 1;
       }
  };

/* Calculation for PwGBS  */

 for (P = 0; P <= N; P++){
      for (Q = 0; Q <= (N - P); Q++){
           Result2 = V[i1];
           Delta2 += Result2;
           i1 += 1;
      }
  };

/* Calculation for PbGBS  */

   for (P = 0; P <= N; P++){
        for (Q = 0; Q <= (N - P); Q++){
             Result3 = V[i1];
             Delta3 += Result3;
             i1 += 1;
        }
   };

  PoGBS = Delta1;
  PwGBS = Delta2;
  PbGBS = Delta3;

  PtGBS = Pfree * (PoGBS*Pg0 + PwGBS*Pwg + PbGBS*Pg);
  NrtrGBS = PtGBS / PsuccGBS()  - 1;
  TdelGBS1 = 1.5 + (NrtrGBS * (1 / (Pg*Pfree))) + ((1 - Pfree) / (Pwg*Pfree));
//printf("\n%g",PoGBS+PwGBS+PbGBS);

  return TdelGBS1;
};

void GETSYSTthr(void)

{
  SYSTthr = (1 - Pfree) + VBRthr + GBSthr;
```

Page 5, listing of REKEN2.CPP, date is 04-07-95, file date is 26-05-95, size is 5604 bytes.

```
};

void Performance (void)

{

SP = SYSTthr - (GBSw * TdelGBS()) - (VBRw * TdelVBR()) -

/* The definition of a penalty function
   this function takes into account that the transitionprobabilities
   must be greater than 0 and smaller then 1                                                  */

     ( .5*pow(2*Pd-1,100) +   .5*pow(2*Pwd-1,100) +
       .5*pow(2*Pg-1,100) +   .5*pow(2*Pwg-1,100)
     );

}
```

Page 1, listing of MARCO2.H, date is 04-07-95, file date is 24-05-95, size is 731 bytes.

```
# include <stdio.h>
# include <math.h>
# include "f:\users\moretti\perfeval\calculat\data.h"
 float V[matmax];
 float CBRthr,CTRLthr;
 float VBRthr,GBSthr;
 float SYSTthr;
 float TdelVBRwait;
 float TdelVBRback;
 float Pd0,Pg0,Pfree,Parr,Inb_outb;
 float SP;
float fact (int n);
float over (int a, int b);
float bin (int a, int b, float PPP);
float PsuccVBR(void);
void GETVBRthr(void);
float PsuccGBS(void);
void GETGBSthr(void);
float TdelVBR(void);
float TdelGBS(void);
void GETSYSTthr(void);
void rowsum (void);
void rowsumv (void);
void readvec (void);
void dumpvector (void);
void values(void);
void dumpresults(void);
void valueassign (void);
void Performance (void);
void data (void);
```

Page 1, listing of GREGOR2.H, date is 04-07-95, file date is 24-05-95, size is 659 bytes.

```
# include "f:\users\moretti\perfeval\calculat\data.h"

float   Pd;
float   Pg;
float   Pwd;
float   Pwg;

extern float    Pd0;
extern float    Pg0,Pfree;
extern float       Parr,Inb_outb;
extern float V[matmax];
extern float SP;

float fact (int n);
float over (int a, int b);
float bin (int a, int b, float PPP);
float PsuccVBR(void);
void GETVBRthr(void);
float PsuccGBS(void);
void GETGBSthr(void);
float TdelVBR(void);
float TdelGBS(void);
void GETSYSTthr(void);
void rowsumv (void);
void dumpvector(void);
void readvec (void);
void values(void);
void dumpresults(void);
void valueassign (void);
void Performance (void);
void data(void);
```

APPENDIX B: THE SIMULATION PROGRAM

This section presents the listing of the simulation program and of the data files.

The simulation program SIMGRAPH.CPP reads the retransmission probability values (P_d, P_{wd}, P_g and P_{wg}) from the data file SIMVAL1.DAT and the system parameters (number of frames to simulate, number of slots per frame, P_{free} and inb_outb ratio) from the data file SIMPAR1.DAT. The results of the simulations are written in the files SIMRES1.DAT (VBR throughput, GBS throughput and System throughput) and SIMRES2.DAT (VBR time delay and GBS time delay).

Here follows the listing of the programs.

The order is:

SIMGRAPH.CPP;

SIMVAL1.DAT;

SIMPAR1.DAT.

Page 1, listing of SIMGRAPH.CPP, date is 04-07-95, file date is 04-07-95, size is 10261 bytes.

