Combined Free/Demand Assignment Multiple-Access (CFDAMA) Protocols for Integrated Data/Voice Satellite Communications

Srikanth Krishnamurthy

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

July 1994

ISBN 0-315-97679-9

Abstract

Combined Free/Demand Assignment Mutliple Access Protocols for Integrated Data/Voice Satellite Communications

Srikanth V. Krishnamurthy

Satellites with wide area coverage are suitable for providing interconnections among a large number of low-load geographically distributed terminals with mixed traffic. Satellite resources have to be dynamically allocated to these terminals so as to cope with real-time traffic demands and maintain the quality of service required by the users. The Combined Free/Demand Assignment Multiple Access (CFDAMA) protocols have been introduced for dynamic resource allocation in packet satellite communications. The studies that have been done earlier are limited to jitter tolerant traffic and large population sizes. In this thesis the performance of these protocols has been analyzed for a wide range of terminal population sizes and in an integrated voice/data environment. The performance is evaluated in terms of the average packet delay for jitter tolerant data and blocking probability for real-time voice. The effect of introducing a Multiple-Frequency Time Division Multiple Access frame format is also considered, and a channel re-assignment strategy for improving channel utilization is suggested.

Acknowledgements

I wish to express my profound gratitude to my thesis-advisor, Dr.Tho Le-Ngoc. He was a constant source of inspiration and his guidance, encouragement and unreserved support continuously helped me throughout the course of my study.

I wish to thank the professors who taught me, and with whom I have been associated with at Concordia University for helping me build the base without which this work could not have been achieved.

I am grateful to the National Sciences and Engineering Research Council of Canada (NSERC) for providing me with financial support during the course of my studies, under the collaborative R & D Grant no. CRD 126885 (with Spar Aerospace Limited).

My heartfelt thanks to my many friends at Concordia University whose help and support was with me throughout and who helped provide a great work environment.

I also wish to thank my parents, my brother and my relatives in Canada and US for their encouragement and support throughout the course of my graduate studies.

This work is dedicated to my late grandparents

K.C. Venkatanarayana Deekshit and Subbamma

Contents

List of Figures

List of Abbreviations

CFDAMA	Combined Free/Demand Assignment Multiple Access
DAMA	Demand Assignment Multiple Access
TDMA	Time Division Multiple Access
FDMA	Frequency Division Multiple Access
MF-TDMA	Multiple Frequency Time Division Multiple Access
OBP	On Board Processor
OBP Satellite	On Board Processing Satellite
CFDAMA-FA	Combined Free/Demand Assignment Multiple Access protocol with Fixed Assigned Request Slots
CFDAMA-PB	Combined Free/Demand Assignment Multiple Access protocol using Piggybacked Reqesting
CFDAMA-RA	Combined Free/Demand Assignment Multiple Access protocol with Random Access Request Slots
LAN	Local Area Network
IM	Intermodulation
RF	Radio Frequency
CPU	Central Processing Unit
TWTA	Travelling Wave Tube Amplifier

List of Symbols Used

N	Terminal population size
τ	Time slot size in seconds
R	Round trip delay
S	Scheduler queue delay in terms of number of slots
M	Number of traffic slots per frame
Q	Number of request slots per frame in CFDAMA-FA
l	Number of arrivals in a superframe (used in the analysis of CFDAMA-FA)
q	Number of packets at a tagged terminal's queue at the beginning of a superframe (used in the analysis of CFDAMA-FA)
η	Normalized population $= \frac{N\tau}{R}$
d	Fraction of demand-assigned slots in a frame
r	Twice the propogation delay from the terminal to the scheduler $r = \begin{cases} R & \text{for an on-board scheduler} \\ 2R & \text{for an on-ground centralized scheduler} \end{cases}$
X	Number of slots in a superframe. $X = aN$ where a is an arbitrary integer.
λ	Arrival rate of jitter tolerant data to a particular terminal
Λ	Arrival rate of jitter tolerant data to the system
Δ	Time distance between arrivals uniformly distributed in a superframe in the analysis of CFDAMA-FA

p	(a) Probability of a particular packet arriving to a non-empty queue being transmitted by a free slot (used in the analysis of CFDAMA-FA) (b) Probabilty of a successful request (used in the analysis of CFDAMA-RA)
p'	(a) Probability of a particular packet arriving to an empty queue being transmitted by a free slot (used in the analysis of CFDAMA-FA) (b) Probability of a particular packet arriving to a non-empty queue being transmitted by a free slot (used in the analysis of CFDAMA-RA)
p''	Probability of a particular packet arriving to an empty queue being transmitted by a free slot (used in the analysis of CFDAMA-RA)
W'	Time taken for a request to be honoured $W' = R + S$ (in the analysis of CFDAMA-FA)
b	Maximum number of free slots a particular terminal can obtain in an interval of r slots (in the analysis of CFDAMA-FA).
P_q	Probability that there are q packets in the terminal's queue at the beginning of a superframe (used in the analysis of CFDAMA-FA).
Ω	Estimated time required for a packet to obtain its due demand assigned slot in the analysis of CFDAMA-FA for the case when $N_T << R$
$G(z)$	Pgf of the service time for a packet in CFDAMA-FA when $N_T << R$
$W(z)$	Pgf of the waiting time in obtaining a slot, for a packet arriving to an non-empty queue
$\tilde{W}'(z)$	Pgf of the waiting time in obtaining a slot, for a packet arriving to an empty queue

H	Random variable representing the time between successful requests in CFDAMA-RA
A_k	Probability that k arrivals occur between successful requests (in the analysis of CFDAMA-RA).
$\bar{\xi}$	Average waiting delay for a packet in the analysis of CFDAMA-RA
β	Fixed time intervals between successive packets in a voice call in progress.
P_b	Blocking Probability for voice
ρ	Total traffic intensity in the system.
α	Fraction of Voice Traffic
λ_v	Poisson arrival rate for voice calls
ρ_d	Traffic intensity of data
ρ_v	Traffic intensity of voice
L	Number of carriers in an MF-TDMA system

Chapter 1

Introduction

Since early 1960s, satellites have been used to provide communication links to geographically distributed terminals [2]. Traditionally they were used for long distance trunk connections in fixed telecommunications networks. The nodes that were linked by the satellite were large in size and power and carried the aggregate traffic from a large number of terminals. However, the increasing digitalization of telecommunications worldwide, and the continuous process of finding more efficient methods of utilizing satellite technology has led to greater use of satellite links in user-oriented applications [2, 15]. In user-oriented applications the satellites link individual small low-load terminals, which have to be situated on or near the user's premises.

It is expected that satellites will continue to play an important role in future global networks in spite of advances in technology in other transmission media such as fiber optic cables because of certain inherent advantages [24]. These advantages may be listed as follows:

- Satellites provide basic telecommunication services to areas where terrestrial alternatives are not feasible due to geographical barriers.
- They are suitable for providing *direct* access between terminals. Users can

have direct control of their communication systems, since there are no intermediate transmission or switching nodes aside from the satellite [2, 41]. In terrestrial networks however, the information may have to pass through several intermediate public nodes before reaching the destination.

• Satellite communications provide the flexibility in adding new terminals, whereas for terrestrial transmission media additional physical connections will have to be provided.

• Satellites also have an edge in providing *direct* mobile services, since there is no need for having physical links [34, 27].

Figure 1.1 shows an envisaged satellite communication system. The satellite can connect large terrestrial networks, small terminals in residential locations, business oriented networks, and land and aeronautical mobile units. It can provide direct links between distant terminals in residential locations, inter-LAN connections in business networks and connections to distant mobile units. A small user-oriented terminal may also communicate with a large terrestrial network through the satellite link.

The changes in satellite applications calls for different criteria in the design of the terminals. Traditional systems that the satellites were serving produced mainly real-time traffic (voice). In user-oriented applications due to the increasing use of computers and exchange of digital information, jitter-tolerant traffic such as data and graphics are generated in large quantities in addition to real-time traffic such as voice. The two different types of traffic have different constraints. Real-time traffic can accept long set-up time but cannot be queued, while jitter-tolerant traffic is bursty and requires fast transfer. In traditional satellite systems, the nodes that satellites linked were large and were situated far away from the user's premises. In user-oriented applications and mobile networks the size of the terminal has to be small for it to be installed near the user's premises. This not only reduces the cost of the terminal but also greatly lowers installation costs. In addition as the terminal becomes bigger in size, it becomes more difficult to install it at the customer's premises since space is likely to be a constraint.

Unlike traditional satellite nodes, which carried the aggregate traffic from a large number of terminals, current satellite nodes are individual low-load terminals. As satellite networks continue to grow, the information processing terminals are increasing at a rapid pace. These large number of terminals have to simultaneously interconnect their respective data and voice communication links through the satellite and share the limited satellite capacity. The terminals require satellite capacity infrequently, but may need high capacities at times. Thus, the power and bandwidth of the satellite have to be shared *dynamically* in order to meet the demands of fluctuating traffic conditions.

1.1 Sharing of the Satellite Resources

For the purpose of sharing the power and bandwidth in a satellite (referred to as the transponder), among the terminals, a multiple access technique is used [4, 10]. The two most common techniques used for sharing the transponder bandwidth are Frequency Division Multiple Access (FDMA) [23] and Time Division Multiple Access (TDMA) [33].

In Frequency Division Multiple Access the transponder bandwidth is divided into a group of non-overlapping channels with smaller frequency bands as shown in Figure 1.2. The channels have different RF carriers, which are used by transmitting terminals [3]. A number of carriers are simultaneously amplified by the high power amplifier (Figure 1.3 (a)) of the terminal and the travelling wave tube amplifier in the satellite transponder (Figure 1.2) when FDMA is used. These amplifiers have non-linear characteristics and this simultaneous amplification of a number of carriers leads to intermodulation (IM) distortion [33, 4, 10]. To reduce the IM distortion the amplifiers

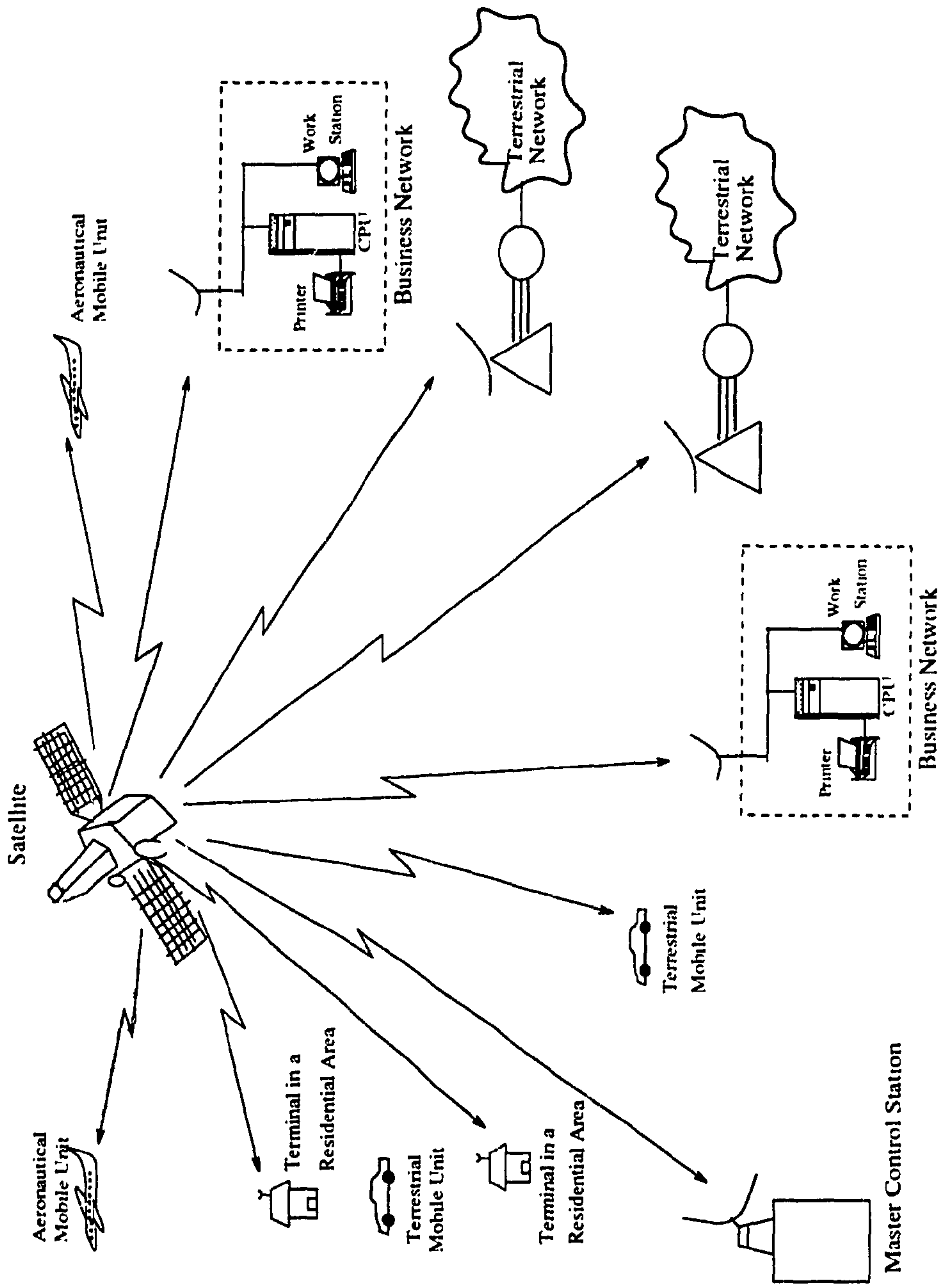


Figure 1.1: A Satellite Communication System

are operated significantly below the saturation point. The effect is a reduction in the capacity and power efficiency of the terminal as well as the transponder. The FDMA scheme is however, simple to implement [33].

Instead of dividing the channel capacity in frequency, we may divide it in time. This is known as the Time Division Multiple Access (TDMA). All the terminals transmit with a single carrier but at different times. In TDMA the term channel refers to a time-slot instead of a frequency slot. Since there is only one carrier at a time (Figure 1.3 (b)), the problem of intermodulation between signals is avoided and the amplifiers in the terminal and the transponder can operate near the saturation level. Therefore compared to the FDMA the capacity and power efficiency of both the terminal and transponder are higher for the TDMA. However, the full transponder bandwidth is shared by all terminals using a single TDMA carrier with a transmission bit rate of the order of 60 Mb/s to 120 Mb/s [41]. Thus a TDMA system requires the terminals to transmit at high transmission bit rates, which in turn requires a bigger antenna and larger power. This is because, the antenna size depends on the transmission rate. As the transmission rate increases, the energy per bit has to increase for a desired output error performance, which in turn requires a higher gain and hence a bigger antenna size. Thus, although TDMA is efficient for interconnections between high-load trunk connections that permit large terminal sizes, it is not suitable for user-oriented applications in business networks, which require very small or portable terminals [15].

In order to obtain a compromise between the loss of capacity and power efficiency in FDMA and large terminal size in TDMA a combined FDMA/TDMA scheme may be used [3, 4, 5, 20, 41]. We call this the Multiple-Frequency Time Division Multiple Access (MF-TDMA). The transponder bandwidth is first divided into a number of frequency sub-bands as in FDMA. Each frequency is then time-shared by the terminals as in TDMA. The MF-TDMA terminal has a single carrier transmitter with a carrier-hopping capability to access any time-frequency slot (Figure 1.3 (b)). Single

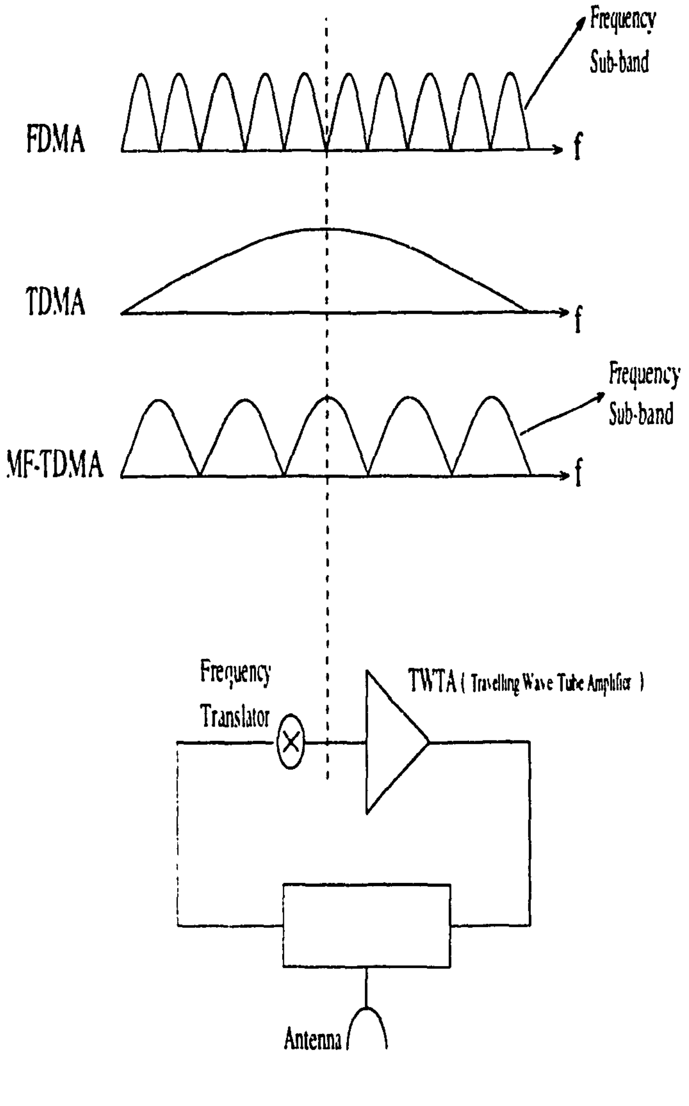
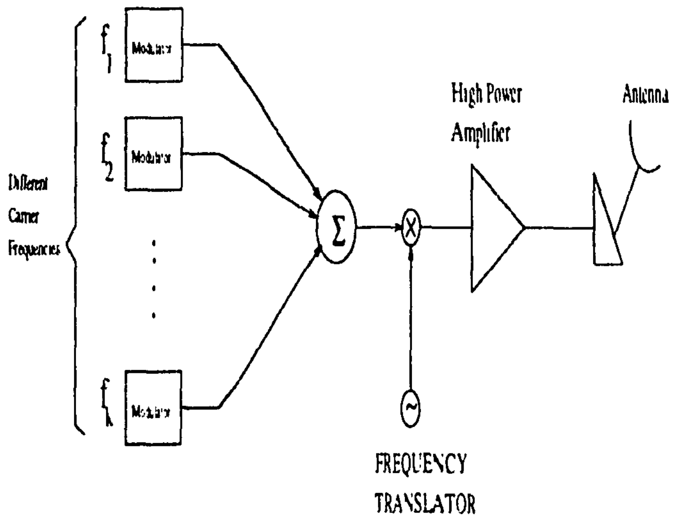
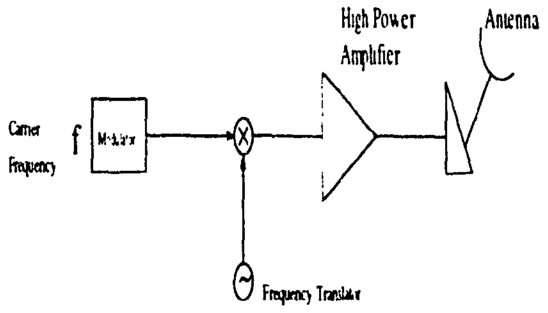


Figure 1.2: A Block Diagram of the satellite transponder and the number of frequency sub-bands in the case of FDMA, TDMA and MF-TDMA



(a) Terminal transmitting with multiple carriers (FDMA)



(b) Terminal transmitting with a single carrier

Figure 1.3: A Block Diagram showing a terminal

(a) Using multiple carriers to transmit (b) Using a single carrier to transmit

Carrier transmission eliminates IM distortion at the high power amplifier of the terminal and thus allows it to operate near saturation for a maximum power efficiency as in the case of the TDMA. Further, an MF-TDMA terminal uses only a portion of the transponder bandwidth at a given time and therefore is required to transmit at much lower transmission bit rates when compared to the TDMA. This allows for smaller terminals as compared to the TDMA. However, when MF-TDMA is used intermodulation effects are present at the satellite transponder, as in the case of the FDMA, due to the presence of multiple frequency carriers [3, 4]. The intermodulation effects increase with the number of carriers that are amplified simultaneously. Due to the combined time/frequency sharing in the MF-TDMA, the number of carriers is greatly reduced when compared to the FDMA (Figure 1.2). Thus, the reduction in the transponder capacity and power is not as much as in the case of the FDMA [3, 20]. The reduction in the size of the terminals makes the MF-TDMA scheme much more cost effective than the TDMA systems [4] in spite of the IM effects of the transponder. The MF-TDMA scheme thus permits inexpensive terminals, and yet provides aggregate throughput comparable to the conventional high-rate TDMA systems.

1.2 Fixed and Dynamic Channel Allocation

The transponder bandwidth that is divided using the FDMA, TDMA or MF-TDMA is to be allocated to the terminals. For this purpose a *Multiple Access Protocol* is used. Multiple access protocols are generally divided into two groups: fixed assignment schemes and dynamic assignment schemes.

In fixed assignment schemes the total available channel capacity is divided into channels by either TDMA, FDMA or MF-TDMA. Each subchannel is then *permanently* assigned to a terminal. Such a fixed allocation is very efficient if the interconnection patterns between terminals are rather static and the network consists of only a few highly loaded nodes with continuous or regular flow, due to the fixed allocation of the capacity in these schemes. However, as the number of terminals increase and the traffic becomes bursty the channel utilization decreases when these schemes are used because of the wasted capacity assigned to idle terminals. Further, these schemes do not allow for changes in traffic flow among the terminals relative to that assumed when the assignment is made.

Traffic in business network applications supporting multimedia services are bursty and vary greatly with time. Each terminal demands satellite capacity infrequently, but when it does it may require relatively high capacity and rapid response. Hence, in this multimedia environment fixed allocation schemes are not efficient. Capacity allocation has to be more dynamic to cope with the varying traffic demands in real-time, so as to effectively share the satellite capacity among a large number of low load terminals. The FDMA, TDMA or MF-TDMA may be used for dividing the available channel capacity into subchannels before dynamic allocation. However, it is easier to change capacity assignment when it is divided in time rather than in frequency. Hence, many of the existing dynamic multiple-access protocols use the TDMA for dividing the available capacity.

Many protocols have been proposed and used for dynamic channel allocation [41, 32]. A brief review of these protocols is presented in the following section.

1.3 Existing Dynamic Multiple Access Protocols and Evolution of CFDAMA

Demand assignment schemes have been widely used for voice communications in satellites. Here, the assignment of channel capacity is done on the basis of demands by terminals. Whenever a terminal has a message it sends in its request for slots on the basis of the number of packets in the message and the scheduler then allocates it the required channel capacity. The scheduler may be an on-board scheduler or a master control station on the ground. This is called centralized control. There could also be a distributed control where in, by the virtue of a broadcast nature of the channel, all the terminals learn the channel requests of every other terminal and maintain a global reservation request queue. In pure demand assignment schemes however, there is an initial set up time between the initial requesting time and the time at which the slot is allotted equal to either one or two round trip delays depending on where the scheduler is situated. The round trip delay is the propagation delay associated in transmitting a message in a satellite communications system and is about 270 ms. For *voice* communications, Demand Assignment Multiple Access (DAMA) schemes are quite acceptable for their offered utilization because the average holding time (or service time) of a voice call is typically about 3 minutes, several times longer than the channel set-up time of about 0.54 seconds[1]. However, for *data* communications, this set-up time becomes too long compared to the average duration of a data message. For example a data message of 8 kilobits can be transmitted within 3.91 ms by a 2.048 Mb/s terminal. Using a DAMA scheme, this message needs to wait for an average of 0.54 seconds in order to use the channel in only 3.91 ms. This large ratio of set-up time to message duration is undesirable for data communications.

Although high utility is achieved in demand assignment schemes the set-up phase is long and may make response time unacceptable as far as short length bursty messages are concerned. Random access schemes ALOHA and Slotted ALOHA [31], [26] were introduced to cater to the needs of bursty traffic and have been used in satellites. The set-up time is eliminated when random access is used. Each slot is

[1]Assuming a centralized on-ground scheduler

open for contention. Whenever traffic arrives it chooses one slot at random to transmit its message. When more than one terminal transmits in a particular time-slot a collision will occur. Then the collided packets will have to be retransmitted. At high channel loading, collisions increase and result in poor channel utility. In Slotted Aloha a maximum channel of utility of only 36 % may be achieved. However, at low throughputs, the random access schemes outperform the demand assignment schemes and the messages can be transmitted with minimum delay. A satellite system using a controlled Slotted Aloha uplink access is described in [9]. While random access protocols can thus be used for transmitting jitter-tolerant data, they are not suitable for transmission of real-time traffic. This is because when packets collide and have to be retransmitted, it is not possible to ensure that the packets arrive in order, whereas a real-time message, such as a voice call, requires its packets to arrive in the proper order to ensure comprehensibility.

Dynamic multiple-access techniques incorporating the advantages of both the demand-assignment and random-access schemes have been introduced in [25, 13, 11]. These schemes give the excellent performance of random access at low throughputs and the advantages of the demand assignment at middle and high throughput ranges. In [25] the slots that are not demand assigned are open for contention. Thus at low throughputs while most of the packets use the random access strategy, at high throughputs they are allocated capacity based on their reservations. Due to possible collisions the scheduler is required to monitor unreserved channels and have a reliable collision detection scheme. This implies the requirement of both high-processing time and reliability. An alternative to combining demand assignment with random access is combining it with *fixed* assignment. Instead of the unreserved slots being made open for contention they may be assigned on a fixed basis [1, 16]. In this scheme a cnannel has two possible states: reserved and unreserved. Reserved channels are those allocated to the requesting terminals in response to their demands. The remaining channels in the frame are in the unreserved state and are assigned to the terminal using a fixed assignment scheme. If the terminal obtaining an unreserved slot has a short traffic message then it can send this message right away and this short message has a very short access time. The unreserved slot is not used however, if the terminal does not have traffic at the instant it is allotted. For this reason, it may be expected that a *dynamic*-assignment of the unreserved slots may be advantageous over the fixed assignment.

In order to make the allotment more dynamic the Combined Free/Demand Assignment Multi-access protocols (CFDAMA) were introduced in [42]. The unreserved slots are allocated based on some heuristic information rather than on a fixed basis, thus increasing the chance that a terminal having traffic obtains these slots. Short length jitter-tolerant messages utilize the free-slots predominantly at low throughputs (since most channels are unreserved) and depend on reservations at high throughputs. Real-time messages make explicit then implicit reservations with pre-releasing [43]. A primary research was done on this scheme for large terminal population sizes with only jitter-tolerant data in [28].

1.4 Contributions and Scope of the Thesis

The Combined Free/Demand Assignment Multiple Access Protocols (CFDAMA) were introduced in [42] for the dynamic allocation of the wide bandwidth satellite channel to a number of bursty terminals. Capacity requesting in CFDAMA protocols can be done in three ways[2], by having fixed assigned request slots after regular intervals (CFDAMA-FA), by piggybacking the requests on information carrying packets (CFDAMA-PB) or by random access (CFDAMA-RA). The studies in [42, 22, 28] were

[2]to be discussed in detail in Chapter 2

limited to jitter tolerant data. The performance of CFDAMA-FA and CFDAMA-PB was analyzed for the case when the round trip delay (*Rseconds*) was close to $N\tau$ (where N is the terminal population size and τ is the time-slot size in seconds) in [28]. It has been shown that the performance of CFDAMA protocols improve as the population size decreases and that the performance of CFDAMA-FA and that of CFDAMA-PB is almost the same. Simulation studies were done for CFDAMA-RA. Further it was shown that when $N\tau$ is comparable to or less than R, the CFDAMA-FA and CFDAMA-PB outperformed the CFDAMA-RA.

In this work the performance of the CFDAMA protocols is further investigated. The effects of scheduler location and different requesting strategies on the performance are examined. The performance of the CFDAMA protocols using *fixed assigned* request slots (CFDAMA-FA) is analyzed for a general population size. The effects of reducing the population size have been examined. The performance of the CFDAMA protocols using a random access requesting strategy (CFDAMA-RA) has been analyzed for a general population size. This scheme has been compared with the CFDAMA-FA for various population sizes. The suitability of each requesting strategy for different population sizes is discussed. Further, with the advent of multimedia communications integrating real-time traffic, such as voice, with jitter tolerant traffic is of particular interest. The performance of the CFDAMA protocols in an integrated voice/data environment has been analyzed assuming constant bit rate voice. The effect of voice on the 'delay throughput' performance of jitter-tolerant data is examined. Further, the effect of introducing a MF-TDMA type frame on the performance of the CFDAMA protocols is investigated.

Queueing theory is employed throughout this research for performance analysis. Markov chain analysis and renewal theory are found to be especially useful. Due to the complexity involved in the analyses a number of simplifying approximations are made. Simulations are performed to support the validity of various approximations.

The rest of the thesis is organized as follows:

An overview of the CFDAMA protocols is presented in **Chapter 2**. The effects of scheduler location, different traffic types and variation in terminal-population size on the performance of CFDAMA protocols is examined. A description of the various requesting strategies that may be employed is presented and the advantagaes and limitations of each strategy discussed.

Chapter 3 presents a performance evaluation of CFDAMA-FA for a generalized population size in a packet satellite system. A mathematical model is formulated. The effect of reducing the population size is examined. Illustrative numerical examples are given comparing analytical results with simulation results. Simplifying approximations are suggested for small population sizes.

The performance of the CFDAMA-RA is analyzed for a general population size, in a packet satellite system in **Chapter 4**. The effect of changing the population size on the performance is examined. Illustrative numerical examples comparing simulation results with analytical results are presented. The performance of CFDAMA-RA is compared with that of CFDAMA-FA for different population ranges.

In **Chapter 5** the performance of the CFDAMA protocols in an integrated voice/data satellite communications system has been analyzed. The effect of the real-time traffic on the performance of jitter-tolerant data in the CFDAMA protocols is examined. The effect of introducing a Multiple Frequency -TDMA frame format, its advantages and disadvantages are discussed. A re-assignment strategy for improving channel utilization, when a MF-TDMA framing is used, is suggested.

Chapter 6 gives a summary of the research and suggestions for further research. A description of the Simulation Models is included in the Appendix.

Chapter 2

The Combined Free/Demand Assignment Multiple Access Protocols: An Overview

A multiple-access scheme essentially performs the task of managing the sharing of the channel capacity in the satellite communications network. Varying traffic demands require the sharing of the satellite capacity to be dynamic in order to achieve a high channel utilization and satisfactory performance measures. Many dynamic multiple-access schemes have been used for sharing the satellite channel capacity as discussed in Chapter 1. In this chapter the Combined Free/Demand Assignment protocols introduced in [42] for dynamic channel allocation are discussed in detail. The effects of scheduler location, different traffic types and different population sizes are examined. The various ways in which the requesting can be done in the CFDAMA schemes are considered and their advantages and limitations discussed.

2.1 The Combined Free/Demand Assignment Strategy

Demand Assignment multiple access schemes suffer from long delays even at low channel utilization. In order to provide *fast* access for low load conditions and short messages, while ensuring the high utilization of satellite capacity at high traffic loads, demand assignment multiple-access schemes have been combined with random access or fixed-assignment. Combined random/reservation access schemes will have to have a reliable collision detection scheme, which implies the need for high processing time and reliability. In combined fixed/demand assignment multiple access schemes slots that are not reserved are allocated by fixed assignment. These unreserved slots may be wasted if the terminal does not have traffic at that moment. Therefore the efficiency of such a combined technique may be increased by increasing the chance that a terminal having traffic obtains the unreserved slots. For this reason, a dynamic-assignment of these unreserved slots (rather than the *fixed*-assignment of such slots) may be expected to improve the performance. The CFDAMA protocols incorporate such a dynamic assignment of unreserved slots by combining demand assignment multiple-access with *free*-assignment. Reserved slots are allocated to requesting terminals in response to their demands. When there are no demands unreserved slots are assigned *dynamically* to terminals based on some heuristic information. The free-assignment may be made by allotting slots in a round robin fashion, by randomly choosing one of the terminals for allocating the next slot, by priority assignment, etc.

2.1.1 Effects of Scheduler Location on CFDAMA

CFDAMA protocols can be applied to both bent-pipe (non-regenerative) and on-board processing (OBP) satellites. The time taken for a requesting terminal to be allocated a channel in response to its explicit requests depends on where the scheduler is located. For new generation satellites with OBP capability a centralized scheduler can be located in the satellite so that it takes the requesting terminal only one round-trip delay (R seconds) plus the scheduler queueing/processing time to receive the reply to its reservation. For bent-pipe satellites the scheduler should be on the ground. We could have a *centralized* (on-ground) scheduler. In this case it takes the requesting terminal two round-trip delays plus the scheduler queueing delay. We could also use a distributed scheduling scheme in which all terminals perform an identical scheduling procedure. This is possible in a global-beam satellite system. Using a distributed scheduling scheme it takes the requesting terminal only one round-trip delay plus queueing/processing time to obtain its allocation. The assignment of *free-assigned* slots however. is not affected by the location of the scheduler. Figure 2.1 shows how a scheduler allocates slots in a CFDAMA protocol.

2.1.2 Effect of the CFDAMA on Various Traffic Types

In the discussions to follow it is assumed that the CFDAMA protocols assign the unreserved slots, which we refer to as the *free-assigned* slots, in a round robin fashion. A free-assigned slot is utilized if the terminal to which it is alloted happens to have a message in its queue at the instant of allotment. At very low throughputs when it may be assumed that there is very little requesting, the messages will be transmitted predominantly by free-assigned slots. The chance that a terminal obtains a free-assigned slot is high. Therefore, its short length jitter-tolerant messages can be transmitted through the *free-assigned* slots with short delays. At high load, since most of the channels are reserved, the chance for a terminal to obtain a free-assigned channel becomes low. As a consequence, at high-load conditions jitter-tolerant traffic is likely to be transmitted over reserved channels with longer delays. In the CF-

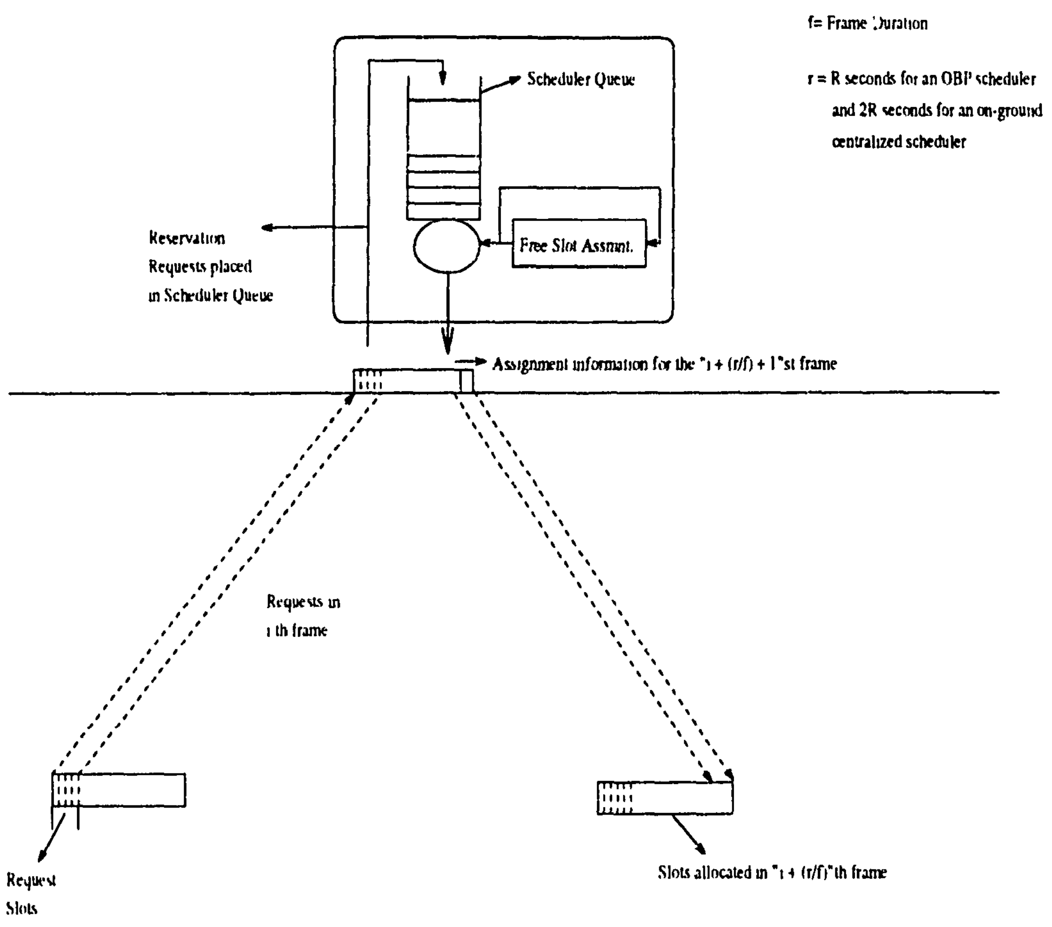

f= Frame Duration
r = R seconds for an OBP scheduler and 2R seconds for an on-ground centralized scheduler
Scheduler Queue
Free Slot Assnmt.
Reservation Requests placed in Scheduler Queue
Assignment information for the "i + (r/f) + 1"st frame
Requests in i th frame
Request Slots
Slots allocated in "i + (r/f)"th frame

Figure 2.1: Slot assignment by the Scheduler in the CFDAMA protocols

DAMA schemes, a short jitter tolerant message may be thus transmitted either by a free-assigned slot or by a demand assigned slot, which is allocated in response to its request. In addition there is a possibility that it may also be transmitted by an *undue* demand assigned slot, which is a slot that may have been reserved by another message of the same terminal, but this other message was transmitted through a free-assigned or undue demand assigned slot.

In [42] and [28], the channel time is divided into contiguous slots, each of which contains a portion for sending information called a *data slot* and a portion for sending

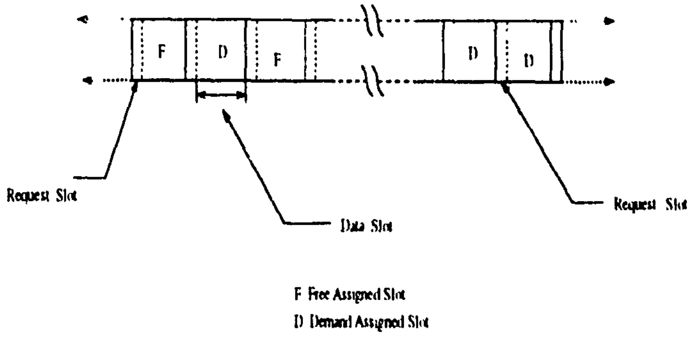
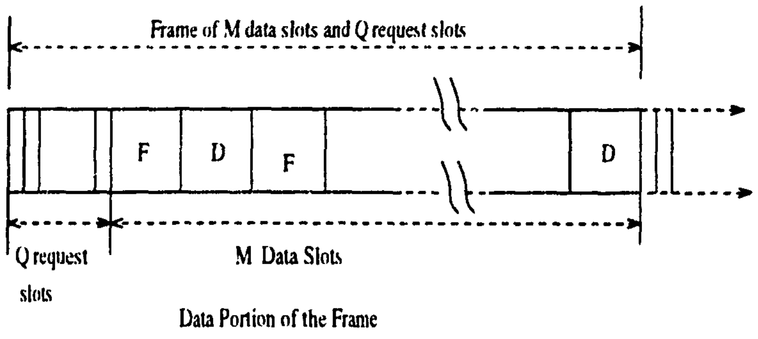


Figure 2.2: Alternating request and data slot format used for jitter-tolerant data

reservations called a *request slot* (Figure 2.2). The data slot and the request slot were assumed to be independent of each other. The request slots are an overhead. The maximum channel utilization is reduced due to the presence of this overhead. Organization of the channel in this manner is suitable if there is only jitter-tolerant traffic, since this kind of traffic is not periodic. However, in the presence of real-time traffic such as voice, which is continous and regular with constant bit rates, there is a need for a *frame structure*, which is repeated after regular periods. We assume that a frame consists of 'M' data slots and 'Q' request slots where M and Q are independent of terminal population N as shown in Figure 2.3. The size of the frame is dependent on the desired transmission bit rates and organization of the channel capacity. If $Q = M$ then the channel assignment will be the same as that followed in [28], which assumes a request slot for every data slot.

An *explicit* requesting strategy is suitable for jitter-tolerant queueable traffic. It is however, not appropriate for stream-time traffic (e.g., voice), which requires channel allocation in a periodic manner. Hence, to support an integrated or mixed traffic in


Frame of M data slots and Q request slots
F
D
F
D
Q request slots
M Data Slots
Data Portion of the Frame

F: Free Assigned Slot
D: Demand Assigned Slot

Figure 2.3: Frame structure used in CFDAMA in a multimedia environment

a multimedia environment the explicit then implicit reservation regime may be used [43]. The CFDAMA protocols may use this unified regime access for real-time and long length jitter tolerant messages. An arriving message has to first place an explicit request for channel capacity. The scheduler will allot channel capacity in response to the explicit request and then the allocated capacity is *implicitly* maintained until a channel release message is received. The message using this scheme cannot be initiated by utilizing a free-assigned slot or a demand assigned slot meant for a short length data message, since the scheduler needs to know that it has to implicitly issue a slot every frame after the first slot. Thus, as far as real-time messages are concerned the CFDAMA behaves as a pure *reservation* scheme. A *pre-releasing* strategy is used so as to improve channel utility by reducing idle time. The terminal sends in a slot-release message to the scheduler one or two round trip delays (depending on whether the scheduler is on the satellite or on the ground) before the end of the message transfer phase. Although the scheduler makes a new allocation when it receives the release request, this channel can still be used by the old terminal until the new capacity allocation information is received by the terminals. The Explicit then Implicit Scheme is illustrated in Figure 2.4 by assuming the message using the regime to be a voice call.

2.2 Requesting Strategies in CFDAMA-Protocols

The performance of the CFDAMA protocols is dependent on the requesting strategy that is used. The requesting may be done through fixed-assigned request slots, which occur after regular intervals, by random access or by piggybacking the requests in the information carrying data packets. The following subsections describe each of these requesting strategies, their advantages and limitations in detail.

2.2.1 Requesting by Fixed-Assigned Slots : CFDAMA-FA

In the fixed assignment type of reservations, pre-assigned, dedicated request slots are allocated to terminals after fixed intervals. The CFDAMA scheme using this fixed-assigned request is called as CFDAMA-FA. A particular terminal can send in its *explicit* request in its *own* request slot only.

We can have a TDMA frame structure for CFDAMA-FA as shown in Figure 2.3. The frame consists of 'M' data slots and 'Q' request slots. Q is usually either less than or equal to M. Increasing Q will give the terminals faster access to request slots. However this will cause an increase in the overhead. A compromise will have to be reached depending on the requirements. In a frame 'Q' of the 'N' terminals will get request slots. Thus for a particular terminal the time between two successive request slots will be $\lceil \frac{N}{Q} \rceil$ frames where $\lceil b \rceil$ denotes the integer immediately greater than or equal to b. Thus on the average a terminal gets a request slot after $\lceil \frac{N}{2Q} \rceil$ frames and hence the time taken for a terminal to get a demand assigned slot increases with the

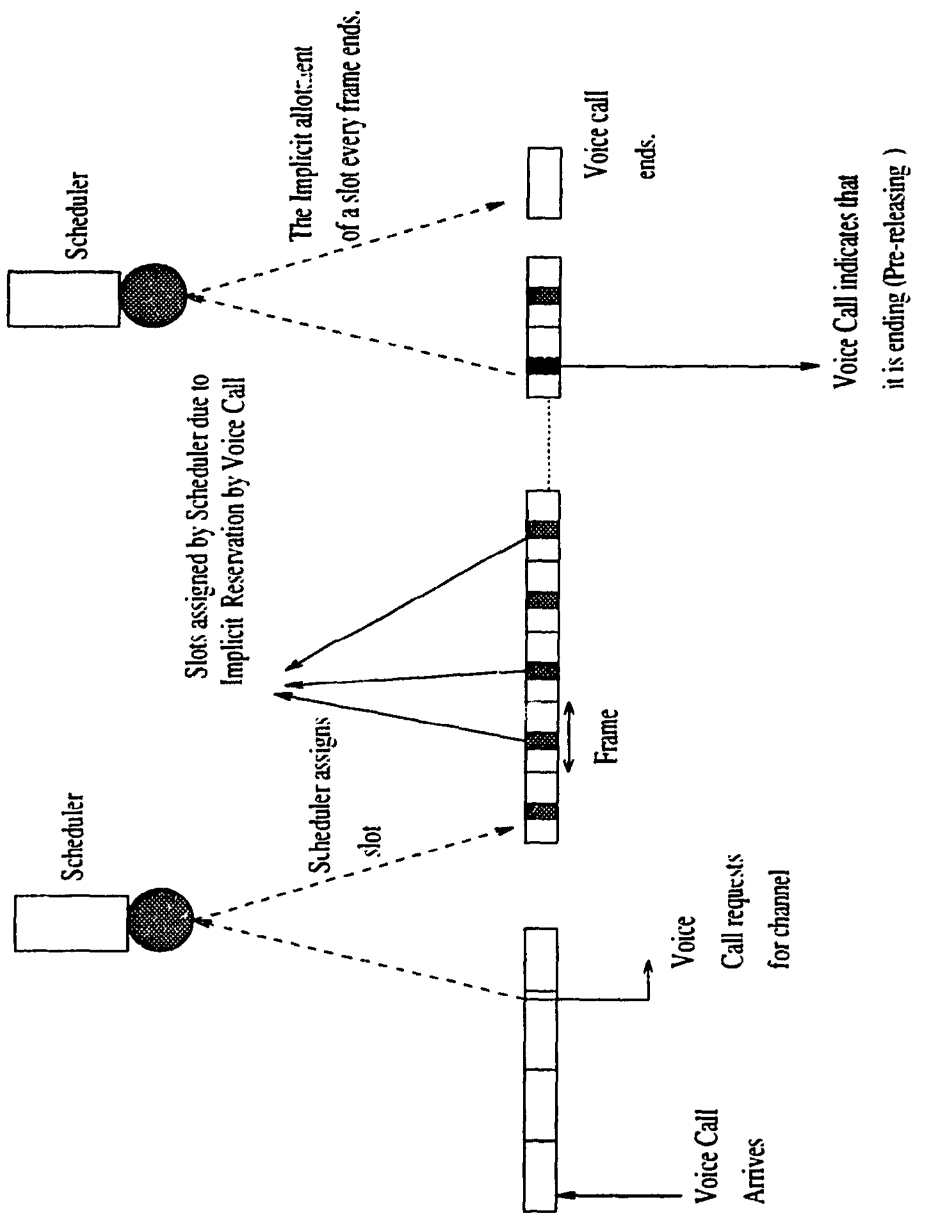

Scheduler
Scheduler
The Implicit allotment of a slot every frame ends.
Voice call ends.
Voice Call indicates that it is ending (Pre-releasing)
Slots assigned by Scheduler due to Implicit Reservation by Voice Call
Scheduler assigns slot
Frame
Voice Call requests for channel
Voice Call Arrives

Figure 2.4: Explicit Then Implicit Reservation

population size. Further, the free-assigned slots for a particular terminal are separated by a minimum of $\lceil \frac{N}{M} \rceil$ frames and thus the time taken to get a free-assigned slot also increases with the terminal population size. The performance of the CFDAMA-FA may be therefore expected to deteriorate as the terminal population size increases.

Let us consider the 'i'th and the '(i+1)' st frames for a tagged terminal. Let the number of packets in the terminal's queue at the beginning of the '(i+1)' st frame be q_{i+1} and that at the beginning of the 'i'th frame be q_i. Let a_i be the number of packets arriving in the 'i'th frame. Let r_i be the total number of slots (both free-assigned and demand assigned together) the tagged terminal *received and utilized* in the 'i'th frame. Then the relationship between these quantities is given by[1]

$$q_{i+1} = [q_i + a_i - r_i]^+ \qquad (2.1)$$

If the distance between successive request slots is given by

$$a = \lceil \frac{N}{Q} \rceil \text{frames}$$

and if the tagged terminal has a request slot in the 'i'th frame, its next requesting slot will be in the '(i+a)' th frame. The number of slots for which the terminal will request is given by the following relation:

$$\text{No of requests} = [q_{i+a} - q_i]^+ \qquad (2.2)$$

1

$$[y]^+ = \begin{cases} y & \text{if } y > 0 \\ 0 & \text{otherwise} \end{cases}$$

[42] and [28] analyze the CFDAMA-FA for the case when $N\tau > R$ where τ and R are the slot size and round trip delay in seconds, respectively. It was shown in [28], that the fixed assignment requesting strategy gave a very good delay vs throughput performance when $N\tau$ is close to or less than the round trip delay.

2.2.2 Piggy-Backed Reservations - CFDAMA-PB

CFDAMA-FA requires the presence of a separate reservation channel, which is an overhead. Thus, the maximum channel utilization is reduced to a certain extent. This overhead may be reduced by piggybacking the requests on the information carrying data slots. This helps achieve a higher channel utilization. The version of CFDAMA using this requesting strategy is called the CFDAMA-PB. The TDMA type frame structure may be used in CFDAMA-PB and will contain only data slots. The frame size may be chosen depending on the desired transmission rates and the organization of the channel.

In CFDAMA-PB, irrespective of whether a request is transmitted or not every data packet is to have a field that is set aside for requesting information. Further, examining each and every data packet for a request requires an increase in processing time. Thus, the overhead for requesting in CFDAMA-PB may be comparable with that of CFDAMA-FA, if the value of 'Q', i.e., the number of request slots per frame in CFDAMA-FA, can be reduced without affecting the performance of the protocol much[2].

In CFDAMA-PB a terminal will have to wait for its *own data slots* for sending its requests. For a system of homogeneous terminals the distribution of the data slots among the terminals may be assumed to be uniform and thus, the time taken for a terminal to get slots (either free-assigned or demand-assigned) may be expected to

[2]This is possible for certain ranges of N τ as may be seen in Chapter 3

increase as the population size increases. The performance of the CFDAMA-PB may be expected to deteriorate with an increase in the terminal population size, as in the case of the CFDAMA-FA. It has been shown in [28] that the Piggybacked reservation strategy is seen to give a very good 'delay vs throughput' performance when N_T is close to the round trip delay R. or is less than R. Further it has been shown that the performance of CFDAMA-PB and the CFDAMA-FA are almost the same.

Let us say that when data slots are alloted for the 'i'th time to a tagged terminal. the 'i'th server is at the terminal in question. Let n_{i+1} be the number of packets in the terminal queue when the $(i+1)$ st server arrives and n_i be the number packets when the i th server arrives. Let σ_i be the number of packets the 'i'th server serves. Then

$$n_{i+1} = [n_i + a_i - \sigma_i]^+ \tag{2.3}$$

and

$$\text{No of requests on the } (i+1) \text{ st server} = [n_{i+1} - n_i]^+ \tag{2.4}$$

Here. a_i is the number of arrivals in the time between the the ith and the $(i+1)$st servers.

In [22] an ID (a unique number associated with each terminal) re-ordering scheme has been introduced in which the scheduler (which has been assumed to be on-board the satellite) maintains an ID table. Whenever any terminal gets a slot either by demand or free-assignment the scheduler moves the ID of the terminal in question to the bottom of the table. The free-assigned slots are assigned according to the status of the ID table. The terminal whose ID is at the top of the ID table is assigned the next free-assigned slot. This is done to ensure fairness. However, the expected delay performance characteristic has been seen to be almost the same for the case with and the case without ID re-ordering [28].

2.2.3 Random Access Requesting - CFDAMA RA

Requesting through fixed-assigned request slots or by piggybacking the request in the traffic packets may be advantageous when the population size N is not very large. It has been shown in [28] that, both have an excellent 'delay vs throughput' performance when $N\tau < R$ or is comparable to R, where τ and R are the time-slot size and round-trip delay, respectively. When $N\tau >> R$ their performance is degraded due to the long waiting time for a request slot in CFDAMA-FA, or for an available data slot in CFDAMA-PB. Another way of requesting is to open the request slots for contention. Unlike in the fixed assignment strategy when a terminal has to wait for its own slot to send in its request, the random access scheme permits a terminal to transmit a request in any request slot. We call this version of CFDAMA as CFDAMA-RA. The 'Slotted Aloha' technique may be used for requesting.

CFDAMA-RA may be expected to be advantageous when $N\tau >> R$ for a fixed system load. At low and medium throughputs the random access proves to be advantageous due to the ability of a terminal to almost immediately access a request slot. However it is disadvantageous at higher throughputs due to an increase in collision rate. At low throughputs, in CFDAMA-FA or CFDAMA-PB packets are transmitted mostly by free-assigned slots. When N is large the distance between these slots is large and thus causes greater delays. In the case of CFDAMA-RA the packets are transmitted by demand assigned slots (due to the large distance between the free-assigned slots) at low throughputs by allowing the terminals to almost immediately access request slots. The packets are however, transmitted through the free-assigned slots at higher throughputs due to greater collision rates.

The CFDAMA-RA may be used with a TDMA type frame structure as in Figure 2.3. The request slots may be grouped together at the beginning of the frame. A particular terminal will choose one of the 'Q' request slots at random to transmit its request. If more than one terminal chooses the same slot for transmitting its request, a collision results and the request is lost. Increasing the number of request slots will decrease the probability of collisions. However, this will result in an increase in overhead and thus limit the maximum channel utilization. Thus, depending on the requirements the number of request slots may be appropriately chosen. If $Q = M$ and the traffic consists of only jitter tolerant data it may prove advantageous to have an alternating data slot, request slot organization as shown in (Figure 2.2). A terminal may transmit a request in the next immediate request slot if there is an arriving message. Thus, the minimum mean delay in this case turns out to be just equal to two round trip delays (if there is an on-board scheduler) if $R << N\tau$.

CFDAMA-RA will be stable (unlike the Slotted-Aloha based Demand Assignment Multiple Access schemes) due to the presence of free-assigned slots. In case of all collisions, packets can still be transmitted by means of free-assigned slots after finite delays. A retransmission strategy may be incorporated, but for a terminal to know that its request has failed it takes at least R seconds (or $2R$ seconds depending on the position of the scheduler). Further, retransmissions at higher throughputs will lead to higher collision rates thereby deteriorating the performance.