```
# include <stdio.h>
# include <stdlib.h>
# include <time.h>
# include <math.h>

# define max_CBR 26
# define nrt     14

float nr_frames;
float nr_slots;
float Pfree;
float Parr;
float inb_outb;

float Pd0;
float Pg0;
float Pd;
float Pwd;
float Pg;
float Pwg;

float P1[nrt];
float P[nrt];    /* vector that memorizes the value of the retransmission probability for each station */
float stat[nrt]; /* variable that indicates the state of the VBR stations */
float perf[nrt]; /* vector that contains the performances of each terminal */

float perf10[nrt];        /* vector that contains the performances of each terminal */
float perf1[nrt];
float perf20[nrt];        /* vector that contains the performances of each terminal */

float flag_wait;
float tot_nr_trx;
float tot_perf;

int temp_perf[nrt];     /* temporary buffer for a single transmission */
int temp_perf10[nrt];   /* temporary buffer for a single transmission for the waiting state*/
int temp_perf20[nrt];   /* temporary buffer for a single transmission for the backlogged state*/


float nr_trx[nrt];   /* counter for the number of successful transmission for each station */
int trx[nrt];        /* vector that indicates if a station is transmitting or not */
int nr_acc;          /* counter for the number of accesses to the channel */
float Pocc;          /* probability for continuous traffic to occupy the channel [Pocc = 1 - Pfree] */
int count_CBR;
int idum = (-1);
float nr_tot;
float a;

float VBR_throughput[10];
float t_delay_VBR[10];
float t_delay_VBRwait[10];
float t_delay_VBRback[10];

float GBS_throughput[10];
float t_delay_GBS[10];
float t_delay_GBSwait;
float t_delay_GBSback;

float Sys_throughput[10];

int prob (float P);
int casuale (void);
float ran0 (int *idum);

float ran0 (int *idum)

{
```

Page 2, listing of SIMGRAPH.CPP, date is 04-07-95, file date is 04-07-95, size is 10261 bytes.

```
        static float y, maxran, v[98];
        static int iff = 0;
        int j;
        unsigned i, k;

        if ((*idum < 0) || (iff == 0)){
            iff=1;
            i = 2;
            do {
                k =i;
                i <<= 1;
               } while (i);
            maxran = k;
            srand(*idum);
            *idum = 1;
            for (j = 1; j <= 97; j++){
                 v[j] = rand();
                }
            y = rand();
          }
          j = 1 + 97.0 * y / maxran;
          if ((j > 97) || (j < 1))  printf ("RANO: THIS CANNOT HAPPEN.");
          y = v[j];
          v[j] = rand();
          return y / maxran;
 }

int prob (float P)     /* function that check if the random number is greater of p */
{
  int res;
  float rand;
  float rand1;

  rand1 = ran0 (&idum);

  if ((P < 0) || (P > 1)){
       res = 0;
     }
  else

  if (P < rand1){
      res = 0;
     }
  else
  if (P >= rand1){
      res = 1;
     }
  return (res);
}

void main(void)
{
  FILE *stream;
  FILE *stream1;
  FILE *stream2;
  FILE *stream3;
```

Page 3, listing of SIMGRAPH.CPP, date is 04-07-95, file date is 04-07-95, size is 10261 bytes.

```
int i;
int j;
int k;
int l;
int flag;       /*if flag = 0 then no transmission at all;
                 if flag = 1 then a successful transmission; if flag > 1 then a collision occurred*/
float s_count;

stream = fopen("f:\\users\\moretti\\perfeval\\simulati\\simres1.dat", "w+");

if (stream == NULL){
    puts("cannot open file1");
    exit(1);
   }

stream1 = fopen("f:\\users\\moretti\\perfeval\\simulati\\simval1.dat", "r+");

if (stream1 == NULL){
     puts("cannot open file val");
     exit(1);
   }
fscanf(stream1,"%f%f%f%f", &Pd, &Pwd, &Pg, &Pwg);

stream2 = fopen("f:\\users\\moretti\\perfeval\\simulati\\simpar1.dat", "r+");