Let us assume a frame format as in Figure 2.3. The number of requests a terminal makes at the beginning of each frame is equal to the number of packets in its queue that do not have requests. The number of requests at the beginning of any frame may be expressed as :

$$\text{No of requests} = [q_{i+1} - q_i]^+ \qquad (2.5)$$

Here q_i refers to the number of packets in the terminal queue at the beginning of the i th frame as in Eq. 2.1 and Eq. 2.2. The relation between q_{i+1} and q_i is the same as in Eq. (2.1).

It was seen that the CFDAMA-PB was advantageous when N_T is close to the round trip delay R or less than R [22]. When $N_T >> R$, it is the CFDAMA-RA that may be preferred. Thus it may prove advantageous to allow the piggybacking and the random access modes of requesting to co-exist. The random access should be used in a controlled manner. Only those terminals that do not any forthcoming demand assigned data slots (as a result of requests made earlier) should be allowed to request using the random access request slots. These slots should be minimal so as to save on overhead. This scheme may prove to be advantageous for all ranges of N_T.

To summarize, the CFDAMA protocols combine *free* assignment with demand assignment effectively to reduce the average transmission delay for jitter-tolerant data. It behaves as a *reservation* based multiple-access scheme as far as real-time voice is concerned. Real-time voice may be transmitted by using the Explicit then Implicit Reservations. The 'delay vs throughput' performance in the CFDAMA protocols may be expected to deteriorate with population size. The requesting in CFDAMA may be made by either pre-assigning request slots, by random access or by piggybacking requests in traffic packets. Different requesting strategies are advantageous at different ranges of terminal population size.

The next two chapters gives mathematical analyses of the CFDAMA-FA and the CFDAMA-RA. The performance of CFDAMA-PB may be expected to be almost the same as that of CFDAMA-FA [42].

We further look at the integration of real-time traffic and the behavior of the CFDAMA in such an environment in Chapter 5.

Chapter 3

Performance of CFDAMA-FA in packet satellite communications

It has been seen in Chapter 2, that the CFDAMA protocols may use different requesting strategies depending on the requirements. The performance of the CFDAMA schemes in a *packet satellite* system using fixed-assigned and piggy-backed requesting strategies has been analyzed in [42, 22]. The analyses however, were limited to large population sizes. In order to obtain a better understanding of the behavior of the CFDAMA schemes, an analysis of the CFDAMA-FA for a general population size is presented in this chapter.

3.1 System Modelling

Consider a Time-Division Multiple-Access (TDMA) frame structure containing 'M' equal size time slots or channels for traffic. A superframe consists of 'X' time-slots where $X = aN$ and 'N' is the terminal population (Figure 3.1). The frame and time-slot sizes are selected on the basis of the required channel capacity and transmission

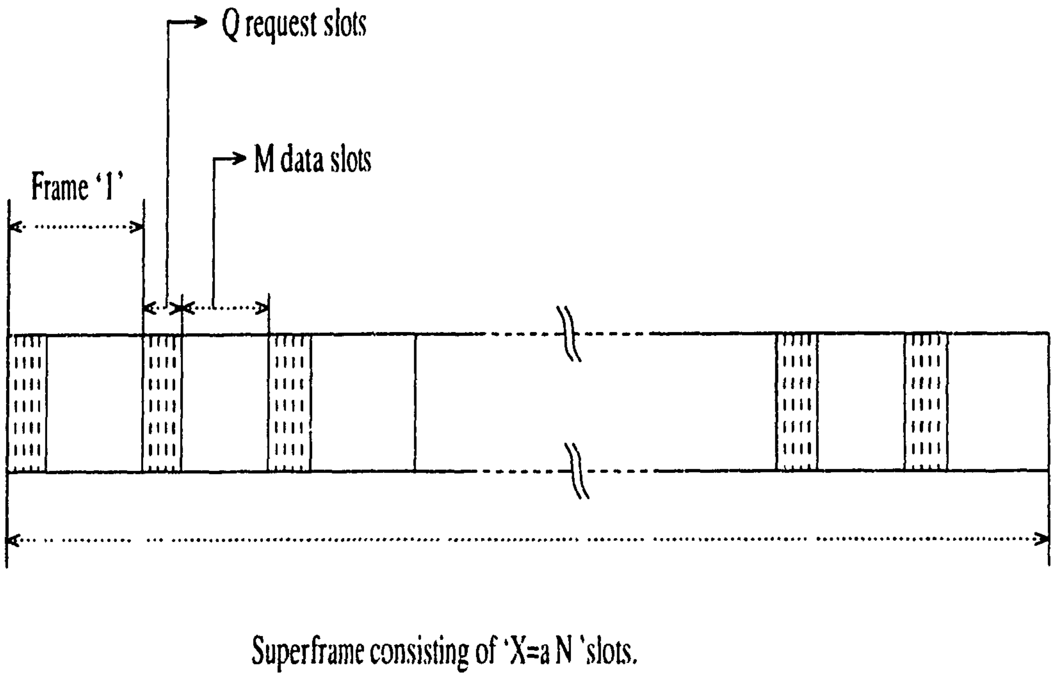


Figure 3.1: Superframe structure used in analysis of CFDAMA-FA

rate in a given application. M, a, X and N are integers. A particular terminal has one pre-assigned request slot in every superframe.

We assume the following:

• The traffic arrival processes are Poisson and the terminals are homogeneous with unlimited buffer capacity.

• The traffic is of jitter tolerant type and the generated messages are of constant length and constitute a single packet.

For convenience we use the time-slot size τ as the time unit in the following discussion. The round trip delay is expressed as R time-slots. We define

$$r = \begin{cases} R & \text{for an on-board scheduler} \\ 2R & \text{for an on-ground centralized scheduler} \end{cases} \tag{3.1}$$

where r is assumed to be an integer smaller than X. This assumption implies that the only packets that may be present at a given instant in the terminal's queue, are those arriving in the current superframe # n, and previous superframe # (n-1).

Consider the activities in the current superframe # n for a particular tagged terminal.

Define

q: number of old packets in the terminal queue at the beginning of superframe # n.

l: number of packets arriving in superframe #n.

For a Poisson traffic source, the probability that l new packets arrive [8] in a superframe is

$$Pr\{\text{l new arrivals}\} = \frac{e^{-\lambda X}(\lambda X)^l}{l!} \tag{3.2}$$

where $\lambda = \Lambda/X$ is the arrival rate per time slot of each terminal. Λ is the system arrival rate. Since $\Lambda < 1$ for a stable system, it can be seen that $Pr\{l > X\}$ is negligible. For this reason we assume that in one time-slot interval, there is at most one packet arriving in the terminal queue[1]. In this case the expected arrival instant of the k th packet is $k\Delta$ where $k \leq l \leq X$ where,

[1]We can calculate the probability of this assumption going wrong as:

$$\begin{aligned} P_e = P\{\text{The number of arrivals is more than one per slot}\} &= \sum_{i=2}^{\infty} \frac{e^{-\lambda}\lambda^l}{l!} \\ &= 1 - e^{-\lambda} - \lambda e^{-\lambda} \end{aligned} \tag{3.3}$$

We can limit this probability to some small quantity ε and determine the minimum number of terminals that should be in the system so that the probability of this assumption not being valid is less than ε.

Thus we have

$$\Delta = \frac{X+1}{l+1} \tag{3.4}$$

and the time origin is taken as the beginning of the superframe as illustrated in Fig 3.2.

The terminal will send a reservation to the scheduler in its pre-assigned request slot at the end of the current superframe # n if its queue (at this instant) is not empty. The expected instant of the first due demand-assigned slot is

$$Y = r + \bar{S} + X \tag{3.5}$$

where r is defined in Eq.(3.1).

$\bar{S}$ is the scheduler queueing delay to be derived later.

When a packet reaches the head of the terminal queue, it can be transmitted via a free-assigned slot if it is available. The fractions of demand-assigned and free-assigned slots are d and $(1-d)$, respectively. For a round-robin free-assignment strategy, the *average* distance between two consecutive *free*-assigned slots, to a particular terminal is $\frac{N}{(1-d)}$. Therefore, the probability that a packet at the head of the terminal queue is transmitted by a free assigned slot is

$$p = \frac{(1-d)}{N} \tag{3.6}$$

if it first came to a non-empty queue.

$$P_e = 1 - e^{-\Lambda/N} - \lambda e^{-\Lambda/N} \leq \epsilon$$

From the above equation taking the equality N can be determined.

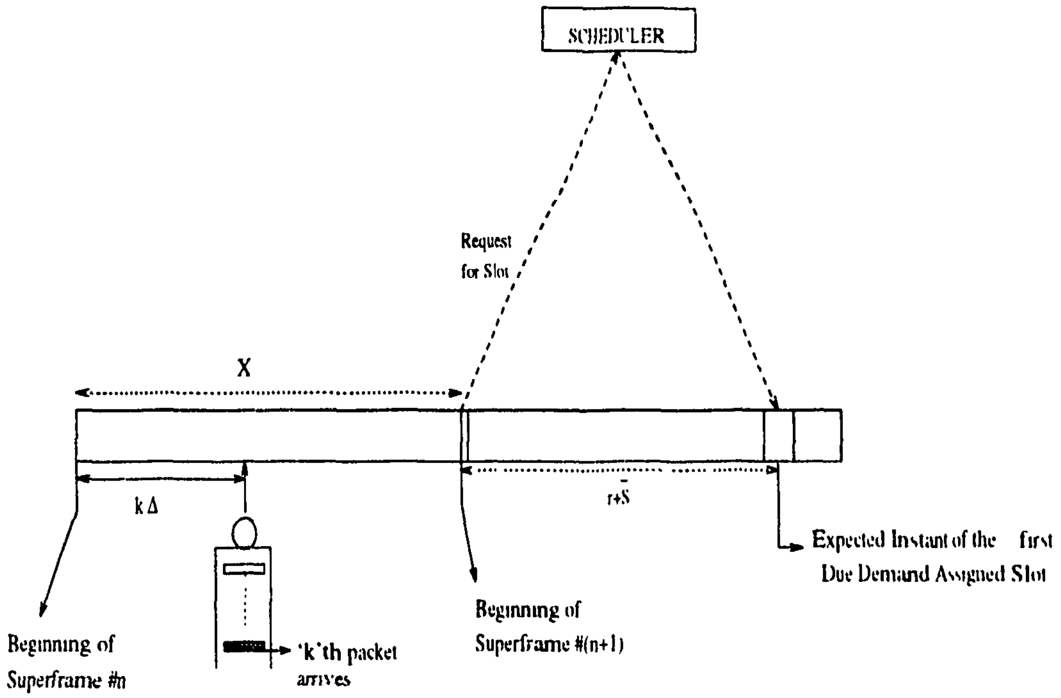

SCHEDULER
Request for Slot
X
kΔ
r+S
Expected Instant of the first Due Demand Assigned Slot
Beginning of Superframe #n
'k'th packet arrives
Beginning of Superframe #(n+1)

Figure 3.2: Timing Relationship

If it first came to an empty queue this probability is doubled, since on the average the packet arrives in the middle of the duration between its free-assigned slots [42], and is equal to

$$p' = 2p = \frac{2(1-d)}{N} \tag{3.7}$$

3.2 Analysis of CFDAMA-FA for a general population size

Let us consider an arbitrary packet say the kth of the l packets arriving to the tagged terminal in the current superframe. We consider two cases :

a) When the terminal queue was empty at the beginning of the superframe i.e., $q = 0$.

b) When there are $q > 0$ old packets lined up at the beginning of the superframe.

In the first case, the expected value of the delay of the kth packet, given that there were l arrivals at the terminal in the superframe, is given by the following expression:

$$\begin{aligned} E(D_k|l, q &= 0) = \\ \sum_{i_1=m_1}^{Y+1} (1-p')^{i_1-m_1} \quad &. \quad p' \sum_{i_2=m_2}^{Y+2} (1-p)^{i_2-m_2}.p \ldots \sum_{i_k=m_k}^{Y+k} (1-p)^{i_k-m_k}.p(i_k - k\Delta) \\ +[1 - \sum_{i_1=m_1}^{Y+1} (1-p')^{i_1-m_1} \quad &. \quad p' \sum_{i_2=m_2}^{Y+2} (1-p)^{i_2-m_2}.p \ldots \sum_{i_k=m_k}^{Y+k} (1-p)^{i_k-m_k}.p] \\ &. \quad (Y + k - k\Delta) \qquad (3.8) \end{aligned}$$

where m_h are defined by the following:

$$m_1 = max(1, \Delta)$$

and when $h \neq 1$

$$m_h = max(i_{h-1}, h\Delta)$$

The first term in Eq.(3.8) represents the delay when the 'k'th packet is transmitted by a free-assigned slot and the second term gives the delay when the 'k'th packet has to be transmitted by its own demand assigned slot[2].

[2] Δ is given by Eq.(3.4).

For the case when $k=1$, Eq.(3.8) becomes

$$E(D_1|l,q=0) = \sum_{i_1=m_1}^{Y+1} (1-p')^{i_1-m_1}.p'.(i_1-\Delta) + [1-\sum_{i_1=m_1}^{Y+1}(1-p')^{i_1-m_1}.p'](Y+1-\Delta) \qquad (3.9)$$

We now consider the case, where the number of packets from the previous superframe q is greater than zero. We take the case when there are 'l' arrivals in the superframe and evaluate the delay of the k th packet.

There are two possibilities, the k th packet arrives after 'r' or before 'r'. Since we assume that the 'l' arrivals occur uniformly in the superframe of 'X' time-slots the number of arrivals before 'r' is given by

$$l_1 = [\frac{lr}{X}]^+; \qquad (3.10)$$

where $[x]^+$ denotes the integer which is immediately greater than or equal to x. Thus if $k \in (1,l_1)$, it could be transmitted by a free-assigned slot, or an undue demand assigned slot, or its due demand assigned slot. On the other hand, if $k \in (l_1,l)$ then it cannot be transmitted by an undue demand assigned slot.

Let 'b' denote the maximum number of free-assigned slots obtained by a terminal in an interval of r slots, i.e.,

$$b = [\frac{r}{N}]^+ \qquad (3.11)$$

The expected delay of the k th packet if $k \in (1,l_1)$ is given by:

$$
\begin{aligned}
E(D_k|k &\in (1,l_1),l,q) = \sum_{i_1=1}^{W'+1} p'(1-p')^{i_1-1} . \sum_{i_2=i_1}^{W'+2} p(1-p)^{i_2-i_1} \dots . \sum_{i_q=i_{q-1}}^{W'+q} p(1-p)^{i_q-i_{q-1}} \\
&. \sum_{j_1=M_1}^{W'+q+1} p(1-p)^{j_1-M_1} . \sum_{j_2=M_2}^{W'+q+2} p(1-p)^{j_2-M_2} \dots \sum_{j_k=M_k}^{W'+q+k} p(1-p)^{j_k-M_k} . \{j_k - k\Delta\} \\
&+ \sum_{j=k}^{b} \begin{pmatrix} b \\ j \end{pmatrix} (1-d)^j d^{b-j} \sum_{i=k}^{min(j,q+k)} \begin{pmatrix} j \\ i \end{pmatrix} \Lambda^i (1-\Lambda)^{j-i} [W' - k\Delta] \\
&+ \sum_{j=0}^{b} \begin{pmatrix} b \\ j \end{pmatrix} (1-d)^j d^{b-j} \sum_{i=0}^{min(j,k-1)} \begin{pmatrix} j \\ i \end{pmatrix} \Lambda^i (1-\Lambda)^{j-i} \\
&. \{ \sum_{i_1=W'}^{Y'+1} p(1-p)^{i_1-W'} . \sum_{i_2=i_1}^{Y'+2} p(1-p)^{i_2-i_1} \dots \sum_{i_{k-i}=i_{k-i-1}}^{Y'+k-i} p(1-p)^{i_{k-i}-i_{k-i-1}} \\
&. [i_{k-i} - k\Delta] \\
&+ [1 - \sum_{i_1=W'}^{Y'+1} p(1-p)^{i_1-W'} . \sum_{i_2=i_1}^{Y'+2} p(1-p)^{i_2-i_1} \dots \sum_{i_{k-i}=i_{k-i-1}}^{Y'+k-i} p(1-p)^{i_{k-i}-i_{k-i-1}}] \\
&. (Y' + k - k\Delta)\} \qquad (3.12)
\end{aligned}
$$

where

$$M_1 = max(i_q, \Delta)$$

$$M_h = max(j_{h-1}, h\Delta) \quad \text{if h} \neq 1$$

$$W' = r + \bar{S}$$

The first term in the above equation represents the case when the tagged packet is transmitted through a free-assigned slot before 'r' slots , the second term covers the case when the tagged packet is transmitted through an undue demand assigned slot occuring after r slots and the third term covers the case when it does not get a free-assigned or an undue demand assigned slot before r slots and therefore has to be transmitted through a free-assigned slot after r slots or through its own demand assigned slot.

In case $k \in (l_1, l)$ then the tagged packet cannot be transmitted before r slots. In this case it arrives after r slots and so has to be transmitted by either a free-assigned slot occuring after r slots or by its own demand assigned slot.

This may be computed as follows:

$$
\begin{aligned}
E(D_k \quad | \quad & k \in (l_1, l), l, q) = \sum_{j=0}^{b} \begin{pmatrix} b \\ j \end{pmatrix} (1-d)^j d^{b-j} \sum_{i=0}^{min(j,d_1)} \begin{pmatrix} j \\ i \end{pmatrix} \lambda^i (1-\lambda)^{j-i} \\
. \quad & \{ (\sum_{i_1=\tilde{m}_0}^{Y+1} p(1-p)^{i_1-\tilde{m}_0} \sum_{i_2=\tilde{m}_1}^{Y+2} p(1-p)^{i_2-\tilde{m}_1} \dots \sum_{i_{k-i}=\tilde{m}_{k-i-1}}^{Y+k-i} p(1-p)^{i_{k-i}-\tilde{m}_{k-i-1}} \\
. \quad & [i_{k-i} - k\Delta]) \\
+ \quad & (1 - \sum_{i_1=\tilde{m}_0}^{Y+1} p(1-p)^{i_1-\tilde{m}_0} \sum_{i_2=\tilde{m}_1}^{Y+2} p(1-p)^{i_2-\tilde{m}_1} \dots \sum_{i_{k-i}=\tilde{m}_{k-i-1}}^{Y+k-i} p(1-p)^{i_{k-i}-\tilde{m}_{k-i-1}} \\
. \quad & [Y + k - k\Delta] \}
\end{aligned}
\tag{3.13}
$$

where

$$\tilde{m}_0 = max(W, (i+1)\Delta)$$

$$\tilde{m}_h = max(i_{h-1}, (i+h)\Delta) \qquad \text{if } h \neq 0$$

The computations for these expressions become cumbersome as k increases. To reduce the computational complexities, beyond a certain value of k, we can make the assumption that the only way that the tagged packet could be transmitted is by its own due demand assigned slot. If $k > a + b$ then this is certainly true because the maximum number of free-assigned slots obtained by one terminal in an interval of $(r + X + \bar{S})$ is $(a + b)$. We can make an approximation that if $q + k > a$, then the tagged packet can be transmitted only by its due demand assigned slot.

The delay in this case will be given by the following:

$$E(D_k|l,q,q+k>a) = X + r + \tilde{S} - k\Delta \tag{3.14}$$

We have to evaluate the probabilities of finding 'q' packets in the tagged terminal's queue at the beginning of the superframe, and 'd', the fraction of the slots that are demand assigned. Knowing these, we could model the scheduler queue as an M/G/1 queue.

We can compute the probabilities P_q of finding q packets in the tagged terminal's queue as follows:

$$P_0 = \sum_{j=0}^{a} \begin{pmatrix} a \\ j \end{pmatrix} (1-d)^j d^{a-j} \sum_{\epsilon=0}^{j} \begin{pmatrix} j \\ \epsilon \end{pmatrix} \Lambda^{\epsilon}(1-\Lambda)^{j-\epsilon} . \sum_{l=0}^{\epsilon} \frac{\epsilon^{-\Lambda a}(\Lambda a)^l}{l!} \tag{3.15}$$

The above equation comes from the fact that in order for the terminal's queue to be empty at the beginning of superframe #n, the number of arrivals in the previous superframe #(n-1) ought to be less than or equal to the utilized free-assigned slots.

If q is non-zero we evaluate P_q by the following approximate expression:

$$P_q = \sum_{l=q}^{q+a} \sum_{j=(l-q)}^{a} \begin{pmatrix} a \\ j \end{pmatrix} (1-d)^j d^{a-j} \begin{pmatrix} j \\ l-q \end{pmatrix} \Lambda^{l-q}(1-\Lambda)^{j-l+q} \frac{\epsilon^{-\Lambda a}(\Lambda a)^l}{l!} \tag{3.16}$$

The above equation arises because if the number of packets queued up at the beginning of the superframe is q, then the number of utilized free-assigned slots in the previous superframe given that there were l arrivals will be $(l-q)$. In addition since the maximum number of free-assigned slots in a superframe is a the number of arrivals cannot exceed $(q+a)$.

The fraction of demand assigned slots is derived as follows:

The expected number of packets that request at the the beginning of a superframe for a terminal is given by

$$E_d = \sum_{j=0}^{a} \left(\begin{array}{c} a \\ j \end{array} \right) (1-d)^j d^{a-j} \sum_{i=0}^{j} \left(\begin{array}{c} j \\ i \end{array} \right) \Lambda^i (1-\Lambda)^{j-i} \sum_{l=max(1,i)}^{X} (l-i) \frac{e^{-\Lambda a}(\Lambda a)^l}{l!} \quad (3.17)$$

The above expression is an approximation, since it is assumed that the number of utilized slots is equal to the number of new packets escaping. This assumption is reasonable since a utilized free-assigned slot before r implies that either a new packet is transmitted or an undue demand assigned slot is created. If a free-assigned slot is utilized after r then a new packet has been transmitted.

Thus the total number of demand assigned slots for a system of N homogeneous terminals in a superframe is

$$\delta = NE_d \quad (3.18)$$

The fraction of demand assigned slots is given by

$$d = \frac{\Lambda\delta}{X} \quad (3.19)$$

The fraction of demand assigned slots, d also represents the arrival rate to the scheduler queue. The scheduler queue delay can be found by the following expression

$$\bar{S} = 1 + \frac{d}{2(1-d)} \quad (3.20)$$

This scheduler queue delay is generally very small in comparison to the round trip delay and may be neglected.

The unconditional average delay is

$$E(D) = \sum_{q=0}^{X} P_q E(D|q) \quad (3.21)$$

where

$$E(D|q) = \frac{1}{1 - e^{-\Lambda q}} \sum_{l=1}^{\infty} \sum_{k=1}^{l} E(D_k|l,q).\frac{a_l}{l}$$

and

$$a_l = \frac{e^{-\Lambda q}(\Lambda q)^l}{l!}$$

The average packet delay $[E(D) + R]$

3.3 Illustrative Results and Discussions

To illustrate the performance of CFDAMA-FA protocols in a wide range of terminal populations we consider the following system parameters:

- $r = R$, equivalent to one round-trip delay of 270 ms.
- Overhead associated with request slots: 0.06 of the system capacity.

Except for the specific values mentioned above the results are represented in terms of the *normalized* terminal population η defined as the ratio of $N\tau$ to R i.e., $\eta = \frac{N\tau}{R}$ (assuming an on-board scheduler[3]). Therefore the results can be applied in different applications in which τ and N are specified. In the examples τ was assumed to be 5 ms.

Simulations have also been done. Figures 3.3 and 3.4 show the analytical and simulation results for $\eta = 0.56$ and 0.37, respectively. They indicate a good agreement between analytical and simulation results. The comparison of the values obtained by simulation and analysis, for the fraction of demand assigned slots d, is shown in Figure 3.5.

[3] $\eta = \frac{N\tau}{2R}$ for a scheduler on ground.

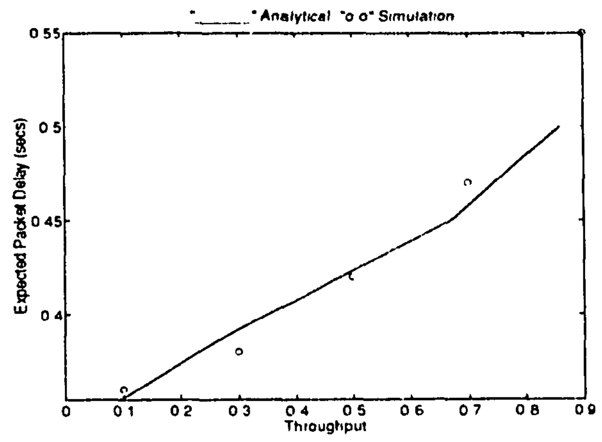


Figure 3.3: Comparison between analytical and simulation results for $\eta = 0.56$

Figure 3.6 shows the delay-throughput performance of CFDAMA-FA for various values of η. At low throughput a low average packet delay is obtained, thanks to the free-assignment aspect of the CFDAMA. Indeed, as indicated in Figure 3.6, the average packet delay for throughputs of 0.1 or lower can be approximately represented by $(1 + 0.5\eta)$ round-trip delays where one round-trip delay is for transmission from source to destination terminals and 0.5η represents the average time spent waiting for a free-assigned slot.

As throughput increases, the average packet delay also increases. However it is important to note, from Figure 3.6, that this increase in average packet delay is slower as η is reduced. For example the increases in average packet delay, as the throughput changes from 0.1 to 0.5, are about 25 ms and 65 ms for $\eta = 0.09$ and $\eta = 0.56$, respectively. This indicates the dominant effects of free-assignment and consequently the advantages of CFDAMA protocols in satellite systems with low terminal population.

Consider the curves for $\eta = 0.09$ and 0.18 in Figure 3.6. For a large range of throughput (up to 0.7), the average packet delays do not exceed 320 ms and 350

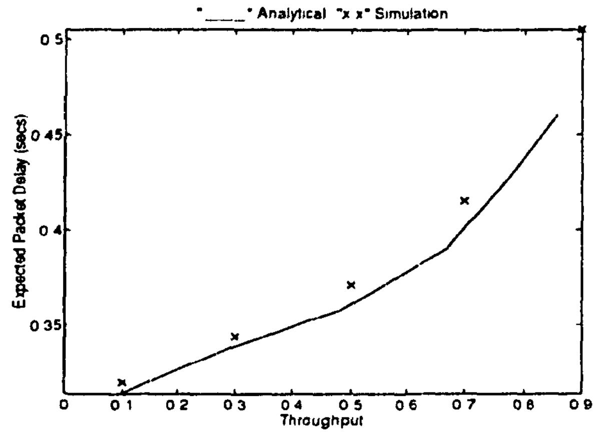


Figure 3.4: Comparison between analytical and simulation results for $\eta = 0.37$

ms for $\eta = 0.09$ and 0.18, respectively. If DAMA were used with $r = R$, the average packet delay would be more than two round trip delays of 540 ms. The above mentioned results also indicate that for satellite systems with low terminal population, the location of the scheduler (e.g., on-ground centralized scheduler or on-ground distributed scheduler or an on-board scheduler) does not have significant effects on the delay-throughput performance of the CFDAMA. Furthermore in this case, the request strategy (i.e., FA, RA or PB) also does not control the delay-throughput performance. This allows system designers a flexibility in selecting the request strategy and location of the scheduler to fit other system requirements without sacrificing the performance of CFDAMA protocols.