if (stream2 == NULL){
     puts("cannot open file par");
     exit(1);
   }
fscanf(stream2,"%f%f%f%f", &nr_frames, &nr_slots, &Pfree, &inb_outb);


stream3 = fopen("f:\\users\\moretti\\perfeval\\simulati\\simres2.dat", "w+");

if (stream == NULL){
    puts("cannot open file2");
    exit(1);
   }

fprintf(stream, "\nParr  VBR throughput  GBS throughput  Sys throughput");
fprintf(stream3, "\nParr     VBR delay     GBS delay");

for (l=0; l < 10; l++){
     Parr = l * 0.1 + 0.1;
     Pd0 = (inb_outb * Parr) / ((nrt -1) * (1 + inb_outb));
     Pg0 = Parr / (1 + inb_outb);

     for (i=0; i < nrt; i++){             /*initializing cycle*/
          temp_perf[i] = 0;
          temp_perf10[i] = 0;
          temp_perf20[i] = 0;

          stat[i] = 0;                       /* When the simulation starts the vector with the station state

          perf[i] = 0;                       /* When the simulation starts the vector with the permanent buf
          perf10[i] = 0;
          perf20[i] = 0;
          perf1[i] = 0;                      /* When the simulation starts the vector with the throughput of

          nr_trx[i] =0;

          P[i] = Pd0;
          P[0] = Pg0;
         }

     s_count = 0;
     flag_wait = 0;
```

Page 4, listing of SIMGRAPH.CPP, date is 04-07-95, file date is 04-07-95, size is 10261 bytes.

```


    for (j=0; j < nr_frames; j++){        /* cycle for the frames */
        count_CBR = 0;


        for (k=0; k < nr_slots; k++){     /* cycle for the slots */
            Pocc = 1 - Pfree;

            for (i=0; i < nrt; i++){

                if (stat[i] != 0){
                    temp_perf[i] += 1;            /* updates the temporary buffers   */
                  }

                if (stat[i] == 1){
                    temp_perf10[i] += 1;          /* updates the temporary buffers   */
                  }

                if (stat[i] == 2){
                    temp_perf20[i] += 1;          /* updates the temporary buffers   */
                  }
              }

            for (i=0; i<nrt; i++){               /* resets the transmission flags   */
                trx[i] = 0;
              }

            flag = 0;                            /* resets flag to zero */
            a = ran0(&idum);

            if (Pocc < a){                  /* No CBR transmission prob(Pocc) = false  */

                for (i=0; i<nrt; i++){

                    if (prob(P[i])){             /* Checks wheater the VBR station transmits or not  */
                        trx[i] = 1;
                        flag += 1;
                      }
                  }

                if (flag == 1){                  /* Check if there is only one transmission */
                    P[0] = Pg0 * trx[0] + P[0] * (1 - trx[0]);   /* update GBS transmission probability  *
                    nr_trx[0] += trx[0];                          /* update number of transmissions of the
                    perf[0]  += temp_perf[0] * trx[0];           /* update the permanent buffer of the GBS
                    temp_perf[0] = temp_perf[0] * (1 - trx[0]) + trx[0];  /* update to one the temporary b
                    stat[0] = stat[0] * (1 - trx[0]);             /* update state of the GBS  */

                    for (i=1; i<nrt; i++){
                        stat[i] = stat[i] * (1 - trx[i]);        /* update state of the station      */
                        P[i] = Pd0 * trx[i] + P[i] * (1 - trx[i]);   /* update transmission probability
                        nr_trx[i] += trx[i];                      /* update number of transmissions of the

                        perf[i]  += temp_perf[i] * trx[i];         /* update the permanent buffer      */
                        temp_perf[i] = temp_perf[i] * (1 - trx[i]) + trx[i];  /* update to one the tempor
                        perf10[i]  += temp_perf10[i] * trx[i];        /* update the permanent buffer
                        temp_perf10[i] = temp_perf10[i] * (1 - trx[i]) ;  /* update to one the temporary
                        perf20[i]  += temp_perf20[i] * trx[i];        /* update the permanent buffer
                        temp_perf20[i] = temp_perf20[i] * (1 - trx[i]);  /* update to one the temporary b
                      }
                  }

                if (flag > 1){                 /*  check if more than one VBR station tried to transmit
                    stat[0] = 2 * trx[0] + stat[0] * (1 - trx[0]);  /* change the state of the GBS to back
                    P[0] = Pg * trx[0] + P[0] * (1 - trx[0]);       /* change the transmission probability

                    for (i=1; i<nrt; i++){
```

Page 5, listing of SIMGRAPH.CPP, date is 04-07-95, file date is 04-07-95, size is 10261 bytes.