3.4 Approximation for $N << R$

The effects of η can also be seen by considering the fraction of demand assigned slots, d, as illustrated in Figure 3.7. As η increases, the benefits of free-assignment are reduced. Therefore the fraction of demand-assigned slots, d, increases with throughput

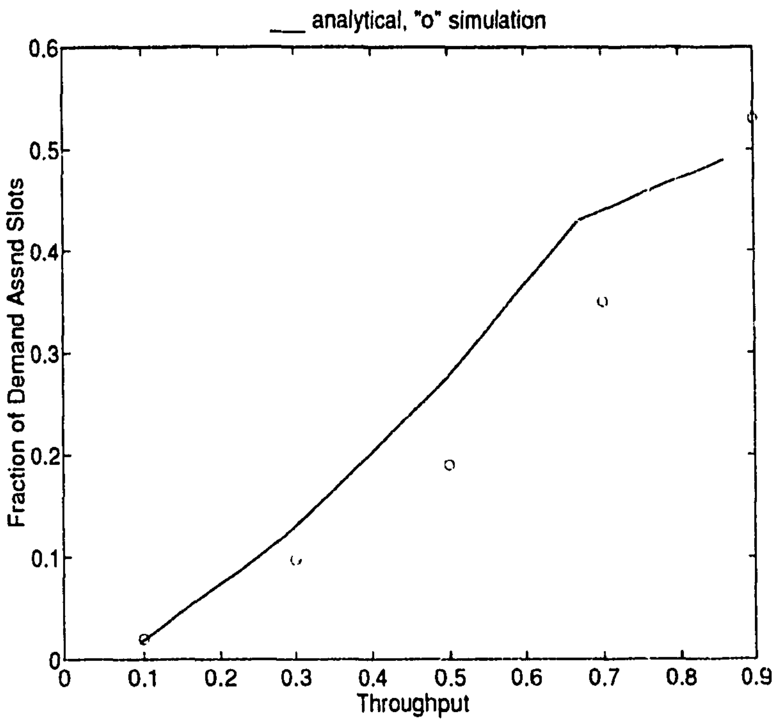


Figure 3.5: Analytical and Simulation results comparing 'd' for η=0.56

and terminal population.

For $\eta \leq 0.1$, the fraction of demand-assigned slots is less than 0.1 for throughputs up to 0.7. This indicates that most incoming packets are not transmitted through their *due demand-assigned* slots.

From Eq. (3.14), the kth arriving packet has to wait for $(r + X + S - k\Delta)$ slots to obtain its own demand assignment. Since $\bar{S}$ is negligible, and $X > R$ and, on the average the expected value of $k\Delta$ is $X/2$, this time interval can be represented as $\Omega = \lceil \gamma R \rceil$ where $\gamma \approx 1.5$ for $r = R$. During this time interval, there are

$$\lceil \frac{\gamma R}{N/(1-d)} \rceil = \lceil \frac{\gamma(1-d)}{\eta} \rceil$$

free-assigned slots available to the tagged terminal. It can be seen that if $(q+k) <$

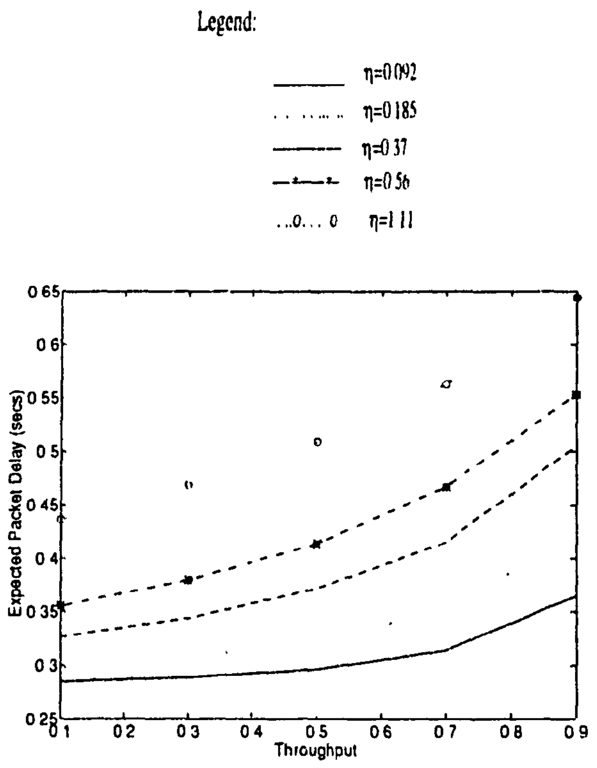


Figure 3.6: Delay of CFDAMA-FA for various values of η

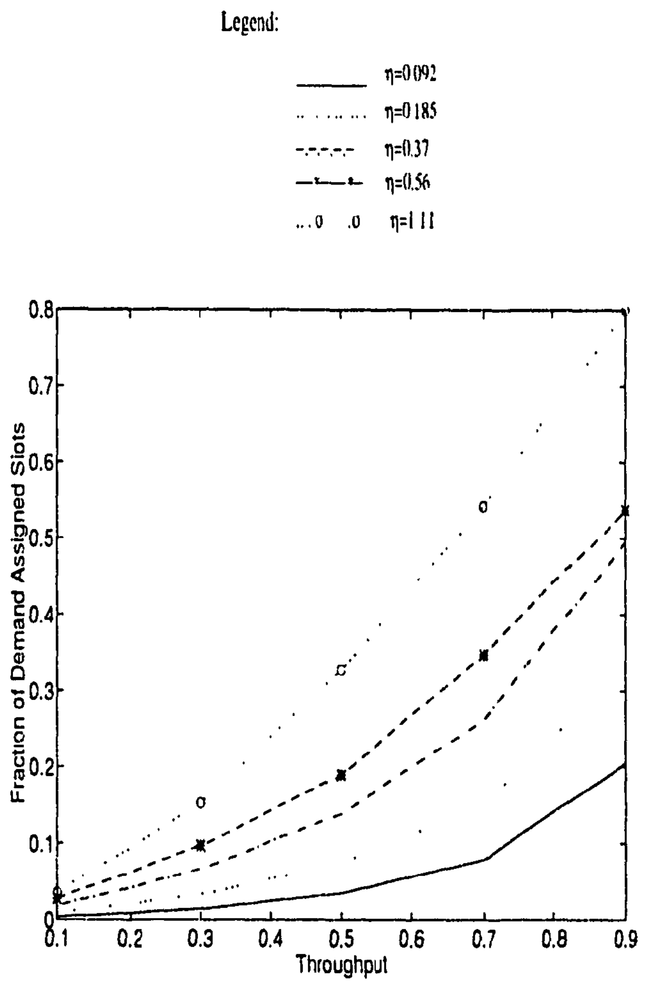


Figure 3.7: Fraction of demand assigned slots versus throughput for different η

$\lceil \frac{\gamma(1-d)}{\eta} \rceil$ then the kth arriving packet will be transmitted by a free-assigned slot. For $N << R$ (i.e., $\eta << 1$), d is also reduced and consequently this event occurs more frequently. Furthermore, the kth arriving packet also has a chance to be transmitted by an undue demand-assigned slot. This undue demand-assigned slot is the result of a request from an antecedent packet that was transmitted through a free-assigned or undue demand-assigned slot.

Consider the average service time of the kth packet when it reaches the head of the terminal queue. This average service time can be expressed in a form similar to Eq (3.9), i.e.,

$$E(\varsigma) \approx \sum_{i=1}^{\Omega} p(1-p)^{i-1} i + [1 - \sum_{i=1}^{\Omega} p(1-p)^{i-1}][\Omega]$$

This is further simplified as:

$$E(\varsigma) = \frac{[1-(1-p)^{\Omega+1}]}{p} - (\Omega+1)(1-p)^{\Omega} + \Omega(1-p)^{\Omega} \qquad (3.22)$$

where

$$p = \frac{1-d}{N} = \frac{(1-d)\gamma}{\eta\Omega}$$

For $\Omega >> 1$ using the relation $\epsilon^{-\nu} \approx (1 - \frac{\nu}{\Omega})^{\Omega}$, $E(\varsigma)$ is simplified to

$$E(\varsigma) \approx \frac{1-\epsilon^{-\nu}}{p} \qquad (3.23)$$

where $\nu = \frac{(1-d)\gamma}{\eta}$.

As η decreases, d also decreases and ν increases. As a result $\epsilon^{-\nu} << 1$. For example with $\eta < 0.1$, $d < 0.1$, $\gamma = 1.5$, $\epsilon^{-\nu} < 1.4 \times 10^{-6}$, which is negligible. This indicates that for a small η, the average service time of a packet at the head of the terminal queue is approximated as

$$E(s) = \frac{1}{p} = \frac{N}{(1-d)} \approx N \text{ for } d << 1 \tag{3.24}$$

The pgf of the service time is approximated as:

$$G(z) = \sum_{i=1}^{\infty} \frac{1}{N}(1 - \frac{1}{N})^{i-1} z^i \tag{3.25}$$

The mean value and the second moment of the service time can be obtained by simply differentiating the above pgf and setting $z = 1$. They are

$$\bar{G} = N \tag{3.26}$$

$$\bar{G^2} = N(N-1) \tag{3.27}$$

Substituting these values in the Pollaczek Khinchin formula for the M/G/1 queue where G is geometric we obtain the total waiting and service time.

$$\bar{T} = N + \frac{\lambda N(N-1)}{2(1-\lambda N)} \tag{3.28}$$

where λ is the arrival rate per terminal in packets/slot. The average packet delay is given by $\bar{D} = \bar{T} + R$. Figure 3.8 shows the simulation and analytical results using Eq (3.28) for $\eta = 0.1$.

The geometric approximation is seen to be in an excellent agreement with simulation results when $N << R$.

The above results suggest that the performance of CFDAMA-FA improves with a decrease in population size. The fraction of slots that are demand assigned becomes

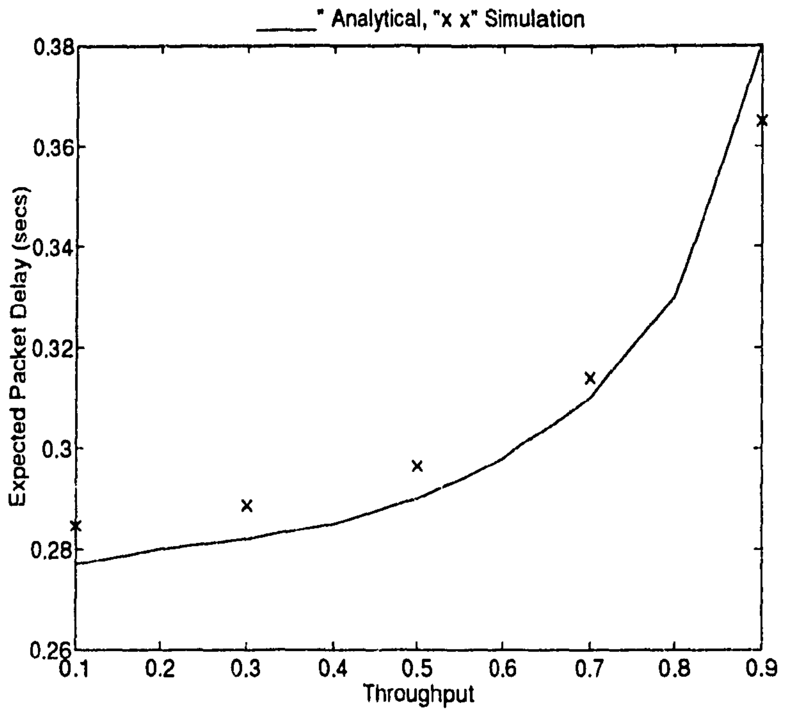


Figure 3.8: Comparison between analytical (geometric service approximation) and simulation results: $\eta = 0.09$

smaller as the population size decreases for a fixed packet size. Further at very low population sizes it has been seen that almost none of the packets are transmitted by their own *due* demand-assigned slots.

CFDAMA-FA can be used in an integrated voice/data traffic environment. Voice calls can use the Explicit then Implicit Reservations, explained in Chapter 2. The CFDAMA schemes behave like reservation based multiple access schemes for voice. The effect of real-time traffic on the jitter tolerant data is examined in Chapter 5.

Chapter 4

Performance of CFDAMA-RA in Packet Satellite Communications

It has been seen that requesting through pre-assigned request slots or by piggybacking the request on traffic packets may be advantageous when the population size N is not *very* large [22]. Both have an excellent delay vs throughput performance when $N\tau < R$ or is comparable to R where τ and R are the time-slot size and round-trip delay, respectively. When $N\tau >> R$ their performance is degraded due to the long waiting time for a request slot in CFDAMA-FA or for an available traffic slot in CFDAMA-PB. In this case a random-access (RA) requesting scheme can provide a faster way for a terminal to send a reservation.

The FA or PB schemes require on the average $\frac{N\tau}{2}$ seconds at very low throughputs to obtain a slot, since the free-assigned slots are predominantly used for transmission, due to the presence of a large number of unreserved slots. They cannot transmit their requests immediately due to the large distance between the request slots. CFDAMA-RA, on the other hand requires on the average R seconds to obtain a slot at low throughputs[1], which is very much less than $\frac{N\tau}{2}$, since the terminal can send in its request as soon as the packet arrives. However at high throughputs the collisions degrade the performance of CFDAMA-RA. The packets will have to predominantly depend on the free-assigned slots for transmission and the performance comes close to that of the synchronous TDMA scheme.

In this chapter a mathematical model is presented for analyzing the performance of the CFDAMA-RA for a general terminal population size [40].

4.1 Modelling and Analysis

We consider a packet satellite system with N homogeneous terminals using a CFDAMA-RA protocol. We assume for convenience, that the scheduler is on-board the satellite[2]. Data traffic can be segmented into independent packets. Each packet can be transmitted in a time-slot of τ seconds. A time-slot also contains a small request slot, which can be randomly accessed by any terminal in the system. If two or more terminals transmit their reservation in the same request-slot, a collision occurs and their requests are discarded. Successful requests are stored in the scheduler's queue and served on a first-come-first-served basis. Whenever the scheduler's queue is empty the scheduler issues free-assigned traffic slot(s) to terminals in a round-robin manner.

A packet therefore can be transmitted via one of three possible types of traffic slots:

- free-assigned slot
- undue demand-assigned slot,1 i.e., a slot assigned to the terminal as a result of the demand of another packet[3]

[1]Assuming an on-board scheduler.

[2]The analysis is valid for an on-ground scheduler as well if we use a generalized $r = R$ or $2R$ depending on the scheduler location as in Chapter 3, as the time taken for a request to be honoured.

[3]While an undue demand assigned slot is only due to an antecedent packet in the case of

- a due demand-assigned slot.

For convenience, in the following analysis the time-unit is taken as the slot size τ. For a Poisson traffic source the system arrival rate per slot Λ should be less than unity to maintain system stability. The terminal arrival rate is $\lambda = \Lambda/N << 1$ for large N. The probability of more than one packet arriving in the terminal during the time-slot is therefore negligible. For this reason we assume that a terminal can produce at most one packet in a given time slot. The probability that this assumption is not valid may be calculated as seen in Chapter 3, and for a large population size, this turns out to be very small.

We consider a tagged packet arriving to a tagged terminal and first find the distribution of the time it takes to get a slot once it reaches the head of the terminal's queue. Then we treat the queue as a M/G/1 queue with the service time for a packet arriving to an empty queue different from the service time for a packet arriving to a non-empty queue [21] and derive the average waiting time ξ.

Let $W(z)$ and $\tilde{W}(z)$ represent the waiting time pgfs when the packet arrives at a non-empty queue and an empty queue, respectively.

Denote the time distance between two successive successful requests from packets arriving to the tagged terminal as a random variable H. Each request will bring one demand assigned slot[a] as a packet makes a request in the next immediate request slot. It follows that the interarrival time between two successive demand-assigned slots is also H (Figure 4.1). Thus, a packet arriving to a *non-empty* queue has to wait for H slots unless it gets a free-assigned slot before a demand assigned slot.

The probability that a particular slot is *free* assigned to the tagged terminal is p' [42].

---

[a]CFDAMA-FA or PB, it may be due to a packet that comes after the tagged packet in the CFDAMA-RA, since the request of the tagged packet may have experienced a collision.

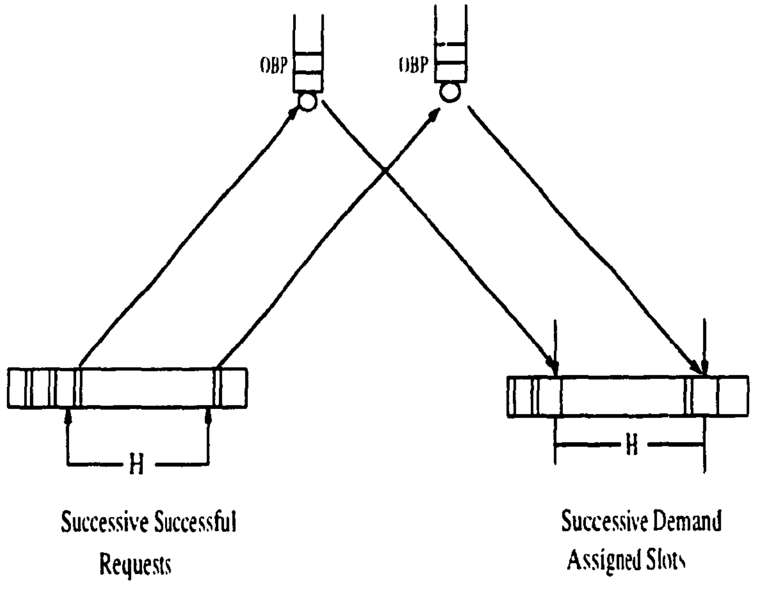

OBP
OBP
H
H
Successive Successful Requests
Successive Demand Assigned Slots

Figure 4.1: The time between successive succesful requests.

$$p' = \frac{1 - d}{N} \tag{4.1}$$

where d is the fraction of the demand assigned slots. This is due to the fact that the distance between two free-assigned slots is on the average $\frac{N}{1-d}$ slots. In our case, d is just the probability that only one terminal transmits in a given request slot, i.e.,

$$d = Ne^{-(N-1)\lambda}(1 - e^{-\lambda}) \tag{4.2}$$

Let k be the number of arrivals between successive successful requests. Therefore the time between two successive successful requests is equal to k/λ on the average, i.e., $E\{H/k\} = k/\lambda$. During this time interval, the arriving packet makes Bernoulli attempts with probability p' to obtain a free-assigned slot. We therefore have the expression for $W(z)$ as follows:

$$\begin{aligned} W(z) &= (\sum_{k=1}^{\infty}\sum_{i=1}^{\frac{k}{\lambda}} z^i p'(1-p')^{i-1} \\ &+ \sum_{k=1}^{\infty}[1-\sum_{i=1}^{\frac{k}{\lambda}} p'(1-p')^{i-1}]z^{\frac{k}{\lambda}}).A_k \end{aligned} \qquad (4.3)$$

The first term deals with the case in which the packet obtains a free-assigned slot whereas the second term is for the case when the packet gets a demand assigned slot. The above equation is independent of whether the previous packet is transmitted by a free-assigned slot or a demand assigned slot because the number of packets arriving before a successful request is geometrically distributed and therefore memoryless.

Here A_k refers to the probability that k arrivals occur between successive successful requests and is given by :

$$A_k = p(1-p)^{k-1} \qquad (4.4)$$

where p is the probability that given the tagged terminal made a request, and no other terminal sends a reservation in the same request slot, i.e.,

$$p = e^{-(N-1)\lambda} \qquad (4.5)$$

Substituting (4.4) in (4.3) and using the well known closed form solution for a geometric series[4] we get the following result for $W(z)$:

[4]We note that the arguments in the geometric series we are summing are each less than 1

$$\begin{aligned} W(z) &= \frac{p'z}{(1 - z(1-p'))} \\ &- \frac{p'pz}{(1 - z(1-p'))} \frac{(z(1-p'))^{1/\lambda}}{(1 - [(1-p)(z(1-p'))^{1/\lambda}])} \\ &+ \frac{p(z(1-p'))^{1/\lambda}}{(1 - [(1-p)(z(1-p'))^{1/\lambda}])} \end{aligned} \tag{4.6}$$

Differentiating with respect to z will yield the values of $\bar{w}$ and $\bar{w^2}$ as follows:

$$w = \frac{1}{p'} - \frac{p(1-p')^{1/\lambda}}{(1 - [(1-p)((1-p'))^{1/\lambda}])}[1 - \frac{p(1-p)}{p'}] \tag{4.7}$$

$$\begin{aligned} \bar{w^2} &= \frac{2(1-p')}{(p')^2}[1 - \frac{p(1-p')^{1/\lambda}}{(1 - [(1-p)((1-p'))^{1/\lambda}])}] \\ &- \frac{2p(1-p')^{1/\lambda}}{\lambda(1 - [(1-p)((1-p'))^{1/\lambda}])p'} \\ &- \frac{2p((1-p')^{1/\lambda})^2}{\lambda(1 - [(1-p)((1-p'))^{1/\lambda}])^2 p'} \end{aligned} \tag{4.8}$$

Now let us consider the packet arriving to an *empty* queue. It immediately transmits its reservation in the next request slot. If this reque.t is successfully received by the scheduler, the due-demand assigned slot will be available after $R + \bar{S}$ slots where R and S represent the round-trip delay and the average scheduler queue delay, respectively. In the meantime, the packet makes Bernoulli attempts for a free-assigned slot. In this case the probability of getting a free-assigned slot is doubled since on the average the packet arrives in the middle of the duration between its free-assigned slots [42] (Chapter 3). Thus the probability of getting a free-assigned slot for a packet arriving to a empty queue is:

$$p'' = \frac{2(1-d)}{N} \tag{4.9}$$

The waiting time pgf $\tilde{W}(z)$ for a packet arriving to an empty queue is derived as

$$\begin{aligned}
\tilde{W}(z) &= p\{\sum_{i=1}^{R+\tilde{S}} z^i p''(1-p'')^{i-1} \\
&+ \sum_{k=1}^{\infty}[1 - \sum_{i=1}^{R+S} p''(1-p'')^{i-1}]z^{R+S}\} \\
&+ [(1-p)\{\sum_{k=1}^{\infty} \sum_{i=1}^{\frac{k}{\lambda}+R+\tilde{S}} z^i p''(1-p'')^{i-1} \\
+\sum_{k=1}^{\infty}[1 &- \sum_{i=1}^{\frac{k}{\lambda}+R+S} p''(1-p'')^{i-1}]z^{\frac{k}{\lambda}+R+S}\}] \\
&\cdot \quad A_k
\end{aligned} \tag{4.10}$$

The first term is due to the fact that with probability "p" the packet immediately gets a request through. The second term with the factor "$(1-p)$" is the case when the request collides and the packet has to wait for either a free-assigned slot or a subsequent packet to succeed in requesting.

The expression (4.10) can be simplified as

$$\begin{aligned}
\tilde{W}(z) &= \frac{p''zp(1-(z(1-p''))^{R+S}}{(1-z(1-p''))} \\
&+ \quad p((1-p'')z)^{R+\tilde{S}} \\
&+ \quad \frac{(1-p)p''pz}{(1-z(1-p''))} \\
&\cdot \quad [\frac{1}{p} - \frac{(z(1-p''))^{(1/\lambda)+R+S}}{(1-[(1-p)(z(1-p''))^{1/\lambda}])}] \\
&+ \quad \frac{(1-p)p(z(1-p''))^{(1/\lambda)+R+S}}{(1-[(1-p)(z(1-p''))^{1/\lambda}])}
\end{aligned} \tag{4.11}$$

We may now differentiate $\tilde{W}(z)$ to obtain the values of $\tilde{w}$ and $\tilde{w}^2$.

$$\bar{\tilde{w}} = \frac{p(1-(1-p'')^{R+S}}{p''}$$

$$+ \quad \frac{(1-p)p}{p''}\{\frac{1}{p} - \frac{((1-p''))^{(1/\lambda)+R+\bar{S}}}{(1-[(1-p)(1-p'')^{1/\lambda}])}\} \tag{4.12}$$

$$\begin{aligned}
\bar{\tilde{w}}^2 \;=\;& \frac{2p}{p''}[\frac{(1-(1-p'')^{R+\bar{S}}(1-p'')}{p''} \\
-\;& (1-p'')^{R+\bar{S}}(R+\bar{S})] \\
+\;& 2p(1-p)[\frac{1}{p} - \frac{(1-p'')^{R+\bar{S}+\frac{1}{\lambda}}}{1-[(1-p)(1-p'')^{\frac{1}{\lambda}}]}]\frac{1-p''}{(p'')^2} \\
-\;& \frac{2p(1-p)^2(1-p'')^{R+\bar{S}+\frac{2}{\lambda}}}{(1-[(1-p)(1-p'')^{\frac{1}{\lambda}}])^2\lambda p''}
\end{aligned} \tag{4.13}$$

The only quantity that we need to determine is $\bar{S}$. The arrival rate at the Scheduler queue of the satellite is Λd since this is the fraction of the total arrivals that make requests successfully. The Scheduler queue of the satellite can be modelled as an M/G/1 queue with average arrival rate Λd per slot. Thus the average delay $\bar{S}$ is given by [21]

$$\bar{S} = 1 + \frac{\Lambda d}{2(1-\Lambda d)} \tag{4.14}$$

We then substitute the terms $\bar{\tilde{w}}$, $\bar{w}$, $\bar{\tilde{w}}^2$, $\bar{w}^2$ in the expression for the average delay for a M/G/1 queue in which the service time for an arrival to an empty queue is different from the service time for an arrival to a non-empty queue ([21], pp 102-104). The average waiting delay for a packet is:

$$\begin{aligned}
\bar{\xi} \;=\;& \frac{\lambda \bar{w}^2}{2(1-\lambda\bar{w})} \\
+\;& (\frac{\lambda[\bar{\tilde{w}}^2 - \bar{w}^2]}{2} + \bar{w})\frac{1}{[1-\lambda\bar{w}+\lambda\bar{\tilde{w}}]}
\end{aligned} \tag{4.15}$$

The average packet transmission delay is

$$\bar{X} = \bar{\xi} + R \tag{4.16}$$

4.2 Illustrative Results and Discussions

We consider a packet satellite system using CFDAMA-RA with $N_T > R$. We define the normalized population η as the ratio of N_T to R, i.e., $\eta = \frac{N_T}{R}$ as in Chapter 3. The round trip delay is assumed to be 270 ms. Figure 4.2 shows the comparison of the simulation and analytical results for η=1.85. In these results the request slot is considered to be 10 % of the time-slot. Consequently the throughput asymptotically approaches 90 % as shown in Figure 4.2.

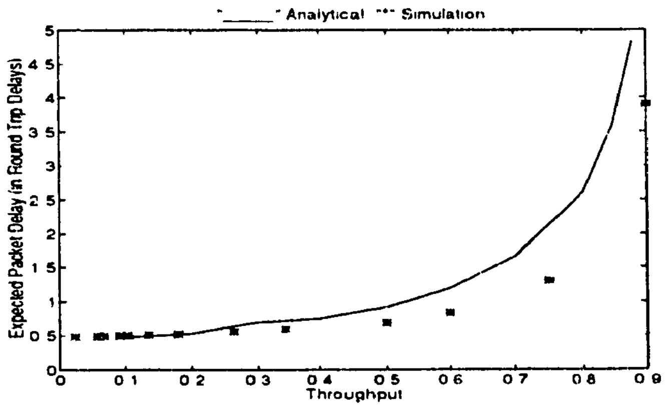


Figure 4.2: Comparison between analytical and simulation results: η=1.85

From Eq.(4.15) it can be easily seen that it is the second term of Eq.(4.15) that is the dominant contributor to the delay. This is because the $(1 - \lambda w)$ factor in this term is the one that approaches zero fastest. However, it can be seen from Figure 5.2 that the value of $\lambda \bar{w}$ is below one for system loads almost close to 1. Thus the CFDAMA-RA is stable for very high throughputs.

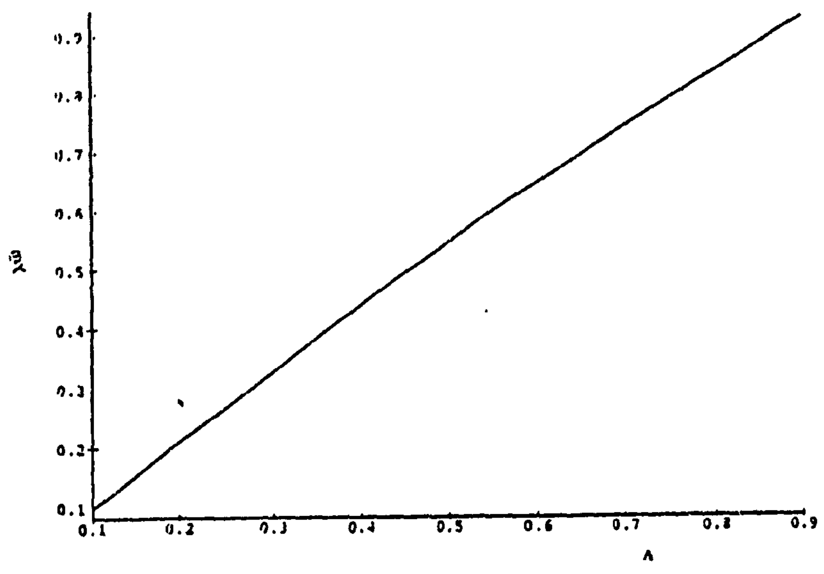


Figure 4.3: $\lambda\bar{w}$ vs System Load

The advantage of using CFDAMA-RA is when η is large. The random access strategy helps access a reservation slot almost immediately at low throughputs with a very low probability of collision. Thus at low throughputs, increase in the population size does not greatly affect the performance of the CFDAMA-RA as seen in Figure 4.4. However, at high throughputs the collision rate will be increased and the distance between free assigned slots becomes larger. Therefore the performance of the CFDAMA-RA deteriorates. As η increases the CFDAMA-RA provides a better performance compared to CFDAMA-FA except at very high throughputs. Figure 4.5 depicts the performance of CFDAMA-RA and CFDAMA-FA for $\eta = 37$. It clearly indicates that CFDAMA-RA outperforms CFDAMA-FA for throughputs up to 0.5 (cross over point in Figure 4.5). We can expect that this cross over point will increase with η.

Thus it is seen that CFDAMA-RA outperforms CFDAMA-FA when $N_T >> R$ except at high throughputs. In [42] it has been shown that, for $N_T < R$, CFDAMA-FA and CFDAMA-PB provide the same performance and both outperform CFDAMA-RA.

The above results indicate that for a wide range of terminal populations, it is desirable to have a hybrid request strategy that combines random-access with piggy-backed requests. In this hybrid CFDAMA-RA/PB scheme, random access request slots are provided. However, a terminal that has assigned traffic slot(s) should use piggy-backing for requests. In this way the random-access request slots are shared by terminals that do not have any assigned traffic slot. This greatly reduces the arrival rate of requests and consequently the collision rate in request slots. The waiting time to send a reservation is also reduced due to the availability of both RA and PB slots.

In CFDAMA-RA the packets that do not have successful requests will have to depend on successful requests of forthcoming packets or free-assigned slots to be transmitted. It is the free-assigned slots which make the system more stable when compared to DAMA schemes where the requesting is by Slotted-Aloha [38, 39].

In the above analysis it has been assumed that the data slots and request slots alternate. Thus, there is a request slot for every data slot. This not only gives a terminal an access to a request slot as soon as a message arrives, but also reduces the probability of collisions. The number of request slots may be reduced in order to reduce overhead and thus increase channel utilization. The request slots may be grouped at the beginning of a frame and a terminal with an arriving message may choose one of these request slots at random to place its request. The number of request slots may be chosen and the position of these slots organized as desired, depending upon the requirement and overhead constraints.

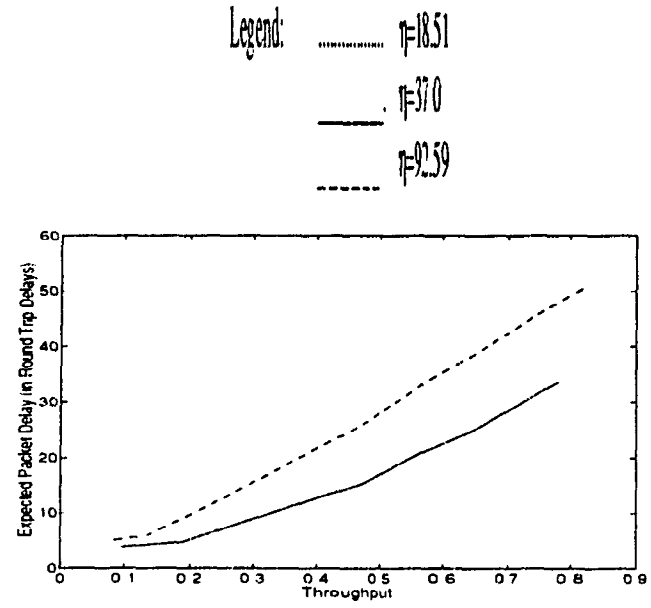


Figure 4.4: Comparison of CFDAMA-RA for various η

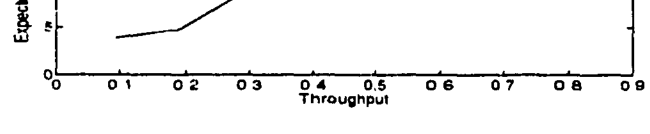


Figure 4.5: Delay Throughput Performance of CFDAMA-RA and CFDAMA-FA for $\eta=37$

Chapter 5

Performance in an Integrated Voice/Data System and Effects of MF-TDMA Framing

The CFDAMA protocols have so far been analyzed with jitter-tolerant type of traffic. Jitter tolerant messages require fast efficient transfer and the average delay is taken as the performance criterion. However, the multi-access scheme will also have to cater to real-time traffic whose characteristics and performance criteria are different from those of the jitter tolerant traffic. Real-Time traffic includes voice, video, etc. Real-Time traffic cannot tolerate random queueing delays and has to be allocated channel capacity almost as soon as the packets arrive [19]. This is because long gaps or restrictive delays may seriously impair the flow and hence the comprehensibility of a real-time message. Further, a real-time message is usually much longer than a jitter-tolerant message. In our discussions we take the example of voice-calls for depicting real-time messages for convenience. Voice is stream type, i.e., once a particular message begins it continues over several frames and requires transmission with a constant bit rate. The flow in stream type traffic is regular and continuous. Packets from a stream type message have to arrive in order for ensuring the comprehensibility of the message. There could be real-time messages that are of variable bit rate but these are not considered in this work.

Voice calls could be either off hook, i.e., ON or on hook, i.e., OFF. Once a call is initiated, i.e., is in the ON state it may be assumed to produce packets after fixed intervals of time [36, 17]. Some of the integrated schemes for multimedia divide a call into talkspurt and silence periods [11, 37]. Packets are generated only during talkspurt periods. The silent periods in a call may be used for transmitting data or other voice calls.

5.1 Multiple-Access in an Integrated Voice/Data Environment

There are various multiple-access schemes in the literature, which deal with an integrated voice/data environment in different ways. Most of them assume a poisson arrival process for voice-calls. The voice traffic may be either circuit switched or be packet switched [29]. Most of the protocols found in the literature use mixed circuit and packet switching in an integrated voice and data environment. One way of catering to the requirements of voice traffic is to prioritize voice over data in the multiple access protocol [37, 45]. Another way is to divide the channel into two subchannels, one for carrying voice using circuit switched techniques and another for carrying packet switched jitter tolerant data. In [12] an integrated access scheme in which the channel is divided into two subframes, one for voice and one for bursty data, is analyzed. The bursty data is transmitted using Slotted Aloha whereas a demand assignment strategy is used for transmission of stream traffic. In [6], Poon and

Suda assume a similar structure and strategy, with an integrated circuit and packet switching technique. A controlling ground radio network accessible to all terminals is present, so as to detect collision and therefore enhance the 'delay performance characteristics' of data traffic. Victor Li and T.Yan, in [44], have introduced an integrated access scheme suitable for both stationary and mobile systems (under some constraints) by dividing the channel into three partitions one for voice, one for data and one for reservations.

Instead of having separate subframes for data and voice we may have a single frame structure and use *the movable boundary strategy*. This involves dividing the network bitrate capacity into two parts. One part is usally treated in an asynchronous, packet switched manner with dedicated slots intended for carrying jitter-tolerant data. The other part is intended for real-time traffic. In this portion of the frame, voice calls have priority over data. If there are slots unused by voice they may be used by jitter-tolerant data packets. The voice calls may not however, occupy the portion of the frame dedicated for jitter-tolerant traffic. Figure 5.1 illustrates the concept of the movable boundary. Many integrated multiple-access schemes which incorporate the movable boundary have been analysed in the literature [45, 35, 14, 11]. In [16], Ahmadi and Stern have extended their work in [1] to integrate voice . The protocol is once again based on fixed and demand assignments. The channel is divided into two subchannels, one using a Fixed-TDMA type of access and the other, a pure demand access. The voice traffic is transmitted via the Fixed-TDMA subchannel in a circuit switched mode while the data is transmitted via the demand assigned channel in a packet switched mode. The slots in the Fixed-TDMA channel unused by voice are used by data.

In the following analysis of the CFDAMA protocols in an integrated voice/data environment it is assumed that a TDMA type frame structure is used. There are no separate channels used for data and voice. Further, no part of the channel is

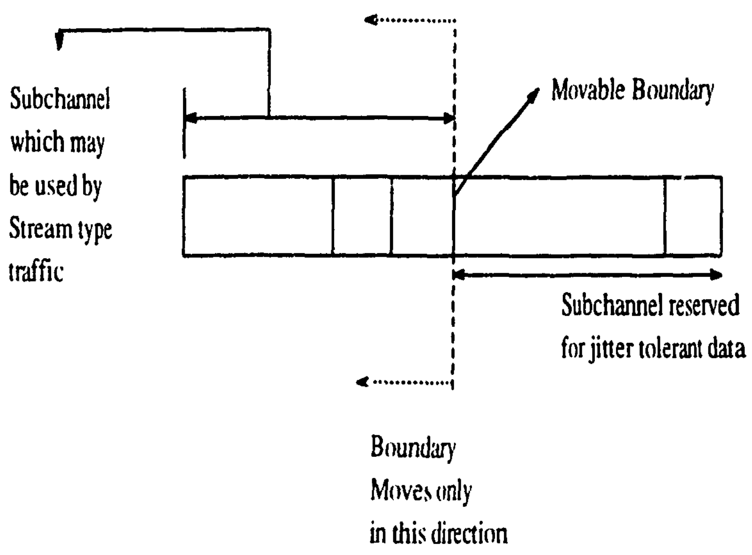

Subchannel which may be used by Stream type traffic
Movable Boundary
Subchannel reserved for jitter tolerant data
Boundary Moves only in this direction

Figure 5.1: The Movable Boundary Concept

dedicated for data traffic. The number of slots that can be occupied in a frame of 'M' slots, by voice calls, is 'M' itself[1]. Slots in a frame are first assigned to voice calls that are in progress. The remaining slots are then allocated to either new voice calls or jitter-tolerant data packets.

5.2 Performance of the CFDAMA protocols in the integrated voice/data environment

We assume that voice calls arrive in a Poisson manner. Voice calls are assumed to be of constant bit rate. Once a call is initiated, it generates packets after fixed intervals of time say β seconds. We assume that the frame size is chosen so as to ensure that a voice call in progress, gets one slot every frame. Thus a frame will occur once every β seconds.

[1]The movable boundary scheme, if incorporated will limit the voice calls to some value $C_v < M$. Then, the portion of the frame $M - C_v$ will be reserved for jitter tolerant data all the time.

When voice is integrated with data in CFDAMA protocols, we use the principle of explicit then implicit reservation (Chapter 2). A voice call sends the reservation in the pre-assigned request slot similar to a data packet. However, the allocated slot will be implicitly kept in the following frames until a release request is received by the scheduler.

When a voice call arrives it makes a request for a channel. If all the 'M' slots in a frame are occupied by on going calls, then the call is blocked. We use the *Lost Call Cleared* technique [19] in which a call finding all the slots occupied is cleared, i.e., assumed to be lost from the system. The blocking probability for voice is simply given by the Erlang B formula [29].

$$P_b = \frac{\frac{\rho_v^M}{M!}}{\sum_{k=0}^{M} \frac{\rho_v^k}{k!}} \tag{5.1}$$

Here we assume that the total traffic ρ is divided into two parts, one for voice and one for data. α is assumed to be the voice fraction of the total traffic. M is the total number of slots per frame as mentioned earlier.

We therefore have the following:

$$\rho_v = \rho\alpha = \frac{\lambda_v}{\mu} \tag{5.2}$$

$$\rho_d = \rho(1-\alpha) = N\lambda \tag{5.3}$$

Here ρ_v is the fraction of the voice traffic, ρ_d is the fraction of the data traffic, λ_v is the total arrival rate of voice calls per second and μ is the average call duration in seconds.

The expected number of slots that the voice calls occupy per frame is given by

$$E(v) = \rho_v M(1-P_b) \tag{5.4}$$

Thus the presence of voice results in occupation of a fraction of the time in the frame given by $\frac{E(v)}{M}$. Thus the fraction of slots used by data is given by $\frac{M-E(v)}{M}$. This may be taken into account in the delay factor by taking the packet duration to be $\frac{M}{M-E(v)}$ slots instead of one slot.

Analytical results match very well with the simulation results as shown in Figures 5.2, 5.3 and 5.4. Figures 5.2 and 5.3 indicate that delay throughput performance of data traffic is affected due to the voice-throughput as if there is an increase in overhead.

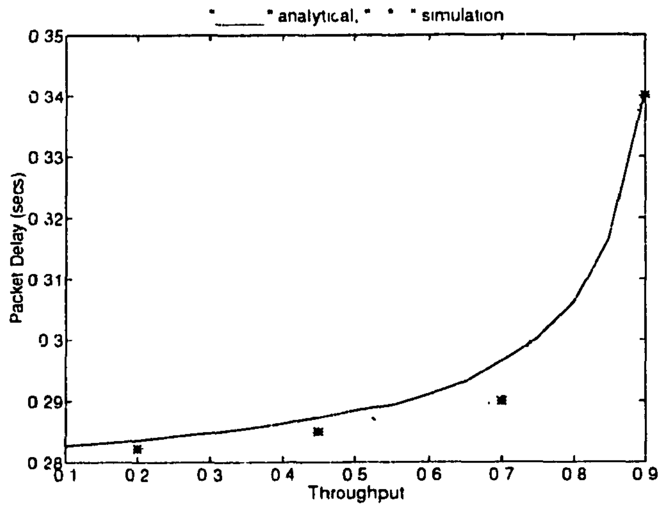


Figure 5.2: Average Packet Delay (seconds) vs Throughput with Voice Fraction = 10 %

The effect of integrating voice calls into the system on the jitter tolerant data is thus, the same as adding to the overhead. By having speech activity detectors we may expect the performance of jitter tolerant data to improve due to a decrease in the channel capacity occupied by voice calls. The voice may be considered to have talkspurt and silence periods and the silent periods may be used for transmitting jitter tolerant data.

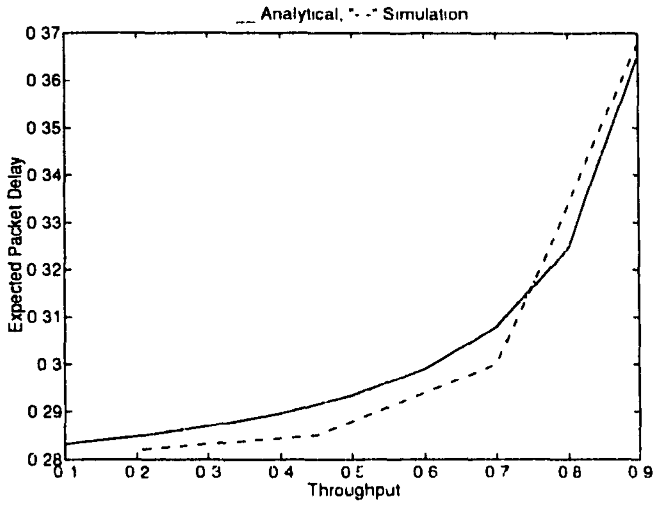


Figure 5.3: Average Packet Delay (seconds) vs Throughput with Voice Fraction = 95 %

5.3 Multiple-Frequency TDMA Framing

In traditional TDMA systems the terminals will be required to transmit at high bit rates. For a desired error performance as the transmission bit rate of a terminal increases the size of the terminal will increase. As satellite applications in business networks grow, there is a need for reducing the size of the terminal so that it becomes cost effective. To reduce the size of the terminals while maintaining the advantages of single carrier TDMA the MF-TDMA framing was introduced.

Multiple-Frequency TDMA (MF-TDMA) is a technique to share satellite resources, in which several carrier frequencies are used in a transponder to form a network of low-cost terminals that have excellent performance in applications varying from low throughput transmission to very high aggregate data rates (upto 130 Mbps) [20]. A brief discussion of the MF-TDMA transponder division technique was presented in Chapter 1. MF-TDMA is a hybrid FDMA-TDMA system whereby each

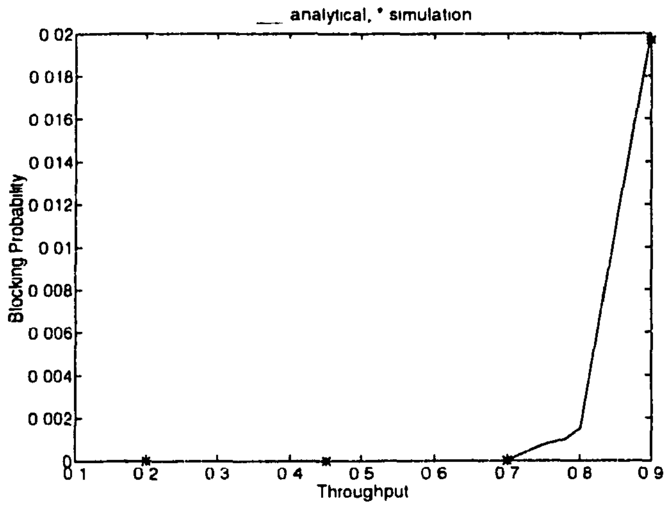


Figure 5.4: Blocking Probability vs Throughput for voice

terminal can transmit bursts of data at a multiplicity of time and frequency slots in a time-frequency map (Figure 5.5). In the MF-TDMA technique, the terminals are required to transmit at much lower bit rates than in the traditional TDMA systems and thus may be made smaller as desired in user-applications [15]. The network however will operate at 'L' frequencies so that L terminals are transmitting simultaneously using single carriers. The network's aggregate data rate will be therefore 'L' times the transmission bit rate of a single terminal. The system cost is therefore considerably less than that of a conventional high-rate single frequency TDMA system. [20] discusses other advantages and disadvantages of a MF-TDMA system as compared to a TDMA system.

From the networking point of view the disadvantage associated with a MF-TDMA system is the restriction that a particular terminal can use only a single carrier in a particular time lot[2]. Thus in a MF-TDMA time-frequency map of 'M x L' slots a

[2]This is to ensure that the terminal's high power amplifier does not encounter intermodulation effects and can function at a high power efficiency [41].

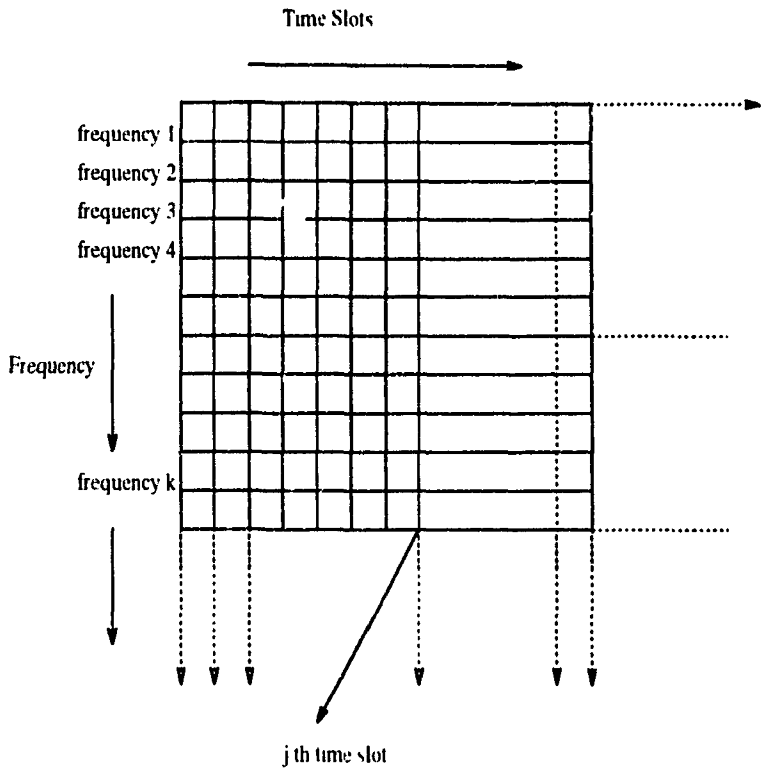
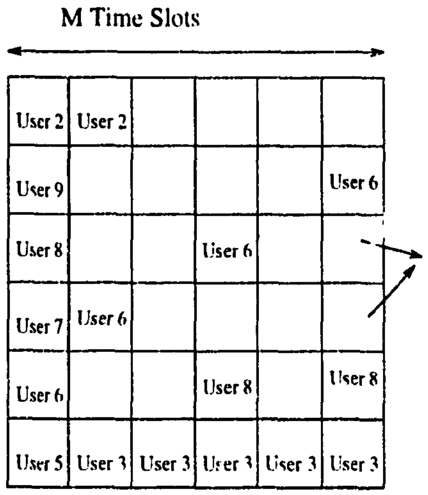


Figure 5.5: Multiple-Frequency TDMA Frame Format

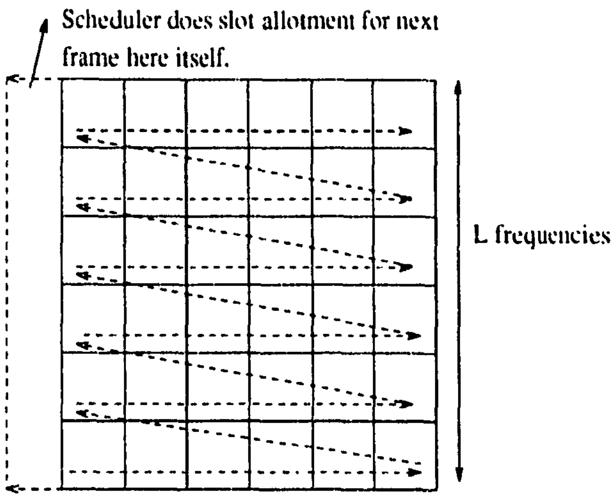
particular terminal could transmit a maximum of M packets. These packets would include jitter-tolerant data packets as well as real-time voice packets. In the absence of a proper re-assignment algorithm, this could lead to *blockage* of packets even when unused slots are available. For messages constrained to fit into single packets this problem will not arise usually, but for voice calls and messages, which may be of several packets in length, it may lead to blockage of calls and subsequent increase in message delay. Figure 5.6 (a) shows how a certain terminal cannot transmit any more packets even when unused slots are available due to blockage of all frequencies in the first time slot. In this example the terminal 'User 3' cannot transmit more packets even though there are vacant slots, since it cannot transmit in more than one frequency per time slot. By moving one of the packets of some other terminal from the first time slot to another time slot 'User 3' can transmit another packet (say for example moving the terminal User 8's packet in time slot 1 to time slot 2). Thus we need a reassignment algorithm, which is run as a preamble before the beginning of each frame in order to ensure a high channel utility. This algorithm should ensure that blocking of the above nature does not occur. In the literature a scheme has been suggested to remove blockage for two way calls [30].

Blockage may be minimized by assigning slots as shown in Figures 5.6 (b) and 5.6 (c). We start allotting slots starting with one terminal and one carrier and proceed in a manner shown in Figure 5.6 (b). Let us assume that there are 'L' frequencies and 'M' time slots. If a terminal starts getting slots from time-frequency slot (i, j), i.e., the ith time slot in the jth frequency, then the next slot it gets would be $(i + 1, j)$ and so on. After (M, j) if it has more packets left it is allotted slot $(1, j + 1)$ and so on until $(i - 1, j + 1)$. Thus a terminal gets a maximum of 'M' slots in a frame. If $j = M$ and all the slots in the frame are occupied, the terminal continues to get slots in the next frame. It is somewhat like a raster scan. In this fashion each terminal can transmit the maximum possible 'M' packets in a frame, if there are unused slots. The 'M' packets will include both jitter-tolerant data and voice calls. When the number of terminals is small there could be a blockage, even when there are unused slots. This occurs if a particular terminal has more than 'M' packets in its queue at the beginning of a frame, since the maximum number of packets a terminal can transmit in a frame is M.

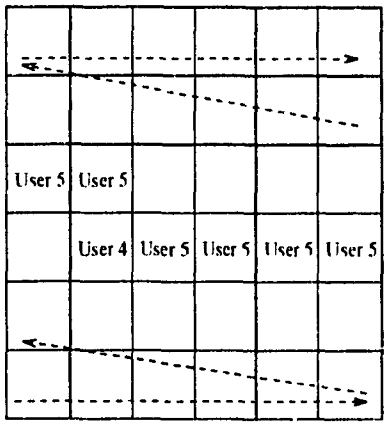
If the distance between requesting instants is an arbitrary 'K' frames then the probability of a blockage of this nature occuring is given by the probability that in 'K' frames a terminal gets more than 'M' packets (assuming the presence of only jitter tolerant data) which is



a) Blocking Due to lack of Assignment Strategy. User 3 cannot transmit another packet even though there are unused slots present.



b) Allocation Strategy suggested for MF-TDMA. Packets are Blocked only if number of packets in the queue > M



c) M=6. User 5 has 6 packets. Allotment for User 4 ends in second time slot. User 5 is allotted slots as depicted. There is no differentiation between free and demand assigned slots. All slots for User 5 are allotted one by one in this fashion before the frame begins.