```
                        stat[i] = 2 * trx[i] + stat[i] * (1 - trx[i]);     /* change the state of the VB
                        P[i] = Pd * trx[i] + P[i] * (1 - trx[i]);          /* change the transmission pr
                      }
                  }
              }

          else

          if (Pocc >= a){                    /* loop when the slot is occupied by a CBR packet  */
              count_CBR += 1;

              for (i=0; i<nrt; i++){

                   if (stat[i] == 0){
                       P1[i] = P[i];
                     }

                   else
                   P1[i] = 0;
                  }

              if (prob(P1[0])){          /* GBS goes into the waiting state    */
                  stat[0] = 1;
                  P[0] = Pwg;                /* Transmission probability of the GBS is updated  */
                 }

              for (i=1; i<nrt; i++){

                   if (prob(P1[i])){
                       flag_wait +=1;
                       stat[i] = 1;             /* VBR terminal goes into the waiting state  */
                       P[i] = Pwd;              /* Transmission probability of the VBR terminal is upda
                     }
                  }
              }
          }                                       /* closes the slos for*/
      s_count += count_CBR;
    }                                             /*closes the frames for*/


nr_tot = 0;
nr_tot = nr_frames * nr_slots;

VBR_throughput[l] = 0;

t_delay_VBR[l] = 0;
t_delay_VBRwait[l] = 0;
t_delay_VBRback[l] = 0;

tot_nr_trx = 0;
tot_perf =0;

for (i=1; i < nrt; i++){              /* Cycle to calculate the throughput for each station */
     perf1[i] = 0;
     perf1[i] = nr_trx[i] / nr_tot;
     VBR_throughput[l] += perf1[i];

     tot_perf += perf[i];


     tot_nr_trx += nr_trx[i];

     t_delay_VBRwait[l] += perf10[i];
     t_delay_VBRback[l] += perf20[i];
    }
```

Page 6, listing of SIMGRAPH.CPP, date is 04-07-95, file date is 04-07-95, size is 10261 bytes.

```
      t_delay_VBR[l] = 0.5 + tot_perf / tot_nr_trx;

      t_delay_VBRwait[l] =  t_delay_VBRwait[l] / tot_nr_trx;
      t_delay_VBRback[l] =  t_delay_VBRback[l] / tot_nr_trx;

      flag_wait = flag_wait / tot_nr_trx;


      GBS_throughput[l] = nr_trx[0] / nr_tot;

      t_delay_GBS[l] = 0.5 + perf[0] / nr_trx[0];
      Sys_throughput[l] = VBR_throughput[l] + GBS_throughput[l] + (1 - Pfree);

      fprintf(stream, "\n\n %g     %f        %f        %f", (float(l) +1)/10, VBR_throughput[l], GBS_throughpu
      fprintf(stream3, "\n\n %g     %f        %f", (float(l) +1)/10, t_delay_VBR[l], t_delay_GBS[l]);
    }

  fclose(stream);
  fclose(stream1);
  fclose(stream2);
  fclose(stream3);

}
```

Page 1, listing of SIMVAL1.DAT, date is 04-07-95, file date is 04-07-95, size is 264 bytes.

```
0.1
0.1
0.1
0.1

************************************************************************************
The order to input the retransmission probability values is:

Pd
Pwd
Pg
Pwg
************************************************************************************
```

Page 1, listing of SIMPAR1.DAT, date is 04-07-95, file date is 04-07-95, size is 496 bytes.

```
10
41
0.85
2.0

****************************************************************************
The order to input the system parameters is:

nr_frames (it indicates the number of frames to simulate)
nr_slots  (it indicates the number of slots per frame)
Pfree     (it is the percentage of the channel available to data traffic)
inb_outb  (it is the ratio between the inbound and the outbound traffic)

****************************************************************************
```