Figure 5.6: Re-Assignment Strategy for MF-TDMA

$$P_{su} = \sum_{i=M}^{\infty} \frac{e^{-\lambda KM}(\lambda KM)^i}{i!}$$

A simulation was done for comparing CFDAMA using a MF-TDMA frame format with eight carriers, with CFDAMA using the traditional TDMA frame format. A packet size of 0.0001875 secs was used. The number of terminals in the system was assumed to be eight. The frame size was 24 ms and the number of slots per frame was 128. 50% of the traffic was voice and the other 50% data. Each terminal had one request slot every frame. The traffic per terminal in the CFDAMA with MF-TDMA was 8 times the traffic in the CFDAMA with the traditional TDMA, since the number of slots per frame in the former was eight times that of the latter. It is seen that since the probability of having 128 packets or more, per frame is not negligible when the MF-TDMA frame structure is used, there is a slight increase in the average delay of less than 3 ms (Figure 5.7). After throughputs of around 0.7 this probability increases and the system is seen to become unstable. However in practice, usually the number of terminals is larger and they have lower loading and this situation does not arise.

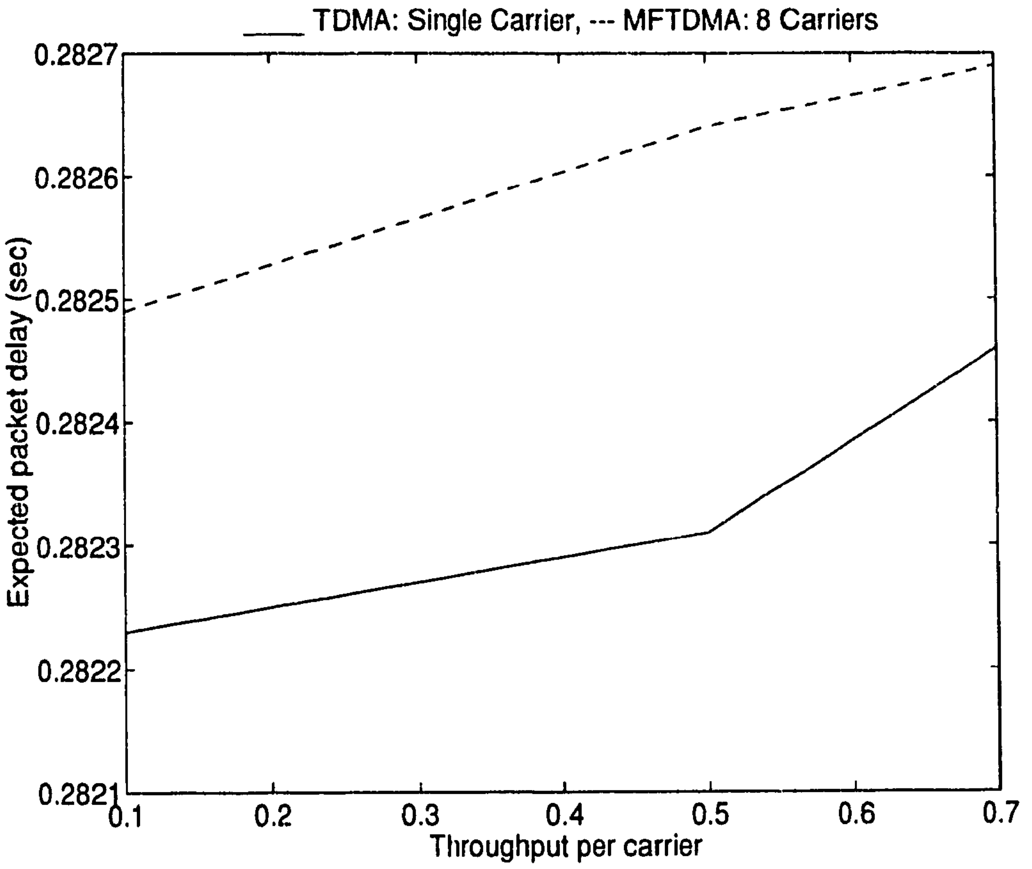


Figure 5.7: Simulation results comparing CFDAMA using a MF-TDMA frame structure with 8 carriers with CFDAMA using a single carrier TDMA type frame structure for equal throughputs per carrier

Chapter 6

Conclusions

The Combined Free/Demand Assignment Multiple-Access Protocols introduced in [42] were examined in detail in this thesis. These protocols were studied beginning with their evolution and the research that was already done on the CFDAMA protocols was reviewed.

In this thesis

• The performance of the CFDAMA-FA was analyzed for a general population size with jitter-tolerant traffic. Effects of reducing population size were considered. Simplifying approximations were suggested for the case when $N\tau << R$[1].

• A performance analysis was done for CFDAMA-RA, for a general population size. The effects of changing the population size were investigated. The performance of CFDAMA-RA was compared with CFDAMA-FA for different population sizes.

• The performance of the CFDAMA protocols in an integrated voice data environment was analyzed. The effects of the integration of real-time voice, with constant bit rates, on the performance of jitter-tolerant data were examined.

[1]N is the terminal population size, τ is the time slot size in seconds and R is the round trip delay in seconds.

• The effects of introducing a MF-TDMA frame format on the CFDAMA protocols were studied and a channel re-assignment scheme for improving channel utilization suggested.

The analytical results in each case have been compared with those obtained from simulation in order to support the validity of the various simplifying approximations that have been used in the analyses.

It was found that the CFDAMA offers great efficiency at all ranges of the population size N. The performance of the CFDAMA protocols was seen to improve as the population size decreased. A normalized terminal population η was introduced as the ratio $\frac{N\tau}{R}$. The results obtained for a particular value of η are valid for different combinations of N and τ. It is seen that when '$\eta << 1$' the free-assigned slots (which were assumed to be allotted in a round robin fashion to the terminals) are used by most of the packets. This allows system designers a flexibility in selecting the request strategy and location of the scheduler [2] to fit other system requirements without sacrificing the performance of CFDAMA protocols. Pre-assigned request slots or piggybacking the request on a traffic packet were shown to be advantageous when the terminal population size η is small, i.e., $\eta < 1$ or close to '1'. A random access mode of requesting was seen to be better when $\eta >> 1$.

In an integrated voice/data environment with constant bit rate voice, it was shown that the CFDAMA protocols behave as a pure demand assignment scheme for voice calls. The blocking probability for voice calls was shown to be given by the 'Erlang B' formula. The occupancy of a portion of the frame by voice calls was seen to act as an additional overhead as far as the jitter tolerant data was concerned.

Finally the Multiple-Frequency TDMA type of frame structure (applied to the CFDAMA) has been studied, its advantages and disadvantages discussed. A strategy

[2]since these factors influence only the demand assignment

has been suggested to avoid blocking when this type of framing is used.

Suggestions for Further Research

In this work it has been assumed that the voice is transmitted at a constant bit rate. However in multimedia applications there is also variable bit rate voice and video. The presence of speech activity detectors will trace talkspurt and silent periods in a voice call. Transmission bit rates for video signals will vary as the image changes. The arrival process of such voice calls or video signals may no longer be Poisson. It may be expected to follow an *Interrupted* Poisson Process or a Poisson Process with a varying arrival rate. Performance investigation of the CFDAMA protocols in such a multimedia environment would be of interest.

It has been seen that CFDAMA-RA outperforms CFDAMA-FA when $N_T >> R$ except at high throughputs. In [42] it has been shown that, for $N_T < R$, CFDAMA-FA and CFDAMA-PB provide the same performance and both outperform CFDAMA-RA. It may prove advantageous to have a hybrid request strategy, which combines random-access with piggy-backed requests to cater to a wide range of terminal population sizes. While the random access requesting would eliminate large delays at low throughputs when the population size is large, the piggybacked requesting would ensure good delay performances at high throughputs. A performance analysis of CF-DAMA using such a hybrid requesting strategy would require further investigation.

Appendix A

Simulation Model Descriptions

A.1 Introduction to OPNET models

The simulations in this research were done using OPNET (Optimized Network Engineering Tools), a software package for simulating computer and communication networks. OPNET provides a user friendly graphical interface. The simulations are based on building Finite State Machines, which are known as Process Models. These Process Models actually host the algorithm or strategy that is to be carried out in order to emulate the process in a queue or a processor. The algorithms are written in "C" using some built in functions provided. The process models are then placed in what are known as the node models. The node models contain functional elements which are generally used in communication networks, such as packet generators, queues, and processors. Various parameters can be specified either at the process model levels or can be *promoted* so that they can be specified later at higher levels (such as the node level). The nodes are then placed in the communication network. OPNET also has a tool, known as the parameter tool, which may be used to format a packet as desired. The following sections describe two process models that were used in the research. 'C' codes that were used in the process models are included and certain variables used in these programs are referred to in the descriptions. A simulation model created for CFDAMA-PB is described in [28]. These process models are responsible for carrying out the entire strategy of the simulations. OPNET gives the output in the form of convenient plots. It is assumed that the reader is familiar with the OPNET simulation package to some extent since it is beyond the scope of this thesis to give a complete description of this package.

A.2 Simulation Model for CFDAMA-RA

In this section the process models developed for simulating the CFDAMA-RA are described. We assume alternating data and request slots as in the analytical model described in Chapter 4. In each request slot, any terminal that has a packet arriving in the previous data slot sends in its request using the Slotted Aloha technique. If more than one terminal requests in a particular request slot then we assume that the requests have collided and are lost. We assume that the scheduler is placed on board the satellite in all the simulation models and refer to it as the On Board Processor (OBP).

The terminals are grouped for convenience. In the network we have a few nodes, each representing a group of terminals. We also assume that the number of terminals in all groups is the same.

A.2.1 Strategy followed for collision detection

In this scheme when a request slot occurs, the OBP interrupts all the terminal groups one by one starting with group 0 and ending with group 9. In the simulation we have 10 terminal groups and the number of terminals in every group can be varied, i.e., this value has to be input at the time of running the simulation.

The strategy followed in collision detection is to have a counter, which is labelled as the *no-in-channel*, which is incremented whenever any terminal transmits its request during a particular request slot. If the value of this counter is one then it means that there is only one terminal that has transmitted in the current request slot and hence there is no collision during this slot. However, in case this *no-in-channel* is more than one then there is a collision. In case of a collision the requests are assumed to be lost, i.e., the OBP does not honour the requests. All packets making requests are transferred to a virtual node called the hub. After a particular request slot the satellite interrupts the hub so that the hub checks the value of *no-in-channel*. If this is one, i.e., there is no collision, then the requests are transferred from the hub to the satellite. Otherwise, the requests are destroyed. In the case of successful transmission, the successful terminal has its requests delivered to the satellite and these requests line up in the satellite's scheduler queue.

A.2.2 Description of the Process Models for CFDAMA-RA

• The OBP Satellite

The process model for modelling the OBP is called **res-aloha-obp**. The OBP has various states (Figure A.1), which can be described as follows.

The state **wait** is a state in which the OBP will remain in the absence of any task. In the wait state the OBP will receive various interrupts such as an arrival to its scheduler queue or the time determined interrupt to indicate to it to allot a data slot or a request slot, etc. On receiving a particular interrupt the OBP will go to a different state and carry out the task it is meant to perform in that state, and then return to the wait state.

The **init** state of the OBP is just to initialize various parameters used in the model. This state is reached when the *begin simulation* interrupt of the OPNET simulation program occurs.

The **req-q-ing** state picks up the incoming requests from various terminals and places them in the scheduler queue of the satellite.

The **dataslotas** state is meant for assigning data slots to the terminals. It first checks the scheduler queue and in case there are some requests queueing up then it allots data slots to the requesting terminals and in case the scheduler queue is empty it allots slots in a round robin fashion. A counter will indicate as to which was the terminal the last free-assigned slot (a slot that is assigned due to this round robin assignment is known as a free-assigned slot) went to and the satellite will allot the current free-assigned slot to the next terminal.

In the **reslotas** state the satellite indicates to each and every terminal that a request slot is allotted.

• The Terminals

The Terminal Process Model is shown in Figure (A.2). The process model is called as **res-aloha-mac**

The **sub-q-ing** state inserts the arriving packets in a sub-queue based on the value of an index, which gives the information as to which terminal generated that particular packet.

The state **send** is reached when the satellite indicates to the terminal that a data slot has been allocated to it, and it can transmit if it has packets waiting in its queue. The terminal checks its queue and in the case where it has a packet it calculates the expected delay etc. by taking into account the round trip delay. All calculated values are written to a scalar output file.

The process model goes to the state **reservn** whenever a request slot is assigned.

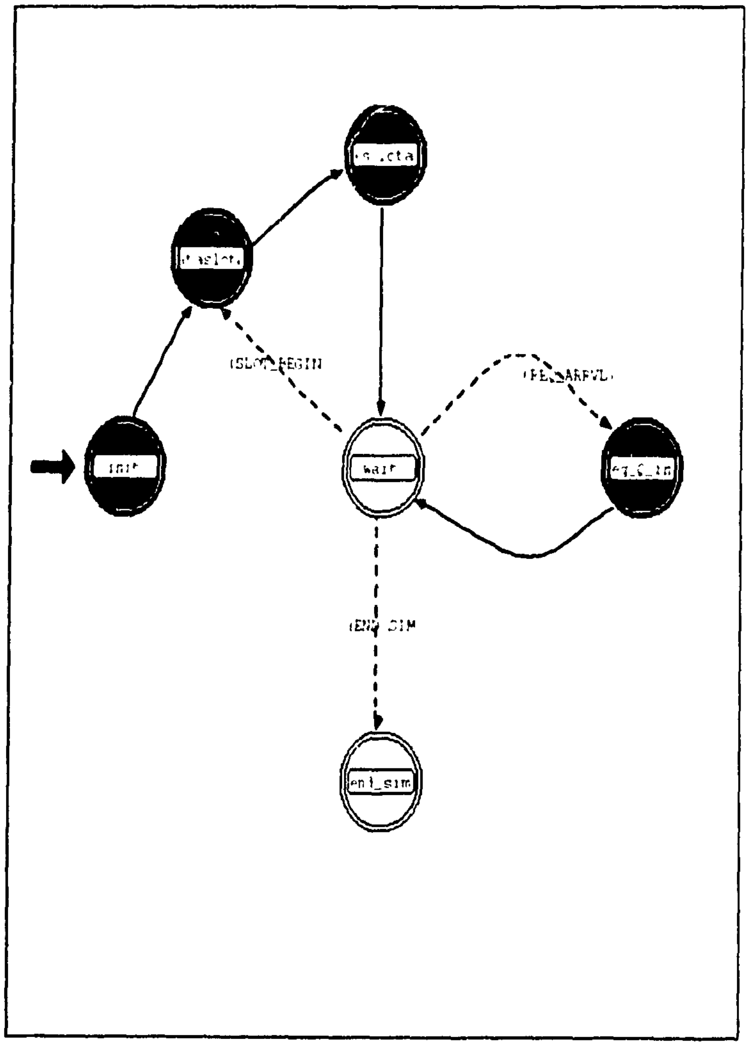


Figure A.1: The Satellite Process Model

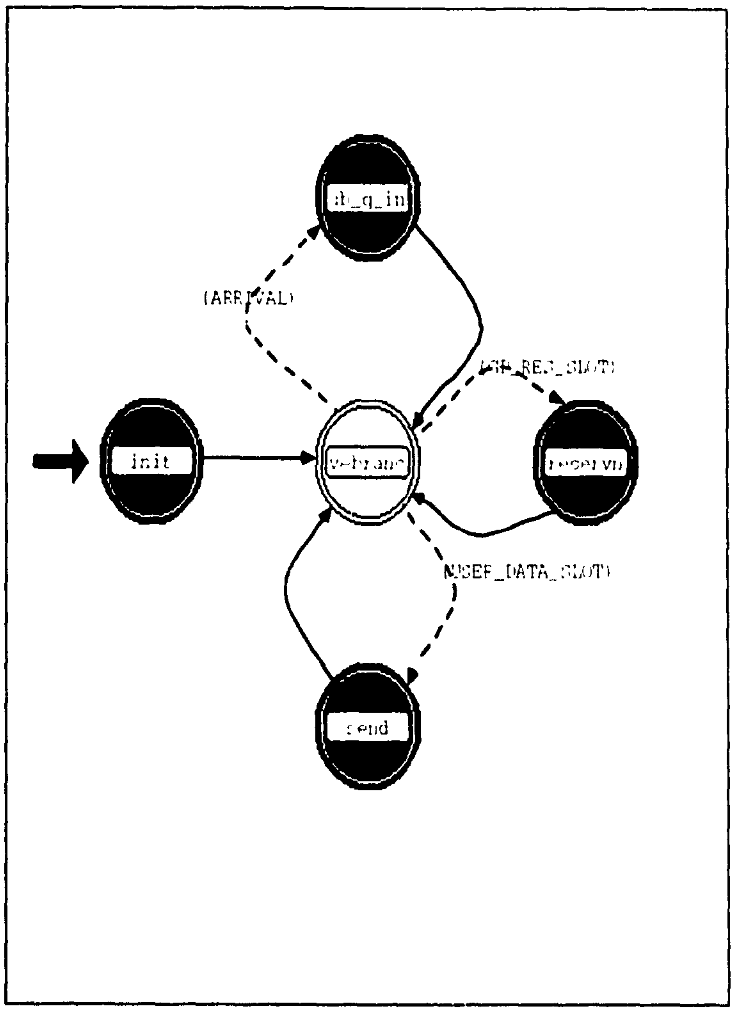


Figure A.2: The Terminal Process Model for CFDAMA-RA

If the subqueue of a particular terminal is not empty, the terminal transmits its requests. The packets for which a terminal makes its requests are transferred from the terminal's first queue (the one we have been talking about so far and to which the generated packets arrive) to a *second queue*, which will therefore contain only packets for which requests have been made. Whenever a data slot is allotted the packets are first removed from this second queue. The requests are transferred (copies of packets making requests) to the hub.

The state **evbranch** is like the **wait** state of the OBP and the process stays in this state when there are no tasks to perform. The **init** state is used to load values for various parameters.

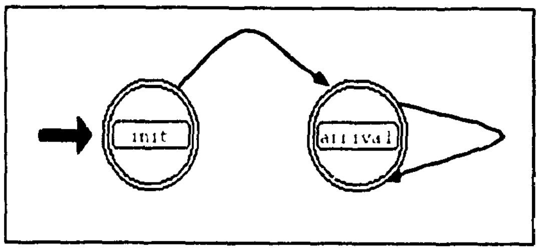
• Packet Source

src-al-mf1 is the process model for the Packet Source in the CFDAMA-RA simulation model. This process model sets various attributes for a packet and obtains (by giving prompts at the time of running the simulation) various unassigned values such as the inter-arrival time and packet size, which are to be used in the simulation. The states in the process model for the Packet Source are shown in Figure A.3 (a).

• The Hub

The Hub is a virtual process model, which has been created for convenience in the simulation model only. This has been created in order to detect collisions. The Hub process model Figure A.3 (b) has three states and is named **hub**.

The **init** state and **wait** state do not have any tasks to perform. As soon as the process gets a **begin simulation** interrupt it goes and waits in the **wait** state. The **q-pkts** state gets requests from the terminals (these requests are in the form of copies of packets that want data slots) and queues them in a global queue. The state **validity** checks the value of *no-in-channel* to see if more than one terminal transmitted a



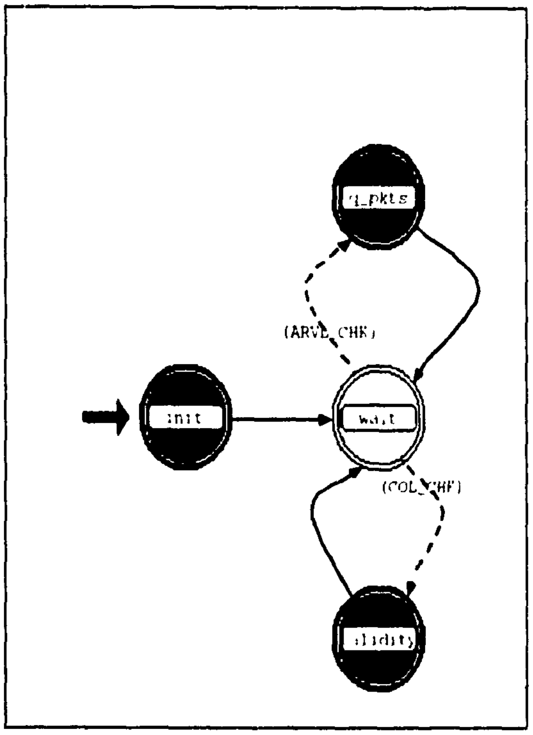


Figure A.3: a) The Source Process Model b) The Hub model for CFDAMA-RA

request in a particular request slot and thus detects collisions. A satellite interrupt, given after a request slot has been issued, takes the hub from the wait state to the state **validity**. Here it checks the collision status. If there has been a collision it destroys the requests. Otherwise, it delivers them to the satellite and they are then queued in the scheduler.

A.3 Simulation model descriptions for CFDAMA-FA with MF-TDMA framing

In this section a description of the process model used for CFDAMA using a MF-TDMA frame format is given. The requesting strategy is based on Fixed Assignment. Both Jitter-Tolerant and Real-Time traffic packets can be present.

In the model we have to have a source that produces both real-time messages and jitter-tolerant messages. Real-Time messages are assumed to be voice calls that arrive in a Poisson manner, but after the arrival of the first packet are assumed to generate one packet every frame for the duration of the call. The duration of the call may be assumed to be exponentially distributed with a mean μ (Refer to Chapter 5). Jitter Tolerant packets also arrive according to a Poisson Process. On going calls get the first priority for slot allotment. Any call that is allotted a slot will be allotted a slot every frame until the call is over (Implicit Reservations: Refer to Chapter 2). The strategy for handling Real Time Traffic is that, instead of generating a packet every frame after a call has been initiated, the first packet is the only one produced and it is held until the call is over. The satellite checks as to how many calls are in progress, at the beginning of a frame, and issues the remaining slots in the frame to jitter tolerant data. The average call duration would have to be limited to 0.03 minutes instead of the widely used 3 minutes to reduce the time of simulation. We group the terminals as in the previous model for convenience.

A.3.1 The Process Model Descriptions

• The OBP Satellite

The satellite has the same finite state model as shown in Figure A.1. However, the requesting strategy and slot allotment strategies are modified. The **init** state is used for initializing various parameters and arrays as in the previous model. The **req-q-ing** state is also similiar to the state with the same name in the CFDAMA-RA OBP model and it just gets the incoming packets and queues them in the scheduler queue. The **end-sim** state, just writes the values of various computed values to scalars, which may then be plotted. The **resslotas** state issues the request slots every frame ('Q' in number - Refer to Chapter 2) according to the fixed assignment strategy. It has a few state variables, which are incremented every time a slot is allotted, and which ensure that request slots are assigned in a round robin fashion. The model takes care to see that after the last terminal the next slot has to be issued to the first terminal. This issuing of request slots is similar to the allocation of free-assigned slots in the **dataslotas** as described in the previous example of CFDAMA-RA. The **dataslotas** state is similar to the same state in the previous example but is modified to accomodate voice traffic and also to take into account that a particular terminal can have at most 'M' (no of time slots– Refer to Chapter 5) frequency time-slots in a particular frame. The process model for the satellite is called **MFTDMA-SAT**

At the beginning of each frame, in the '**dataslotas**' state, an interrupt is issued to the hub so that the hub can take care of the status of various voice calls that are going on in the system. Then a global variable **v-calls**, which indicates the number of voice calls in progress, in the system, at the current time, is checked. Then the slots in the frame that are not occupied by voice calls are allotted to data or new voice calls. The requests are taken from the scheduler queue. **Voice-Tab** is an array, which keeps count of the number of voice calls, a given terminal has in progress. **Indice** is an array, which keeps count of the number of slots (other than for voice calls in progress) allotted to a terminal in the current frame. A terminal can have a maximum of 'M' (number of time slots in a frame per carrier) slots in a frame. If a request comes to the head of the scheduler queue and its terminal already has M slots allotted to it, then the request cannot be accomodated in that frame. For further assignment in that frame, the satellite then starts serving requests from, not the head of the queue, but the next position. Then if this position is also occupied by a request similar to the one described, the satellite starts serving requests from position 3 leaving the first two requests as they are. This is taken care of by the index "of" in the simulation model. The scheduler starts issuing slots to the packet at the head of the queue again beginning with the next frame. The allocation of slots is once again similar to the previous case (CFDAMA-RA) except that free-assigned slots cannot be issued to a terminal if it has already been allotted M slots in a frame. The terminal is then assumed to lose that free-assigned slot. For assigning a demand assigned slot the satellite checks whether the packet demanding is of type real-time or jitter tolerant and issues interrupts with different codes for both. These codes are properly interpreted at the terminal models and proper action is taken.

• The Terminal

The process model for the terminal in this simulation model is named **MF-TDMA-MAC**. The terminal is once again similar to the one used in the CFDAMA-RA, except for changes to accomodate voice traffic. The state **init** is again for initialization of certain parameters. The state **sub-q-ing** gets the packets from the source, checks whether the packet is of jitter tolerant type or of real-time type (from a field

status that each packet has). It then puts the packet in the proper queue. Jitter tolerant data and Real-time traffic have different queues. The state **resns** is similiar to the state **reservn** in CFDAMA-RA. However, only the terminal to which a particular request slot is issued can send its request in that slot. Both real-time traffic and jitter tolerant data send in their requests. The packets that make requests are shifted to a new queue as in CFDAMA-RA to differentiate between them and packets with no requests. The two queues together form a First in First out compound queue. The state **vc-slot** is reached when the satellite interrupt carries a code indicating that the slot is meant for a new voice call. The packet is then transferred to the hub where it is retained until the call-duration is over. If the number of voice calls exceeds the maximum permissible, then the voice call is said to be blocked, and is lost. The state **send** is exactly the same as the state "**send**" in the CFDAMA-RA terminal and is used for transmitting a jitter tolerant data packet. Figure A.4 shows the process model for the MF-TDMA terminal.

- **The Source**

The source process model is called **MF-TDMA-SRC**. The source for the CF-DAMA for integrated voice and data has to generate two types of traffic, real-time traffic and jitter tolerant traffic. Various parameters such as the total traffic load and the fraction of voice traffic (Refer to Chapter 5, ρ and α) can be input at the time of running the simulation. The state **init** gets all the simulation attributes, i.e., the various parameters. It then computes various values like the inter-arrival time for voice, inter-arrival time for data, etc. It also initializes various other variables before going to the state **idle**. The process model just stays in the state **idle** until some packet arrives. Depending on whether the packet is of real-time type or jitter-tolerant type the finite state machine goes to either the state **vc-arrvl** or **data-arrvl**. Then the packet is formatted and various fields are set before the packet is sent out to the

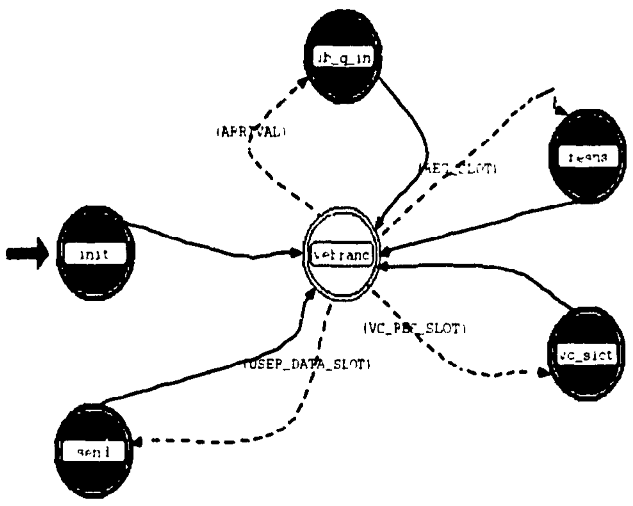


Figure A.4: The Terminal Process Model for CFDAMA-FA with MF-TDMA framing

terminal's queue. Figure A.5 shows the process model for the source.

• The Voice Hub

This is a process model (Figure A.6), which is not present in the actual system but created in the simulation model for convenience. We call this model as **MF-TDMA-HUB-REORD**. The Voice calls are modelled by just holding the first packet for the duration of the call rather than generating a packet every frame. The packet is held in the hub. After a newly arriving voice call is allotted a slot (from then on it will get one slot every frame as a result of implicit reservations), it is moved to the hub. The number of subqueues in the hub represent the number of slots in a frame. Therefore each non-empty subqueue represents a voice call. The **init** state just initializes the number of calls in the system to zero. The state **vc-bulk** is reached whenever a voice call is allotted a slot. The packet is transferred from the terminal to the hub and it occupies the first non-empty subqueue available. There is an algorithm to check if the number of voice calls in the system is still less than the maxmimum permissible and whether the particular terminal has less than M calls of its own. If not the voice call is destroyed. Otherwise, it is placed in a non-empty queue. The packet has two fields "ref-time" and "call-dur". The value in call-dur represents the duration of a voice call and is exponentially distributed with the average value μ. The value in ref-time refers to the time when the call was initiated. These fields are set in the packet.

The state **frm-bgn** is reached due to an interrupt from the satellite at the beginning of every frame. This state, then checks the status of each voice call by accessing the fields "ref-time" and "call-dur" to see which of the voice calls have ended. Then those packets are destroyed and the channel capacity that they were occupying freed for new voice calls or data packets.

The 'C' program listi..gs of the process models described in this appendix are included.

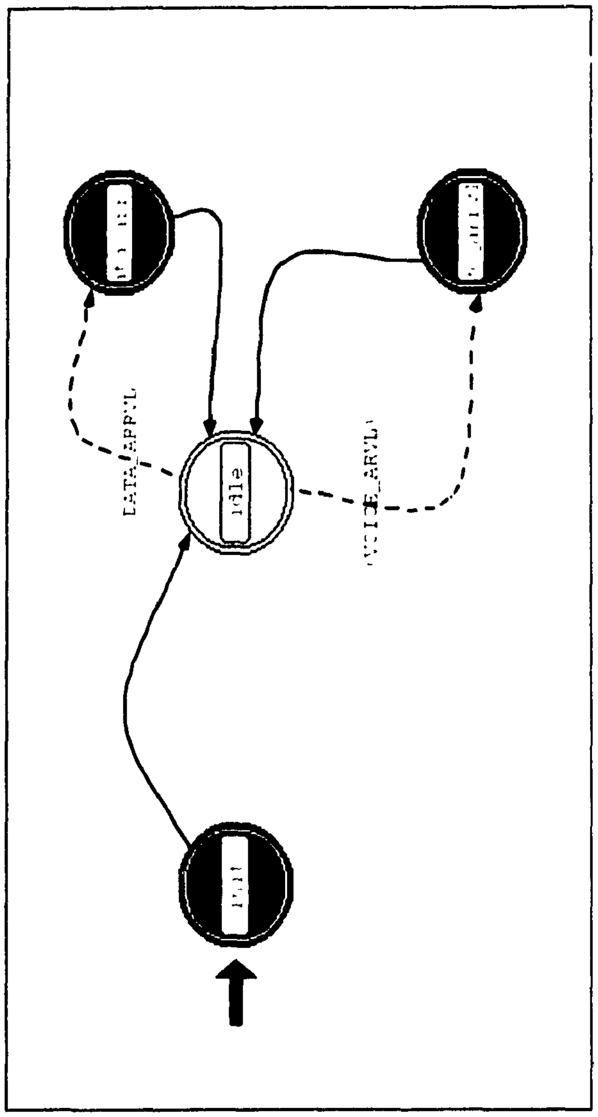


Figure A.5: The Source Process Model for CFDAMA-FA with MF-TDMA framing

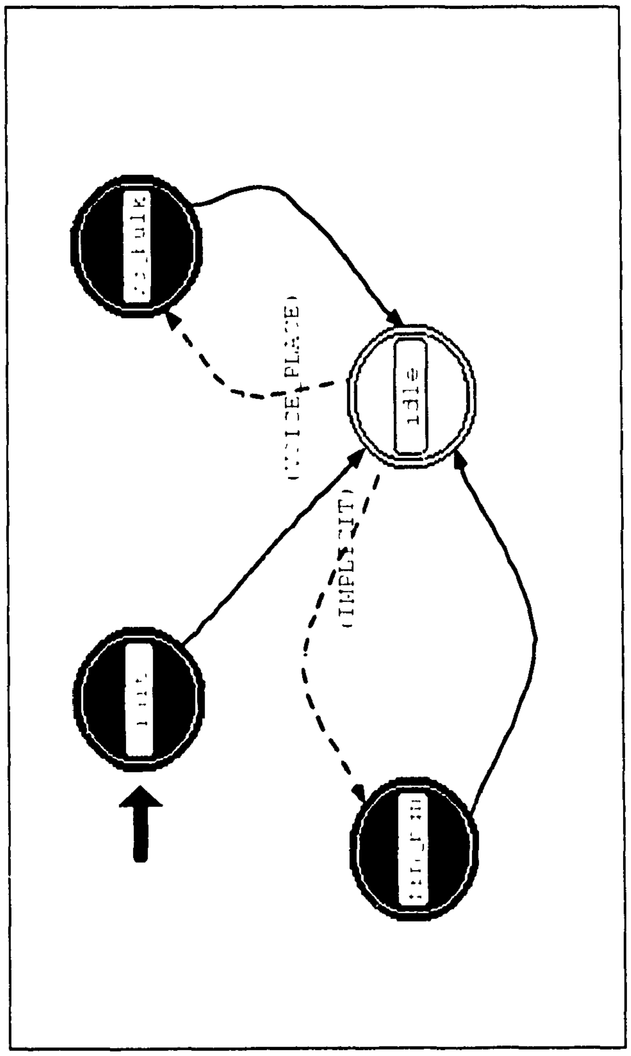


Figure A.6: The Voice Hub Process Model for CFDAMA-FA with MF-TDMA framing

Process Model Report: **res_aloha_obp**	Mon May 16 10:28:54 1994	Page 1 of 5
...		
...		

Header Block

```
#define SLOT_BEGIN op_intrpt_type() == OPC_INTRPT_SELF && \
                op_intrpt_code() == 2
#define REQ_ARRVL  op_intrpt_type() == OPC_INTRPT_STRM
#define END_SIM  op_intrpt_type() == OPC_INTRPT_ENDSIM
extern int USER_GROUP_SIZE;
#define NO_OF_GROUPS 10
#define OBP_IN_STRM 0
/*global variables*/
int no_pk_success;
double acc_pk_delay;
int RES_SLOT;
extern double packet_size , tiat;
Objid obp_id,obp_node_id,subnet_id;
int no_in_chnl;
/*extern Objid subnet_id;*/
extern double packet_size;
extern double res_chnl_cap;
int NO_REQ_SLOTS;
int pkts_lost;
extern int total_gen_pkts;
extern Objid hub_id;
```

State Variable Block

```
int \gp_id,\count, \user_id, \free_slot_owner, \free_slot_owner_gp;
double \chnl_cap;
int \j, \k, \i, \w;
```

Temporary Variable Block

```
Packet *pkptr, *respk, *ptrpk;
Objid origin_obj_id , node_id, node_q_id, free_slot_owner_gp_id;
Objid free_slot_owner_gp_q_id;
int slot_owner, num_subq;
double alpha;
Objid hub_node_id;
```

forced state Init

attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

...
...

enter execs **init**

```
gp_id = 0;
RES_SLOT = USER_GROUP_SIZE+1;
count = USER_GROUP_SIZE-1;
obp_id = op_id_self();
obp_node_id = op_id_parent( op_id_self());
subnet_id = op_id_parent(obp_node_id);
/*op_ima_sim_attr_get (OPC_IMA_DOUBLE , "packet_size_sec" , &packet_size);*/
no_pk_success = 0;
acc_pk_delay =0;
free_slot_owner = 0;
free_slot_owner_gp = 0;
pkts_lost=0;
```

transition **init -> dataslotas**

attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

unforced state **wait**

attribute	*value*	*type*	*default value*
name	wait	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition **wait -> req_Q_ing**

attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition	REQ_ARRVL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

transition **wait -> dataslotas**

attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition	SLOT_BEGIN	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition **wait -> end_sim**

attribute	*value*	*type*	*default value*
name	tr_4	string	tr

...
...

condition	END_SIM	string	
executive		string	
color	RGB331	color	RGB333
drawing style	spline	toggle	spline

forced state **req Q ing**

attribute	*value*	*type*	*default value*
name	req_Q_ing	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **req Q ing**

```
respk = op_pk_get(OBP_IN_STRM);
op_subq_pk_insert (0,respk,OPC_QPOS_TAIL) ;
```

transition **req Q ing -> wait**

attribute	*value*	*type*	*default value*
name	tr_6	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

forced state **resslotas**

attribute	*value*	*type*	*default value*
name	resslotas	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **resslotas**

```
op_intrpt_schedule_self(op_sim_time()+packet_size,2);
hub_node_id= op_id_from_userid(subnet_id, OPC_OBJTYPE_NODE_FIXED, 10);
hub_id=op_topo_child(hub_node_id, OPC_OBJTYPE_QUEUE, 0);
op_intrpt_schedule_remote(op_sim_time()+(packet_size/2),0,hub_id);
for(j=0;j<NO_OF_GROUPS;j++)
   {
   node_id = op_id_from_userid (subnet_id, OPC_OBJTYPE_NODE_FIXED, j);
   node_q_id = op_id_child (node_id,OPC_OBJTYPE_QUEUE,0);
   op_intrpt_force_remote(RES_SLOT ,node_q_id);
   }
```

...
...

transition **resslotas -> wait**

attribute	*value*	*type*	*default value*
name	tr_7	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

forced state **dataslotas**

attribute	*value*	*type*	*default value*
name	dataslotas	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **dataslotas**

```
if (!op_subq_empty(0))
 {
 pkptr = op_subq_pk_remove(0,OPC_QPOS_HEAD);
 origin_obj_id = op_pk_stamp_mod_get(pkptr);
 op_pk_nfd_get(pkptr,"user_index",&slot_owner);
 op_intrpt_schedule_remote(op_sim_time()+0.135,slot_owner,origin_obj_id );
/*has to be refined*/
 op_pk_destroy (pkptr);
 }
else
 {
  free_slot_owner_gp_id = op_id_from_userid(subnet_id,OPC_OBJTYPE_NODE_FIXED,free_slot_owner_gp);
  free_slot_owner_gp_q_id = op_id_child (free_slot_owner_gp_id,OPC_OBJTYPE_QUEUE,0);
  op_intrpt_schedule_remote(op_sim_time()+0.135,free_slot_owner,free_slot_owner_gp_q_id );

if (free_slot_owner == USER_GROUP_SIZE -1)
    {
    free_slot_owner = 0;
    ++free_slot_owner_gp;
    }
  else
              ++free_slot_owner;
           if(free_slot_owner_gp == NO_OF_GROUPS)
         free_slot_owner_gp = 0;
       else
          ;
 }
```

transition **dataslotas -> resslotas**

attribute	*value*	*type*	*default value*
name	tr_8	string	tr
condition		string	
executive		string	

...
...

color	RGB300	color	RGB333
drawing style	spline	toggle	spline

unforced state **end sim**

attribute	*value*	*type*	*default value*
name	end_sim	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

enter execs **end sim**

```
chnl_cap = (1-res_chnl_cap);
alpha=(pkts_lost/total_gen_pkts);
op_stat_write_scalar("Throughput" , chnl_cap*((1-alpha)*packet_size/tiat));
op_stat_write_scalar("Expected Packet Delay(sec)", (acc_pk_delay/no_pk_success));
```

...
...

Header Block

```
#define ARRIVAL                         \
      (op_intrpt_type() == OPC_INTRPT_STRM)
#define GP_RES_SLOT                        \
      (op_intrpt_type() == OPC_INTRPT_REMOTE) &&   \
      (op_intrpt_code() == RES_SLOT )
#define USER_DATA_SLOT                       \
      (op_intrpt_type() == OPC_INTRPT_REMOTE) &&   \
      (!(op_intrpt_code() == RES_SLOT))
extern int USER_GROUP_SIZE;
extern Objid obp_id ,obp_node_id, subnet_id;
#define OBP_IN_STRM_INDEX 0
/*#define ARBITRARY (USER_GROUP_SIZE+2)
#define RES_SLOT 11*/
/*global stat*/
extern double acc_pk_delay;
extern int no_pk_success,no_in_chnl;
extern double packet_size;
extern int RES_SLOT;
extern int NO_RES_SLOTS;
extern Objid hub_id;
```

State Variable Block

```
double \pk_delay;
double \res_slot_time;
int \user_turn , \i, \j;
int \user_index_res, \ARBITRARY;
double \num_in_subq;
/*user_index_res q has the pkts expecting slot*/
```

Temporary Variable Block

```
Packet *pkptr, *respkptr, *pk;
int user_index, slot_owner;
double packet_delay, create_time;
```

unforced state **evebranch**

attribute	*value*	*type*	*default value*
name	evebranch	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition **evebranch -> sub_q_ing**

attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	ARRIVAL	string	
executive		string	

color	RGB033	color	RGB333
drawing style	spline	toggle	spline

transition **evebranch -> reservn**			
attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition	GP_RES_SLOT	string	
executive		string	
color	RGB331	color	RGB333
drawing style	spline	toggle	spline

transition **evebranch -> send**			
attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition	USER_DATA_SLOT	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

forced state **reservn**			
attribute	*value*	*type*	*default value*
name	reservn	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **reservn**

```
for(i=0;i<USER_GROUP_SIZE;i++)
 {
 if (!op_subq_empty(i))
 {
  ++no_in_chnl;
  num_in_subq = op_subq_stat (i, OPC_QSTAT_PKSIZE);
  for(j=0;j<num_in_subq;++j)
  {
   pkptr = op_subq_pk_remove(i,OPC_QPOS_HEAD);
   respkptr = op_pk_copy (pkptr);
   op_subq_pk_insert ((USER_GROUP_SIZE + i),respkptr,OPC_QPOS_TAIL);
   op_pk_deliver(pkptr,bub_id,0);
   }
  }
  else
  ;
 }
```

transition **reservn -> evebranch**			
attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition		string	

...
...

executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

forced state **sub q ing**			
attribute	*value*	*type*	*default value*
name	sub_q_ing	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **sub q ing**

```
pkptr = op_pk_get(op_intrpt_strm() );
op_pk_stamp (pkptr);
op_pk_nfd_get (pkptr,"user_index", &user_index);
op_subq_pk_insert (user_index, pkptr, OPC_QPOS_TAIL);

```

transition **sub q ing -> evebranch**			
attribute	*value*	*type*	*default value*
name	tr_5	string	tr
condition		string	
executive		string	
color	RGB033	color	RGB333
drawing style	spline	toggle	spline

forced state **send**			
attribute	*value*	*type*	*default value*
name	send	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **send**

```
slot_owner = op_intrpt_code();
if(!op_subq_empty(slot_owner+USER_GROUP_SIZE))
 {
 pk = op_subq_pk_remove (slot_owner+USER_GROUP_SIZE, OPC_QPOS_HEAD);
 op_pk_nfd_get(pk, "create_time", &create_time);
 packet_delay = op_sim_time() + 0.27 + packet_size - create_time;
 acc_pk_delay += packet_delay;
 ++no_pk_success;
 op_pk_destroy (pk);
 }
else
 {
 if(!op_subq_empty(slot_owner))
   {
   pk = op_subq_pk_remove (slot_owner, OPC_QPOS_HEAD);
```

...
...

```
  op_pk_nfd_get(pk, "create_time", &create_time);
  packet_delay = op_sim_time() + 0.27 + packet_size - create_time;
  acc_pk_delay += packet_delay;
  ++no_pk_success;
  op_pk_destroy (pk);
  }
else
  ;
}
```

transition **send -> evebranch**

attribute	*value*	*type*	*default value*
name	tr_6	string	tr
condition		string	
executive		string	
color	RGB331	color	RGB333
drawing style	spline	toggle	spline

forced state **init**

attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **init**

```
user_turn = 0;
/*op_ima_sim_attr_get(OPC_IMA_DOUBLE, "packet_size_sec" , &packet_size);*/
ARBITRARY = USER_GROUP_SIZE + 2;
no_in_chnl = 0;
```

transition **init -> evebranch**

attribute	*value*	*type*	*default value*
name	tr_7	string	tr
condition		string	
executive		string	
color	RGB331	color	RGB333
drawing style	spline	toggle	spline

...
...

Header Block

```
int USER_GROUP_SIZE;
int USER_SIZE_INDEX;
#define NO_OF_USER_GROUPS 10
#define OUT_STRM 0
double tiat;
double packet_size;
double res_chnl_cap;
int total_gen_pkts;
```

State Variable Block

```
Distribution *\gpiat_dist_ptr;
Distribution *\index_dist_ptr;
Objid \gp_node_id;
int \gp_id;
```

Temporary Variable Block

```
Packet *pkptr;
int user_index;
double gpiat ;
```

unforced state **init**

attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

enter execs **init**

```
op_ima_sim_attr_get (OPC_IMA_DOUBLE , "packet_size_sec" , &packet_size);
op_ima_sim_attr_get (OPC_IMA_DOUBLE , "total_i_a_t" , &tiat );
op_ima_sim_attr_get (OPC_IMA_INTEGER , "user_group_size", &USER_GROUP_SIZE);
op_ima_sim_attr_get (OPC_IMA_DOUBLE , "res_chnl_size", &res_chnl_cap);
USER_SIZE_INDEX = USER_GROUP_SIZE - 1;
gpiat = tiat*NO_OF_USER_GROUPS;
index_dist_ptr = op_dist_load("uniform_int", 0, USER_SIZE_INDEX );
gpiat_dist_ptr = op_dist_load("exponential" , gpiat , 0.0);
op_intrpt_schedule_self (op_dist_outcome(gpiat_dist_ptr),0);
gp_node_id = op_id_parent(op_id_self());
op_ima_obj_attr_get(gp_node_id,"user id",&gp_id);
/*user_id and gp_id are same*/
/*beg_sim_intpt should be enabled*/
total_gen_pkts=0;
```

...
...

transition **init -> arrival**

attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

unforced state **arrival**

attribute	*value*	*type*	*default value*
name	arrival	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

enter execs **arrival**

```
pkptr = op_pk_create_fmt ("packet");
user_index = op_dist_outcome (index_dist_ptr);
op_pk_nfd_set (pkptr, "gp_id",gp_id);
op_pk_nfd_set (pkptr, "user_index",user_index);
op_pk_nfd_set (pkptr, "create_time",op_sim_time());
op_pk_send(pkptr,OUT_STRM);
total_gen_pkts++;
op_intrpt_schedule_self (op_sim_time () + op_dist_outcome(gpiat_dist_ptr),0);
```

transition **arrival -> arrival**

attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

...
...

Header Block

```
Objid hub_id:
#define ARVL_CHK \
(op_intrpt_type()== OPC_INTRPT_STRM)
#define COL_CHK \
 (op_intrpt_type()==OPC_INTRPT_REMOTE)
extem Objid obp_id;
extern int no_in_chnl;
```

Temporary Variable Block

```
Packet *pkptr;
int numb_q,j;
```

forced state **init**

attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **init**

transition **init -> wait**

attribute	*value*	*type*	*default value*
name	tr_0	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

unforced state **wait**

attribute	*value*	*type*	*default value*
name	wait	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition **wait -> q_pkts**

attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	ARVL_CHK	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

...
...

transition **wait -> validity**			
attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition	COL_CHK	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state **q_pkts**			
attribute	*value*	*type*	*default value*
name	q_pkts	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **q_pkts**

```
pkptr=op_pk_get(0);
op_subq_pk_insert(0,pkptr,OPC_QPOS_TAIL);
```

transition **q_pkts -> wait**			
attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state **validity**			
attribute	*value*	*type*	*default value*
name	validity	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **validity**

```
if(no_in_chnl==1)
{
numb_q=op_subq_stat(0,OPC_QSTAT_PKSIZE);
for(j=0;j<numb_q;++j)
{
pkptr=op_subq_pk_remove(0,OPC_QPOS_HEAD);
op_pk_deliver_delayed(pkptr,obp_id,0,0.135);
}
}
else
{
numb_q=op_subq_stat(0, OPC_QSTAT_PKSIZE);
op_subq_flush(0);
```

...
...

```
15  }
    no_in_chnl=0;
```

transition **validity -> wait**			
attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

...
...

Header Block

```
#define SLOT_BEGIN op_intrpt_type() == OPC_INTRPT_SELF && \
            op_intrpt_code() == 2
#define REQ_ARRVL op_intrpt_type() == OPC_INTRPT_STRM
#define END_SIM op_intrpt_type() == OPC_INTRPT_ENDSIM
extern int USER_GROUP_SIZE;
#define NO_OF_GROUPS 8
#define OBP_IN_STRM 0
extern double rho, alpha;
/*global variables*/
int no_pk_success;
extern int respk;
int total_slots;
int no_req_pkts;
double acc_pk_delay;
int RES_SLOT;
extern double packet_size , tiat;
Objid obp_id,obp_node_id,subnet_id;
int no_in_chnl;
/*extern Objid subnet_id;*/
extern double packet_size;
extern double res_chnl_cap;
int NO_REQ_SLOTS;
int pkts_lost;
extern int tot_gen_pkts;
extern Objid voice_q_id;
extern double frame_time;
extern int no_succ_calls;
extern int no_calls;
int indice[1000];
int voice_tab[1000];
extern Objid hub_id;
extern int L, M, C;
int pkts_carried_ovr;
extern int v_calls;
extern double call_time;
extern double tiat_p, tiat_v;
extern int Q;
extern int blocked;
```

State Variable Block

```
int \gp_id,\count, \user_id, \free_slot_owner, \free_slot_owner_gp;
double \chnl_cap;
int \j, \k, \i, \w;
int \res_slot_owner, \res_slot_owner_gp
double \blocking;
```

Temporary Variable Block

```
Packet *pkptr, *respk, *ptrpk, *pkptr1;
Objid origin_obj_id , node_id, node_q_id, free_slot_owner_gp_id;
Objid free_slot_owner_gp_q_id, hub_id;
int slot_owner, num_subq;
int z1, z2;
int of;
```

...
...

```
int slot_id;
int ind, Z;
int nq;
int mark;
int p_calls;
double blk_pkts;
int stat;
int vc_code;
```

forced state **init**			
attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **init**

```
gp_id = 0;
RES_SLOT = USER_GROUP_SIZE+1;
count = USER_GROUP_SIZE-1;
obp_id = op_id_self();
obp_node_id = op_id_parent( op_id_self());
subnet_id = op_id_parent(obp_node_id);
/*op_ima_sim_attr_get (OPC_IMA_DOUBLE , "packet_size_sec" , &packet_size);*/
no_pk_success = 0;
acc_pk_delay =0;
free_slot_owner = 0;
free_slot_owner_gp = 0;
pkts_lost=0;
res_slot_owner = 0;
res_slot_owner_gp=0;
for(z1=0;z1<1000;z1++)
{
indice[z1]=0;
voice_tab[z1]=0;
}
of=0;
pkts_carried_ovr=0;
```

transition **init -> dataslotas**			
attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

unforc⁻ᵈ state **wait**			
attrib' .e	*value*	*type*	*default value*
name	wait	string	st
enter execs	(empty)	textlist	(empty)

exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition wait -> req Q ing			
attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition	REQ_ARRVL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

transition wait -> dataslotas			
attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition	SLOT_BEGIN	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition wait -> end sim			
attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition	END_SIM	string	
executive		string	
color	RGB331	color	RGB333
drawing style	spline	toggle	spline

forced state req Q ing			
attribute	*value*	*type*	*default value*
name	req_Q_ing	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs req Q ing

```
respk = op_pk_get(OBP_IN_STRM);
op_subq_pk_insert (0,respk,OPC_QPOS_TAIL) ;
```

transition req Q ing -> wait			
attribute	*value*	*type*	*default value*
name	tr_6	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

...
...

forced state **resslotas**			
attribute	*value*	*type*	*default value*
name	resslotas	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **resslotas**

```
frame_time=M*packet_size;
op_intrpt_schedule_self(op_sim_time()+frame_time,2);
for(j=0;j<Q;j++)
  {
  node_id = op_id_from_userid (subnet_id, OPC_OBJTYPE_NODE_FIXED, res_slot_owner_gp);
  node_q_id = op_id_child (node_id,OPC_OBJTYPE_QUEUE,0);
  op_intrpt_schedule_remote(op_sim_time()+0.135,((4*USER_GROUP_SIZE)+res_slot_owner), node_q_id);
   if(res_slot_owner == USER_GROUP_SIZE-1)
    {
     res_slot_owner = 0;
     ++ res_slot_owner_gp;
    }
  else
    ++ res_slot_owner;
  if(res_slot_owner_gp == NO_OF_GROUPS)
   {
    res_slot_owner_gp=0;
   }
else
;
}
```

transition **resslotas -> wait**			
attribute	*value*	*type*	*default value*
name	tr_7	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

forced state **dataslotas**			
attribute	*value*	*type*	*default value*
name	dataslotas	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **dataslotas**

```
hub_id=op_id_from_userid(subnet_id, OPC_OBJTYPE_NODE_FIXED, 9);
voice_q_id=op_topo_child(hub_id,OPC_OBJTYPE_QUEUE,0);
op_intrpt_schedule_remote(op_sim_time()+0.135,0,voice_q_id);
p_calls=v_calls;
```

...

...

```
total_slots+=p_calls;
of=0;
for(Z=0;Z < (C-p_calls);Z++)
{
nq=op_subq_stat(0,OPC_QSTAT_PKSIZE);
if (nq > of)
 {
 pkptr = op_subq_pk_access(0,of);
 op_pk_nfd_get(pkptr,"user_index",&slot_owner);
 op_pk_nfd_get(pkptr,"gp_id",&slot_id);
 ind=(slot_id*USER_GROUP_SIZE) + slot_owner;
 if((indice[ind]+voice_tab[ind]) < M)
 {
pkptr=op_subq_pk_remove(0,of);
op_pk_nfd_get(pkptr, "status", &stat);
 origin_obj_id = op_pk_stamp_mod_get(pkptr);
if(stat==0)
{
++no_req_pkts;
++total_slots;
 op_intrpt_schedule_remote(op_sim_time()+0.135,slot_owner,origin_obj_id);
}
else
{
vc_code=6*USER_GROUP_SIZE + slot_owner;
 op_intrpt_schedule_remote(op_sim_time()+0.135,vc_code,origin_obj_id);
}
/*has to be refined*/
 op_pk_destroy (pkptr);
indice[ind]++;
 }
else
{
of++;
++pkts_carried_ovr;
}
}
else
 {
 mark=0;
 ind=(free_slot_owner_gp*USER_GROUP_SIZE)+free_slot_owner;
 while(mark !=1 )
{
if((indice[ind]+voice_tab[ind]) < M )
{
 free_slot_owner_gp_id = op_id_from_userid(subnet_id,OPC_OBJTYPE_NODE_FIXED,free_slot_owner_gp);
 free_slot_owner_gp_q_id = op_id_child (free_slot_owner_gp_id,OPC_OBJTYPE_QUEUE,0);
  op_intrpt_schedule_remote(op_sim_time()+0.135,free_slot_owner,free_slot_owner_gp_q_id );
  mark = 1;
 ++ total_slots;
 }
else
;
if (free_slot_owner == USER_GROUP_SIZE -1)
   {
   free_slot_owner = 0;
   ++free_slot_owner_gp;
   }
  else
```

...
...

```
          ++free_slot_owner;
        if(free_slot_owner_gp == NO_OF_GROUPS)
      free_slot_owner_gp = 0;
    else
        ;
}
}
 }
of=0;
for(z2=0;z2<1000;z2++)
{
indice[z2]=0;
}
```

transition **dataslotas -> resslotas**

attribute	*value*	*type*	*default value*
name	tr_8	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

unforced state **end sim**

attribute	*value*	*type*	*default value*
name	end_sim	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

enter execs **end sim**

```
chnl_cap = (1-res_chnl_cap);
blk_pkts=((double)pkts_carried_ovr/(double)tot_gen_pkts);
/*op_stat_write_scalar("Throughput_Data" , ((chnl_cap*packet_size)/(tiat_p*L)));*/
op_stat_write_scalar("Expected Packet Delay(sec)", (acc_pk_delay/no_pk_success));
printf("No_succ_calls=%d", no_succ_calls);
printf("\n No_CALLS = %d", no_calls);
printf("The Number Blocked= %d", blocked);
printf("\n Packets carried Over= %d", pkts_carried_ovr);
printf("The number who reserved=%d", respk);
blocking = (double)blocked/(double)no_calls;
op_stat_scalar_write("Blocking Probability", blocking);
/*op_stat_scalar_write("Throughput_Voice",(rho*alpha));*/
op_stat_scalar_write("Throughput", rho);
op_stat_scalar_write("Fraction of DA slots", ((double)no_req_pkts/(double)total_slots));
```

...
...

Header Block

```
#define ARRIVAL                              \
      (op_intrpt_type() == OPC_INTRPT_STRM)
#define RES_SLOT                             \
      (op_intrpt_type() == OPC_INTRPT_REMOTE)  &&  \
       (op_intrpt_code() >= 4*USER_GROUP_SIZE) &&  \
       (op_intrpt_code() < 6*USER_GROUP_SIZE)
#define VC_RES_SLOT                          \
      (op_intrpt_type() == OPC_INTRPT_REMOTE)  &&  \
      (op_intrpt_code() >= 6*USER_GROUP_SIZE)
#define USER_DATA_SLOT                          \
      (op_intrpt_type() == OPC_INTRPT_REMOTE)  &&  \
      (op_intrpt_code() < 4*USER_GROUP_SIZE)
extern int USER_GROUP_SIZE ;
extern Objid  obp_id ,obp_node_id, subnet_id;
#define OBP_IN_STRM_INDEX 0
extern double acc_pk_delay;
extern int no_pk_success;
extern double packet_size;
extern int VOICE_ADV;
extern Objid voice_q_id;
int no_calls, no_succ_calls;
extern int v_calls,Cv;
int blocked;
extern int L, M, C;
int sq_acc_pk_delay;
int respk;
```

State Variable Block

```
Distribution *\cor;
Objid \m_id;
```

Temporary Variable Block

```
Packet *pkptr,*pk,*respkptr, *pk1,*pkt, *pkz, *pky, *rpk, *pko;
int user_index, slot_owner;
double packet_delay, create_time;
int num_in_subq , i, vsq, oa;
int owner;
int num_q,n_q,w,v,m;
int status;
int index;
int owner_v;
int res, req;
int pl;
Objid t_id;
int s_id, n_id,oi;
int fol, fl;
int ocome;
int x;
int v_code;
double time;
int vrq;
```

...
...

unforced state **evebranch**

attribute	*value*	*type*	*default value*
name	evebranch	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition **evebranch -> sub_q_ing**

attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB033	color	RGB333
drawing style	spline	toggle	spline

transition **evebranch -> resns**

attribute	*value*	*type*	*default value*
name	tr_28	string	tr
condition	RES_SLOT	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition **evebranch -> vc_slot**

attribute	*value*	*type*	*default value*
name	tr_30	string	tr
condition	VC_RES_SLOT	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition **evebranch -> send**

attribute	*value*	*type*	*default value*
name	tr_33	string	tr
condition	USER_DATA_SLOT	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state **sub_q_ing**

attribute	*value*	*type*	*default value*
name	sub_q_ing	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **sub_q_ing**

```
pkptr = op_pk_get(op_intrpt_strm());
op_pk_nfd_get(pkptr, "status", &status);
op_pk_nfd_get(pkptr, "op_id", &m_id);
```

...
...

```
if(status==0)
{
op_pk_stamp (pkptr);
op_pk_nfd_get (pkptr,"user_index", &user_index);
op_subq_pk_insert (user_index, pkptr, OPC_QPOS_TAIL);
}
else
{
if(status==1)
{
no_calls++;
op_pk_stamp(pkptr);
op_pk_nfd_get(pkptr,"user_index", &user_index);
pl=3*USER_GROUP_SIZE+user_index;
op_subq_pk_insert(pl,pkptr,OPC_QPOS_TAIL);
}
else
{
printf("HEY STATUS NOT FOUND");
printf("status = %d", status);
}
}
```

transition **sub_q_ing -> evebranch**

attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition		string	
executive		string	
color	RGB033	color	RGB333
drawing style	spline	toggle	spline

forced state **send**

attribute	*value*	*type*	*default value*
name	send	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **send**

```
slot_owner = op_intrpt_code();
if(!(op_subq_empty(slot_owner+USER_GROUP_SIZE)))
 {
 pk = op_subq_pk_remove ((slot_owner+USER_GROUP_SIZE) , OPC_QPOS_HEAD);
 op_pk_nfd_get(pk, "create_time" , &create_time);
 packet_delay = op_sim_time() + 0.27 + packet_size - create_time;
 sq_acc_pk_delay += packet_delay*packet_delay;
/* op_stat_write_global(0, "packet_delay", packet_delay); */
 acc_pk_delay += packet_delay;
 ++no_pk_success;
 op_pk_destroy (pk);
 }
else
```

...
...

```
{
if(!(op_subq_empty(slot_owner)))
  {
  pk = op_subq_pk_remove(slot_owner, OPC_QPOS_HEAD);
  op_pk_nfd_get(pk, "create_time" , &create_time);
  packet_delay = op_sim_time() + 0.27 + packet_size - create_time;
/* op_stat_write_global(0, "packet_delay", packet_delay);*/
   sq_acc_pk_delay += packet_delay*packet_delay;
  acc_pk_delay += packet_delay;
  ++no_pk_success;
   op_pk_destroy (pk);
  }
 else
  ;
 }
```

transition **send -> evebranch**

attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition		string	
executive		string	
color	RGB331	color	RGB333
drawing style	spline	toggle	spline

forced state **init**

attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **init**

```
no_calls=0;
no_succ_calls=0;
respk=0;
```

transition **init -> evebranch**

attribute	*value*	*type*	*default value*
name	tr_27	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

...
...

forced state **resns**			
attribute	*value*	*type*	*default value*
name	resns	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

***enter execs* resns**

```
fl=0;
while(op_subq_empty(fl) && (fl < 4*USER_GROUP_SIZE + 1))
{
fl++;
}
if(fl<=4*USER_GROUP_SIZE)
{
pkz=op_subq_pk_access(fl, OPC_QPOS_HEAD);
op_pk_nfd_get(pkz, "gp_id", &s_id);
n_id=op_intrpt_code()-(4*USER_GROUP_SIZE);
if(!op_subq_empty(n_id))
{
num_in_subq=op_subq_stat(n_id,OPC_QSTAT_PKSIZE);
for(m=0;m<num_in_subq;++m)
{
pk = op_subq_pk_remove(n_id,OPC_QPOS_HEAD);
respkptr=op_pk_copy(pk);
op_subq_pk_insert((USER_GROUP_SIZE+n_id),respkptr,OPC_QPOS_TAIL);
op_pk_deliver_delayed(pk,obp_id,0,0.135);
}
}
else
;
oi=3*USER_GROUP_SIZE+n_id;
vrq=oi+USER_GROUP_SIZE;
if(!op_subq_empty(oi))
{
vsq=op_subq_stat(oi, OPC_QSTAT_PKSIZE);
for(oa=0;oa<vsq;++oa)
{
pky=op_subq_pk_remove(oi,OPC_QPOS_HEAD);
pko=op_pk_copy(pky);
op_subq_pk_insert(vrq, pko, OPC_QPOS_TAIL);
op_pk_deliver_delayed(pky,obp_id,0,0.135);
++respk;
}
}
else
;
}
else
;
```

transition **resns -> evebranch**			
attribute	*value*	*type*	*default value*
name	tr_29	string	tr

condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state **vc_slot**

attribute	*value*	*type*	*default value*
name	vc_slot	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **vc_slot**

```
v_code=op_intrpt_code();
slot_owner=v_code-(6*USER_GROUP_SIZE);
if(!(op_subq_empty(slot_owner+4*USER_GROUP_SIZE)))
{
pk1=op_subq_pk_remove((slot_owner+4*USER_GROUP_SIZE), OPC_QPOS_HEAD);
op_pk_nfd_get(pk1, "create_time", &create_time);
t_id=op_id_from_userid(subnet_id, OPC_OBJTYPE_NODE_FIXED,9);
voice_q_id=op_topo_child(t_id,OPC_OBJTYPE_QUEUE,0);
time=op_sim_time()-create_time;
if((v_calls < Cv))
{
op_pk_deliver(pk1, voice_q_id,0);
no_succ_calls++;
}
else
{
blocked++;
op_pk_destroy(pk1);
}
}
else
;
```

transition **vc_slot -> evebranch**

attribute	*value*	*type*	*default value*
name	tr_32	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

...
...

Header Block

```
int USER_GROUP_SIZE;
int USER_GROUP_SIZE_INDEX;
#define NO_OF_GROUPS 8
#define OUT_STRM 0
double rho;
int Q;
double talk_dur;
double tiat_p;
double call_time;
double tiat_v;
double packet_size;
double vc_intarrvl_time;
#define DATA_ARRVL                    \
 (op_intrpt_type()==OPC_INTRPT_SELF) &&   \
 (op_intrpt_code()== 0)
#define VOICE_ARVL                     \
 (op_intrpt_type()==OPC_INTRPT_SELF) &&   \
 (op_intrpt_code()==1)
int no_calls;
int M,L;
int no_succ_calls;
int tot_gen_pkts;
int C, Cv;
double frame_time;
double alpha;
double res_chnl_cap;
```

State Variable Block

```
Distribution *\gpiat_dist_ptr;
Distribution *\index_dist_ptr;
Distribution *\voice_intrvl_ptr;
Objid \gp_node_id;
int \gp_id;
double \[illegible];
```

Temporary Variable Block

```
Packet *pkptr;
int user_index;
double gpiatd;
double gpiatv;
double gpiat;
```

forced state init			
attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

...
...

enter execs **init**

```
/*op_ima_sim_attr_get(OPC_IMA_DOUBLE,"packet_size",&packet_size);*/
op_ima_sim_attr_get(OPC_IMA_DOUBLE,"Total Load",&rho);
op_ima_sim_attr_get(OPC_IMA_DOUBLE,"fraction_of_voice_traffic(alpha)",&alpha);
op_ima_sim_attr_get(OPC_IMA_INTEGER,"user_group_size",&USER_GROUP_SIZE);
op_ima_sim_attr_get(OPC_IMA_INTEGER,"no_of_frequencies", &L);
/*op_ima_sim_attr_get(OPC_IMA_INTEGER,"no_of_request_slots", &Q);*/
/*op_ima_sim_attr_get(OPC_IMA_INTEGER, "no_of_time_slots", &M);*/
/*op_ima_sim_attr_get(OPC_IMA_INTEGER,"no_voice_slots_per_frame",&Cv);*/
/*op_ima_sim_attr_get(OPC_IMA_DOUBLE,"voice_call_size_min",&talk_dur);*/
packet_size=0.0001875;
M=128;
Q=4;
talk_dur=0.03;
C=L*M;
Cv=C;
call_time=talk_dur*60;
erlang=rho*alpha*M*L;
frame_time=M*packet_size;
USER_GROUP_SIZE_INDEX=USER_GROUP_SIZE-1;
tiat_v=call_time/erlang;
tiat_p=packet_size/(((1-alpha)*rho)*L);
gpiatd=NO_OF_GROUPS*tiat_p;
gpiatv=NO_OF_GROUPS*tiat_v;
printf("tiatp=%g, tiat_v=%g", tiat_p, tiat_v);
index_dist_ptr=op_dist_load("uniform_int", 0, USE[illegible]ROUP_SIZE_INDEX);
gpiat_dist_ptr=op_dist_load("exponential",gpiatd,0.0);
voice_intrvl_ptr=op_dist_load("exponential", gpiatv,0.0);
op_intrpt_schedule_self(op_dist_outcome(gpiat_dist_ptr),0);
op_intrpt_schedule_self(op_dist_outcome(voice_intrvl_ptr), 1);
gp_node_id=op_topo_parent(op_id_self());
op_ima_obj_attr_get(gp_node_id, "user id", &gp_id);
no_calls=0;
tot_gen_pkts=0;
res_chnl_cap=0.06;
```

transition **init -> idle**

attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

unforced state **idle**

attribute	*value*	*type*	*default value*
name	idle	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

...
...

transition **idle -> vc arrvl**

attribute	*value*	*type*	*default value*
name	tr_5	string	tr
condition	VOICE_ARVL	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition **idle -> data arrvl**

attribute	*value*	*type*	*default value*
name	tr_10	string	tr
condition	DATA_ARRVL	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state **vc arrvl**

attribute	*value*	*type*	*default value*
name	vc_arrvl	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **vc arrvl**

```
pkptr=op_pk_create_fmt("multi");
user_index=op_dist_outcome(index_dist_ptr);
op_pk_nfd_set(pkptr, "resn", 0);
op_pk_nfd_set(pkptr, "gp_id", gp_id);
op_pk_nfd_set(pkptr, "user_index", user_index);
op_pk_nfd_set(pkptr, "status", 1);
op_pk_nfd_set(pkptr, "create_time", op_sim_time());
op_pk_send(pkptr, OUT_STRM);
op_intrpt_schedule_self(op_sim_time()+op_dist_outcome(voice_intrvl_ptr),1);
```

transition **vc arrvl -> idle**

attribute	*value*	*type*	*default value*
name	tr_9	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state **data arrvl**

attribute	*value*	*type*	*default value*
name	data_arrvl	string	st
enter execs	(See below.)	textlist	(See below.)

...
...

exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **data arrvl**

```
++tot_gen_pkts;
pkptr=op_pk_create_fmt("multi");
user_index=op_dist_outcome(index_dist_ptr);
op_pk_nfd_set(pkptr,"gp_id", gp_id);
op_pk_nfd_set(pkptr,"user_index", user_index);
op_pk_nfd_set(pkptr,"status", 0);
op_pk_nfd_set(pkptr,"create_time",op_sim_time());
op_pk_send(pkptr, OUT_STRM);
op_intrpt_schedule_self(op_sim_time()+op_dist_outcome(gpiat_dist_ptr), 0);
```

transition **data arrvl -> idle**

attribute	*value*	*type*	*default valu*
name	tr_11	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

...
...

Header Block

```
int v_calls;
extern int Cv;
#define IMPLICIT                    \
  (op_intrpt_type()==OPC_INTRPT_REMOTE)
#define VOICE_PLACE                 \
  (op_intrpt_type()==OPC_INTRPT_STRM)
extern double packet_size;
extern Objid subnet_id;
extern double call_time;
extern int USER_GROUP_SIZE;
extern double frame_time;
Objid voice_q_id;
extern int no_succ_calls;
extern int indice[1000];
extern int voice_tab[1000];
extern int L, M;
int blk[8][128];
extern int blocked;
extern int C;
```

State Variable Block

Temporary Variable Block

```
int h,call_owner,gp_call_owner;
Packet *pkd, *pktalk, *pkptr, *pkr, *rpk;
Objid own_id,own_q_id;
double temp;
int y;
int no_pkts;
int f_r;
int c;
double ref_time,call_dur;
double now;
Distribution *call_size;
double timing;
int ind, indi;
int u1,u2;
int p1, p2;
int length;
int h1;
```

forced state **init**

attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

...
...

enter execs **init**

```
v_calls=0;
```

transition **init -> idle**

attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state **vc_bulk**

attribute	*value*	*type*	*default value*
name	vc_bulk	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(See below.)	textlist	(See below.)
status	forced	toggle	unforced

enter execs **vc_bulk**

```
pkptr=op_pk_get(0);
op_pk_nfd_get(pkptr,"gp_id",&gp_call_owner);
op_pk_nfd_get(pkptr,"user_index", &call_owner);
indi=gp_call_owner*USER_GROUP_SIZE+call_owner;
if((v_calls <= C) && (voice_tab[indi] <= M))
{
voice_tab[indi]++;
v_calls=v_calls++;
y=0;
while(!(op_subq_empty(y)))
{
 y++;
}
call_size=op_dist_load("exponential",call_time,0.0);
temp=op_dist_outcome(call_size);
op_pk_nfd_set(pkptr,"ref_time",op_sim_time());
op_pk_nfd_set(pkptr,"call_dur",temp);
op_subq_pk_insert(y, pkptr, OPC_QPOS_TAIL);
}
else
{
op_pk_destroy(pkptr);
no_succ_calls--;
blocked ++;
}
```

exit execs **vc_bulk**

```
...
...
```

transition **vc_bulk -> idle**

attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

unforced state **idle**

attribute	*value*	*type*	*default value*
name	idle	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition **idle -> vc_bulk**

attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition	VOICE_PLACE	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition **idle -> frm_bgn**

attribute	*value*	*type*	*default value*
name	tr_5	string	tr
condition	IMPLICIT	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state **frm_bgn**

attribute	*value*	*type*	*default value*
name	frm_bgn	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **frm_bgn**

```
h=0;
while(h<=Cv)
{
if(!(op_subq_empty(h)))
{
pkd=op_subq_pk_access(h,OPC_QPOS_HEAD);
op_pk_nfd_get(pkd, "user_index", &call_owner);
op_pk_nfd_get(pkd, "gp_id", &gp_call_owner);
op_pk_nfd_get(pkd, "ref_time", &ref_time);
op_pk_nfd_get(pkd, "call_dur", &call_dur);
now=op_sim_time();
timing=now-ref_time;
if(timing >= call_dur)
{
pkr=op_subq_pk_remove(h,OPC_QPOS_HEAD);
op_pk_destroy(pkr);
ind=(gp_call_owner*USER_GROUP_SIZE)+call_owner;
voice_tab[ind]--;
v_calls=v_calls--;
}
else
;
}
else
;
h++;
}
```

transition **frm bgn -> idle**

attribute	*value*	*type*	*default value*
name	tr_6	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

Bibliography

[1] H. Ahmadi and T.E. Stern, "A New Satellite Multiple Access Technique for Packet Switching Using Combined Fixed and Demand Assignments", *Conf. Records of NTC*, pages 70.4.1–70.4.3, 1980.

[2] B. Ackroyd, *"World Satellite Communications and Earth Station Design"*, BSP Professional Books, 1990.

[3] B. Jabbari, "Combined FDMA-TDMA: A Cost Effective Technique for Digital Satellite Communication Networks", *Proc. IEEE Int'l Conf. Commun.*, pages 7F.4.1–7F.4.5, 1982.

[4] B. Jabbari, "Cost-Effective Networking Via Digital Satellite Communications", *Proc. IEEE*, 72(11):1556–1563, November 1984.

[5] B. Jabbari and D. McDysan, "Performance of Demand Assignment TDMA and Multicarrier TDMA Satellite Networks", *IEEE J. Select. Areas Commun.*, 10(2):478–486, February 1992.

[6] C.C.K. Poon and T. Suda, "Delay Analysis of an Integrated Voice and Data Access Protocol with Collision Detection for Multimedia Satellite Networks", *American Institute of Aeronautics and Astronautics Inc.*, AIAA-92-1828-CP:193–201, 1992.

[7] D.R. Cox, *"Renewal Theory"*, Methuen and Co. Ltd., London, 1962.

[8] E. Cinlar, *"Introduction to Stochastic Processes"*, Prentice-Hall,Inc., Englewood Cliffs, New Jersey, 1975.

[9] G. Benelli et al, "Performance of Uplink Random-Access and Downlink TDMA Techniques for Packet Satellite Networks", *Proc. IEEE*, 72(11), November 1984.

[10] J.G. Puente et al, "Multiple-Access Techniques for Commercial Satellites", *Proc. IEEE*, 59(2):218–229, February 1971.

[11] S. Kotikalapudi et al, "Discrete-Time Analysis of Integrated Voice/Data Multiplexers with and without Speech Activity Detectors", *IEEE J. Select. Areas Commun.*, SAC-1(6):1124–1132, December 1983.

[12] T. Suda et al, "Performance Evaluation of an Integrated Access Scheme in a Satellite Communications Channel", *IEEE J. Select. Areas Commun.*, SAC-1(1):153–164, January 1983.

[13] E.W.M. Wong and T. Yum, "A Controlled Multiple Access Protocol for Packet Satellite Communications", *IEEE Trans. Commun.*, COM-39:1133-1140, July 1991.

[14] G.F. Williams and A. Leon-Garcia, "Performance Analysis of Integrated Voice and Data Hybrid Switched Links", *IEEE Trans. Commun.*, COM-32(6):695-706, June 1984.

[15] T.T. Ha, *"Digital Satellite Communications"*, McGraw Hill Inc., 1990.

[16] H. Ahmadi and T.E. Stern, "A Combined Fixed and Demand Assignment Satellite Multiple Access Protocol for Integrated Circuit and Packet Switching", *Proc. IEEE Int'l Conf. Commun.*, pages 17.5.4.1–17.5.7, 1986.

[17] H. Heffes and D.M. Lucantoni, "A Markov Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance", *IEEE J. Select. Areas Commun.*, SAC-4(6):856–868, September 1986.

[18] J.Y. Hui, *"Switching and Traffic Theory for Integrated Broadband Networks"*, Kluwer Academic Publishers, Norwell, Massachusetts, 1990.

[19] J. Bellamy, *"Digital Telephony"*, John Wiley and Sons, Inc., 1991.

[20] J.E. Ohlson and R.J. Huff, "Multi-Frequency TDMA for Satellite Communications", *Proc. IEEE Int'l Conf. Commun.*, pages D1.3.1–D1.3.5, 1983.

[21] J.F. Hayes, *"Modeling and Analysis of Computer Communications Networks"*, Plenum Press, New York, 1984.

[22] J.I. Mohammed and T. Le-Ngoc, "Performance Analysis of Combined Free/Demand Assignment Multiple Access (CFDAMA) Protocol for Packet Satellite Communications", *Proc. IEEE Int'l Conf. Commun.*, pages 869–873, May 1994.

[23] J.L. Dicks and M.P. Brown Jr, "Frequency Division Multiple Access (FDMA) for Satellite Communication Systems", *IEEE-EASCON Record '74*, pages 167–178, October 1974.

[24] G.A. Codding Jr, *"The Future of Satellite Communications"*, Westview Press Inc., 1990.

[25] H.W. Lee and J.W. Mark, "Combined Random/Reservation Access for Packet Switched Transmission over a Satellite with On-Board Processing: Part-I Global Beam Satellite", *IEEE Trans. Commun.*, COM-31:1161–1171, October 1983.

[26] L.G. Roberts, "Aloha packet system with and without slots and capture", *Computer Communications Review*, 5:28–42, April 1975.

[27] J.H. Lodge, "Mobile satellite communications systems: Toward global personal communications", *IEEE Communications Mag.*, 29(11):24-30, November 1991.

[28] J.I. Mohammed, *"Combined Free/Demand Assignment Multiple Access Protocols for Packet Satellite Communications"*, M.ASc Thesis, Concordia University, Montreal, CANADA, June 1993.

[29] M. Schwartz, *"Telecommunications Networks: Protocols, Modelling and Analysis"*, Addison Wesley Publishing Company, 1987.

[30] M. Yabusaki, "Reassignment Algorithm in Multiple Carrier Hopping Demand Assigned TDMA System", *8th Int.Conf.Digital. Sat. Comm*, 1989.

[31] N. Abramson, "The ALOHA System: Another alternative for Computer Communications", *Proc. AFIPS Conf.*, 37:281–285, 1970.

[32] N. Abramson, "Fundamentals of Packet Multiple Access for Satellite Networks", *IEEE J. Select. Areas Commun.*, 10(2):309–316, February 1992.

[33] O.G. Gabbard and P. Kaul, "Time Division Multiple Access", *IEEE-EASCON Record '74*, pages 179–184, October 1974.

[34] P. Wood, "Mobile satellite services for travellers", *IEEE Communications Mag.*, 29(11):32–35, November 1991.

[35] R.K. Goel and A.K. Elhakeem, "A Hybrid FARA/CSMA-CD Protocol for Voice Data Integration", *Computer Networks Journal*, 9(3):223–240, March 1985.

[36] S. Kotikalapudi and W. Whitt, "Characterization of Superposition Arrival Processes in Packet Multiplexers for Voice and Data", *IEEE J. Select. Areas Commun.*, SAC-4(6):833–846, September 1986.

[37] S. Li and J.C. Majithia, "Performance Analysis of a D-TDMA Local Area Network for Voice and Data", *Computer Networks Journal*, 8(2):81–91, April 1984.

[38] S. Tasaka, *"Performance Analysis of Multiple Access Protocols"*, The MIT Press, 1986.

[39] S. Tasaka and Y. Ishibashi, "A reservation protocol for satellite packet communication - a performance analysis and stability considerations", *IEEE Trans. Commun.*, COM-32:920-927, August 1984.

[40] S.V. Krishnamurthy and T. Le-Ngoc, "Performance of CFDAMA-RA in Packet Satellite Communications", *to appear in ITS'94, Rio De Janeiro, Brazil*, August 1994.

[41] T. Le-Ngoc, "Dynamic Resource Allocation Schemes for Multimedia Satellite Communications", *The Fourth International Symposium on Personal, Indoor and Mobile Radio Communications, (PIMRC-93), Yokohoma, Japan*, September 1993.

[42] T. Le-Ngoc and J.I. Mohammed, "Combined Free/Demand Assignment Multiple Access (CFDAMA) Protocols for Packet Satellite Communications", *IEEE International Conference on Universal Personal Communications*, October 1993.

[43] T. Le-Ngoc and Y. Yao, "CREIR, A Multiple Access Protocol for On-Board Processing Satellite Systems", *Can. Conf. on ECE, Quebec*, 1991.

[44] V. Li and T. Yan, "An Integrated Voice and Data Multiple-Access Scheme for a Land-Mobile Satellite System", *Proc. IEEE*, 72(11), November 1984.

[45] W. Lidinsky and D. Vlack, *"Perspectives on Packetized Voice and Data Communications"*, IEEE Press, 1990.